МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЧЕРКАСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ім. Богдана Хмельницького

Авраменко В.С., Салапатов В.І.

ВСТУП ДО ПРОГРАМНОЇ ІНЖЕНЕРІЇ

Том 1. Історія розвитку. Основні поняття

Навчальний посібник

Друге видання

Черкаси 2014

УДК 004(075.032)
ББК 018*32.973я723

Рецензенти:
Чл.-кор. АН України, д.т.н., проф. *Васильєв В.В.*
д.т.н., проф. *Литвинов В.В.*

Рекомендовано до друку Вченою радою
Черкаського національного університету імені Богдана Хмельницького
(протокол №2 від 15.11.2011 р.)

Авраменко В.С., Салапатов В.І.

Вступ до програмної інженерії. Том 1. Історія розвитку. Основні поняття. Навчальний посібник. Друге видання. – Черкаси: ЧНУ імені Богдана Хмельницького, 2014, – 500 с.
ISBN 978-966-353-261-5

Мета даного посібника – сформувати уявлення про професію програміста та закласти основи для отримання навичок в області обчислювальної техніки, комп'ютерних наук та програмування, розвинути професійний світогляд майбутніх програмістів та адміністраторів комп'ютерних систем та мереж, орієнтувати їх у бурхливому вирі нових ідей та технологій апаратного та програмного забезпечення, операційних систем і мов програмування.

У посібнику розглядаються вміст основних понять програмування, історії їхнього розвитку, початкові відомості з інформатики, основні елементи мов програмування, принципи структурного програмування та основи програмування у середовищі Turbo Pascal.

Посібник розбитий на два томи. В першому томі надається еволюція розвитку комп'ютерної техніки та основи інформатики. У другому томі йдеться про основи програмування взагалі та конкретне застосування мови Pascal при розробці програм зокрема. Наводяться численні приклади перевірених програм.

Призначено для студентів ВНЗ, які навчаються за спеціальностями по напрямах підготовки "Комп'ютерні науки", "Програмна інженерія", "Комп'ютерна інженерія", "Інформатика та обчислювальна техніка", для студентів інших споріднених спеціальностей, а також учнів старших класів шкіл, технікумів, коледжів з поглибленим вивченням інформатики.

У другому виданні посібника виправлені деякі помилки, а також доопрацьовані, доповнені та додані нові розділи.

УДК 004(075.032)
ББК 018*32.973я723

ISBN 978-966-353-261-5

© ЧНУ ім. Б. Хмельницького, 2014
© Авраменко В.С., Салапатов В.І., 2014

ЗМІСТ

ЧАСТИНА І. ІСТОРІЯ РОЗВИТКУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 15
Вступ..... 15
1. Передвісники комп'ютерної ери 16
1.1. Комп'ютерна програма – що це?..... 16
1.2. Доелектронна історія обчислювальної техніки..... 18
1.3. Можливості двійкового коду 27
1.4. Розвиток двійкової системи..... 27
1.5. Винахід перших комп'ютерів..... 30
1.6. Гарвардська архітектура 34
1.7. Архітектура фон Неймана 36
1.8. Створення зрозумілих людині кодів 43
1.9. Крок на благо програмування 45
1.10. Можливості програмного управління..... 46
2. Нові мови програмування..... 49
2.1. Поневіряння пакетної обробки..... 50
2.2. Універсальна мова програмування 51
2.3. Усунення неоднозначності 53
2.4. Заклик до дотримання математичної строгості..... 54
2.5. Пошук та усунення помилок 55
2.6. Нелегке мистецтво програмування 55
2.7. Обчислювальна техніка та програмування в СРСР 56
3. Розквіт та хаос програмного забезпечення..... 67
3.1. Місце народження хакерів..... 70
3.2. Два чародії програмування..... 71
3.3. Перші промислові стандарти..... 72
3.4. Дружній інтерфейс..... 83
3.5. Прообраз сучасного «ноутбука» 87
4. Болісний шлях розвитку програмування..... 90
4.1. Плануюче обчислення 90
4.2. Внесок Великої Британії..... 91
4.3. Програмування англійською мовою 92
5. Три комерційні гіганти..... 94
5.1. Перша комерційна мова програмування 94
5.2. Обчислювальна техніка приходить у бізнес..... 95
5.3. Народження CODASYL..... 96
5.4. Конференція в Цюріху..... 98
5.5. На шляху до сумісності комп'ютерів 99
5.6. Розбіжності Нового Світу..... 101
6. Десятиліття динамічного розвитку 104
6.1. Перші кроки непроцедурної мови..... 104

6.3. Алфавітне хрещення 106
6.4. Успіх та суперечки 107
6.5. Інженерний підхід 108
6.6. Структурний підхід 109
6.7. Поява мови “Ада” 110
7. Програмування приходить у наші домівки 113
7.1. Розквіт Бейсіка 113
7.2. Поява мови Модула-2 114
7.3. Музикальний француз 115
7.4. Довгожитель Lisp – інструмент функціонального програмування 116
7.5. Prolog – нездійснена мрія ЕОМ V покоління 117
7.6. Революція на ім’я Java 120
8. Історія і шляхи розвитку супер-ЕОМ 125
8.1. Усе починалося з менфреймов 125
8.2. Напрями розвитку обчислювальної техніки 128
8.3. Розвиток елементної бази. Закон Мура 129
8.4. Вдосконалення архітектури 132
8.5. Топ-500 – рейтинг супер-ЕОМ 137
9. Сучасний стан та перспективи розвитку програмування 142
9.1. Криза у програмуванні 143
9.2. Методологія процедурно-орієнтованого програмування 143
9.3. Методологія об’єктно-орієнтованого програмування 144
9.4. Методологія об’єктно-орієнтованого аналізу та проектування 145
9.5. Технології програмування 146
9.6. CASE –засоби 148
9.7. Методологія RAD 149
9.8. Microsoft Visual Basic 151
9.9. Borland Delphi 152
9.10. Clarion 153
9.11. Microsoft Visual Studio LightSwitch 2012 155
9.11.1. Знайомство с LightSwitch 156
9.11.2. Архитектура LightSwitch 157
9.11.3. Створення проекту в Microsoft Visual Studio LightSwitch 158
9.11.4. Дванадцять основних переваг LightSwitch 159
9.12. Огляд WinDev 161
9.12.1. ПризначенняWinDev 161
9.12.2. Деякі характеристики WLanguage 164
9.13. Технологія MODEL CHECKING 165
9.14. NeoBook – программирование для непрограммистов 167
9.14.1. Вступ для секретарок 168
9.14.2. NeoBook 168
9.14.3. Можливості та області застосування 169
9.15. Файлові системи найближчого майбутнього 171
9.15.1. Зетта-повінь настає 171

9.15.2. Файлова система ZFS 174
9.15.3. Файлова система Btrfs 174
9.15.4. Файлова система Hammer 175
10. Історія операційних систем 177
10.1. Послідовна обробка даних 177
10.2. Прості пакетні системи 178
10.3. Багатозадачні пакетні системи 180
10.4. Системи з режимом розподілу часу 182
10.5. Основні досягнення 183
10.6. Сучасні системи UNIX 192
10.7. OS/2. Битва двох гігантів 192
Список літератури 194
ЧАСТИНА ІІ. ІСТОРІЯ КОМП'ЮТЕРНИХ МЕРЕЖ 196
Вступ 196
1. Передісторія сучасних комп'ютерних мереж 198
1.1. Покоління комп'ютерних мереж 198
1.2. Перші кроки телеобробки 198
1.3. Телеобробка у 60-і та 70-і роки 199
1.4. Особливості мереж із комутацією каналів 200
2. Мережі пакетної комутації 202
2.1. Великий ідеолог Internet 202
2.2. Принцип комутації повідомлень та пакетів 204
2.3. День народження першої комп'ютерної мережі (70-ті роки) 206
2.4. Розвиток мереж пакетної комутації 208
2.5. Виникнення Internet (80-ти роки) 210
2.6. Комплексне інформаційне обслуговування в Internet (90-і роки) 211
2.7. Internet нового покоління 213
3. Локальні обчислювальні мережі 215
3.1. Широкомовні мережі 215
3.2. Технологія Ethernet 217
4. Мережні послуги 219
4.1. Віддалений доступ до ЕОМ 219
4.2. Передача файлів 221
4.3. Електронна пошта 222
4.4. Групи новин, форуми 223
4.5. Чат та миттєві повідомлення 225
4.6. Портали 227
4.7. Передача мультимедіа 227
4.8. Використання інформаційних ресурсів 230
4.9. Послуги у вигляді cookies 231
5. Web-революція 233
5.1. Ідея гіпертекстових інформаційних систем 233
5.2. Універсальне сховище знань 234
5.3. Реалізації документальних гіпертекстових систем 234

5.4. Народження Web ... 236
5.5. Перші версії браузерів ... 241
5.6. Веб-сайт ... 243
5.7. Війна браузерів ... 244
6. Пошук в Internet ... 246
6.1. Internet-каталоги ... 246
6.2. Пошукові системи ... 247
6.2.1. Пошукові машини ... 247
6.2.2. Пошукова система Google ... 249
6.2.3. Пошукова система Яндекс ... 255
6.2.4. Пошукова система Rambler ... 258
6.2.5. Синтаксис мови запитів ... 259
6.2.6. Додаткові можливості пошукових систем ... 261
6.2.7. Яка із пошукових систем шукає краще ... 261
7. Internet-торгівля ... 263
8. Електронна пошта ... 266
9. Сервіси сучасного Internet ... 268
9.1. Блоги ... 268
9.2. Internet-пейджери ... 270
9.3. Skype ... 271
9.4. Дистанційне навчання ... 271
9.5. Пошук роботи ... 272
9.6. Робота через Internet ... 272
9.6.1. Оформлення резюме ... 273
9.6.2. Структура резюме ... 273
9.6.3. Про що писати не потрібно ... 274
9.6.4. Супровідний лист ... 274
9.7. Оплата через Internet ... 275
9.8. Заміна десктопних додатків ... 276
9.9. Словники ... 276
9.10. Газети та новини. Журнали. Карти ... 277
10. Що ще необхідно знати про Інтернет? ... 278
10.1. Правила хорошого тону ... 278
10.2. Приватна власність ... 278
10.3. Сервер, сайт, домашня сторінка ... 279
11. Як зробити сайт самому ... 281
11.1. Зробити сайт самому на безкоштовному хостингу ... 281
11.2. Зробити сайт самому, але з невеликими вкладеннями ... 282
11.3. Розкрутка сайту ... 283
11.3.1. Пошукові системи ... 283
11.3.2. Інші способи розкрутки ... 284
11.3.3. Платні послуги ... 285
12. Для чого потрібен торрент? ... 287
12.1. Що таке торрент-клієнт? ... 287

12.2. Як працює торрент? 287
12.3. Що таке торрент-трекер 288
Список літератури 288
ЧАСТИНА ІІІ. ОСНОВИ ІНФОРМАТИКИ 290
Вступ 290
1. Загальні відомості про комп'ютер 292
1.1. Як влаштований комп'ютер 292
1.2. Архітектура комп'ютера 295
1.3. Монітори 296
1.4. Відеокарти 300
1.5. Процесори 302
1.6. Системна (материнська) плата 307
1.7. Пам'ять комп'ютера 310
1.8. Дисководи 317
1.9. CD-ROM, CD-RW, DVD 318
1.10. Порти 319
1.11. Переносні запам'ятовуючі пристрої 320
1.12. Принтери 323
1.13. Миша 326
1.14. Сканери та камери 328
1.15. Модеми 329
1.16. Звукова карта. Аудіо система 330
1.17. Мережна карта 331
1.18. Бездротові з'єднання (Wi-Fi-мережі, Bluetooth) 332
1.19. Плоттери (графічні пристрої) 333
1.20. Графічний планшет 334
1.21. Проекційна техніка 334
1.22. Системна шина 335
1.23. Кнопки та клавіші 337
2. Класифікація комп'ютерів 342
2.1.Класифікація комп'ютерів за поколіннями 342
2.2. Класифікація комп'ютерів за застосуванням 344
2.2.1. Основні області застосування комп'ютерів 344
2.2.2. Гіпертекст 346
2.2.3. Мультимедіа 346
2.2.4. Комп'ютерні мережі 347
3. Поняття інформатики та інформації 349
3.1. Що таке інформатика? 349
3.2.Що таке інформація? 351
3.3. Інформаційна модель 354
4. Дані 357
4.1.Носії даних 357
4.2. Операції з даними 357
4.3. Кодування даних 358

4.3.1. Кодування даних двійковим кодом 359
4.3.3. Універсальна система кодування текстових даних 360
4.3.4. Кодування графічних даних 361
4.3.5. Кодування звукової інформації 362
4.4. Основні структури даних 363
5. Файли та файлова структура 365
5.1. Одиниці вимірювання даних 365
5.2. Одиниці зберігання даних 366
5.3. Поняття про файлову структуру 367
6. Системи числення 374
6.1. Загальні уявлення 374
6.2. Римська система числення 375
6.2.1. Читання римських цифр 376
6.2.2. Запис римських цифр 376
6.3. Системи числення, що використовуються в обчислювальній техніці 377
6.4. Перетворення між системами числення 379
6.4.1. Перетворення цілих чисел за схемою Zp → Z1 → Zq 379
6.4.2. Перетворення цілих чисел за схемою $Z_p \rightarrow Z_{10} \rightarrow Z_q$ 380
6.5. Перетворення між двійковою, 8-ковою та 16-ковою системами 383
6.6. Перетворення правильної дробі 384
6.7. Швидкий алгоритм перетворення 387
6.8. Перетворення нормалізованих чисел 388
6.9. Кодовані позиційні системи числення 393
6.10. Поняття економічності системи числення 395
7. Арифметичні операції в позиційних системах числення 399
7.1. Додавання 399
7.2. Віднімання 400
7.3. Множення 401
7.4. Ділення 409
8. Подання чисел у комп'ютері 413
8.1. Поняття машинного слова 413
8.2. Подання цілих чисел у комп'ютері 415
8.2.1. Цілі числа без знаку 415
8.2.2. Цілі числа зі знаком 417
8.3. Числа з фіксованою крапкою 425
8.4. Числа з плаваючою крапкою 426
8.5. Внутрішні формати дійсних даних 428
8.6. Виконання арифметичнх дій над нормалізованими числами 437
8.6.1. Додавання та віднімання 437
8.6.2. Множення 438
8.6.3. Ділення 438
8.7. Перетворення цілих та дійсних даних при введенні/виведенні у внутрішнє уявлення 439
8.7.1. Перетворення цілих даних при введенні у внутрішнє подання 439

8.7.2. Перетворення цілих даних із внутрішнього подання у зовнішнє при виведенні 439
8.7.3. Перетворення дійсних даних при введенні у внутрішнє подання 440
8.7.4. Перетворення дійсних даних із внутрішнього подання у зовнішнє при виведенні 441
8.8. Округлення і точність представлення чисел в ЕОМ 442
8.8.1. Похибки наближених чисел 442
8.8.2. Округлення і точність представлення чисел 444
9. Поняття алгоритму 449
9.1. Що являє собою алгоритм? 449
9.2. Що являє собою "Виконавець алгоритму"? 451
9.3. Які властивості притаманні алгоритмам? 451
9.4. Способи завдання запису алгоритмів 452
9.5. Основні види обчислювальних процесів 456
10. Програмне забезпечення комп'ютера 458
10.1. Системне програмне забезпечення 458
10.2. Що являє собою операційна система? 460
10.3. Програми для роботи з текстами 461
10.4. Графічні редактори 463
10.5. Музичні та звукові редактори 464
10.6. Бази даних, електронні таблиці, бухгалтерські програми 466
10.7. Телекомунікаційні програми 467
10.8. Ігри 468
11. Програмування та мови програмування 471
11.1. Що являє собою програмування? 471
11.2. Мови програмування 473
11.2.1. Як «влаштована» мова програмування 475
11.2.2. Типи мов програмування 478
11.2.3. Парадигми програмування 480
11.3. Як навчитися програмувати? 485
11.3.1. Програмування – це просто? 485
11.3.2. Основні прийоми програмування 486
12. Технологія розв'язання завдань за допомогою комп'ютера 490
12.1. Етапи розв'язання завдань за допомогою комп'ютера 490
12.2. Постановка завдання 490
12.3. Побудова математичної моделі 491
12.3. Які основні етапи містить процес розробки програм? 492
12.4. Як проконтролювати текст програми до виходу на комп'ютер? 493
12.5. Для чого потрібні відлагодження та тестування? 494
12.6. У чому полягає відлагодження? 494
12.7. Що являє собою тест та тестування? 495
12.8. Якими повинні бути тестові дані? 495
12.9. Які характерні помилки програмування? 496

12.10. Чи є відсутність синтаксичних помилок доказом правильності програми? 497
12.11. Які помилки не виявляються транслятором? 497
12.12. У чому полягає супровід програми? 498
Список літератури 499

Передмова

Можна з упевненістю сказати, що сьогодні знання основ інформатики та обчислювальної техніки стали необхідними кожній людині як у переважній більшості областей професійної діяльності, так і в побуті, та по-справжньому є загальноосвітніми. Тому мета цього посібника – звернутися до минулого, зануритися в історію розвитку апаратного та програмного забезпечення і зрозуміти, як воно розвивалося і чому ще й досі не існує загальних технологій, які дозволили б усім розробникам створювати надійне програмне забезпечення з мінімальними витратами і за прийнятний проміжок часу, полегшити вивчення та використання основ інформатики та мов програмування.

Цей посібник за своїм змістом істотно відрізняється від підручників і посібників для початківців. Він виник на основі курсу лекцій для студентів першого курсу факультету обчислювальної техніки, інтелектуальних та управляючих систем Черкаського національного університету ім. Б. Хмельницького «Вступ до програмної інженерії» та «Основи програмування та алгоритмічні мови», які обрали програмування та адміністрування комп'ютерних систем та мереж своєю професією.

Студенти першого курсу після закінчення школи в цілому обізнані з побудовою комп'ютера та основами програмування, мають навички роботи з типовими пакетами програм. Вони із захопленням читають спеціальну літературу, сперечаються про переваги та недоліки новітніх мікропроцесорів та програм. Проте, в цілому, їх знання поверхневі та носять уривчастий характер, оскільки вони відірвані із загального контексту історії розвитку обчислювальної техніки та ідей інформатики та програмування.

Метою курсу «Вступ до програмної інженерії» і цього посібника зокрема є розвиток професійного світогляду майбутніх програмістів та адміністраторів комп'ютерних систем і мереж, орієнтація їх у бурхливому вирі нових ідей та технологій апаратного та програмного забезпечення, операційних систем та мов програмування.

Звідси витікає обраний нами історичний підхід до викладення матеріалу. Дивлячись на новинки, які з'являються щодня, легко розгубитися. Неосвідченій людині може здатися, що вся ця нескінченна різноманітність хаотична і абсолютно непередбачувана. Насправді це далеко не так. Найдивовижніше в інформатиці – не швидка мінливість, а, навпаки, дивовижна стійкість фундаментальних концепцій.

Електронним обчислювальним машинам не більше шестидесяти п'яти років. За цей час вони невпізнанно змінилися зовні, у мільйони разів збільшилася швидкість їх роботи та розмір пам'яті, проте основні принципи апаратного та програмного забезпечення залишилися багато у чому практично незмінними.

Темпи розвитку ЕОМ воістину фантастичні. Ще в 1984 р. американські газети писали:

«У 1953 р. ЕОМ з пам'яттю 64 Кбайт коштувала 1 млн дол, зараз вона коштує менше 1 тис. дол. Якби автомобілі розвивалися протягом останніх 20 років тими ж темпами, як комп'ютери, то сьогодні роллс-ройс коштував би 3,0 дол, проходив мільйон миль на галоні бензину, розвивав потужність лайнера «Queen Elisabeth» і 2 автомобілі поміщалися б на кінчику пера».

У 21 столітті темпи ще вище. Наприклад, якщо мікропроцесор Pentium IV «Northwood» (2002 р.) містив 42 млн транзисторів, то Pentium IV «Prescott» (2004 р.) – 125 млн [17].

Програмне забезпечення є «тінню» свого «старшого брата». Судіть самі: найпопулярніші мови програмування та алгоритми компіляції були розроблені на початку 60-х років минулого століття. Тоді ж були сформульовані принципи роботи операційних систем. Системи управління базами даних (СУБД) з’явилися на початку 70-х років і з тих пір майже не змінилися. Навіть наймодніша нині концепція об’єктно-орієнтованого програмування була запропонована, виявляється, ще в середині 60-х років минулого століття (мова Smalltalk). Наведені приклади свідчать про те, що принципово нові ідеї з’являються в інформатиці, як, втім, і в інших науках, відносно рідко, і в найновітнішому та розрекламованому пакеті програм, якщо добре придивитися, можна побачити добре забуте старе.

Правда, за останній час, завдяки новим технологіям, широке використання баз даних різними категоріями користувачів призвело, з одного боку, до створення інтерфейсів, що вимагають мінімум часу на освоєння коштів управління системою, а з іншого – до побудови потужних, гнучких СУБД, що мають у тому числі розвинені засоби захисту даних від випадкового або навмисного руйнування. З'явилися і засоби автоматизації розробки, що дозволяють створити базу даних будь-якому користувачеві, навіть якщо він не володіє основами теорії БД.

Можливості накопичувати і оперативно обробляти великі обсяги інформації, що характеризують діяльність підприємств за досить тривалі періоди і в різних аспектах, дали новий імпульс до розвитку аналітичних систем. Такого роду системи підтримки прийняття рішень зазвичай використовуються для оцінки та вибору альтернативних рішень, прогнозування, ідентифікації об'єктів і станів тощо.

У першій частині посібника висвітлюється історія обчислювальної техніки та еволюція розвитку програмного забезпечення, як основного базису інформатики – від простих механічних пристроїв до сучасних комп’ютерів, від програмування на комутаційних дошках (пультах) до сучасних технологій програмування.

У другій частині розглядаються принципи побудови та напрями еволюції сучасних комп’ютерних мереж, коротко описана їх історія, основні поняття теорії та тенденції розвитку комп’ютерних мереж.

У третій частині описуються загальні відомості про комп'ютер, інформація, алгоритми та мови програмування.

Четверта частина присвячена основам програмування в середовищі Паскаль, як початкового (базового) засобу навчання програмуванню для студентів. Ми вважаємо, що вивчення будь-якої мови програмування проходить найуспішніше, коли кожне питання розглядається з різних боків. Тому цей посібник містить опис не тільки різних базових елементів мови Паскаль, але і велику кількість прикладів (близько 170) для закріплення матеріалу, самоконтролю, експериментування та аналізу своїх помилок.

На початковому етапі задачі навчання пов'язані не стільки з мовою, скільки з виробленням алгоритмічного мислення. Ці завдання багатогранні та достатньо складні. Потрібне розуміння нових понять: змінних та типів даних. Потрібне вміння описувати процес обчислення обмеженим набором управляючих структур – вибору, циклів, рекурсії та лінійних блоків. Потрібне знання основ подання даних та класичних алгоритмів. І, нарешті, потрібно не тільки знати, але й вміти писати програми!

Мета даної частини посібника – не зробити з вас високопрофесійних програмістів, бо для цього необхідно цілеспрямовано вчитися і створювати багато різних програм, а засвоїти основні поняття програмування, набути початковий досвід самостійної розробки програм та ознайомитись з деякими класичними завданнями та алгоритмами.

Велика увага приділена стилю програмування. Питання дійсно дуже важливе особливо для програміста-початківця. Немає нічого дивного у тому, що різні програмісти дотримуються різних точок зору при написанні програм. Деякі прагнуть добитися, щоб програма містила мінімальну кількість операторів або виконувалася за мінімальний час. Точка зору на програмування, що наводиться у даному посібнику, ґрунтується на іншому критерії якості програм. Хороша програма відповідно до цього критерію – це програма, яка написана так, щоб вона була зрозумілою іншим людям і водночас, по можливості, містила мінімальну кількість операторів та виконувалася за мінімальний час. Більш докладно це означає, що програміст повинен писати програму так, щоб її було легко читати, і, щоб її було легко використовувати.

Цілі, які відмічені вище, можуть показатися зрозумілими, проте у реальному житті вони не завжди легко досягаються. Основне завдання, яке полягає у написанні хорошої програми, вимагає для свого розв'язку серйозного та зваженого ставлення. При розробці програми часто виникає спокуса принести ясність у жертву незначному підвищенню ефективності. На щастя, мова Паскаль містить гарні засоби проектування зрозумілих програм. Її систематизована структура накладає серйозні обмеження на дисципліну програмування, і у той же час дозволяє розширювати мову за рахунок введення нових можливостей, які задовольняють спеціальним застосуванням. Проте, слід пригадати «аксіому» програмування: «**Немає, не було, та, скоріше за все, не буде мови програмування, яка б гарантувала нас від написання поганих програм**».

Ми намагались написати цей посібник без складних слів та виразів, звичною мовою, яка зрозуміла кожному. Робота з посібником не вимагає попередніх знань з інформатики та програмування.

Автори усвідомлюють також те, що цей посібник не повністю відображає матеріал, який необхідний для вивчення основ інформатики. За його межами опинився опис цікавих інформаційних технологій, але цим матеріалом довелося пожертвувати через обмежений об’єм посібника.

У другому виданні посібника виправлені деякі помилки, доопрацьовані, доповнені та додані нові розділи.

ЧАСТИНА І. ІСТОРІЯ РОЗВИТКУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Вступ

Програмне забезпечення (ПЗ) розробляють вже більше шістдесяти років, але й дотепер програми, які мають багато помилок, залишаються нормою, а якісні рішення – рідкісним виключенням. Різноманітність дефектів вражаюча: від проблеми 2000 року щодо вад захисту, не кажучи вже про безліч катастрофічних програмних помилок. Нам слід звернутися до минулого, зануритися в історію розвитку ПЗ та зрозуміти, як воно розвивалося, і чому на сьогодні не існує загальних технологій, які дозволили б усім розробникам створювати надійне програмне забезпечення з мінімальними витратами та в прийнятний час.

На початку XXI століття є сенс проаналізувати 60 років, що промайнули. Перші експерименти, які можна віднести до сучасного програмування, проводилися ще під час Другої світової війни. Але саме 50-і роки стали першим десятиліттям розвитку програмування як галузі. За цей період, включаючи початок нового тисячоліття, буквально на наших очах кардинально змінилося коло завдань, яке здатне вирішувати програмне забезпечення, та форми подання таких рішень.

У не меншій мірі змінилися методи роботи та відношення до програмування самих розробників. Технологічні досягнення в апаратному забезпеченні, операційних системах та мовах програмування допомогли сформувати середовище розробки. Проте соціальні та економічні чинники зіграли важливішу роль, оскільки саме вони визначали, яким чином галузь адаптувала ці досягнення, хто, зрештою, став їх використовувати, і як вони впливають на можливість створювати якісне програмне забезпечення.

Ретроспективний огляд програмного забезпечення неможливий без огляду апаратного забезпечення. Тому паралельно з оглядом ПЗ ми розглянемо основні історичні досягнення в розвитку обчислювальної техніки.

Історія обчислювальних машин (тепер прийнято говорити – комп'ютерів) налічує також близько шести десятків років, проте у цій історії є передісторія завдовжки у декілька століть. Прості пристрої, які полегшують обчислення, з'явилися у глибокій давнині, декілька тисячоліть тому. По мірі розвитку людської цивілізації вони поволі еволюціонували, безперервно вдосконалюючись. Але тільки у 40-і роки XX століття було покладено початок створенню комп'ютерів із сучасною архітектурою та сучасною логікою. Саме ці роки можна по праву вважати часом народження сучасних обчислювальних машин. З тих пір за історично дуже короткий термін комп'ютери, завдяки величезним успіхам мікроелектроніки, пройшли такий шлях у своєму технічному вдосконаленні та масштабах застосування, з яким не порівняється жоден інший винахід людини, включаючи атомну енергію та космічну техніку. Та й останні не отримали б такого могутнього розвитку без досягнень в галузі обчислювальної техніки. Але жодні великі винаходи не народжуються на порожньому місці, у них завжди є попередники та прабатьки. **Відомо, що прогрес розвивається по спіралі, і все нове – це добре забуте старе**.

1. Передвісники комп'ютерної ери

1.1. Комп'ютерна програма – що це?

Придбавши комп'ютер, багато непрофесіоналів обчислювальної техніки цікавляться питанням, яким чином вдається примусити комп'ютер вирішувати різноманітні задачі? Не випадково однією з найперших назв цього пристрою було *універсальна цифрова обчислювальна машина*. При всій своїй громіздкості така назва точніше передає сенс та призначення комп'ютера, підкреслюючи, що, в принципі, на ньому можна розв'язувати будь-яку задачу (зрозуміло, якщо існує спосіб її розв'язання). Проте, на тільки-но виготовленому (або купленому) комп'ютері нової моделі, мало що можна зробити, хоча потенційно він є дуже гнучким та могутнім інструментом.

Але що наділяє комп'ютер здатністю розв'язувати різноманітні задачі, що робить цю розумну машину несхожою на всі інші машини та механізми, що були створені людьми за довгу історію людської цивілізації? Потужність комп'ютера полягає у його **програмному забезпеченні**, і щоб він почав діяти, потрібна ще досить велика робота з написання серії програм (або з придбання готових), які здатні надихнути життя у комп'ютер. Причому для кожного нового завдання доводитися складати свою програму. Ця робота (робота програміста) вельми специфічна і не має аналогів у жодній іншій області діяльності людини. Насправді, об'єм програм, що забезпечують вирішення комп'ютером будь-якого комплексу завдань, сягає сотень мільйонів бітів, і достатньо лише в одному місці випадково переплутати 0 та 1, як все піде шкереберть: комп'ютер або відмовиться розв'язувати задачу, або почне видавати різну нісенітницю, або "безмовно" працюватиме і працюватиме, поки його не зупинять.

У перші десять – двадцять років існування комп'ютера основні витрати фірм-виробників були пов'язані із розробкою та виробництвом власно машин. У ті часи питанням програмування приділялося значно менше уваги, вони залишалися так би мовити на другому плані. Потім картина різко змінилася. З одного боку, грандіозні успіхи електроніки ведуть до здешевлення самих машин, а з іншого – значні сфери застосування комп'ютерів вимагають все більшого обсягу робіт з програмування. В результаті витрати на розробку програмного забезпечення сягають нині вже **90%** [2, 14] від загальної вартості обчислювальних систем і частка їх продовжує зростати, а професія програміста у наші дні стала однією з найпопулярніших та престижніших, що мають постійний попит у переважній більшості застосувань людської діяльності.

Складні програми дозволяють комп'ютеру управляти польотами космічних апаратів, виступати в ролі експерта, а також виконувати складні математичні розрахунки. Програма – це засіб, за допомогою якого людина диктує машині свою волю, примушує її виконувати потрібну їй роботу. Але щоб вступити в діалог з машиною, людині довелося створити спеціальні мови – так звані **мови**

програмування, які стали універсальним засобом спілкування програмістів усього світу.

Якщо представити комп'ютер як музичний інструмент, то ПЗ – це нотний запис. Термін **"програмне забезпечення" (software**) широко увійшов до вжитку з початку 60-х років 20 ст. Це було пов'язано з необхідністю провести чітку грань між командами керування комп'ютером та його фізичними компонентами, або **"апаратним забезпеченням"** ("залізо" – **hardware**), яке, власно, і складає саму машину. Вдале слово **software** (буквально – "**м'який вироб**"), яке підкреслює рівнозначність програмного забезпечення і самої машини, разом з тим говорить про його гнучкість, здатність модифікуватися, пристосовуватися, розвиватися.

В сучасних комп'ютерах команди, що записані у вигляді електромагнітних імпульсів, зберігаються на магнітних носіях. Спеціальний блок комп'ютера зчитує інформацію з магнітних носіїв та перетворює її у спеціальну форму, яка придатна для збереження у внутрішній пам'яті та подальшого виконання.

Переналаштування комп'ютера з одного виду діяльності на інший (наприклад, з розробки стандартного контракту на створення карти погоди земної кулі і т.і.) не завжди було такою простою справою. Для програмування "монстрів", що сповістили своєю появою у 40-50-і роки початок комп'ютерної ери, були потрібні не тільки математичні здібності, але і фізична витривалість. Серед перших програмістів машини ENIAC (1946 р., ENIAC – Electronic Numerical Integrator and Computer – електронний цифровий інтегратор та обчислювач), яка була створена у Вищому технічному училищі Пенсільванського університету Джоном Моучлі (Мочлі) та Преспером Еккертом, були тільки математики (більшість з них були жінки). Машина ENIAC призначалася для обчислення балістичних таблиць, яких гостро потребувала армія США під час другої світової війни. Такі таблиці давали артилеристам можливість обирати належний кут вертикального наведення гармати при заданій відстані до цілі та певній вазі снаряду з урахуванням атмосферних умов і швидкості вітру. При використанні електромеханічного настільного арифмометра кваліфікований фахівець витрачав на обчислення однієї траєкторії близько трьох днів, а для комбінації гармати та снаряда було потрібно 2-4 тис. таких розрахунків.

Для підготовки ENIAC до виконання цих операцій (зміна програми) доводилося проробляти запаморочливу процедуру з установки тисяч перемикачів та з'єднання сотень кабелів. У ENIAC використовувалося 17468 електронних ламп, сполучні кабелі були змонтовані на 40 панелях.

Програмісти, що обслуговували ENIAC, бралися за вивчення креслень, з'ясовували, як треба перемикати сотні кабелів, та міркували, в які положення встановити приблизно 6 тис. тумблерів і перемикачів. В середньому на підготовку машини для обчислення однієї таблиці витрачалося два дні. Не дивлячись на невелику продуктивність ENIAC та труднощі з підготовкою машини, саме за допомогою цих перших машин були закладені **основи програмування**.

1.2. Доелектронна історія обчислювальної техніки

Машини військового часу у конструктивному плані будувалися на ідеях минулого століття. Наприклад, машину ENIAC згодом модифікували так, щоб, принаймні, частково її можна було програмувати за допомогою перфокарт. Перфокартковий метод управління механізмом пов'язаний з ім'ям французького винахідника Жозефа Жаккара, що створив у 1804 р. ткацький верстат, для управління яким застосовувалися перфокарти, що були сполучені одна з одною у вигляді стрічки.

Історія обчислень уходить своїм корінням у глибину століть так само, як і історія розвитку людства. Накопичення запасів, ділення здобичі, обмін – всі ці дії пов'язані з обчисленнями. Для підрахунків люди використовували власні пальці, камінчики, палички, вузлики тощо.

Все почалося з пальців, які в стародавнії століття за сумісництвом працювали ще й рахунковими приладами. Правда, для складних обчислень ні пальців, ні інтелекту у стародавніх людей не вистачало – зате саме їх можна вважати першовідкривачами двійковій системи рахунку! Адже палець може бути або загнутий (логічний 0), або розігнутий (логічна одиниця). Адже «на пальцях» можна було показати всі числа від 0 до 1023 (2^{10} - 1). Тобто – в діапазоні від 0000000000 до 1111111111.

Однак цей спосіб рахунку явно здався важкуватим. Для елементарних підрахунків цілком вистачало пальців на одній руці. Тому реалізація нових проектів в галузі інформатики була відкладена на невизначений термін.

Термін визначився приблизно до п'ятого століття до нашої ери, коли у світі активно почала розвиватися торгівля. І найактивніші торговці – стародавні єгиптяни і греки – швидко виявили, що колишній «пальцевий комп'ютер» застарів і для розрахунків більш непридатний.

Потреба пошуку рішень все більш і більш складних завдань і, як наслідок, все складніших та триваліших обчислень поставила людину перед необхідністю шукати засоби, винаходити пристосування, які змогли б їй у цьому допомогти. Історично склалося так, що в різних країнах з'явилися свої грошові одиниці, міри ваги, довжини, об'єму, відстані та ін. Для переведення з однієї системи вимірювання в іншу були потрібні обчислення, які зазвичай могли проводити лише спеціально навчені люди, які досконало знали всю послідовність дій. Їх нерідко запрошували навіть з інших країн. І абсолютно природно виникла потреба у винаході пристроїв, що допомагають такому обрахунку. Так поступово стали з'являтися механічні помічники. До наших днів надійшли свідчення про багатьох таких винаходів, що назавжди увійшли до історії техніки.

Найважливішими арифметичними операціями з погляду освіченої людини стародавнього світу були додавання та віднімання. Важко уявити собі, як з ними могли б впоратися, скажімо, римляни, користуючись тільки прийнятою у них непозиційною системою числення. Саме тоді був створений **абак** (V-IV століття до н. е.), перший механічний обчислювальний пристрій на основі

примітивних кам'яних «процесорів» – лічильних каменів, які розміщувалися на розрядних лінійках.

Первинно це була дошка, посипана тонким шаром дрібного піску або порошку з блакитної глини. На ній загостреною паличкою можна було писати букви, цифри. Згодом абак був вдосконалений і обчислення на ньому вже проводилися шляхом переміщення кісток та камінчиків (з латині “камінчик” – “**calculus**”, звідси походження слів “калькуляція”, “калькулятор”) у подовжніх поглибленнях (лінійках), відповідно до окремих розрядів числа у позиційній системі числення, а самі дошки почали виготовляти з бронзи, каменю, слонячої кістки тощо.

Кожна лінійка мала значення на порядок більшу, ніж її сусідка знизу: камінчик в першій лінійці позначав 1, в другій –10, у третій – вже 100. А оскільки лінійок було багато, то можливостей абака вистачало купцям для підрахунку навіть крупнооптових партій товару.

З часом ці дошки стали розкреслюватися на декілька смуг і колонок. В Греції абак існував ще у V столітті до н. е., у японців цей прилад називався "серобян", у китайців – "суан-пан".

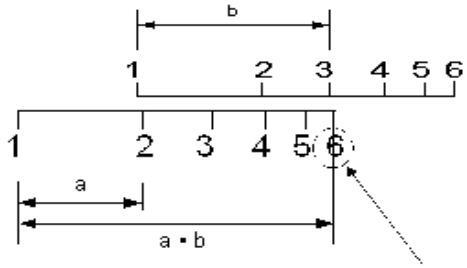
Арабські купці рознесли абак по всьому світу. Практичні китайці нанизали камінчики на спиці, і вставили їх у дерев’яну раму.

У Стародавній Русі при підрахунку застосовувався пристрій схожий на абак, і називався він "російський щот". У XVII столітті цей прилад вже мав вид звичних російських рахівниць, які ще можна зустріти і у наші дні.

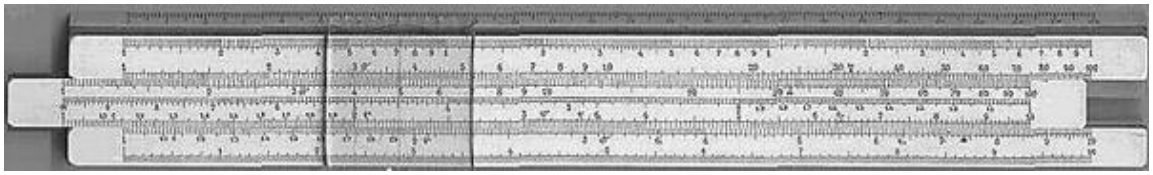
Після винаходу абака у світі настала нова смуга спокою – майже на п'ять тисячоліть. За цей час з'явилися тисячі різновидів абака – від стаціонарних до портативних, які можна було носити в кишені камзола. Однак прогрес не стояв на місці: ще в другому столітті до нашої ери у Стародавній Греції існували обчислювальні механізми, які були схожі з арифмометрами, що з'явилися через тисячу років!

Черговий технологічний прорив почався в XVII столітті – на початку «епохи науки». На зміну торговцям прийшли вчені – саме вони і стали ініціаторами створення нових обчислювальних пристроїв нового покоління. Першим із них гідний згадки шотландський математик Джон Непер (1550-1617), винахідник логаріфмів та таблиць логаріфмів (у 1590-х роках), який відмітів, що множення та ділення чисел может бути виконане додаванням та відніманням, відповідно, логаріфмів ціх чисел (log(a*b) = log(a) +log(b)). В якості додатку до свого відкриття він розробив прилад, який назвав «рахунковими паличками».

Дійсні числа можуть бути представлені інтервалами довжини на лінійці, і це лягло у 1620 році в основу обчислень за допомогою **логарифмічної лінійки**, що дозволило виконувати множення та ділення набагато швидше. Першу логарифмічну лінійку створив вже після смерті Непера англієць Роберт Біссакар.



Логарифмічні лінійки використовувалися декількома поколіннями інженерів та інших професіоналів, аж до появи кишенькових калькуляторів. Існує легенда, що інженери програми «Аполлон» відправили людину на Місяць, виконавши на логарифмічних лінійках всі обчислення, багато з яких вимагали точність в 3-4 знаки.



Логарифмічна лінійка

На відміну від абаку, на логарифмічній лінійці представляється не цифровий запис числа, а деякий його фізичний аналог. Такі аналоги є безперервними фізичними величинами (аналоговими значеннями), тому представлення завжди не точне, а наближене, оскільки неможливо точно виставити рушій лінійки на цифру, та і самі шкали мають деяку похибку. Таким чином, логарифмічна лінійка є простим прикладом **аналогової обчислювальної машини**.

Винахід Непера не відкрив нову епоху, а лише закрив стару – на зміну «пальцьовому двигуну» вже йшли перші механічні пристрої на основі зубчастих коліс. Ці пристрої на основі шестерен і зубчатиих коліс були здатні виконувати вже не дві, а чотири арифметичні дії і називалися «арифмометрами».

Цифрові (**дискретні**) та **аналогові обчислення** – це дві постійно співіснуючі та конкуруючі гілки математики. Математика стародавнього світу була переважно пов’язана з аналоговими (геометричними) побудовами. Строго кажучи, циркуль та лінійка – це теж якнайдавніша аналогова обчислювальна машина.

З винаходом символічних алгоритмів арифметики роль аналогових обчислень стала слабшати, оскільки цифрові методи здатні забезпечити вищу точність обчислень. Проте не завжди і не скрізь потрібна астрономічна точність, тому, паралельно з цифровою, розвивалася і аналогова обчислювальна техніка. У XIX та XX століттях для аналогових обчислень використовувалися різноманітні фізичні процеси: механічні, гідравлічні, а після винаходу радіоламп та транзисторів – електронні. Так, у 1930 р. професор Массачусетського технологічного інституту (MTI) Ванневар Буш (1890-1974)

побудував “диференціальний аналізатор” – велику механічну аналогову машину для розв’язку диференціальних рівнянь. Обов’язки оператора на цій машині виконував молодий аспірант **Клод Шеннон** – майбутній автор теорії інформації. З тих пір МТІ став найбільшим центром досліджень в області обчислювальної техніки та інформатики.

Ще у 1960-х роках популярність аналогових електронних обчислювальних машин була порівнянна з цифровими, їх вивчення було обов’язковим на всіх технічних факультетах. Проте після винаходу мікропроцесорів (1970-і роки), коли цифрові ЕОМ різко зменшилися за габаритами, подешевшали і стали більш потужними, аналогові ЕОМ сильно поступились у конкурентній боротьбі з ними. Стало простіше переводити аналоговий сигнал у цифрову форму, виконувати необхідні обчислення, а на виході перетворювати результат обчислень у фізичну (аналогову) величину. Таким шляхом, наприклад, пішли звукозапис та телебачення.

Ми так детально зупинилися на аналогових обчислювальних машинах для того, щоб до цієї теми більше не повертатися. Весь подальший матеріал буде присвячений проблемам інформатики та цифрових машин – їх минулому, сьогоденню та майбутньому.

Перший у світі ескізний малюнок 13-розрядного десяткового пристрою додавання на основі коліс з десятьма зубцями належить Леонардо да Вінчі. Він був описаний в одному з його щоденників.

Блез Паскаль

На початку XVII сторіччя, коли математика стала відігравати ключову роль в науці, все гостріше відчувалася необхідність у винаході рахункової машини. До цього часу відноситься створення у 1623 році **Вільгельмом Шикардом** рахуючого годинника – першого механічного калькулятора, який міг виконувати чотири арифметичні дії. Рахуючим годинником пристрій був названий через те, що як і у справжньому годиннику робота механізму була заснована на використанні зірочок та шестерінок. Практичне застосування цей винахід знайшов в руках друга Шикарда, філософа та астронома Іоганна Кеплера.

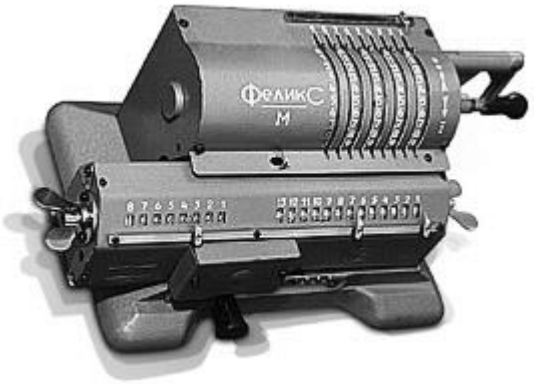
Але все ж таки першим справжним механічним рахунковим пристроєм, який існував не на папері, а працював, була побудована у 1642 році молодим французьким математиком і фізиком **Блезом Паскалем** у віці 19 років (1623-1662) рахункова машина (арифмометр), яка була названа **Паскаліной** і виконувала операції додавання та віднімання.

Головний недолік абака при виконанні розрахунків – необхідність ручного перенесення одиниці переносу у старший розряд в разі переповнення попереднього. Після довгих роздумів Паскаль винайшов 10-ти зубчасте колесо з нанесеними цифрами від 0 до 9. Поворот колеса на один зубець відповідає збільшенню значення розряду на одиницю, а при повному оберті колеса спеціальний, досить складний, механізм передачі десятків повертає наступне колесо рівно на один зубець. Кількість коліс визначала кількість розрядів – так, два колеса дозволяли рахувати до 99, три – вже до

999, а п’ять коліс робили машину такою, що «знає» навіть такі великі числа як 99999. Кажучи сучасною мовою, Паскаль винайшов богаторозрядний десятковий механічний лічильник обертів, який і дотепер використовується у спідометрах автомобілів, електролічильниках і т.і. Більш того, лічильник Паскаля з деякими вдосконаленнями став центральним пристроєм – суматором усіх подальших механічних та електромеханічних обчислювальних машин аж до середини XX століття, поки не стала застосовуватись двійкова система числення.

У 1670 – 1680 роках німецький математик Готфрід Лейбніц (1646-1716) сконструював рахункову машину, яка виконувала всі чотири арифметичні дії. Ввівши один раз перший множник, можна багато разів вводити його із зсувом у лічильнику, проводячи операцію множення. Операція ділення виконувалася аналогічно, шляхом повторного віднімання з діленого, для чого рукоятку приладу слід крутити в інший бік. Арифметична машина Лейбніца удосконалювалася у деталях, але не в принципі, впродовж XVIII, а потім і XIX століть, вона отримала назву **арифмометра** і стала виготовлятися багатьма фірмами.

Приблизно у 1820 році Charles Xavier Thomas створив перший вдалий механічний, який серійно випускався, калькулятор – **Арифмометр Томаса**, який міг додавати, віднімати, множити та ділити. В основному, він був заснований на роботі Лейбніца. Протягом наступних двохсот років було знайдено та побудовано ще декілька подібних рахункових пристроїв, які через низку недоліків не набули широкого розповсюдження.



Арифмометр Фелікс

Лише у 1878 році російський учений П. Чебишев сконструював рахункову машину, яка виконувала додавання та віднімання багатозначних чисел. Найширшого розповсюдження у той час набув **арифмометр**, який був сконструйований петербурзьким інженером Однером у 1874 році. Головною деталлю пристрою було колесо Однера із змінною кількістю зубців. Конструкція приладу виявилася вельми вдалою, оскільки дозволяла досить швидко виконувати всі чотири арифметичні дії.

Після еміграції Однера до Швеції у 1917 р. його арифмометри досконалішої конструкції випускалися на заводі ім. Дзержінського під маркою «Фелікс». У 1969 р. їх було вироблено 300 000 шт.

Ці рахункові пристрої застосовувалися декілька десятиліть і були основним технічним засобом, що полегшував працю людей, які були пов’язані з обробкою великих об’ємів числової інформації. На них виконували навіть дуже складні розрахунки, наприклад, розрахунки балістичних таблиць для артилерійських стрільб. Існувала і спеціальна професія – лічильник – людина,

що працює з арифмометром, швидко і точно виконуючи певну послідовність інструкцій (згодом їх стали називати програмою). Але багато розрахунків виконувалися дуже повільно – навіть десятки лічильників повинні були працювати по декілька тижнів і місяців.

Так, одному із авторів цього посібника (Авраменко В.С.) «пощастило» виконувати лабораторні та курсові роботи саме на арифмометрах «Фелікс».

З усіх винахідників минулих століть, що зробили внесок у розвиток обчислювальної техніки, найближче до створення комп'ютера у сучасному уявленні підійшов англійський математик **Чарльз Беббідж** (1791-1871). Чарльз Беббідж увійшов до історії як винахідник першої обчислювальної машини – прообразу сучасних комп'ютерів. У 1812 р. він почав працювати над так званою "**різницевою"** машиною. Робота моделі ґрунтувалася на принципі, відомому у математиці як "метод кінцевих різниць". Даний метод дозволяє обчислювати значення многочленів, використовуючи лише операцію додавання і не виконувати множення та ділення,.

Попередні обчислювальні машини Паскаля та Лейбніца виконували тільки арифметичні дії. Беббідж же прагнув сконструювати машину, яка б виконувала певну програму, проводила розрахунок числового значення заданої функції. Як основний елемент різницевої машини Беббідж використовував зубчасте колесо для запам'ятовування одного розряду десяткового числа. В результаті він зміг оперувати 18-розрядними числами. До 1822 року він побудував невелику модель, яка діяла, і розрахував на ній таблицю квадратів.

Але все таки репутація Беббіджа як першовідкривача в області автоматичних обчислень завойована в основному завдяки іншому, досконалішому пристрою. Удосконалюючи різницеву машину та використовуючи ідею Жаккара, Беббідж приступив у 1832 році до розробки чудового пристрою – перший у світі комп'ютер загального призначення, який він назвав **Аналітичною машиною**. Вона повинна була відрізнятися від різницевої машини більшою швидкістю і простішою конструкцією, та не просто вирішувати математичні завдання одного певного типу, а виконувати різноманітні обчислювальні операції відповідно до інструкцій, що задаються оператором.

Августа Ада Байрон

Аналітична машина була задумана як суто механічний апарат з трьома основними блоками. У конструкцію аналітичної машини входив також пристрій для введення початкових даних та друку отриманих результатів. Передбачалося, що машина діятиме за програмою, яка задавала б послідовність виконання операцій і передачі чисел з пам'яті в арифметичний блок і назад. Програми, у свою чергу, повинні були кодуватися і переноситься на перфокарти.

Багато з того, що відоме про цю машину, дійшло до нас завдяки науковим працям обдарованого математика-любителя **Августи Ади Байрон (**графиня Лавлейс,

Lovelace; уроджена Байрон). Дочка знаменитого поета лорда Байрона, який після її народження назавжди покинув Британію, Ада ніколи не знала свого батька. Освіту вона здобула, займаючись з приватними викладачами, а потім самостійно, у чому їй допомагав Август Де Морган, професор математики Лондонського університету. У 1835 р. Ада вийшла заміж за барона Уїльяма Кінга, а після отримання ним титулу графа Лавлейса у 1838 р., **Августа Ада Байрон** стала графинею Лавлейс.

У 1840 році Беббідж їздив на запрошення італійських математиків в Турин, де читав лекції про свою машину. Луїджі Менабреа, викладач туринської артилерійської академії, створив і опублікував конспект лекцій французькою мовою.

У 1843 р. Ада Лавлейс переклала на англійську мову та анотувала статтю про Аналітичну машину, яка була написана по лекціях Беббіджа Федеріко Менабреа «Елементи аналітичної машини Чарльза Беббіджа», забезпечивши її власними докладними коментарями про потенційні можливості та інструкціями з програмування до неї.

Аду Лавлейс, одного з небагатьох сучасників Беббіджа, хто зумів оцінити Аналітичну машину, по праву вважають першим у світі програмістом. Вона теоретично розробила деякі прийоми управління послідовністю обчислень, які використовуються у програмуванні і до цього дня. Наприклад, вона описала команди, що забезпечують умовну передачу управління та повтору певної послідовності кроків доти, доки не виконувалась задана умова. Тепер така конструкція називається **циклом**. Вона також написала для цієї машини програму обчислення **чисел Бернуллі**. Ця програма містила 25 кроків, була складена дуже витончено, мінімізуючи пам’ять та перфокарти. Для того щоб перевірити, наскільки правильно була написана ця програма (відлагодити її на машині автор не мала можливості), у 1978 році в інституті прикладної математики (Москва) був поставлений експеримент. Програму переклали мовою Фортран, для чого було потрібно 85 операторів, та протестували на комп’ютері. Виявилось, що у програмі Ади Лавлейс міститься лише одна алгоритмічна помилка. Після її виправлення програма запрацювала правильно.

Уява Беббіджа малювала Аналітичну машину не як пристрій, який призначений для вирішення якогось одного конкретного обчислювального завдання, а як універсальний засіб розв’язку широкого класу подібних завдань. По суті він задумав та спроектував універсальний програмований комп’ютер, втіливши у ньому багато рис, які притаманні сучасним машинам. Хоча Беббіджу так і не вдалося здійснити свої задуми, за його ретельно розробленим проектом можна бачити, що Аналітична машина була задумана як чисто механічний апарат з трьома основними блоками.

Перший блок – пристрій для зберігання чисел на регістрах із зубчастих коліс і система, яка передає ці числа від одного вузла до іншого. У сучасній термінології – це пам’ять, тобто свого роду аналог Оперативного Запам’ятовуючого Пристрою (ОЗП) сучасного комп’ютера. Другий блок – пристрій, який дозволяв виконувати арифметичні операції. Беббідж назвав його

"млином". Третій блок призначався для управління послідовністю дій машини. У конструкцію аналітичної машини входив також пристрій для введення початкових даних та друку отриманих результатів.

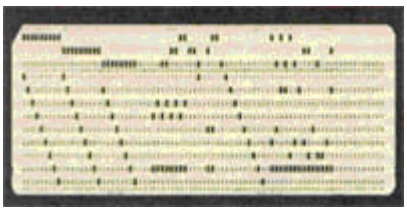
Передбачалося, що машина діятиме за програмою, яка задавала б послідовність виконання операцій та передачі чисел з пам’яті у млин та навпаки. Програми, у свою чергу, повинні були кодуватися та переноситися на перфокарти, які оператор міг би міняти на свій розсуд. Для створення комп'ютера в сучасному розумінні залишалося лише розробити схему з програмою, яка б зберігалася в ОЗП, що було зроблено 100 років потому Цузе (Німеччина), а потім Айкеном, Еккертом, Мочлі і фон Нейманом (США).

Сам Чарльз Беббідж у 1864 році написав: «Пройде, ймовірно, півстоліття, перш ніж люди переконаються, що без тих цінностей, які я залишаю після себе, не можна буде обійтися». У своєму припущенні він помилився на 30 років. Тільки через 80 років після цього висловлювання була побудована машина МАРК-1 (1944 р.), яку назвали «здійсненою мрією Беббіджа». Архітектура МАРК-1 була дуже схожа з архітектурою Аналітичної машини. Говард Айкен насправді серйозно вивчав публікації Беббіджа і Ади Лавлейс перед створенням своєї машини, причому його машина ідеологічно незначно пішла вперед порівняно з недобудованою Аналітичною машиною. Продуктивність МАРК-1 виявилася всього в десять разів вище, ніж розрахункова швидкість роботи Аналітичної машини.

Беббіджу так і не вдалося повністю втілити свій проект у життя через недостатній розвиток технології, а також тому, що в 1837 році математик і астроном Джордж Байдел Ейрі заявив Британському Міністерству Фінансів, що Аналітична машина «марна», і проект Беббіджа повинен бути зупинений. Незабаром після цього уряд заморозив проект. Світ забув про цю ідею до 1938 року, вона була вбита скептицизмом і забута. Ось, що писав Беббідж в 1851 році: «Всі розробки, які пов'язані з Аналітичної машиною, виконані за мій рахунок. Я провів цілий ряд експериментів і дійшов до межі, за якою моїх можливостей не вистачає. У зв'язку з цим я змушений відмовитися від подальшої роботи».

Хоча проект Беббіджа не був повністю реалізований, проте, його роботи мали важливе значення. Багато подальших винахідників скористалися ідеями Беббіджа, що були закладені в основу сконструйованих ним пристроїв, тоді як Джордж Ейрі запам'ятається, як короткозорий вчений, що не бачить далі свого носа.

Лише через 19 років після смерті Беббіджа один з принципів, що був покладений в основі ідеї Аналітичної машини, – використання перфокарт – знайшов втілення у діючому пристрої. Необхідність автоматизувати обчислення для перепису населення в США підштовхнула американського винахідника Генріха (Германа) Холлеріта (1860-1929) до створення у 1890 році пристрою, названого **електромеханічною рахунковою машиною – табулятором**, в якому інформація, що наносилась на перфокарти, розшифровувалася електромеханічним способом.



Перфокарта

Цей пристрій дозволив обробити дані переписи населення США у 1890 р. всього за 3 роки, замість восьми років, що витрачалися раніше. На карті табулятора Холлеріта було 12 рядків, в кожному з яких можна було пробити по 20 отворів (пізніше – 80), які відповідали таким даним, як вік, стать, кількість дітей, місце народження, сімейний стан та інші відомості, що були включені в опитувальник перепису населення. Табулятор отримав таке широке визнання, що для задоволення замовлень Холлеріту, довелося заснувати у 1924 році власну фірму для серійного випуску табуляторів. Незабаром фірма перетворилася на знамениту корпорацію **ІБМ (IBM, International Business Machines)**, яка зробила перфокарти стандартним засобом для програмування комп'ютерів. Отже, саме Холлерита можна вважати в деякому роді «хрещеним батьком» наших сучасних ЕОМ і персоналок, що з'явилися на світ саме завдяки цій компанії.

Табулятори здійснили дуже великий вплив на подальший розвиток обчислювальної техніки. Перші покоління електронних комп'ютерів успадкували конструкцію їх пристроїв введення-виведення, перфокарта Холлеріта довгий час, до появи дисплеїв та пізніше, залишалася основним носієм інформації і символом інформатики у цілому. На даний час ширина рядка на алфавітно-цифровому дисплеї все ще дорівнює 80 символам, якраз за розміром перфокарти. Формат багатьох мов програмування, наприклад, Фортрану або Коболу, так і залишився орієнтованим на перфокарти.

Величезний вплив на розвиток обчислювальної техніки здійснили теоретичні розробки таких математиків як англієць А. Тьюринг та американець Е. Пост, що працював незалежно від нього. "Машина Тьюрінга (Поста)" – прообраз програмованого комп'ютера. Ці вчені показали принципову можливість розв'язання за допомогою автоматів будь-якої проблеми за умови, якщо її можна представити у вигляді алгоритму, який орієнтований на виконанні машиною низки операцій.

З моменту виникнення ідеї Беббіджа про створення аналітичної машини до її реального втілення в життя пройшло більше як півтора сторіччя. Чому ж таким великим виявився розрив у часі між народженням ідеї та її технічним втіленням? Це обумовлено тим, що при створенні будь-якого пристрою, у тому числі і комп'ютера, дуже важливим чинником є вибір **елементної бази**, тобто тих деталей, з яких збирається вся система.

З іншого боку, навряд чи хтось вважатиме можливим оскаржити той факт, що історія виникнення та вдосконалення комп'ютерних технологій – це історія розвитку цивілізації нашої планети. Найстрашніша ганьба людства – Друга світова війна підштовхнула творчу думку десятків та сотень інженерів та математиків, основною метою яких стало всіма силами добитися максимальної технологічної переваги над супротивником. Цілком природно і те, що вчені наполегливо вирішували поставлені політичною обстановкою того часу завдання по обидві сторони лінії фронту. Саме тому відсутність як власно

Internet, так і в бібліотечних архівах інформації на російській та англійській мовах про високі технології фашистської Німеччини є обставиною невеликої кількості відомостей, які були здобуті науковими комісіями, та оцінені союзниками тільки після закінчення війни.

1.3. Можливості двійкового коду

У липні 1969 р., коли американський астронавт Ніл Армстронг ступив на поверхню Місяця, звістка про цю подію, миттєво подолавши 400 тис. км. космічного простору, досягла міста Х'юстон у штаті Техас, а потім облетіла весь світ. Телебачення показувало цю подію у мільйонах квартир, а телетайпи передавали підробиці у тисячі редакцій газет та журналів по всьому світу. Значна частина цієї інформації подорожувала від машини до машини у вигляді спеціального коду, що складався з імпульсів – електронного еквівалента нулів та одиниць.

Те, що зв'язок між людиною на Місяці та Землею здійснювався за допомогою нулів та одиниць, глибоко символічно і закономірно. За їх допомогою було закодовано все – від команд, що надсилалися космічному кораблю при зльоті, до інструкцій, завдяки яким апарат експедиції Армстронга повернувся на Землю.

Ідея використання лише двох символів для кодування інформації стара, як світ. Барабани, якими користувалися деякі африканські племена, передавали повідомлення у вигляді комбінацій дзвінких та глухих ударів. Інший, сучасніший приклад двосимвольного кодування – азбука Морзе (поєднання крапок та тире).

Одним з перших зацікавився двійковою системою геніальний німецький вчений Готфрід Вільгельм Лейбніц. У 1666 р., закінчуючи університет, 20-річний Лейбніц накидав роботу "Мистецтво складання комбінацій", яку скромно охарактеризував як "твір школяра". У цій роботі були викладені основи загального методу, який дозволяє звести думку людини, – будь-якого вигляду і на будь-яку тему – до абсолютно точних формальних висловів. Таким чином, відкривалася можливість перевести логіку (або, як називав її Лейбніц, закони мислення) із царини слова у царину математики, де відносини між об'єктами або висловами визначаються абсолютно точно. На додаток до своєї пропозиції зробити все раціональне математично строгим, Лейбніц закликав до ухвалення загальної мови, яку винайти або побудувати дуже важко, проте буде легко зрозуміти.

1.4. Розвиток двійкової системи

Сучасники Лейбніца залишили роботу молодого вченого без уваги, та і сам Лейбніц не став розвивати ідею нової мови. Проте десятиліттям пізніше він зайнявся дослідженням строгих математичних законів стосовно нової області – двійкової системи числення. На це його надихнув старовинний рукопис –

коментар з приводу знаменитої китайської книги “Ай чинг” (“Книга змін”), в якій робилися спроби описати Всесвіт за допомогою філософських категорій протилежностей – наприклад, таких понять, як темнота і світло, чоловічий та жіночий початок. Підбадьорений цим співзвуччям зі своїми математичними концепціями Лейбніц терпляче досліджував нескінченні комбінації нулів та одиниць, формулюючи знайдені ним закономірності та закладаючи основи сучасної **двійкової системи.**

Проте при всій своїй геніальності Лейбніц так і не зміг знайти корисного застосування отриманим результатам. Винайдений ним механічний калькулятор призначався для роботи з десятковими числами. Надалі Лейбніцу приходила думка використовувати двійкові числа в обчислювальному пристрої, але він навіть не намагався побудувати таку машину.

Якщо у Лейбніца і виникала думка, що двійкова система може стати тією універсальною логічною мовою, про яку йшлося у його творі 1666 р., він не висловлював її вголос. Але пізніше більше ста років після смерті Лейбніца (1716) англійський математик-самоучка **Джордж Буль** енергійно взявся за пошуки такої універсальної мови.

Примітно, що для досягнення цієї мети зайнялася людина такого скромного походження, як Буль. Він був родом з бідної робочої сім’ї, що жила в місті Лінкольні (Англія). У школі він здобув початкову освіту, проте його батько, що самостійно опанував деякими пізнаннями у математиці, передав ці знання своєму здібному синові. Надалі Буль продовжував своє навчання самостійно, і вже у 14 років вільно володів латинською, грецькою, французькою, німецькою та італійською мовами.

У 1831 р. у віці 16 років Буль поступив на роботу на посаду помічника вчителя, але через чотири роки відкрив власну школу. Зрозумівши, що йому слід поглибити свої пізнання в математиці, щоб перевершити учнів, він приступив до вивчення математичних журналів. Вивчивши гори наукових публікацій, він опанував складними математичними теоріями свого часу. У нього виникли власні оригінальні ідеї. Впродовж наступного десятиліття його роботи регулярно друкувалися в журналах. Врешті-решт, діяльність Буля отримала таку високу оцінку, що він, не дивлячись на відсутність формальної освіти, був запрошений працювати на математичний факультет Королівського коледжу в Ірландії.

У 1847 р. Буль написав важливу статтю “Математичний аналіз логіки”, а у 1854 – розвинув свої ідеї в роботі “Дослідження законів мислення”. Ці основоположні праці внесли справді революційні зміни до логіки як науки. Буль винайшов своєрідну алгебру – систему позначок та правил, яка могла застосовуватись до всіляких об’єктів, від чисел та букв до речення.

Починаючи з 1867 р., американський логік **Чарльз Пірс** впродовж двох десятиліть модифікував та розширив бульову алгебру. Він усвідомив, що бінарна логіка Буля добре підходить для опису електричних схем перемикачів. І хоча пізніше він розробив просту електричну логічну схему, вона так і не була зібрана. Проте, Пірс посіяв насіння, яке дало багаті сходи півстоліття потому.

У 1936 р. 21-річний випускник американського університету **Клод Шеннон** зумів ліквідувати розрив між теорією алгебри логіки та її практичним застосуванням. Перейшовши одразу ж на роботу у Массачусетський технологічний інститут (MTI), Шеннон взявся за вивчення логічної організації машини ”Диференціальний аналізатор”, яка була побудована у 1930 р. його науковим керівником професором В. Бушем, як тема докторської дисертації. Пригадавши бульову алгебру, яку він вивчав ще студентом, Шеннон здивувався, як свого часу і Пірс, її схожістю з принципами електричних схем. Поступово у Шеннона стали вимальовуватися контури пристрою комп’ютера. Якщо побудувати електричні кола відповідно до принципів булевої алгебри, то вони могли б відображати логічні відносини, визначати істинність тверджень, а також виконувати складні обчислення.

Свої ідеї щодо зв’язку між двійковими числами, булевою алгеброю та електричними схемами Шеннон розвинув у докторській дисертації, яка було опублікована у 1938 р. Ця робота по праву вважається поворотним пунктом в історії розвитку сучасної інформатики та обчислювальної техніки.

Потреба у зручній машині, яка була б здатна вирішувати диференційні рівняння, була настільки великою, що ще троє дослідників – двоє у США та один у Німеччинії – розвивали одні і ті самі ідеї практично одночасно.

Поки Шеннон працював над дисертацією, професор фізики **Джон Атанасов (**1903-1995**)**, американець болгарського походження, міркував про ті самі проблеми у коледжі шт. Айова. У 1938 р., після двох років роздумів про оптимальну конструкцію комп’ютера, Атанасов вирішив створити машину на основі двійкової системи числення. До осені 1939 р. він побудував грубу модель і став шукати фінансової допомоги для продовження роботи.

Багатьом відомо, що першу електронну обчислювальну машину ENIAC побудували у США Еккерт та Моучлі у 1945г., проте не всі знають, що з пріоритетом тут не все так просто. Суперечки з приводу авторства продовжувалися майже 30 років, поки після 7-річного розгляду федеральний окружний суд у місті Міннеаполісі у 1973 році не виніс наступної ухвали: “Еккерт і Моучлі не винайшли першими електронну цифрову обчислювальну машину, а витягли сутність концепції з винаходу доктора Джона Атанасова”.

Для прискорення обчислень Атанасов спочатку намагався використовувати Диференціальний аналізатор Буша, але точність при цьому виходила недостатньою. Тоді Атанасов вирішив створити принципово нову швидкісну машину на електронних лампах. Фінансувати роботу погодилася лише експериментальна агрохімічна станція штату Айова. У 1939 р. Атанасов разом з аспірантом Кліффордом Бері приступив до спорудження машини, яка повинна була розв’язувати системи алгебраїчних рівнянь з 30 невідомими (проект “ABC” – Atanasoff-Berry Calculator).

До весни 1942 р. проект був близький до завершення, проте обстановка військового часу не дала можливості його успішно закінчити, і через деякий час машина була розібрана. Вона так і не потрапила б в історію, коли б не випадкова зустріч Атанасова на конференції у Філадельфії у грудні 1940 року з

доктором Джоном Моучлі. Моучлі так зацікавився роботами Атанасова, що у червні 1941 р. спеціально приїхав у гості до Атанасова, щоб детально познайомитись з принципом роботи та конструкцією машини. Потім, у листі від 30 вересня 1941 року Моучлі питав Атанасова: "Чи не будете Ви заперечувати, якщо я займуся розробкою обчислювального пристрою, що містить деякі особливості Вашої машини?".

У цей же час на іншому кінці країни **Джордж Стібіц**, математик з фірми "Белл телефон лабораторіз" створив пристрій, в якому використовувалися логічні вентилі, які керувались електричним струмом і виконували операції двійкового додавання. Це був перший пристрій подібного типу у США. У наші дні двійковий суматор як і раніше залишається одним з основних компонентів будь-якого цифрового комп'ютера.

Ще через пару років Стібіц разом з іншими співробітниками розробив пристрій, який був здатний проводити операції віднімання, множення та ділення, а також додавання комплексних чисел, який назвали **калькулятором комплексних чисел**.

Проте ще до того, як Шеннон закінчив дисертацію, а Стібіц почав збирати модель калькулятора, подібною роботою зайнявся їх побратим по духу **Конрад Цузе**, який жив у Берліні і працював практично у повній ізоляції на квартирі своїх батьків.

1.5. Винахід перших комп'ютерів

Програмовані комп'ютери вперше почали використовувати для вирішення військових завдань, що стояли перед США, Німеччиною, Англією у Другій світовій війні. Такі пристрої були потрібні для найрізноманітніших цілей, від обчислення траєкторій бомб до дешифровки ворожих радіопередач. Саме війна стимулювала створення якісніших і швидших засобів обчислень. До вирішення цієї проблеми були залучені найблискучіші фахівці. Першими «комп'ютерами» були люди, головним чином жінки. Вони складали довгі «конвеєри» з механічних рахункових машин. Уявіть собі «роботу» програми таким чином: кожна з жінок, що сидять у ряд, працює на своїй станції, виконуючи певну частину обчислень траєкторії бомби, і передає відповідь своїй сусідці для виконання наступного кроку.

Але у військових умовах швидкість та точність однаково важливі, а комп'ютер гарантував і те, і інше. Крім того, війна відкривала фінансові можливості, які в інших умовах могли і не виникнути. Комп'ютер військових років вже був більшою мірою електричною, ніж механічною машиною, використовуючи електрику не тільки для управління механічними компонентами, але і для самих обчислень. Ця машина могла не тільки отримувати результати, але і автоматично використовувати ці результати для інших обчислень.

Після війни комп'ютери не змінили свого амплуа, тобто застосовувалися також, у першу чергу саме для оборонних завдань, особливо в СРСР. Вони

створювалися для розв’язку серйозних математичних проблем і першими програмістами у більшості своїй були люди, які складали та виводили рівняння. Фізики та математики ретельно розробляли алгоритми. Вони готували докладну і точну документацію, аналізували рішення своїх колег і шукали математичні докази. Ніколи більше в історії розробки програмного забезпечення для вирішення задачі програмування фахівці не підходили так методично.

Проте, за сучасними мірками проблеми, які вирішували ці талановиті першопроходці, були відносно простими. З іншого боку, не можна сказати, що алгоритми і математичні обчислення були тривіальними. Але на той час застосовувалися лише базові команди і операції. Сучасних операційних систем з тисячами вбудованих функцій та служб тоді ще не існувало.

Незважаючи на поєднання старих добрих зубчастих коліс з новомодними електромеханічними реле, «обчислювачі» залишалися всього лише обчислювачами. Потрібна була поява принципово нової схеми роботи, щоб вивести ці пристрої на якісно новий рівень. І ця схема була розроблена німецьким інженером Конрадом Цузе.

Одним з перших комп’ютерів, який по праву можна вважати **програмованим**, був створений німецьким інженером **Конрадом Цузе** (1910-1995). Ще у школі, він створив машину для розміну грошей. Починаючи з 1934, будучи студентом технічного ВНЗ, він приступив до створення універсальної обчислювальної машини, яка була б програмованою, і могла вирішувати задачі будь-якого рівня складності.

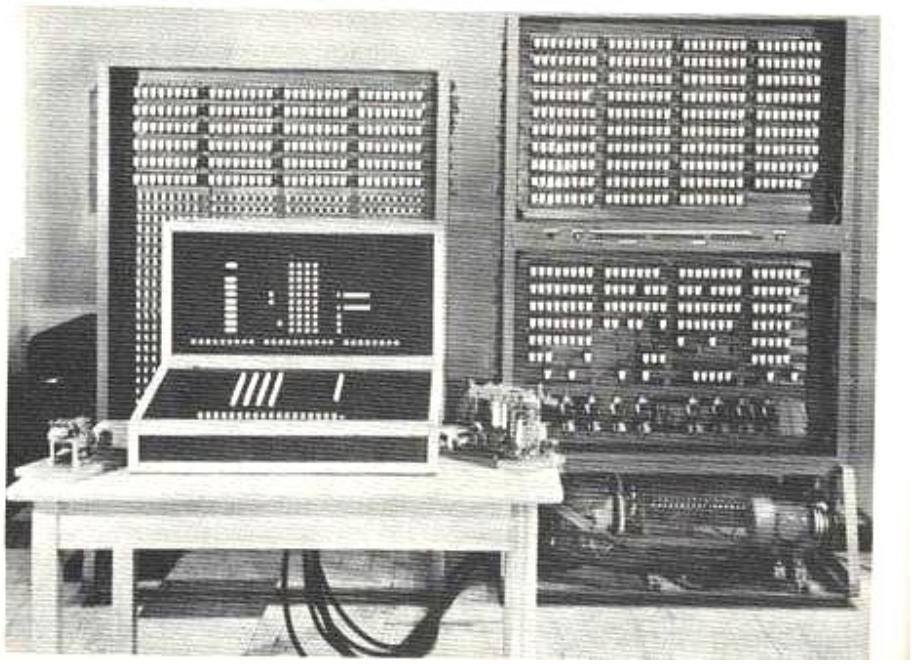
Конрад Цузе

Почавши спочатку з десяткової системи, Цузе віддав перевагу все ж таки двійковій системі. І тут він проявив неабиякі здібності. Знаючи про роботи Буля не більше, ніж про машину Беббіджа, він, проте, використовував у задуманому ним комп’ютері принципи булевої алгебри. В кінці 30-х – початку 40-х років ним було побудовано у домашніх умовах декілька рахункових машин і комп’ютерів для здійснення складних інженерних розрахунків (Z1, Z2 – перша буква від прізвища Zuse). У роботі йому допомагали лише кілька друзів, знайомі виділили Цузе невелику суму грошей на його дослідження. Слід зазначити, що тоді як за розробниками американських автоматичних обчислювальних систем стояли цілі університети, Міністерство оборони США і такі могутні компанії, як IBM, Цузе діяв самостійно, на свій страх і ризик.

Машина Цузе складалася з кількох блоків: керуючий пристрій, обчислювальний блок на основі електромеханічних реле, пристрій введення-виведення і, нарешті, пам'ять! Останнє варто відзначити особливо: саме Цузе створив перший зразок механічної «оперативної пам'яті» (на основі рухливих металевих стрижнів) і отримав на нього патент в 1936 р. Таким чином, його пристрій був здатний зберігати у своїй пам'яті проміжні результати

розрахунків, а значить, і виконувані ним операції могли бути набагато сложніші.

Машини Цузе управлялися за допомогою програми на перфорованій стрічці, яка виготовлялась з бракованої кіноплівки (через дефіцит паперу у роки війни). Як основні елементи машин використовувалися електромеханічні реле.



Перша в історії працююча програмнокерована універсальна обчислювальна машина Z-3 (1941 р.)

Конрад Цузе згадував, що машинні команди можуть зберігатися в тій самій пам'яті, що і дані – передбачивши тим самим те, що пізніше стало відомо як архітектура фон Неймана і було вперше реалізовано тільки в 1949 р.

Після таких успіхів Цузе отримав офіційну підтримку німецького уряду на розробку нового комп'ютера для військових цілей – конструювання літаків та ракет. У 1941 році Цузе організував власну фірму Zuse Ingenieurburo und Apparatebau – першу у світі комерційну компанію, яка ставила на меті тільки розвиток комп'ютерних технологій. У цьому ж році (1941), на два роки випередивши Ейкена, Цузе **перший серед винахідників комп'ютерів** зумів побудувати третю діючу машину – Z3, яка була заснована на електромеханічних реле, яка працювала у **двійковій системі числення**, замість звичної нам десяткової системи числення. Z3 складалася з 600 реле рахункового пристрою та 2000 реле пристрою пам'яті. Числа можна було «записати» у пам'ять (64 слова по 22 біта) та «прочитати» звідти за допомогою електричних сигналів, які проходили через реле. Реле або пропускали сигнал, або не пропускали. Машина читала програму механічно крок за кроком (лінійно) і проводила від 15 до 20 обчислювальних операцій за секунду.

Тим самим, Z3 стала першим працюючим комп'ютером, який керувався програмою. У багатьох відношеннях Z3 була подібна сучасним машинам, в ній вперше була представлена низка нововведень, таких як арифметика з

плаваючою крапкою. Заміна складної в реалізації десяткової системи на двійкову, зробила машини Цузе більш простими і, а значить, більш надійними.

Ще у 1937 році ***Конрадом Цузе*** для збільшення діапазону чисел, що подані в арифметиці двійкових чисел, а також для підвищення точності цього подання чисел було запропоновано подання чисел у плаваючій, нормалізованій формі – число x подається у вигляді: $x = m \times p^k$, де m – **мантиса числа**, k – цілий **порядок числа**, $p^{-1} \le |m| < 1$, p – основа системи числення**.**

К. Цузе також були запропоновані арифметичні операції над числами з плаваючою крапкою. Наприклад, нехай дано два числа: $x = m \times p^k$ і $y = n \times p^l$ і ($k > l$). Тоді можна перевірити, що результати виконання операцій будуть однакові:

$$x + y = (m + n \times p^{l-k}) \times p^k, \quad x - y = (m - n \times p^{l-k}) \times p^k$$

$$x \times y = (m \times n) \times p^{k+l}, \qquad x / y = (m / n) \times p^{k-l}.$$

Для завдання складних алгоритмів обчислень Z3 дозволяла використовувати розроблений Конрадом Цузе так званий «набір інструкцій», що включав близько десяти основних та кілька десятків додаткових команд, і був справжнісінькою мовою програмування. У грудні 1941 року Z3 був введений в експлуатацію і одразу же взятий на озброєння виробниками військових літаків. Саме за допомогою Z3 обраховувалися аеродинамічні та балістичні характеристики перших німецьких ракет.

Реставрована Z3 в музеї

У цей же час Цузе приступив до спорудження Z4, в якій всі механічні частини повинні бути замінені на електронні лампи. Під час бомбардувань Берліна всі машини Цузе, окрім Z4, були знищені. Роботи над Z4 продовжувалися три роки, і до грудня 1944 року цей проект був практично завершений. Проте до цього моменту хід війни змінився далеко не на користь Німеччини.

Під час одного з нальотів радянської авіації на Берлін перший екземпляр ЕОМ Z3 був повністю зруйнований (вчені відновили його тільки в середині 60-х років), а коли у травні 1945 року до столиці Німеччини увійшла радянська армія, Цузе був вимушений тікати разом з вже зібраним комп'ютером Z4 на південь країни. У післявоєнні роки Цузе, не маючи ні засобів, ні можливості продовжувати роботу над комп'ютером, винайшов ефективний спосіб програмування комп'ютерів – систему числових та символьних позначок, яка була побудована на основі логіки. Лише у 1949 Цузе приступив до комерційного виробництва комп'ютерів на основі Z4.

Уникнувши, таким чином, плачевної долі свого «молодшого» побратима, Z4, що розроблялася для військових потреб гітлерівської Німеччини, у 1950 році зайняла своє місце в офісі одного з цюріхських банків, де протягом п'яти років проводила цілком мирні фінансові обрахунки та обробляла дивіденди на рахунках вкладників.

Британські та американські військові експерти уважно ознайомилися з розробками Цузе, і незабаром він був запрошений до співпраці американськими

виробниками ЕОМ. Проте Цузе ввічливо відмовився від пропозиції працювати на своїх колишніх супротивників і відновив власну компанію Zuse Ingenieurburo und Apparatebau, яка під маркою Zuse KG налагодила випуск комерційних комп'ютерів. Цією фірмою були створені та широко використовувалися у 50-х роках в Німеччині ЕОМ Z5, Z11, Z22 і Z23. Але Zuse KG почала зазнавати фінансових труднощів і у 1962 році була продана компанії Brown Boveri and Co., а трохи пізніше увійшла до складу корпорації Siemens.

Конрад Цузе називав себе аполітичною людиною як під час війни, так і після неї. Справою всього свого життя він вважав розвиток комп'ютерних технологій у Німеччині і висловлювався з деякою гіркотою про те, що йому так і не вдалося здійснити свою мрію – створити переносний персональний комп'ютер для ділових людей, шкодуючи про те, що у цьому його зуміли випередити американські розробники. Конрад Цузе помер в Німеччині, в Хюхнфельде, у 1995 році, доживши до вісімдесяти п'яти років. З його смертю закінчилася ціла епоха у розвитку європейських інформаційних технологій.

Через війну роботи Цузе довгий час залишалися невідомими за межами Німеччини. Але по обидва боки Атлантики були і інші вчені, які, як і Цузе, зрозуміли ефективність двійкової системи числення та символьної логіки для проведення обчислень, проте перехід від десяткової системи до двійкової не одразу отримав визнання. Але очевидні переваги та перспективність автоматизованих обчислювальних систем, що «воювали» під час Другої світової на боці Німеччини, не могли залишитися непоміченими колишніми країнами-союзниками, що вже неабияк відстали від німців у цій сфері технологій. Проте в Радянському Союзі подальший розвиток ЕОМ був фактично зупинений з ідеологічних міркувань: уряд СРСР зарахував кібернетику у розряд псевдонаукової єресі, внаслідок чого еволюція комп'ютерів у даній окремо взятій країні на деякий час стала нездійсненною. І пальма першості в області високих технологій перейшла до американців.

По суті, ідеї Цузе, що були закладені у комп'ютерах Z1-Z4, є основними у комп'ютерних технологіях, які будуть пов'язані з ім'ям Джона фон Неймана.

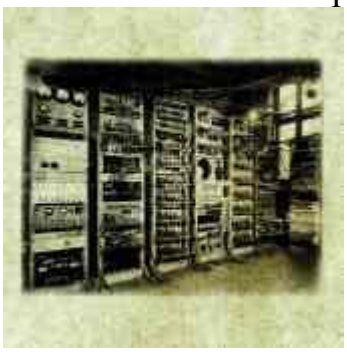
1.6. Гарвардська архітектура

Поки Цузе продовжував свої дослідження в Німеччині, американські вчені працювали над двома проектами, які залишили помітний слід в історії програмованих комп'ютерів. Ще у 30-х роках уряд США доручив Гарвардському та Прінстонському університетам розробити архітектуру комп'ютера для військово-морської артилерії. Перемогла розробка Прінстонського університету (відоміша тепер як архітектура фон Неймана, про неї мова буде далі), оскільки вона була простіша в реалізації. А так звана Гарвардська архітектура не використовувалася аж до кінця 70-х років (за винятком комп'ютера «Марк-1»), де вона відродилася у вбудованих процесорах, де швидкість обробки сигналів є найбільш критичною.

Типові операції (додавання та множення) вимагають від будь-якого обчислювального пристрою декількох дій: вибір інструкції, вибірку двох операндів, виконання інструкції і, нарешті, збереження результату. Ідея, яка була реалізована Айкеном, полягала у фізичному розділенні ліній передачі команд та даних, що було передбачено у Гарвардській архітектурі. У першому комп'ютері Айкена «Марк-1» для зберігання інструкцій

використовувалася перфорована стрічка (карта), а для роботи з даними – електромеханічні регістри. Це дозволяло одночасно пересилати та обробляти команди та дані, завдяки чому значно підвищувалася загальна швидкість роботи комп'ютера.

У Гарвардському університеті математик **Говард Айкен** (Ейкен, 1900-1973) з групою інженерів фірми IBM завершили роботу над машиною "MARK-1". Приводом для розробки цієї машини послужила робота над дисертацією, теоретична частина якої була присвячена методикам розв'язку нелінійних диференціальних рівнянь. Саме тоді у нього народилася думка про створення автоматичного обчислювального пристрою, який міг би значно полегшити проведення подібних математичних розрахунків.



***Марк*-1**

Побудована у 1944 році обчислювальна машина з автоматичним управлінням послідовністю операцій, MARK-1 за сучасними мірками навряд чи могла називатися повноцінним комп'ютером. Тоді як ентузіаст Конрад Цузе ввів в експлуатацію повністю електронну ЕОМ «Z2» ще у 1940 році, створений Айкеном чотирма роками пізніше MARK-1 був наполовину механічним, реле застосовувалися лише для автоматизації управління елементами його конструкції.

Довжина цього електронно-механічного чудовиська складала 17 метрів, висота – 2,5 метра, вага 5 т, споживало воно 160 кіловат електроенергії, і коштувало $500 тис. (Z2 Конрада Цузе була значно компактною), призначалася для балістичних розрахунків у ВМС США. Конструкція MARK-1 включала 750 000 різних елементів сполучених дротами загальною довжиною більше 800 метрів. Електронна начинка, як і машина Цузе, складалася з електромеханічних реле (3304 шт.), які у процесі своєї роботи видавали жахливий шум і часто-густо виходили з ладу, і управлялася за допомогою команд, що були закодовані на паперовій перфострічці.

Числа, які оброблялись MARK-1, зберігалися у спеціальних регістрах, які були реалізовані у вигляді металевих зубчастих коліщаток, які приводилися у рух особливим механізмом. Кожен регістр включав 24 колеса, 23 з яких використовувалися для представлення самих розрядів числа, а 24-е – його знаку. Крім того, кожен регістр мав пристрій, який дозволяв зберігати значення десяткових долей та передавати результат обчислень в інший регістр. Всього архітектура MARK-I налічувала 72 регістри для обробки цифрових значень та 60 додаткових регістрів для зберігання математичних постійних – в них за допомогою системи перемикачів вручну записувалися незмінні у процесі обчислень константи. Множення двох 23-розрядних чисел машина виконувала за 3 сек.

Крім основного математичного блоку, у складі MARK-I було декілька окремих модулів, які призначались для виконання операцій множення, ділення, підрахунку ступеня числа, значення синусу та обчислення логарифму. Операторський пульт був панеллю, що містить 420 механічних перемикачів, не враховуючи декількох контрольних панелей, що дозволяли операторові управляти режимами роботи машини. Хоча незабаром інші машини перевершили "MARK-1", він, тим не менше, продовжував використовуватися аж до 1959 р.

1.7. Архітектура фон Неймана

Доля розпорядилася так, що в широкому масштабі була однозначно і беззаперечно прийнята архітектура фон Неймана. У ній постулювались три основні принципи.

1. Програмне управління. Програма складається з послідовності кодів машинних команд, які вибирались з пам'яті за допомогою лічильника команд. Лічильник – це звичайний регістр, він або автоматично збільшується на довжину коду поточної команди після закінчення виконання цієї команди, або його стан міняється примусово при виконанні команд розгалуження.

2. Однорідність пам'яті. І програми, і дані зберігаються у спільній пам'яті; над кодами команд можна виконувати такі самі дії, що і над кодами даних. Отже, програму можна модифікувати у процесі виконання, наприклад можна управляти виконанням циклів та підпрограм; програма може бути результатом дії іншої програми, на цьому засновані методи компіляції (переведення символічних програм у машинний код).

3. Адресація. Пам'ять складається з перенумерованих комірок і процесору у будь-який момент часу доступна будь-яка комірка.

До появи цієї архітектури привів химерний ланцюжок, здавалося б, ніяк не пов'язаних між собою подій. Спробуємо прослідкувати за ними.

Тоді як напівмеханічні обчислювальні машини поволі доживали свого віку, технології не стояли на місці: поступовий перехід від релейних схем та зубчатих коліс до прогресивніших вакуумних ламп був неминучий. У 1943 році Головне управління артилерії США уклало з університетом Пенсільванії контракт на розробку та спорудження спеціальної електронно-обчислювальної машини, яка б призначалася для автоматичного розрахунку балістичних таблиць. Керівником проекту призначили знайомого нам професора факультету електротехніки Пенсільванського університету **Джона Моучлі** (Мочлі, 1907-1980), головним інженером – молодого вченого і любителя електроніки **Джона Преспера Еккерта** (1919-1995), а технічним куратором від міністерства оборони – капітана Гольдштейна (Голдстейна). Офіційне ж подання побудованого ними за допомогою 10 інженерів та 200 техніків «електронно-цифрового інтегратора і обчислювача» (Electronical Numerical Integrator and Calculator, скорочено – ENIAC) відбулося 15 лютого 1946 року.

Комп'ютер "Еніак", 1946 р.

Машина ENIAC, про яку згадувалося раніше, відрізнялася вельми незграбним способом завдання програми. Як і "MARK-1", вона оперувала десятковими, а не двійковими числами, проте замість електромеханічних реле в ній використовувалися електронні лампи, що дозволило у 1000 разів підвищити її швидкодію. Правда, за рік роботи

машини довелося замінити близько 19 тис. ламп, але, проте, машина продемонструвала, що майбутнє саме за електронним обчислювальним пристроєм.

У конструкції ENIAC використовувалося більше 18 тис. вакуумних ламп та близько 1500 реле, машина займала окреме приміщення площею 85 квадратних метрів, важила 30 тонн і споживала 150 кіловат електоенергії, якій, навіть не дивлячись на те, що це складало на 10 кіловат менше «потреб» MARK-I, вистачило б на задоволення потреб невеликого підприємства.

З появою електроніки стала практично реальною внутрішня пам'ять. Замість зубчастих коліс для зберігання числових значень в ENIAC були передбачені замкнені кола з 10 спеціальних електронних перемикачів. Але, розроблена в умовах військового поспіху, внутрішня пам'ять була вельми обмеженою. Її **суматор** (внутрішній пристрій на електронних лампах, призначений для зберігання оброблюваних даних) міг запам'ятовувати лише 20 10-значних десяткових чисел. З 12 таких кіл ланцюжків, 10 – для зберігання самого значення числа та 2 – для представлення його знаку, утворювали один регістр пам'яті машини. Всього розробники передбачили 20 таких регістрів, що використовувались для виконання операцій додавання та віднімання (комп'ютер виконував одну таку операцію за 0,2 мілісекунди), множення та ділення проводилося в окремих блоках машини.

Саме у конструкції ENIAC вперше використовувався принципово новий вид носія інформації, на який можна було записати послідовність виконуваних комп'ютером операцій, – перфокарти – набір спеціальних діелектричних пластин з пробитими в них отворами, кожний з яких замикав певне електричне коло при попаданні у нього контактної щітки зчитуючого пристрою. Перевага перфокарт у порівнянні з перфострічкою була очевидною: якщо стрічку, що несподівано порвалася, доводилося або ретельно та копітко склеювати, або викидати та готувати нову, то у даному випадку фахівці просто замінювали одну перфокарту, що прийшла у непридатність, з кількох сотень.

Проте, не дивлячись на всю прогресивність та революційність запропонованого Моучлі та Еккертом технічного рішення, ENIAC не був позбавлений серйозних недоліків. По-перше, введення інформації у машину здійснювалося за допомогою перемикання контактних комутаторів на 40 набірних дошках, кожна з яких мала декілька тисяч дротів, а загальна їх кількість складала 6000.

Для того щоб переналагодити комп'ютер з виконання однієї послідовності математичних операцій на іншу, доводилося перемикати всі ці дроти вручну, на що деколи витрачалось до кількох днів. До того ж, ENIAC відрізнявся вкрай низькою надійністю: варто було перегоріти одній з 18 тисяч вакуумних трубок, а перегорали вони постійно, як уся робота зупинялася, і техніки починали кропіткий пошук пошкодження, який міг затягнутися на добу і навіть більше.

ENIAC став до ладу у 1945 р., але за два роки до цього Моучлі та Еккерт вже почали обмірковувати створення більш досконалої машини, яка була б здатна зберігати не тільки велику кількість даних, **але і команди програми**, що

керують її роботою. Їх ідея створення машини з програмою, що зберігається у пам’яті, змінила принципи організації обчислень, підготувавши підґрунтя для появи сучасних мов програмування. Життя ENIAC було недовгим: він застарів вже до 1949 р., коли на світ з'явився його спадкоємець – комп'ютер EDVAC, перша машина, здатна зберігати програму у своїй пам'яті.

Свою нову машину Моучлі та Еккерт назвали EDVAC (EDVAC – Electronic Discrete Variable Automatic Computer – електронний автоматичний обчислювач з дискретними змінними). У якості внутрішньої пам’яті вони запропонували застосовувати ртутні лампи затримки, які використовувалися у роки війни в радіолокаторах. Ще одна особливість машини полягала в її орієнтації на роботу з **двійковими числами**, що дозволило спростити конструкцію арифметичного пристрою.

Поки Моучлі і Екерт трудилися над новим проектом, їх ідеї в узагальненому вигляді були викладені в епохальній роботі, що належить перу видатного математика угорського походження **Джона (Яноша) фон Неймана** (1903-1957).

В кінці 1944 р. на допомогу Моучлі та Еккерту Джон фон Нейман був направлений як консультант. Джону фон Нейману, який у 41 рік вже здобув популярність блискучого математика, призначено було зробити величезний вплив на розвиток обчислювальної техніки у післявоєнні роки. Фон Нейман розумів, що комп’ютер – це щось більше, ніж простий калькулятор – він повинен бути універсальним інструментом для наукових досліджень. Інтерес фон Неймана до комп’ютерів певною мірою був пов’язаний з його участю у надсекретному Манхетенському проекті по створенню атомної бомби, який розроблявся в Лос-Аламосе (штат Нью-Мексіко). Тепер він міркував про значно могутнішу зброю – водневу бомбу, створення якої вимагало дуже складних розрахунків.

Робота над бомбою вимагала виконувати величезну кількість розрахунків, тому у коло зору фон Неймана попали всі рахункові машини, що існували на той час. Він був знайомий з роботами і особисто з Ванневаром Бушем, Джорджем Стібліцем і Говардом Айкеном. Проте можливості створених ними механічних пристроїв, що являли собою кінець кінцем розвиток машини Беббіджа, його не задовольняли, але найголовніше полягало у тому, що він мав підстави передбачати існування інших, продуктивніших машин.

Джон (Янош) фон Нейман

Інформацію про теоретичну можливість альтернативного підходу до автоматизації розрахунків він отримав завдяки одній випадковості. Фон Нейману казково повезло, без всяких на те зусиль він отримав шанс познайомитися та подружитися з самим Аланом Тьюрінгом та його гіпотетичною машиною.

Втім, матеріалізація машини Тьюрінга могла б і не відбутися, а комп’ютерна історія могла піти абсолютно іншим шляхом, не будь ще одної випадковості, а саме –

зустрічі фон Неймана з лейтенантом Германом Гольдштейном.

Декілька слів про Гольдштейна. Цей зовсім не типовий армійський офіцер, професор математики з Мічіганського університету, після виклику на військову службу отримав чин лейтенанта і керував розрахунком артилерійських таблиць в балістичній лабораторії при полігоні. Роботу виконували сотні жінок, які використовували механічні арифмометри, у ті часи саме жінок-обчислювачів і називали комп'ютерами. Обсяг необхідних обчислень був жахливим: лише для розрахунку однієї траєкторії польоту снаряда потрібно було виконати 750 операцій, а для передачі військовим гармату необхідно було забезпечити як мінімум розрахунками 3 тис. траєкторій.

Гольдштейн, що відповідав за цю роботу, активно шукав шляхи вирішення проблеми прискорення розрахунку, і одного дня один з колег показав йому статтю «Використання високошвидкісних електронних ламп для обчислювальних пристроїв». Її написав у 1942 році співробітник Муровської школи Джон Мочлі (сьогодні Moore School of Electrical Engineering входить до складу університету штату Пенсільванія). Оцінивши прочитане, Гольдштейн негайно поїхав у Муровську школу і зробив все можливе, перш за все забезпечивши фінансування проекту, який спочатку називався Project PX, а потім став іменуватися ENIAC.

Фон Нейман усвідомив, що необхідно скористатися ідеєю машини Тьюрінга, для чого зрівняти у правах програми та дані і перенести гіпотетичну тьюринговську безкінечну стрічку із записаними на ній кодами команд в оперативну пам'ять комп'ютера. У такому разі програмування зводиться до запису послідовності команд у пам'ять. Ця процедура помітно простіша за ручну комутацію сотень або тисяч дротів. Крім того, у текст програм можна вставляти команди управління, а це означало – виконувати переходи та цикли. Створення машини з програмою, яка зберігається у пам'яті, поклало початок тому, що ми сьогодні називаємо програмуванням.

Менше ніж за рік після приєднання до групи фон Нейман своє відкриття виклав в одному з самих неоднозначних у всій історії комп'ютерної науки документі. Він підготував звіт на 101 сторінці машинописного тексту, в якому узагальнив плани роботи над машиною EDVAC. Цей звіт, що мав назву "Попередня доповідь, про машину EDVAC", був прекрасним описом не тільки самої машини, але й її логічних властивостей. Фон Нейман розглядав цю роботу лише як нарис, призначений для обговорення членами робочої групи EDVAC. Проте 1 червня 1945 р. один з членів групи, Герман Гольдштейн, військовий представник, який запропонував фон Нейману взяти участь у проекті, розіслав цей звіт більш ніж 30 фахівцям під тим самим заголовком – «Попередня доповідь про машину EDVAC», причому фон Нейман фігурував як єдиний автор звіту.

У своєму історичному звіті Джон фон Нейман виділив та детально описав **п'ять ключових компонентів** того, що нині називають "**архітектурою фон Неймана**". Щоб комп'ютер був ефективним та універсальним інструментом, він повинен включати такі структури: Центральний арифметико-логічний

пристрій, Центральний пристрій управління, що “диригує” операціями, Запам’ятовуючий пристрій, або Пам’ять, а також Пристрій введення-виведення інформації. Фон Нейман також відзначав, що ця система повинна працювати з двійковими числами, бути електронним, а не механічним пристроєм, і виконувати операції послідовно, одну за одною.

Побачивши, скільки галасу наробив звіт фон Неймана, Моучлі та Еккерт були глибоко обурені. Свого часу з міркувань секретності вони не змогли опублікувати ніяких повідомлень про свій винахід. І раптом Гольдштейн, порушивши секретність, надав трибуну людині, яка тільки-тільки приєдналася до проекту.

Передчасне розповсюдження звіту стало причиною багаторічного конфлікту і кінець кінцем призвело до того, що Моучлі та Еккерту так і не вдалося відстояти своє право на патент. З низки причин (у тому числі і історія із звітом) Моучлі та Екерт у березні 1946 р. покинули Пенсільванський університет, внаслідок чого довелося припинити роботи над машиною EDVAC.

Поширення звіту відкрило можливість для розробки комп’ютерів в інших країнах, перш за все в цьому досягла успіху Британія. Безумовно, документ став відомий в СРСР і сприяв початку вітчизняних робіт зі створення ЕОМ.

Серед тих, хто отримав екземпляр звіту про проект EDVAC, був співробітник британської Національної фізичної лабораторії **Дж. Уомерзлі**. Його запросили до США для знайомства з роботами над машинами ENIAC і “Марк-1”. Повернувшись до Англії, він привіз з собою примірник звіту та пристрастне бажання почати аналогічні розробки у своїй країні.

Для роботи над проектом він запросив математика **Алана Тьюрінга** (1912-1954), який під час війни брав участь в створенні надсекретної англійської декодуючої машини “Колос” (Colossus, 1943 р.), в якій замість електромеханічних реле містилося близько 2000 електронних вакуумних ламп.

Свій внесок в теорію обчислень Тьюрінг зробив десятиліттям раніше, коли у віці 25 років (у 1936 р.), будучи студентом Кембріджського університету, опублікував статтю “Про обчислювальні числа”, якій призначено було зіграти важливу роль у розвитку обчислювальної математики та інформатики. У ній описувався **гіпотетичний (абстрактний) пристрій**, придатний для вирішення будь-якого вирішуваного математичного або логічного завдання. Мета Тьюрінга полягала не у винаході комп’ютера, а в описі завдань, що не мають рішення.

Алан Тьюрінг

Разом з тим в гіпотетичній машині Тюрінга були передбачені деякі риси сучасних комп’ютерів. Так, наприклад, **Універсальна Машина Тьюрінга є прообразом універсальної ЕОМ**, в якій алгоритм (машина *T*) у вигляді програми та початкових даних поміщаються в оперативному пристрої ЕОМ, яка відіграє роль нескінченної вхідної стрічки Машини Тьюрінга.

«Колос» став першим повністю електронним обчислювальним пристроєм. У ньому використовувалася

велика кількість електровакуумних ламп, введення інформації виконувалось з перфострічки. «Колос» можна було налаштувати на виконання різних операцій булевої логіки, але він не був Тьюрінг-повною машиною. Крім Colossus Mk I, було зібрано ще дев'ять моделей Mk II. Інформація про існування цієї машини трималася в секреті до 1970-х рр. Через свою секретність, «Колос» не згадано в багатьох працях з історії комп'ютерів.

Завдяки Тьюрінгу машина "Колос" прекрасно впоралася з розшифровкою донесень німецької машини "Загадка" ("Енігма"). В «Енігме» була алфавітна клавіатура та набір з 26 лампочок. Натиснення, кожної літери викликало загоряння певної лампочки відповідної іншій літері. Якщо ви набирали літерами слово «dog», запалювалися лампочки, що відповідали буквам «rlu». Відправник посилав ці букви азбукою Морзе, одержувач друкував їх на алфавітній клавіатурі своєї «Енігми», і там спалахували лампочки, що означають слово «dog».

Всередині «Енігми» був набір шестерінок, які забезпечували зв'язок між вхідними та закодованими літерами. Плюс – пристрій, що дозволяв міняти код, тобто, міняти співвідношення літер та лампочок. Його міняли кожну добу, опівночі. Щоб розшифрувати такий код перерахунком, потрібно було перевірити логічні ланцюжки у кількості 15-ти квінтилліонів (одиниця з вісімнадцятьма нулями). Особливість німецької системи шифрування за допомогою «Енігми» полягала у тому, що **навіть володіння зразком самої машини не забезпечувало розшифровки**.

Погубив «Енігму» німецький педантизм – Тьюрінг помітив, що в німецьких шифровках регулярно повторюються декілька коротких однакових фраз. На подібних повторах дешифровщики почали будувати і перевіряти припущення, що зводилися всього лише до мільйона задач, які могла швидко вирішити машина Т'юринга – "Колос".

Під час роботи машина "Колос" видавала характерний цокаючий звук – коли перемикалися реле, перевіряючі один логічний ланцюжок за іншим (відбувався звичайний перебір комбінацій). За цей звук дешифровщики прозвали машину Тьюрінга «Бомбою». Одна «Бомба» могла зламувати добовий код «Енігми» за одну годину.

Тисячі перехоплених за день ворожих повідомлень вводилися у пам'ять "Колоса" у вигляді закодованих символів на перфострічці. Стрічку вводили у фотоелектричний зчитуючий пристрій, який сканував її з дивовижною швидкістю, – 5000 символів за секунду, після чого у пошуках відповідності машина співставляла інформаційне повідомлення з вже відомими кодами "Загадки". Кожна машина мала п'ять зчитуючих пристроїв, в результаті за секунду оброблялося близько 25000 символів.

Деякі автори статей вважають, що Тьюрінга часто помилково називають автором архітектури машини "Колос". У тому, що Тьюрінг мав до цієї машини лише непряме відношення, переконує факт, що вона була зроблена не за його схемою. Архітектура "Колос" ближче до Гарвардської архітектури – звідси її фантастична продуктивність.

Хоча використання вакуумних ламп зробило значний крок вперед у розвитку обчислювальної техніки, "Колос" все ж таки був спеціалізованою машиною, застосування якої обмежувалося розшифровкою секретних кодів.

Приблизно у той же час Конрад Цузе звернувся до уряду Німеччини з проханням про виділення коштів на створення аналогічного пристрою для дешифровки англійських секретних кодів, нічого не знаючи про машину “Колос”, але отримав відмову.

Отримавши завдання Уомерзлі (вивчити проект фон Неймана та побудувати комп’ютер, подібний “Едвак”), Тюрінг розробив план створення машини **ACE** (Automatic Computing Engine – автоматичний обчислювальний пристрій). Проте проект ACE натрапив на цілу низку бюрократичних перепон, що змусило Тюрінга відмовитися від роботи задовго до того, як у 1950 р. вийшов зменшений варіант машини.

Наступну “зупинку” Тьюрінг зробив у Манчестерському університеті, де під керівництвом англійського математика **Макса Ньюмена** велися роботи над комп’ютерним проектом “Марк-1” (“тезкою” американського “Марк-1”). Внутрішня пам’ять “Марк-1” була побудована на 6 електронно-променевих трубках. Офіційно англійська машина називалася EDSAC (Electronic Delay Storage Automatic Calculator – електронний автоматичний калькулятор з пам’яттю на лініях затримки). 21 червня 1948 р. манчестерський “Марк-1” реалізував **першу у світі програму, що зберігалася у пам’яті машини**, – пошук найбільшого співмножника заданого числа.

Поява комп’ютерів, які забезпечені внутрішньою пам’яттю та схемами обробки двійкових даних, впритул наблизила обчислювальну науку до усвідомлення величезних можливостей подібних універсальних машин. Проте Алан Тьюрінг та інші фахівці чітко розуміли, що для більшості програмістів спілкування з машиною на мові ланцюжків нулів та одиниць (тобто за допомогою так званого машинного коду) виявиться достатньо важкою справою. Першою спробою полегшити роботу програміста можна вважати розроблене Тьюрінгом кодування довгих рядків двійкових нулів та одиниць, де використовувалися комбінації телетайпних символів, які вводилися у комп’ютер з паперової перфострічки.

Ймовірно, Тьюрінг міг би ще багато чого досягти у цій області, але цьому заважала його ексцентричність. Тьюрінг захопився проблемами машинного інтелекту, почав працювати в області біології. У 1950 році Алан Тьюрінг запропонував свій оригінальний тест як спосіб визначити, чи може комп'ютерна програма думати як людина. У ході випробування кілька суддів повинні спілкуватися одночасно з комп'ютером і людиною, і за результатами спілкування визначити, хто із співрозмовників робот, а хто – людина. Якщо судді повірять комп'ютерній програмі, значить, вона дійсно володіє штучним інтелектом.

Перший тест Тьюринга був проведений 8 листопада 1991 в Музеї комп'ютерної техніки Бостона. Серед учасників було вісім комп'ютерних програм, які спілкувалися з суддями через модеми, і дві людини, також доступні в режимі віддаленого зв'язку. Судді бачили тільки номер терміналу (від 1 до 10) і назви тем для спілкування. Ось деякі з тем: «Сухий мартіні», «Шекспір», «Романтичні відносини». Теми для розмови були запропоновані доктором Робертом Епштейном, одним з організаторів заходу. П'ять з десяти суддів вирішили, що під ніком (*nick, nickname* – прізвисько, анонім) Whimsical Conversation (Ексцентрична бесіда) ховається людина.

Насправді це був комп'ютер під управлінням програми PC Therapist III. І це була перша часткова перемога, але поки що жодна машина, яка володіє штучним інтелектом, не пройшла повністю **тест Тьюрінга**.

Заслуги Алана Тьюрінга були гідно оцінені: після розгрому Німеччини він отримав орден, був включений в наукову групу, що займалася створенням британської електронно-обчислювальної машини. У 1951 р. у Манчестері почав працювати один з перших у світі комп'ютерів. Тьюрінг займався розробкою програмного забезпечення для нього. Тоді він написав і першу шахову програму для ЕОМ. Це був тільки алгоритм, тому що комп'ютера, здатного виконати цю програму, ще не існувало.

Життя Тьюрінга зламав випадок. Все звалилося буквально в один день. У 1952 р. квартиру Тьюрінга обікрали. У ході розслідування з'ясувалося, що це зробив один з друзів його сексуального партнера. Вчений ніколи не приховував своєї «нетрадиційної сексуальної орієнтації», але й зухвало себе не вів. Проте скандал з крадіжкою отримав широкий розголос, і в результаті звинувачення в «непристойній поведінці» було висунуто проти самого Тьюрінга. 31 березня 1953 відбувся суд. Вирок припускав вибір: або тюремне ув'язнення, або ін'єкції жіночого гормону естрогену (спосіб хімічної кастрації). Він вибрав останнє. З Департаменту кодів його звільнили. Позбавили допуску до секретних матеріалів. Правда, колектив викладачів Манчестерського університету взяв Тьюринга на поруки, але він і в університеті майже не з'являвся.

Складнощі виникли у нього і в особистому житті. У 1954 р., займаючись грою, яку він назвав "нежилий острів", сенс якої полягав у виготовленні хімічних речовин із звичайних побутових продуктів, Тьюрінг отримав ціаністий калій і 7 червня 1954 р. розчин ціаніду Тьюринг впорснув в яблуко. Надкусивши його, він помер. Кажуть, саме цей плід, знайдений потім на нічному столику Алана, і став емблемою знаменитої комп'ютерної фірми «Еппл». Експертиза кваліфікувала його смерть як самогубство. Він помер у віці 41 року.

На честь видатного англійського вченого Алана Тьюрінга, що отримав перші глибокі результати щодо обчислюємості задовго до появи перших електронних обчислювальних машин, Асоціацією обчислювальної техніки була заснована **Премія Тьюрінга**.

Премія щорічно вручається одному або кільком фахівцям у галузі інформатики та обчислювальної техніки, чий внесок у цій області зробив сильний і тривалий вплив на комп'ютерне співтовариство. Премія може бути присуджена одній людині не більше одного разу. У сфері інформаційних технологій премія Тьюрінга має статус, аналогічний Нобелівської премії в академічних науках. Вперше Премія Тьюрінга була присуджена в 1966 році Алану Перліс за розвиток технології створення компіляторів.

На даний час премія спонсорується корпораціями Intel і Google і становить 250 000 доларів США.

1.8. Створення зрозумілих людині кодів

У 1949 р. Джон Моучлі спільно з Еккертом, покинувши Пенсильванський університет, працювали над створенням двох нових комп'ютерів: "**Бінак"**

(BINAC) та “**Юніак-1” (**UNIAC, Universal Automatic Computer). Моучлі доручив своїм програмістам створити програму, яка б дозволяла машині сприймати рівняння алгебри в традиційній формі. Так, рівняння вигляду A = B + C вводилося у формі 00 S0 03 S1 07 S2, де змінні A, B, C позначалися S0, S1, S2, а операції присвоєння та додавання – 03 і 07 (комбінація 00 визначає початковий номер рядка програми).

Система кодування, що була запропонована Моучлі, з’явилася справжнім відкриттям для програміста з його компанії – енергійної персони на ім’я **Грейс Мюррей Хопер** (1906-1992).

Трапилося так, що через сто років після Ади Лавлейс другим в історії програмістом знов стала жінка. Здобувши математичну освіту та ступінь доктора математики, у роки другої світової війни Хопер була направлена на роботу у Гарвардський університет, до відділу, що займався обчисленнями для потреб артилерії на “Марк-1”. Слово “***програміст”*** ще не дійшло до них з Англії, тому у цьому відділі фахівців – математиків називали **кодувальниками** (така назва зберіглася у США і до цього дня, так називають програмістів низької кваліфікації, що займаються кодуванням (написанням програм) деяких завдань за готовими алгоритмами).

Хопер та її колеги Роберт Кемпбел та Річард Блок вперше у світі почали створювати **підпрограми**. Під цим розумілися послідовності команд, які багаторазово використовувались та побудовані таким чином, щоб з них можна було компонувати крупніші сегменти або блоки всередині програм. Хоча само слово “***підпрограма”*** було вигадане дещо пізніше, можна вважати, що один з головних прийомів сучасного програмування вперше застосовували ще у 1944 р.

Через програмну несумісність у кожній ЕОМ були свої унікальні бібліотеки. Наприклад, для М-20, що випускалась в СРСР, Михайло Романович Шура-Бура (ІПМ – Інститут прикладної математики) з колегами розробили чудову на ті часи (60-і роки) інтерпретуючу систему ІС-2 з багатющою колекцією стандартних програм, що охоплювала всі області обчислювальної математики. Створення бібліотек різко підвищило продуктивність праці програмістів, оскільки з’явилася можливість спертися на працю попередників і не програмувати кожне нове завдання з нуля.

Наступного року Хопер та її колеги ввели інше – таке ж фундаментальне поняття діяльності програміста. Одного разу спекотним літнім днем таємнича несправність змусила зупинитися “Марк-1”. Після ретельної перевірки програмісти встановили, що контакти одного з реле були заблоковані рештками метелика. Цю подію зафіксували у журналі. Супровідний напис свідчив: “Перший достовірний випадок виявлення комахи (**bug**)”. Пізніше Хопер згадувала: “Коли до нас зайшов офіцер, щоб дізнатися, чим ми займаємося, ми відповіли, що шукаємо несправність, очищаємо комп’ютер від комахи (**debugging**)”. Термін “**дебаггін” (відлагодження)** з тих пір прижився і почав використовуватися серед програмістів.

А ось для іншого комп’ютерного терміну “**глюк**” (glitch), який, як правило, використовується як синонім терміну “збій”, існує низка припущень. Деякі припускають, що цей термін походить від єврейського слова “glitshen”, яке означає “ковзати”. Існує й інша думка з цього приводу. В середині 60-х років програма освоєння космосу йшла повним ходом. Вранці в день запуску ракети з

астронавтами одна з компаній, яка займалася виробництвом бортових комп'ютерів та систем для управління польотом, розмістила на сторінках найпопулярніших газет рекламні матеріали, в яких йшлося про те, що комп'ютер благополучно керує польотом астронавтів по ходу їх місії. Того ж дня комп'ютер дав збій, що призвело, практично, до повної втрати контролю над апаратом. Наступного ранку одна з газет ознаменувала цю подію таким чином – "найгучніша невдача в історії компанії", що англійською мовою звучало як "**Greatest Lemon in the Company's History**"

У 1949 р. Хопер перейшла в комп'ютерну фірму Еккерта та Моучлі, де взяла участь у створенні машини "Юніак-1". За час своєї роботи вона невплинно відстоювала думку, що для програмування слід використовувати **мови високого рівня**, тобто мови, у певному значенні якомога ближчі до природної мови.

Запропонована Моучлі система кодування зовсім не була панацеєю. Програму, що була записана в системі кодування Моучлі, треба було переглядати та виконувати рядок за рядком за допомогою спеціальної **програми-перекладача**, яка звалась **інтерпретатором**, і кожна окрема операція перед її виконанням піддавалася повному посимвольному розбору, навіть якщо вона до цього вже не раз виконувалася.

У жовтні 1951 р. Грейс Хоппер отримала завдання створити набір стандартних математичних підпрограм. До цього при переписуванні фрагментів програм та підпрограм з однієї програми в іншу припускалися помилки, і в програмі доводилося міняти всі числа, що задавали адреси конкретного місцерозташування програми у пам'яті машини. Очевидно, для подібної роботи був потрібний якийсь пристрій, єдиним призначенням якого було б акуратне виконання операцій копіювання та об'єднання.

1.9. Крок на благо програмування

Людям властиво помилятися. Це стало початковою передумовою появи **програми-компоновника** або **редактора зв'язків**. Компоновник за заданим ідентифікатором, тобто умовним кодом, здійснював вибірку потрібної підпрограми (вона зберігалася на магнітній стрічці), прочитував та записував її у відведене місце оперативної пам'яті, перераховував відповідні адреси і формував з них єдину програму. Компоновник, що постачався разом з комп'ютером "Юніак", дозволяв створювати так звані ***автоматичні програми***. Це стало для програмістів справжньою знахідкою, оскільки купити програмне забезпечення на той час було ніде. Треба зазначити, що спочатку програми створювались як абсолютні, тобто були прив'язані до конкретних адрес у пам'яті. В подальшому програми стали такими, що переміщуються у пам'яті. Причому остаточне налаштування програм на конкретні адреси здійснювалось під час їхнього завантаження в оперативну пам'ять. Таким чином з'явилась можливість створювати бібліотеки стандартних підпрограм у переміщуваному форматі. Для компоновки окремих модулів в єдину програму (EXE-файл) компоновник виконує налаштування переміщуваних адрес з урахуванням розміру усіх модулів та їх уточнюючих параметрів.

Інші підприємці, відмітивши комерційний успіх “Юніак”, швидко включилися у комп’ютерний бізнес – так почалася технологічна революція. У неї були два рушійні чинника: прогрес в області розробки швидкодіючої електроніки та безперервне вдосконалення мови спілкування людини з машиною. Безмежні можливості обчислювальної техніки, які більш ніж століття тому передбачали Чарльз Беббідж та Ада Лавлейс, почали втілюватися у життя.

1.10. Можливості програмного управління

Сучасні досягнення в області мініатюризації електронних схем дозволили створити пристрої, які розміщуються на кременевій платівці розміром з кінчик пальця, але за обчислювальною потужністю вони багато у чому перевершують гігантські машини 50-х років. Це зробило комп’ютери доступними практично для будь-якої людині. Більш того, сучасні машини здатні вирішувати завдання, про які навряд чи можна було навіть мріяти за часів комп’ютерів на електронних лампах та перфокартах.

Секрет такої універсальності криється у програмному забезпеченні, яке дозволяє використовувати одну і ту саму машину для вирішення великої кількості найрізноманітніших завдань. На відміну від програм ранніх моделей, що вводилися у машину за допомогою рутинного перемикання безлічі тумблерів та цифро-набірних пристроїв, програмне забезпечення, що надихнуло життя у сучасний комп’ютер, записується на магнітні (жорсткі) диски та вступає у роботу одразу після включення машини.

Як відомо засновниками накопичувачів на жорстких магнітних дисках є компанія IBM, всім також відомо, що перший HDD був розроблений у 1973 році, і називався він IBM 3340, і саме тоді почали HDD називати вінчестерами, але не всі знають, що історія HDD почалася майже на 20 років раніше.

Отже, рік 1956-й, вересень місяць. Компанія IBM надає свою нову розробку – комп’ютер 305 RAMAC, який став революційним проривом в області комп’ютерів і дав величезний поштовх для подальшого розвитку та впровадженню комп’ютерної техніки.

У тому ж 1956 році американська фірма IBM вперше продає пристрій для зберігання інформації на магнітних дисках – RAMAC (Random Access Method of Accounting and Control – метод з довільною вибіркою розрахунків та контролю). Незабаром RAMAC став індустріальним стандартом, а сам IBM 305 – став першим у світі комп’ютером з використанням жорстких магнітних дисків.

З цього часу можна було вже забувати про магнітні стрічки – на зміну їм прийшов новий герой – **350 Disk File**. Саме так іменувався жорсткий диск у комп’ютері 305 (також можна зустріти такі назви, як IBM RAMAC, 350 RAMAC Disk File, RAMAC storage device або просто RAMAC – це все 350 Disk File).

Загальне уявлення про зовнішній вигляд накопичувача, можна отримати, якщо добре придивитися до фото. Сам 350 Disk File важив біля тони і займав місце двох великих холодильників. Конструктивно накопичувач складався з 50-ти алюмінієвих дискових поверхонь, кожна з яких мала діаметр – 24 дюйми. Поверхні з обох боків були покриті оксидом заліза. В цілому конструкція схожа з сучасними жорсткими дисками. Всі 50 поверхонь кріпилися на шпинделі і були герметизованими. За допомогою важеля вибірки дві голівки читання/запису могли переміщуватися на потрібну поверхню. Поверхні оберталися із швидкістю 1200 обертів за хвилину. Загальна ємкість накопичувача складала 5 000 000 символів, це приблизно 4,4 Мбайт. Щільність запису була – 2000 біт на квадратний дюйм. Вартість одного Мб на той час складала 10000 доларів.

Але все ж таки принципи сучасної технології виготовлення жорсткого диску були розроблені у 1973 тією ж фірмою IBM (модель 3340, з 14-ти дюймовими пластинами). Новий пристрій, який міг зберігати до 16 мегабайт інформації, мав 30 циліндрів (сукупність магнітних смужок з однаковими номерами на різних поверхнях) для запису, кожен з яких був розбитий на 30 секторів. Тому, за однією з версій, воно отримало назву 30/30. Відомі гвинтівки «вінчестер» мають калібр 30/30, тому жорсткі диски теж почали називатися «вінчестерами». В літературі зустрічається ще декілька суперечливих даних щодо характеристик диска (30/30), ось деякі з них:

- диск мав ємність 30 Мбайт, час доступу 30 мілісекунд;

***Пакети дисків ємністю 7,25 (29)* Мбайт**

- накопичувач складався з двох дисків по 30 Мбайт кожен (разом 60 Мб).

Сумнівна версія, оскільки накопичувач на жорстких магнітних дисках – основний пристрій зовнішньої пам’яті ЕОМ 3-го покоління (1964 р), в якому ємкість пакету дисків складала всього від 7,25 до 29 Мбайт.

За іншою, найбільш імовірною версією [Велика енциклопедія Кирила та Мефодія, 2005], він отримав таку назву через м. Вінчестер (Англія), де проходили перші розробки жорсткого диска філією фірми IBM. Пройшов час, але по суті команди комп’ютера – хоча вони і задаються тепер набагато зручнішим способом – помітних змін не зазнали.

Оскільки вже пішла розмова про диски, то треба сказати, що фізичний розмір накопичувачів на ранніх етапах розвитку комп’ютерної індустрії був не надто найважливішим чинником при їх проектуванні та виробництві, оскільки основний наголос робився на ємність та швидкість. Проте з часом досить гостро встало питання про зменшення розмірів жорстких дисків. І ось через 6

років після випуску "Вінчестера", у 1979 році, фірмою IBM була анонсована модель 3310 – перший жорсткий диск з 8-ми дюймовими платівками.

Цей форм-фактор прийшов на зміну 14-ти дюймовому, який був стандартом де-факто впродовж майже 10-ти років. Практично навздогін за цією подією (у 1980 році) Seagate представила жорсткий диск ST-506, що мав форм-фактор 5,25 дюйма (5 мегабайт, 4 поверхні). Досягнення настільки малих на ті часи розмірів накопичувачів дозволило використовувати їх у перших персональних комп'ютерах. Даний форм-фактор набув небувалої популярності і використовувався у ПК впродовж багатьох наступних років. Навіть поява у 1984 році першого жорсткого диску з прогресивнішим розміром 3,5", що став надалі одним з найважливіших стандартів усієї індустрії інформаційних технологій, практично не вплинуло у той момент на ситуацію на ринку, де продовжували домінувати пристрої 5,25". Компанією, яка представила перший 3,5" жорсткий диск була невелика шотландська фірма Rodime plc. Модель називалася RO352, мала ємність 10 мегабайт і дві 3,5" пластини.

За минулі роки в результаті розвитку програмного забезпечення виділено декілька основних типів програм. Центральне місце у них займає спеціальний комплекс програм, що має назву **операційна система**, яка координує роботу комп'ютера та інших програм, а також управляє розміщенням програм та даних в оперативній пам'яті. Вона дає вказівку комп'ютеру, як розподіляти апаратні ресурси для виконання завдання, як управляти периферійними пристроями. Вона також забезпечує можливість безпосередньої взаємодії людини та комп'ютера, виконуючи, наприклад, такі дії, як видача на дисплей панелей діалогу з елементами управління для продовження роботи комп'ютера за певним сценарієм, зберігання програм та файлів даних тощо.

Якщо розглядати операційну систему як "режисера" комп'ютерної дії, то **прикладні програми** відіграють роль "акторів". Саме завдяки таким програмам, як **текстовий процесор (word processor), електронні таблиці (spreadsheet) та ігри**, комп'ютер набуває дивовижної різноплановості.

Поняття "***Програмне забезпечення"*** включає також такі програми, як **транслятори та утиліти** (або **службові програми**). Транслятори допомагають програмістам створювати прикладне програмне забезпечення, спочатку створене на мовах програмування високого рівня. Утиліти виконують корисні (**utility – корисність**), часто вкрай необхідні рутинні одноманітні дії (функції).

2. Нові мови програмування

В середині 50-х років, коли обчислювальна техніка міцно вкорінялася в університетах та науково-дослідних центрах США та Європи, наступив час прогресу в області програмування, а в 60-і програмування стає загальнодоступним. Проте нові розробки зовсім не заперечували усьому тому, що було зроблене раніше. Навпаки, вони спиралися на вже побудований фундамент. Транслятори для так званих мов **Асемблера** – символічних мов машинних команд (такі мови вимагають від програміста глибокого знання апаратури) залишилися важливим засобом програмування для будь-якого комп'ютера і до цього часу. Проте їх роль поступово зменшувалась. Посередником між програмістами і машинами стали мови програмування – мови нового типу. Від мови асемблера вони відрізнялися більшою гнучкістю та можливістю використання конструкцій, схожих на речення. Але мова Асемблер ще й досі використовується як проміжна мова для компіляторів з мов високого рівня.

З появою мов високого рівня програмісти одержали можливість більше часу приділяти вирішенню конкретної проблеми, не відволікаючись на вельми тонкі питання організації самого процесу виконання завдання на комп'ютері. Крім того, поява цих мов ознаменувала перший крок на шляху створення програм, які вийшли за межі науково-дослідних центрів. Транслятори для цих мов програмування, які перетворювали символічні програми у машинний код, отримали назву – **компілятори**. При цьому кожен оператор мови високого рівня породжував низку машинних команд на мові Асемблер.

Університети почали пропонувати навчальні програми за новою технологією, і кількість виробників апаратного забезпечення швидко зростала. Раптово комп'ютери і присвячені ним навчальні курси сталі доступними всім, або, принаймні, тим, хто відвідував університет.

У той же час комп'ютери стали значно ефективнішими. Коло проблем, які вони могли вирішувати, серйозно змінилося як за рівнем, так і за складністю. Мови програмування ставали могутнішими та простішими у використанні. 60-і роки були періодом феноменального зростання комп'ютерних технологій і завдали тон тій частині сторіччя, що залишилася.

Тим часом, у ці роки розвиток галузі міг піти тупиковим шляхом. Менш талановиті люди бралися за вирішення складніших завдань. Дійсно, як можна було знайти ще талановитіших розробників, ніж програмісти 50-х років? Ситуація вела до катастрофи, але катастрофи не відбулося. Програмне забезпечення, яке було розроблене у 60-х роках, відрізняється такою ж високою якістю, як і програми, що були створені десятиліттям раніше.

Цей парадокс має просте, хоча і неочевидне, пояснення. Програмісти у 60-і роки були добросовісними, а якість програм високою, завдяки відсутності персональних засобів компіляції.

Компіляція у 60-і роки була справою нелегкою. Здебільшого компанія або університет мали тільки один величезний комп'ютер. Компілятор на цьому комп'ютері, по-перше, знаходився досить далеко від робочого місця програміста, по-друге, був настільки завантажений, що час роботи на ньому доводилося резервувати заздалегідь, і, по-третє, він був вкрай чутливий до помилок у синтаксисі та конструкціях мови програмування.

Іншими словами, компіляція недостатньо досконалої програми вимагала величезних непродуктивних витрат часу та сил, і, в результаті, могла вимагати значних доопрацювань. Уявіть, що вам потрібно пройти декілька десятків метрів, нести в руках купу перфокарт заввишки 20-30 см тільки для того, щоб машина видала назад цю пачку з повідомленням про «неузгодження типів» на 30-ій перфокарті. І могло статися так, що до компілятора наступного разу вас допустять тільки через декілька днів.

Процес компіляції примушував програмістів годинами просиджувати за робочими столами з олівцем у руках, ретельно перевіряючи свої програми, звертатися за допомогою до колег і знову читати і читати свої перфокарти доти, доки не будуть вичерпані всі можливі засоби їх перевірки. Ніякі заходи не здавалися зайвими, оскільки ціна недбалості була дуже висока.

2.1. Поневіряння пакетної обробки

Оскільки комп'ютери на початку 50-х років були нечисленні і страшенно дорогі, то й безпосередній доступ до них вкрай обмежувався. Програмістові треба було підготувати завдання на перфокартах та передати його в обчислювальний центр операторові. Цей посередник вводив карти у машину великим пакетом з іншими програмами.

Потім починалося очікування. Щонайменшої описки, наприклад, пропуск дужки, було достатньо, щоб машина припинила виконання програми. Виправивши відповідний рядок програми, програміст вимушений був знову чекати своєї черги. Навіть дуже хороші програмісти часто витрачали тижні на пошук та виправлення всіх помилок у програмі. Проте проблеми не обмежувалися лише незручністю **пакетної обробки**. Програмування на асемблері по суті мало у чому відрізнялося від програмування у двійковому коді. Якщо програма поводилася "дивно", то єдиним способом пошуку помилки було вивчення так званого **дампу пам'яті** – довгої паперової роздруківки, що містить таємничі числові коди.

На початку 60-х років молоді математики Дартмутського коледжу (США) **Кемені** (за походженням угорець) та **Курц** висунули ідею: навчати азам програмування всіх студентів незалежно від їх спеціалізації, будь то природничі або гуманітарні науки. На відміну від своїх колег з інших навчальних закладів, яких цілком влаштовувала методика навчання студентів шляхом читання лекцій, вони хотіли, щоб студенти опановували комп'ютер, створюючи для нього реальні програми. Але спочатку необхідно було подолати деякі перешкоди. Перш за все – машина в коледжі була всього одна.

Спочатку вони почали шукати, чим замінити пакетний режим, і вирішили використовувати **систему розділення часу CTSS** (Compatible Time Sharing System – Сумісна система розділення часу), яка була вперше запропонована в Массачусетському технологічному інституті (MTI) у 1959 р. чудовим та різностороннім математиком **Джоном Маккарті**.

Одному з авторів пощастило слухати доповідь Д. Маккарті на 2-ій Всесоюзній Конференції Програмістів, що проходила у 1970 р. у м. Новосибірськ. На все життя запам'ятався його вислів (постулат програміста), який був надрукований величезними літерами на довгій паперовій стрічці за допомогою АЦПУ (Алфавитно Цифровое Печатающее Устройство) на наймогутнішому на той час комп'ютері **БЭСМ-6** (Большая Электронная Счетная Машина – серія вітчизняних ЕОМ, що були створені під керівництвом радянського вченого – академіка Сергія Олексійовича Лебедева): **"*У кожній "готовій", ретельно відлагодженій програмі є хоча б одна помилка. Якщо Ви її виправили, то в цій програмі все одно є хоча б одна помилка…".*** Про радянську школу створення ЕОМ та програмування, оскільки торкнулися цієї теми, ми розповімо трохи пізніше.

Система з розділенням часу припускала підключення до одного головного **процесора (mainframe)** кількох терміналів з алфавітно-цифровою клавіатурою. Спеціальні програми дозволяли процесору по черзі підключатися до кожного терміналу, виконуючи при цьому невелику частину кожного завдання протягом певного періоду (або кванта) часу (зазвичай мілісекунди). Оскільки процесор обробляв інформацію з дуже великою швидкістю, у кожного програміста створювалося враження, що машина працює тільки з ним одним. Насправді програмісти розділяли між собою час роботи комп'ютера.

Наступна їхня ідея полягала у створенні мови програмування, що складається з простих слів природної мови. За словами Кемені, вони хотіли зробити мову настільки простою, щоб студенти могли використовувати її вже після трьох годин навчань.

2.2. Універсальна мова програмування

Оскільки передбачалося, що студенти будуть самі набирати програми на терміналі, то й мова була сконструйована з розрахунком на інтерактивну взаємодію. Така взаємодія дозволяє змінювати програму у процесі її роботи. Крім того, мова передбачала єдине представлення чисел, що відрізняло її від мови Фортран, де використовувалися два типи чисел – цілі і так звані числа з плаваючою крапкою або дійсні числа (ці числа складаються з цілої та дробової частини, і комп'ютер справді інтерпретує їх як числа, в яких десяткова крапка може переміщуватися). Тому студентові не треба було думати, якого типу вийде число в результаті ділення 2 на 3.

Для нової мови було обрано дуже вдалу назву – **універсальний символічний код для початківців (BASIC, Beginner's All-Purpose Symbolic**

Code). Мова дійсно була універсальною. На такій мові можна було складати прості програми для розв'язання завдань у будь-якій проблемній області.

Влітку 1963 р. почалася розробка першої версії компілятора для мови BASIC. Восени того ж року два студенти приступили до проектування та кодування операційної системи для машин, на яких Кемені та Курц хотіли реалізувати свої ідеї. Це були машини "Дженерал електрик-225" і 1 травня 1964 р. усі їхні зусилля злилися воєдино.

Незабаром робота в режимі розділення часу переважала майже в усіх крупних обчислювальних центрах. Сам же BASIC став справді масовою мовою програмування, а пізніше (у 70-і роки) став основною мовою мікрокомп'ютерної революції.

На початку 60-х років усі існуючі мови програмування високого рівня можна було перерахувати по пальцях. Проте незабаром їх число (з урахуванням різноманітних діалектів) досягло кількох сотень. Тому були зроблені різні спроби створити універсальну мову програмування. Але жодна з них не увінчалася повним успіхом. Серед десятка мов, які найширше використовувались, кожна орієнтувалася на розв'язання специфічних завдань. Наприклад, BASIC як і раніше широко вживався для написання простих програм для мікрокомп'ютерів.

Першою з мов високого рівня був **Фортран (FORTRAN,** від **FORmula TRANslator – перекладач формул)**. Цю мову розробила у кінці 50-х років група програмістів IBM. Фортран спеціально призначався для роботи з формулами, які використовувались у математиці, фізиці та інших науково-технічних дисциплінах. Фортран – з його чітко певними правилами виконання арифметичних дій – був класичною мовою програмування, найбільш відповідною для виконання природничо-наукових, математичних та інженерних розрахунків.

Мова програмування **Кобол (COBOL, Common Business Oriented Language – загальна мова, орієнтована на ділові завдання)**, створена у 1960 р. об'єднаним комітетом виробників та користувачів, була задумана як мова для масової обробки даних у сферах управління і бізнесу.

Інші мови були ще більш спеціалізовані. Мова програмування **Алгол (ALGOL, ALGOrithmic Language – алгоритмічна мова**), як і Кобол – плід комітету, але не національного, а міжнародного. Перша версія Алгол-58 призначалась для запису алгоритмів, які будуються у вигляді послідовності процедур, і була розроблена у ході напруженої восьмирічної наради у 1958 р. Основним ідеологом і творцем мови був **Джон Бекус**.

Програмісти далеко неоднозначно прийняли мову Алгол-58 і його наступника Алгол-60/68, широкого схвалення ця мова так і не отримала. Тим не менше, вплив Алголу на розвиток інших мов програмування виявився вельми значним.

Серед мов, при створенні яких ставилося завдання поліпшити Алгол, була мова **Паскаль**. Ця мова була розроблена в кінці 60-х років швейцарським вченим **Никлаусом Віртом**.

Джон Бекус

Оскільки у мові Паскаль визначення різних типів даних задавалися явним чином, то у Паскаль-програмах порівняно небагато помилок і їх простіше зрозуміти та виправити програмістові, що не є автором програми. Це зробило Паскаль відповідним інструментом для створення великих програм. У 1963 р. він був оголошений офіційною мовою програмування для американських спеціалізованих середніх шкіл та університетів.

Не дивлячись на стрімке зростання кількості мов програмування, програмне забезпечення істотно відставало у своєму розвитку від технічної бази комп'ютерів. Тоді як в області виробництва апаратного забезпечення спостерігалося неухильне зростання технологічності, продуктивності та зниження ціни, вартість програмного забезпечення продовжувала збільшуватися. Це, природно, викликало незадоволеність інженерів, які вважали, що програмне забезпечення фактично перетворилося на чинник стримування як для розвитку, так і для застосування комп'ютерів.

Никлаус Вірт

Головна причина, яка зробила програмне забезпечення свого роду каменем спотикання, криється у його зростаючій складності. Проте складність програмного забезпечення лише частково обумовлена заплутаністю вирішуваного завдання. Багато у чому вона пов'язана із самою сутністю процесу програмування, який вимагає величезної уваги та акуратності при перекладі конкретного завдання у терміни, які доступні комп'ютеру.

2.3. Усунення неоднозначності

У програмуванні неприпустимо покладатись на випадок. Користувач, який заворожений магією комп'ютера, не володіючи мистецтвом програмування, схильний не брати до уваги, що ті дії, які він вважає само собою зрозумілими, машині треба повідомляти в самих найдрібніших подробицях. Для комп'ютера будь-яку заздалегідь заплановану дію треба розбити на елементарні кроки так, щоб в результаті вийшов **алгоритм**. У простому випадку алгоритм нічим не відрізняється від рецепту приготування пирога. Але програміст ще повинен визначити і ті елементарні кроки, які інші просто не помічають. Наприклад, що у рецепті яйця обов'язково треба його розбити і використовувати тільки їх вміст, а не шкаралупу.

Оскільки, щоб передбачити будь-яку випадковість, треба написати сотні, а то і тисячі команд, природно, зростають витрати і вкрадаються помилки. Все це породжує, за словами голландського вченого **Едсгера Дейкстри**, "величезну безліч прикрих дрібниць". Дейкстра – науковий співробітник фірми "Бароуз" (BURROUGHS), один з найбільш авторитетних теоретиків програмування. Не один рік він стверджував, що більшості помилок можна уникнути, якщо вести

боротьбу з недбалим стилем у програмуванні, яке розвивалося в основному не як строга наука, а як мистецтво, засноване на інтуїції та особистому досвіді.

Свій плідний внесок в теорію програмування Дейкстра зробив у 1968 р. у віці 38 років. У невеликій роботі під назвою "Замітки по структурному програмуванню" він доводив, що більшість програм невиправдано складні через відсутність в них чіткої математичної структури. Стан справ в області проектування програмного забезпечення викликав неспокій багатьох фахівців з обчислювальної техніки. Все частіше роботи зі створення нових систем програмного забезпечення вартістю у мільйони доларів не вкладалися в заздалегідь встановлені строки, а потім у процесі використання в них виявлялися тисячі помилок. Якраз у той час, коли Дейкстра писав свою роботу, проходила міжнародна конференція, на якій виникла ситуація, яка отримала назву "Криза програмного забезпечення".

2.4. Заклик до дотримання математичної строгості

Головним об'єктом критики Дейкстра обрав команду безумовного переходу, або оператор GOTO. Даний оператор призначений для безумовної передачі управління з однієї частини програми в іншу. В деяких випадках він вельми корисний, але у той же час цей оператор ускладнює сприйняття програми людиною. Відстеження цього оператора нагадує ситуацію, в якій ви опинилися б, якби при читанні роману вам доводилося весь час перегортати книгу на декілька сторінок то вперед, то назад. Перериваючи логічну послідовність алгоритму, оператор GOTO практично виключає можливість добре розібратися в структурі програми для всіх, окрім самого її автора.

Замість оператора GOTO Дейкстра запропонував використовувати три типи управляючих структур: **просту послідовність** (тобто групу операторів, що виконуються один за одним), **альтернативу** (конструкцію, що дозволяє вибрати один з двох або більш можливих операторів) та **повторення** (конструкцію, що дозволяє виконувати задану групу операторів доти, доки виконується деяка умова). На думку Дейкстри, використовуючи ці три структури, програміст може взагалі обходитися без оператора GOTO.

Спочатку ідеї Дейкстри викликали лише усмішку серед програмістів, проте на початку 70-х років група співробітників фірми IBM під керівництвом фахівця із системного програмування **Харлана Міллса** застосувала структурне програмування для створення інформаційного банку даних газети "Нью-Йорк Таймс". Робота над проектом йшла на подив гладко, а закінчена програма практично не містила помилок.

З тих пір структурне програмування почало робити помітний вплив на розвиток програмного забезпечення всіх рангів.

2.5. Пошук та усунення помилок

Едсгер Дейкстра, зробивши свій внесок до перегляду принципів складання програм і створення самих мов програмування, на цьому не заспокоївся. Не менше, ніж оператор GOTO, його дратували відлагодження та тестування – складові завершального етапу створення будь-якого програмного забезпечення. На тестування та відлагодження часто потрібно більше часу, ніж на інші фази розробки програми, разом узяті. До того ж, як вказував Дейкстра, **тестування може показати лише наявність помилок, але не їх відсутність**. Навіть коли здавалося, що програма повністю відлагоджена, нові вхідні дані могли викликати відмову, або, як говорять програмісти, “аварію”.

Замість того, щоб займатися тестуванням програм, Дейкстра запропонував перевіряти їх математичними методами. Стосовно маленьких програм йому вдалося добитися певних успіхів, проте великі програми не піддавалися таким прийомам. Дейкстра визнав, що це завдання важко досяжне, але не обов’язково нереальне.

Останнім часом широко звстосовується технологія попереднього моделювання та верифікації і розробки програм за допомогою скінченних автоматів – MODEL CHECKING. При цьому кожна верхівка таких автоматів передбачає опис дій, які мають виконуватись у відповідному стані автомату. Вони описуються спеціальним чином за допомогою так званих структур Крипке із застосуванням темпоральної логіки. Остання визначає порядок виконання певних дій і забезпечує синхронізацію виконання окремих гілок (потоків) програм у часі. Ця технологія широко застосовується для створення розподілених та паралельних програм і на даний час має реальне промислове впровадження. Автори цієї технології – Е. Кларк, О. Грамберг та Д. Пелед нагароджені престижною премією імені А. Тюринга в галузі комп'ютерних наук.

2.6. Нелегке мистецтво програмування

Написання комп’ютерних програм вимагає особливої точності. Це справедливо для програмного забезпечення будь-якого рангу – від простої до складної програми, наприклад, для управління повітряним рухом у районі аеропорту.

Складання програм було б особливо складним, якби для цього використовувалися нулі та одиниці двійкового коду. На самому **нижньому рівні** це – єдиний доступний комп’ютеру спосіб спілкування. Проте програмісти створили спеціальні мови, які дозволяють конструювати набори машинних команд, не оперуючи рядками, що складаються з двійкових символів. Існує безліч мов високого рівня, які призначені для розв’язку певного класу завдань. Жодна з цих мов не може бути повністю визнана універсальною. Наприклад, мова для розробки наукових програм-додатків погано підходить

для програми обробки текстової інформації. Хоча в принципі її можна було б пристосувати для нового завдання, але це зажадало б певних витрат та зусиль.

Таким чином, вибір відповідної мови високого рівня – це перший крок у процесі створення програмного забезпечення. Наступний крок полягає в описі як самої роботи, що покладається на комп'ютері, так і методів її виконання. Такий опис робиться у вигляді алгоритму, тобто складання покрокового плану дій. **Алгоритм повинен бути бездоганний** з логічної точки зору – інакше в результаті виконання програми або виходять неправильні результати, або програма зовсім не працюватиме.

Нарешті, програміст повинен проаналізувати які дані підлягають обробці, підібравши для них якнайкращі методи обробки. Якщо програма пов'язана з процесами, де вирішальну роль видіграє чинник часу, то найкращими методами вважаються ті, які забезпечують максимальну швидкість.

2.7. Обчислювальна техніка та програмування в СРСР

Зараз, на початку 21 століття, молодим людям, що звикли бачити навколо себе тільки іноземну радіотехніку та персональні комп'ютери (ПК), важко уявити собі, що в Радянському Союзі колись була самостійна електронна промисловість, власні оригінальні комп'ютери та системи програмування. Тим не менше, але це так. Більш того, на початковому етапі розвитку СРСР майже не відставав від Заходу, а за рівнем ідей іноді випереджав іноземні розробки. Але роботи радянських вчених майже не знайшли віддзеркалення у популярній або професійній літературі. В основному це було пов'язано з тим, що історія становлення вітчизняної обчислювальної техніки та системного програмного забезпечення (ПЗ) була тісно пов'язана з розвитком оборонної промисловості. Практично всі кращі зразки обчислювальної техніки працювали на "оборонку" і були засекречені.

Історія радянської обчислювальної техніки починається з 1948 року і пов'язана з іменами трьох батьків-засновників. Видатного вченого Сергія Олексіевича Лебедєва (1902 – 1974), який багато років очолював Інститут точної механіки та обчислювальної техніки АН СРСР (ITMiOT, з 1953 року і до кінця життя був директором цього інституту), який зараз носить його ім'я, Ісаака Семеновича Брука (1902-1974) і Башира Іскандеровича Рамєєва (1918-1994).

У 1945 р. С.О. Лебедєв створює першу аналогову обчислювальну машину для розв'язання системи звичайних диференціальних рівнянь. У 45 років, будучи вже відомим вченим, він зайнявся зовсім новим напрямком – створенням обчислювальної техніки.

У 1946 р. С. О. Лебедєв був запрошений в Академію наук України на посаду директора Інституту енергетики. Через рік Інститут енергетики розділився на два, і С. О. Лебедєв став директором Інституту електротехніки АН України.

С.О. Лебедєв

Перша ЕОМ, яка була створена під керівництвом Лебедєва С.О., отримала назву МЭСМ (Малая Электронная Счетная Машина, 1948 р.). МЕСМ була зібрана у секретній лабораторії за 15 км. від Києва у Феофанії. Там було відновлено зруйновану війною будівлю колишнього монастирського готелю, виділена прилегла до неї ділянка лісу та садиби.

Тут і були розгорнуті роботи по створенню МЕСМ (на даний час його адреса: вул. Академіка Лебедева, б. 19). Почала працювати МЕСМ з 1951 р. у Києві. Вона містила 6000 електронних ламп і займала ціле крило двоповерхової будівлі. Оперативна пам'ять налічувала 94 16-розрядних слова, швидкодія складала близько 50 оп/с. У машині Лебедєв незалежно від фон Неймана реалізував основні принципи класичної архітектури: зберігання програм в оперативній пам'яті, двійкову систему числення. У 1952 та 1953 рр. МЕСМ була єдиною ЕОМ в колишньому СРСР, на якій розв'язувалися найважливіші задачі того часу: в основному обчислення з області термоядерних процесів, космічної та ракетної техніки. Саме тоді розпочався розвиток радянської школи програмування.

Опис МЕСМ став першим підручником з обчислювальної технікі. Лабораторія С. О. Лебедєва виконала роль організаційного зародка обчислювального центру АН України, на базі якого згодом був створений один з найбільших у світі Інститут кібернетики імені В.М. Глушкова НАН України.

У 1952 р. на МЕСМ вирішувалися найважливіші науково-технічні завдання з області термоядерних процесів (Я. Б. Зельдович), космічних польотів і ракетної техніки (М. В. Келдиш, А. А. Дородніцин). У 1950 р. С. О. Лебедєв почав розробку Великої (згодом перейменованої в швидкодіючу) електронно-лічильної машини – БЕСМ АН СРСР.

У 1950 році Брук приступив до практичної реалізації проекту створення ЕОМ. За два роки зусиллями 9 осіб була побудована ЕОМ М-1, яка налічує всього 750 ламп. М-1 стала першою у Москві працюючою ЕОМ і одразу викликала великий інтерес в наукових колах. Працюючи із швидкістю 15-20 операцій за секунду, вона виконувала серйозні розрахунки для відомства академіка Курчатова і для КБ під керівництвом Корольова.

Подальший розвиток обчислювальної техніки в Радянському Союзі був тісно пов'язаний з військовими розробками і оточений атмосферою таємничості. Ініціатором урядових рішень був академік Михайло Олексійович Лаврентьєв, майбутній організатор новосибірського Академмістечка. У 1950 році він звернувся до Сталіна з листом, в якому звертав увагу на велику важливість обчислювальних машин для оборони країни. Реакція вождя була негайною: ухвалою уряду було наказано почати паралельну розробку ЕОМ в Академії наук СРСР та Міністерстві машинобудування і приладобудування. Така практика була типовою для найважливіших військових замовлень, вона створювала конкуренцію між розробниками. В результаті в 1952-1953 році майже одночасно на світ з'явилися дві повномасштабні вітчизняні ЕОМ.

У Москві, в ІТМіОТ (Інститут Точної Механіки і Обчислювальної Техніки) була пущена перша черга **БЭСМ** (Большая Электронная Счетная Машина), проектувати яку С. О. Лебедєв почав ще у Києві, але закінчив після переїзду до Москви. БЭСМ-1 була для свого часу самою швидкодіючою машиною в Європі (8-10 тисяч операцій за секунду, у 1955 р. на міжнародній конференції в Дармштадте БЕСМ була визнана кращою в Європі ЕОМ). Закладений С. А. Лебедєвим ще у МЕСМ принцип розпаралелювання процесу обробки інформації був при цьому суттєво розвинений і залишається до цих пір одним з основних при побудові супер-ЕОМ. «Вміти дати напрям – ознака геніальності», – сказав про таких людей німецький філософ Ф. Ніцше.

«**Стріла» –** перша вітчизняна серійна ЕОМ середньої продуктивності зі швидкодією близько 2000 оп/с – була створена у московському СКБ-245 під керівництвом Б. І. Рамеєва. Всього було побудовано 7 примірників цієї машини.

У 1955 р. С.О. Лебедєв почав розробку М-20 (цифра в назві вказувала на очікувану швидкодію – 20 тис. операцій в секунду). Такої швидкості обчислень тоді не мала жодна машина у світі. Ідеологію і структуру М-20 розробляв С.О. Лебедєв, систему команд – М.Р. Шура-Бура, схемотехнику елементної бази – П.П. Головистиков.

“Лебедєвські” БЭСМ-1, БЭСМ-2 та М-20 (1958 р., 20 тис. опер./с.) були засновані на серійних вітчизняних електронних лампах. Потім були створені їх напівпровідникові варіанти БЭСМ-3М, БЭСМ-4, М-220 (200 тис. опер./с) та М-222.

Лампова ЕОМ М-20 з рекордною для машин того часу швидкодією 20 000 оп./с, об’ємом пам’яті у 4096 45-розрядних слів була відома цілим рядом цікавих нововведень, авторами яких був головний ідеолог машини Лебедєв та його заступник, видатний математик і один з перших радянських програмістів Михайло Романович Шура-Бура. Треба відзначити, що перші кроки по створенню основ системного програмного забезпечення Лебедєв зробив у машині М-20, де були реалізовані можливості написання програм у **мнемокодах** (пізніше – на **Автокоді** або **Асемблері**). І це значно розширило коло фахівців, які змогли скористатися перевагами обчислювальної техніки.

Шура-Бура разом з Лебедєвим розробляв систему команд і керував створенням математичного забезпечення М-20. Це була перша радянська ЕОМ із власним системним програмним забезпеченням – пакетом стандартних підпрограм ІС-2. Машина вважалася секретною, оскільки вона обслуговувала радянські космічні і атомні програми, але потім її розсекретили і почали встановлювати у ВНЗ та наукових організаціях.

З появою М-20 у Сибірському відділенні Академії наук СРСР (Новосибірськ), Андрій Петрович Ершов у 1960 р. заснував неформальну школу програмістів, в якій за його участю були розроблені такі мови, як “Альфа”, перша оптимізуюча система програмування для складних мов, потім Альфа-6 та транслятори до них. У 1970-х роках Єршов розробив типову схему трансляції для багатьох мов (Алгол, Симула 67, Паскаль, Ада та Модула-2) та

універсальний програмуючий процесор «Бета», а також обчислювальну систему "АІСТ" (Автоматична Інформаційна Станція). Проект АІСТ поєднував у собі створення як багатопроцесорної архітектури, так і розвиненого програмного забезпечення, що використовує ті можливості, які надавала ця архітектура. Усе системне програмне забезпечення АІСТ було написане на власній мові системного програмування Епсилон.

Найвищим досягненням колективу С.А. Лебедєва з'явилася розробка у 1966 році напівпровідникової ЕОМ БЭСМ-6. Модель БЭСМ-6 була спроектована з використанням попереднього імітаційного моделювання роботи її операційної системи, що дозволило знайти безліч оригінальних технічних рішень. У розробці архітектури БЭСМ-6 активну участь приймали програмісти зі створеної за ініціативою Лебедєва лабораторії математичного забезпечення.

БЭСМ-6

Довгий час БЭСМ-6 із швидкодією 1 млн. операцій за секунду вважалася однією з кращих ЕОМ у світі. Рекордна швидкодія цієї машини була досягнута не за рахунок швидкісних елементів, а завдяки досконалій архітектурі процесора. Технічні рішення, які були знайдені при проектуванні машини, були настільки оригінальними, що ця модель побила всі рекорди живучості. Покоління ЕОМ змінювали одне одного, а легендарна БЭСМ-6 продовжувала випускатися і надійно працювати впродовж більш ніж 25 років!

Академік Корольов говорив, що без своєчасно зроблених Лебедєвим машин було б складно почати освоювати космос. Навіть у знамениту формулу 3К – так журналісти називали засекречених учених І.В. Курчатова, С.П. Корольова та М.В. Келдиша – знаючі люди і самі конструктори додавали букву Л (ім'я С.О. Лебедєва також трималося в секреті). Правомірність формули «3К+Л» не викликає сумнівів, всі розуміли, що без ЕОМ не могло б бути таких досягнень. Сергію Олексійовичу Лебедєву вдалося сформувати вітчизняну школу досліджень і розробок, яка багато років за низкою напрямків утримувала лідируючі позиції у світі. Тільки з середини 70-х рр. XX в. почалося поступове відставання від західних розробників. Багато в чому це було пов'язано з копіюванням серії IBM, а також з намітившимся розривом в області елементної бази.

В кінці 60-х рр. С. О. Лебедєв, В. А. Мельников та А. А. Соколов як головні конструктори очолили роботи із створення багатомашинної обчислювальної системи АС-6, до складу якої входила і БЕСМ-6. Вона втілила у собі багато нових ідей, які в подальшому склали основу майбутніх супер-ЕОМ. Ця система використовувалася в космічній програмі «Союз-Аполлон» та наступних запусках космічних кораблів. Вся інформація оброблялася часто значно раніше, ніж у колег із США.

Міжнародне визнання прийшло до Лебедєва через багато років після його кончини. Міжнародне комп'ютерне товариство IEEE Computer Society удостоїло С.О. Лебедєва своєю вищою нагородою – медаллю «Computer Pioneer Award» за видатні новаторські роботи в області створення обчислювальної техніки. На медалі написано: «Сергій Олексійович Лебедєв. Розробник і конструктор першого комп'ютера в Радянському Союзі. Нагороджений у 1996».

В одному з самих засекречених і закритих міст атомної галузі – Снєжинське (Челябінськ-70) в Російському федеральному ядерному центрі, – науково-дослідному інституті технічної фізики (НДІТФ) під керівництвом Володимира Федоровича Тюріна було розроблено ряд операційних систем для ЕОМ М-20, М-220 (В.Ф. Тюрін, В.С. Авраменко) та БЕСМ-6.

Розробниками операційної системи "ДІСПАК" (диспетчер пакетної обробки завдань) для родини машин БЕСМ-6, у якій вперше в СРСР була реалізована робота з магнітними дисками, були В. Ф. Тюрин, М. І. Шулепов, Ю. В. Озорнин, Л. В. Шинкарева, С. А. Зельдинова, І. Д. Бокова та В. І. Зуєв. Три розробники ОС були удостоєні Державної премії.

Плід праці системних програмістів вийшов настільки вдалим, надійним, стабільним у роботі і при цьому простим в експлуатації, що завод-виготівник вирішив використати ОС в якості основної серійної операційної системи для усього Радянського Союзу. Через чотири роки на базі ДИСПАКа була створена нова версія ОС, названа ДИАПАК (діалогово-пакетний режим роботи), яка дозволяла автоматизувати практично усі дії оператора і розділити управління проходженням завдань через ЕОМ на функціональні ділянки. Основний внесок у створення системи зробили Ю. В. Озорнин, Н. І. Шулепов, Л. В. Шинкарева, В. К. Корякин та Г. П. Охрименко. Робота була виконана на рівні світових стандартів системного програмування.

Для машини БЕСМ-6 була розроблена велика кількість пакетів прикладних програм, за допомогою яких були вирішені задачі найбільш передових напрямів науково-технічного прогресу. Був накопичений величезний фонд програм користувачів та досвід його експлуатації. Можна цілком зрозуміло сказати, що на базі БЕСМ-6 був створений "золотий фонд" програмного забезпечення, значення якого важко переоцінити.

Лебедєв розробив також основи створення багатопроцесорних комплексів, обчислювальних мереж, структурно-програмних операційних систем, алгоритмічних мов програмування. Як наслідок, вже на початку 80-х років його учнями, вперше в світі, були створені могутні багатопроцесорні суперскалярні комплекси ЕЛЬБРУС-1 (1978 р., 10 млн. опер./с, ЕЛББРУС-2 (1985 р.). 1990 рік ознаменувався випуском та введенням в експлуатацію векторно-конвеєрної супер-ЕОМ ЕЛЬБРУС-3.1 (розробники – Г.Г. Рябов, А.А. Соколов, А.Ю. Бяков), з продуктивністю у однопроцесорному варіанті – 400 мегафлоп (команд обробки дійсних даних).

Успіхи в обчислювальній техніці та програмуванні дали поштовх і для наукової діяльності. Так, до 1978 року програмістам в СРСР, які розробляли навіть чудові комплекси, практично було дуже важко захистити дисертацію на здобуття вченого ступеня кандидата (доктора) фізико-математичних наук, якщо вона не містила досліджень в області теоретичної математики або фізики.

Одного разу, а точніше 4 квітня 1978 року, на засіданні Вченої ради Обчислювального центру АН СРСР ім. С.А. Лебедєва йшов захист дисертації на здобуття вченого ступеня кандидата фізико-математичних наук. Тема

дисертації була пов'язана зі штучним інтелектом. В процесі обговорення дисертації виникло питання, а чи може представлена дисертація претендувати на здобуття ступеня кандидата фізико-математичних наук. Математичних досліджень немає, теорем немає, а є якісь евристичні оцінки дій, що здійснюються «штучним інтелектом» і є програма, написана на мові ЛІСП, яка дозволяє визначати дії «штучного інтелекту».

Слід зазначити, що ОЦ Академії наук, не дивлячись на свою прикладну назву, був зовсім не рахунковою фабрикою, обслуговуючи інші інститути академії. Це був (і є) першокласний науково-дослідний інститут, в якому у ті роки працювали такі видатні математики, як академіки А.Дородніцин, А.Марков, Г.Поспелов. Це далеко не повний список імен, якими може пишатися російська наука і які працювали у той час у ОЦ АН.

Між членами вченої ради виникла жвава дискусія про правомірність захисту дисертації на ступінь кандидата фізико-математичних наук, а не на ступінь кандидата технічних наук. Ступінь кандидата технічних наук присуджується за роботи, в яких основою дисертації є вирішення прикладних завдань. Думки розділилися, і захист дисертації опинився під питанням. Вирішальною виявилася думка, яка була висловлена директором інституту академіком А. Дородніциним. Він висловив тезу, що кожну програму можна розглядати як теорему про можливість розв'язання деякої задачі. І якщо програма працює, то можна стверджувати, що теорема доведена. Ця теза була підтримана А. Марковим, який запропонував тільки уточнити тезу. У більшості випадків неможливо визначити, чи завжди програма дає правильну відповідь. Тому пропонувалося прийняти наступне твердження: **кожна програма є теоремою, справедливість якої доведена для тих випадків, в яких програма дає правильну відповідь**.

А. Марков – великий російський математик, саме він дав одне з найпоширеніших визначень поняття алгоритму, яке отримало його ім'я – нормальні алгорифми Маркова. **Теза Дородніцина–Маркова** одноголосно була прийнята Вченою радою як аксіома, постулат. Дисертація була успішно захищена, а претендент отримав ступінь кандидата фізико-математичних наук.

Пензенська школа сформувалася на базі створеній там у 1955 році філії СКБ-245, пізніше перейменованого в НДІ математичних машин (НДІММ). Отримавши Державну премію за машину «Стріла», туди з групою учнів поїхав Б.І. Рамеєв. Очолюваний ним колектив створював універсальні ЕОМ під фірмовою маркою «Урал». Модель «Урал-1» була завершена вже у 1954 р. Це була перша мала серійна ЕОМ, вона мала 800 електронних ламп і розміщувалася в одній стійці. Оперативна пам'ять розміром 1024 слова була виконана на магнітному барабані, тому швидкодія машини визначалася швидкістю обертання барабана і складала всього 100 оп./с («Урал-2», 200 оп./с).

Подальші моделі цієї серії – «Урал -11, -14, -16» виготовлялися на напівпровідниках, вони мали вельми пристойну продуктивність (50, 45 і 100 тис. оп./с відповідно).

Чудова особиста доля Б.І. Рамеєва. Цей талановитий інженер не мав формальної вищої освіти, він був виключений з інституту як «син ворога народу» і не мав права викладати у ВНЗ. Тільки на початку 60-х років йому без захисту дисертації був присуджений вчений ступінь доктора наук.

Київську школу очолив радянський вчений – академік Віктор Михайлович Глушков (1923-1982). У 1962 р. на базі лабораторії обчислювальної техніки і математики Української Академії наук, в якій під керівництвом С.А.Лебедєва створювалася перша радянська ЕОМ, він організував перший в країні Інститут кібернетики, який незабаром став провідним науковим центром в області кібернетики, інформатики та обчислювальної техніки.

Внесок В.М. Глушкова в інформатику величезний. Він не тільки розробляв абстрактні теоретичні питання конструювання ЕОМ, але і безпосередньо керував їх технічним проектуванням та виготовленням, створював алгоритми та методики застосування комп'ютерів у найрізноманітніших областях.

В.М.Глушковим та його колегами були створені декілька оригінальних моделей комп'ютерів: машина загального призначення «Київ», управляюча ЕОМ «Дніпро», мала напівпровідникова ЕОМ «Промінь». Найвидатнішою розробкою київської школи стала машина для інженерних розрахунків «МІР», яка перша у світовій практиці, мала апаратний інтерпретатор мови програмування високого рівня – АЛМІР.

В **Білорусії** був побудований інститут і завод, де під керівництвом Віктора Володимировича Пржіялковського (1960 р.) розроблялися та випускалися ЕОМ серії «Мінськ»: лампова «Мінськ-1», потім перша масова вітчизняна напівпровідникова «Мінськ-2» і, нарешті, «Мінськ-32», яка мала модульну структуру та операційну систему.

З 1969 р. Мінською філією Науково-дослідного центру електронної обчислювальної техніки (НДЦЕОТ) керував Р. П. Лопато, академік Національної академії наук Білорусії, член-кореспондент РАН, лауреат Державної премії СРСР. У 1972 р. став директором НДІЕОМ, який був перетворений у філію НДЦЕОТ. За 28 років під керівництвом Р. П. Лопато в НДІЕОМ було створено 15 моделей ЕОМ "Мінськ", п'ять моделей ЄС ЕОМ, моделі ПЕОМ, 6 обчислювальних комплексів спеціального призначення, інші засоби обчислювальної техніки та системи програмування. Р. П. Лопато по праву вважається одним із засновників Мінської школи обчислювальної техніки.

В **Єреванському** інституті математичних машин було створено дві родини машин: загального призначення «Раздан» та інженерних розрахунків «Наїрі», в яку також був вбудований апаратний інтерпретатор.

Оцінюючи у цілому позитивно розвиток обчислювальної техніки в СРСР у 50-і – 60-і роки, слід визнати, що, не дивлячись на багато оригінальних розробок, за загальним рівнем СРСР істотно відстав від передових країн. Причому відставання постійно збільшувалося, склавши на початок 70-х років ціле покоління ЕОМ. Тому було декілька причин, як об'єктивних, так і суб'єктивних.

По-перше, у кінці 40-х – початку 50-х років Радянський Союз одразу після Другої світової війни був втягнутий в «холодну» війну, кинувши всі ресурси на гонку озброєнь. Обстановка тотальної таємничості, відсутність реальної конкуренції, орієнтація виключно на військові застосування виявилися згубними для науково-технічного прогресу.

По-друге, холодна війна призвела до самоізоляції і зустрічної міжнародної ізоляції СРСР у світовій спільноті. Зарубіжні наукові публікації доходили до радянських вчених з великим запізненням, поїздки за кордон були практично неможливими.

На початку 70-х авторитетна комісія, проаналізувавши зарубіжний досвід, прийшла до невтішних висновків – за якістю і кількістю обчислювальної техніки СРСР відстав від цивілізованого світу на 8-10 років, до того ж не було сформовано середовище впровадження ЕОМ – не вистачало інженерів-електронщиків, програмістів, не було спеціальної літератури. І тоді уряд вирішив зробити "великий стрибок" у справі комп'ютеризації країни. Щоб виграти час, вирішено було не розвивати далі вітчизняні розробки, а копіювати конструкції передових на ті часи зарубіжних ліній ЕОМ. Була мобілізована технічна розвідка, сконцентровані сили оборонних галузей промисловості, організований міжнародний розподіл праці в рамках Ради економічної взаємодопомоги (РЕВ).

Передбачалося налагодити масове виробництво двох родин обчислювальних машин. Перша називалася Єдиною системою ЕОМ (ЄС ЕОМ) і повинна була відтворити IBM S/360. Наймолодшу модель ЄС-1010 планувалося виробляти в Угорщині, модель ЄС-1020 – в Болгарії, ЄС-1040 – в НДР, а моделі ЄС-1030, 1050 і найстаршу ЄС-1060 – в СРСР.

Друга родина повинна була покрити потребу народного господарства у міні-комп'ютерах і позначалася СМ ЕОМ (Система малих ЕОМ). Оскільки кожне з міністерств наполегливо пропонувало свій зразок для успадкування, до єдиної думки прийти не вдалося, в результаті під маркою СМ-1 та СМ-2 ховалися аналоги міні-ЕОМ фірми Hewlett-Packard, а моделі СМ-3-4 відтворювали популярні комп'ютери PDP-11 фірми DEC.

Лінію супер-ЕОМ, враховуючи передовий рівень БЭСМ-6, вирішили не припиняти і не копіювати, а залишити за вітчизняними розробниками. В рамках цього проекту в ITMiOT згодом були розроблені багатопроцесорні ЕОМ серії «Ельбрус». Розробники «Ельбрусов» пишаються тим, що за роки, коли у нас займалися переважно відтворенням західної архітектури, їм вдалося зберегти властиву лебедєвскому колективу творчу атмосферу і створити повністю самостійний, оригінальний проект. Висока продуктивність багатопроцесорного комплексу досягалася за рахунок оригінальних архітектурних рішень.

Другою знаковою фігурою в галузі обчислювальної техніки є учень С.О. Лебедєва академік В.С. Бурцев. Всеволод Сергійович зробив значний внесок у створення систем управління об'єктами в реальному часі.

В.С. Бурцев почав свою діяльність під керівництвом видатного вченого академіка С.О. Лебедєва ще до закінчення Московського енергетичного інституту. Темою його дипломної роботи була система управління першої радянської швидкодіючої електронної машини – БЕСМ АН СРСР. Вже на дипломному проектуванні він став одним з провідних розробників у створенні цієї системи.

У 1973-85 рр., будучи головним конструктором многопроцессороного обчислювального комплексу (МОК) «Ельбрус-1» і «Ельбрус-2», Бурцев В.С. поряд з принциповими схемотехническими питаннями велику увагу приділяв конструктивно-технологічним питанням, питанням системи охолодження та підвищення інтеграції елементної бази, а також питанням автоматизації проектування. У процесі створення МОК «Ельбрус-2» з його ініціативи та з безпосередньої участі були створені нові швидкодіючі інтегральні схеми, високочастотні групові роз'єми, багатокристальні і великі інтегральні схеми, мікрокабелі, прецизійні багатошарові друковані плати.

У 1980 р. були закінчені роботи по створенню МОК «Ельбрус-1» загальною продуктивністю 15 млн операцій за секунду. У 1985 р. успішно завершені Державні випробування десятипроцессорного МОК «Ельбрус-2» продуктивністю 125 млн операцій за секунду. Обидва комплекси були освоєні в серійному виробництві.

У 1986-98 рр. було розроблено структуру супер-ЕОМ, яка була заснована на новому не фон-неймановскому принципі і забезпечувала суттєве розпаралелювання обчислювального процесу на апаратному рівні. Ця архітектура використовувала новітні принципи оптичної обробки інформації, і дозволила досягти продуктивності 10^{10}-10^{12} операцій за секунду.

Ідеолог архітектури «Ельбрусів» Борис Арташесовіч Бабаян, який багато років також працював в колективі Лебедєва, та її головний конструктор вважали їх першими у світі комерційними (тобто такими, що вийшли за рамки наукових лабораторій) суперскалярними машинами. Більшість з цих властивостей значно пізніше будуть реалізовані у західних комп'ютерах. Перша комерційна машина такого типу на заході з'явилася лише у 1992 році (через 14 років!).

Хоча «Ельбруси» створювалися для потреб радянської системи протиракетної оборони та атомної галузі, ці машини вважалися універсальними і могли застосовуватися не тільки для оборонних потреб, але й у великих ОЦ, які працювали на науку та промисловість. Проте основними користувачами таких високих технологій залишалися ВПК та радянські атомщики. В них і досі стоять та працюють «Ельбруси».

Підводячи підсумки, можна сказати, що в Радянському Союзі були значні досягнення у використанні обчислювальної техніки та створенні програмних комплексів. Так, політ вже перших штучних супутників Землі (і, природно, усіх подальших, у тому числі і першого у світі космонавта Юрія Гагаріна) супроводжувався оперативними розрахунками траєкторій, аналізом та обробкою телеметричних вимірювань, що проводилися за допомогою ЕОМ. Літако- та ракетобудування, створення атомної та водневої зброї та інші роботи проводилися виключно за допомогою різних типів вітчизняних ЕОМ (Стріла, Весна (1 прим.), БЭСМ-4, М-20, М-220, БЭСМ-6, ЕЛЬБРУС).

Розробки ядерної і термоядерної зброї в СРСР сприяли не лише швидкому розвитку обчислювальної техніки, але і обчислювальної математики. У реалізацію атомного проекту СРСР великий вклад внесли видатні математики академіки АН СРСР М.В. Келдиш, С.Л. Соболев, А.А. Дородніцин, М.А. Лаврентьєв.

У зв'язку з цим на закінчення слід привести висловлювання заступника наукового керівника Всеросійського науково-дослідного інституту експериментальної фізики (ВНДІЕФ) і керівника математичного відділення професора Івана Денисовича Софронова (радянський і російський учений в області прикладної математики), який був опонентом у Авраменко В. С. при захисті дисертації к.ф-м.н. у ІПМ АН СРСР імені М. В. Келдиша, і давши на неї дуже хороший відгук (система МОДІС, яка була створена Авраменко В. С., впроваджена у ВНДІЕФ).

«Математичні розрахунки при створенні атомної зброї грають особливу роль. Ну, а головна наша гордість – те, що уся ядерна зброя, яка стоїть на

бойовому чергуванні, була розрахована нашими математиками, за нашими методиками, нашими програмами з використанням тільки вітчизняної обчислювальної техніки. Слід зазначити, що наше відставання від американців по продуктивності обчислювальних машин було нижче на два порядки або більше. Проте, паритет у зброї був досягнутий. І заслуга в тому наших математиків та програмістів, наших фізиків-теоретиків і конструкторів.

Як ні парадоксально це звучить, але наше відставання в обчислювальній техніці зробив великий вплив на інтенсивність математичних розробок, на ефективність створюваних математичних методів. Саме воно примушувало нас винаходити економічні алгоритми рішення задач, розробляти досконалішу технологію розрахунків, нарешті, залучати до роботи більші, ніж в США, математичні колективи. Треба зазначити, що сьогодні у нас немає конкурентів в рішенні таких складних прикладних завдань. Ми маємо в розпорядженні найбільший пакет програм і методик для вирішення, у тому числі і тривимірних задач математичної фізики, механіки суцільного середовища, нейтронної фізики тощо. У нас найбільший математичний колектив в країні. І ми непогано озброєні, потужність нашого машинного парку мало кому в Росії поступиться.

Ядерна зброя, проте, вимагає до себе пильної уваги, її потрібно правильно зберігати, вона має бути безпечна і готова до застосування, її потрібно удосконалювати, що не можна робити без випробувань, які нині заборонені. Де ж вихід? У математичних розрахунках.

Зовсім нещодавно це вважалося неможливим. Але вже минулого року президент США Клінтон зібрав своїх генералів і розробників ядерної зброї і оголосив, що у них створена віртуальна ядерна бомба. Тобто американці заявили, що вони уміють робити зброю без випробувань. Для цього п'ять років тому була придумана ініціатива СКІ і з'явилися терафлопні машини. Американці вже готові робити, можливо, не дуже сучасні, але заряди без випробувань. Поки, правда, жодного подібного заряду на озброєння не поступило. Але, проте, вони до цього наполегливо йдуть. Я думаю, що максимум років через п'ять вони оголосять, що можуть це робити, і почнуть ставити на озброєння цілком сучасні заряди без випробувань. Думаю, що ми підемо таким самим шляхом. Для цього нам необхідно створити супер-ЕОМ. Зараз в нашій країні є дві закінчені розробки – машина МВС, створена НДІ «Квант» і Інститутом прикладної математики РАН, і наша машина МП-Х-Y. Вони відрізняються по архітектурі, складу елементної бази і програмному забезпеченню. Ми починали розробляти машину разом з ІПМ РАН, але у нас виявилися різні точки зору, і кожен став робити свою систему.

Нещодавно ми зайвий раз переконалися, що їх підхід до проектування ЕОМ, призначених для вирішення наших завдань, не годиться. Двічі намагалися ми рахувати свої завдання на машинах, розроблених конкурентами, і обидва рази дива не сталися – ефективність їх машин для наших потреб виявилася неприйнятно низькою. Тому рахувати завдання другого ядерного століття ми на них не стали. Сам факт, що спочатку ми повинні були

відповідати за свої розрахунки і прораховувати тисячі варіантів виробу, змусив нас працювати відповідально і великими колективами.

Задачі в 1000 годин машинного часу для нас нічого рекордного навіть 20 або 30 років тому не представляли. Були задачі і в 10 000 годин безперервної роботи машини. І це з урахуванням того, що беззбійність роботи вітчизняної обчислювальної техніки оцінювалася по ТУ не більше ніж 10 годин. Нагальною необхідністю при цьому виявилося розпаралелювання обрахунку задач, тобто рішення однієї задачі відразу на кількох машинах. У 1993-1994 рр. у ВНДІЕФ створена мультипроцесорна система з продуктивністю 1 гигафлоп, в якій використовуються спеціально розроблені і виготовлені на вітчизняній елементній базі комутатори з пропускною спроможністю 90 Мб за секунду. На цих екземплярах мультипроцесорів нами розроблені паралельні програми для вирішення низки виробничих задач – тривимірні задачі адіабатичної газової динаміки, газової динаміки з урахуванням теплопровідності, тривимірні задачі перенесення нейтронів в кінематичному наближенні, двомірні задачі газової динаміки з урахуванням теплопровідності на нерегулярній сітці тощо.

Один зразок нашої машини був куплений Ливерморской національною лабораторією США для відладки тих програм, які ми розробляємо за їхнім замовленням. Зараз ведуться інтенсивні роботи із створення наступної мультипроцесорної системи з продуктивністю 120-240 гигафлоп, з наступним нарощуванням її до потужності в 1 терафлоп. Будучи базовою моделлю у ряді МП-Х, цей мультипроцесор дозволить створити наступного року аналог сучасних супер-ЕОМ США типу ASCI RED і Origin 2000. Зараз вони аналогів в Росії не мають».

Авторам цього ретроспективного огляду пощастило розробляти програмні комплекси різної складності майже на всіх вітчизняних ЕОМ, зокрема 1963-1964 р. – Урал-2; 1965-1990 р. – М-20, М-220, Мир, БЭСМ-4, БЭСМ-6, ЕЛЬБРУС-1; 1976-1997 р. ЄС ЕОМ (1022, 1040, 1060), СМ ЕОМ.

3. Розквіт та хаос програмного забезпечення

Розквіт програмного забезпечення припадає на 70-і роки, але для поборників якості програмного забезпечення вони виявилися далеко не кращими. Проблеми 60-х років – складніші завдання і менш кваліфіковані програмісти – в 70-х роках тільки посилилися. З іншого боку, недоступна та трудомістка компіляція пішла у минуле. Поява ПК змінила правила гри, знявши обмеження, які змушували досягати високої якості програм у 60-і роки.

Настільні обчислення перетворили комп'ютер на інструментарій, дійсно доступний для всіх, а не тільки для математиків, університетських вчених та військових. Нікому більше не доводилося годинами, днями та тижнями чекати, поки буде надана можливість скористатися компілятором, оскільки вбудований компілятор мав кожен ПК. Компілятор можна було запустити у будь-який час, коли програміст хотів швидко перевірити синтаксис. Навіщо, сидячи за столом, ретельно аналізувати кожен рядок, коли можна запустити компілятор – хай перевіряє він?

Можливо подібну «лінь» програмістів стимулював той факт, що змінився і вид завдань, для яких створювалося програмне забезпечення. Програмісти, в основному, більше не займалися кодуванням математичних алгоритмів. Вони створювали системи, які дозволяли працювати швидше і ефективніше. Вони писали програми, про які раніше не доводилося і мріяти.

У 60-х роках компетентні розробники самі виконували весь аналіз та тестування. Але у 70-х, коли почався бум у створенні автоматизованих рішень для нових завдань (і в додаванні нових можливостей до існуючих систем), з'явився великий попит на програмістів. Тому кожен, хто хоч щось знав про програми, почав займатися програмуванням, і в цьому ажіотажі про тестування просто забули.

Звичайно, організації, що займалися розробкою програмного забезпечення, не хотіли зовсім відмовлятися від тестування, але багато хто з них віддав його на відкуп нефахівцям, по суті перетворивши на тестувальників підсобний персонал. Зараз тестувальники, як і раніше, вимушені миритися з непрестижністю своєї роботи, коріння якої занурюється у ті роки.

Код, що був написаний у 70-х роках – найгірше, що є у сучасному програмуванні. Для нього навіть існує спеціальна назва: «успадкований код». Він лякає, він заплутаний, і з ним дуже складно працювати. Більшість фахівців прагнуть усіма можливими засобами уникнути необхідності його підтримувати. Врешті-решт, чужий код іноді зрозуміти дуже складно, і одна помилка, яка зроблена у модифікованому коді, може породити непередбачувані побічні ефекти, незалежно від того, наскільки ретельно цей код протестований.

Нарешті, ще одним важливим нововведенням 70–х років стали **метрики –** показники, що, як передбачалося, повинні характеризувати «якість» коду, але часто їх інтерпретували дуже суб'єктивно. Десятиліття хаосу виявилося не самим кращим часом для введення метрик. Ця теорія будується на кількісних

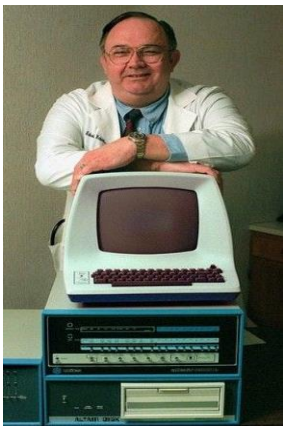
аспектах початкових текстів – кількості циклів, розгалужень, умовних виразів, кількості перфорядків тощо. Замість того, щоб намагатися визначити, чи є дане програмне забезпечення функціонально коректним, розробники могли просто підрахувати кількість елементів у коді, щоб встановити його складність.

У 70-х роках це було цікаве заняття, і, можливо, давало багатьом розробникам відчуття задоволеності своїм кодом. Проте на даний час використання метрик залишається виключенням з правил. Більшість розробників ігнорують параметри, розуміючи, що хороші програмісти можуть створити дуже хороший код, у якого окремі метрики опиняться далеко не кращими, а погані програмісти можуть написати поганий код, який за своїми метриками буде просто блискучим. Отже, на жаль, «репутація» метрик була зіпсована, оскільки спочатку вони не відображали дійсність. Хороші, сучасні метрики функціональної коректності і зараз не користуються довірою через асоціації з метриками складності коду, які були запропоновані у 70-х роках минулого століття.

В цілому про це десятиліття розвитку програмного забезпечення можна сказати, що воно було орієнтоване на код, а не на якість. До кінця 70-х років стало зрозуміло, що зміни в галузі просто необхідні. І перші книги з тестування програмного забезпечення (Glenford Myers, The Art of Software Testing, Wiley, 1979; Р. Майерс, Надійність програмного забезпечення. М. “Мир”, 1980) з’явилися у кінці цього десятиліття. Це була вірна ознака того, що насуваються зміни.

У 1975 р. січневий номер журналу Popular Electronics розбурхав ентузіастів комп’ютерної техніки. У цьому номері була опублікована стаття, в якій був описаний перший у світі набір для зборки міні-комп’ютера, що був здатний конкурувати з комерційними моделями, – “Альтаїр-8800”. У наступні дні сотні чеків, кожен на суму 397 дол. (сьогодні це близько 1,5 тисяч доларів), прийшли на адресу компанії Micro Instrumentation and Telemetry System, яка розташована у м. Альбукерк, шт. Нью-Мексіко. “Альтаїр” побудували офіцер ВПС США з дипломом інженера-електронщика **Едвард Робертс** та **Джим Байбі** на основі мікропроцесора “Intel-8080”, поклавши початок цілій галузі – роздрібній торгівлі комп’ютерами для приватних та юридичних осіб. Мікропроцесор “Intel-8080” – перший 8-розрядний мікропроцесор, на якому було розміщено декілька інтегральних мікросхем на одному кремнієвому кристалі, створив у 1970 р. співробітник компанії Intel **Едвард Хофф**. Це революційний винахід кардинально перевернув уявлення про комп’ютери як про громіздких, ваговитих монстрів.

Перший ПК Altair-8800 фирми MITS



Едвард Робертс

Ім’я “Альтаїр” вигадала дочка редактора журналу Popular Electronics Леслі Соломона, на честь зірки з улюбленого нею фільму “Зоряний похід”. Треба віддати цим хлопцям належне, вони старалися щосили, майже одразу відкривши для себе і попутно для всієї земної кулі раніше нерозвідану маркетингову нішу. Вони здогадалися пропонувати своїм покупцям не тільки готові комп’ютери, але й комплектуючі до них, заклавши базові основи сучасних принципів торгівлі ПК. Напевно, саме в їх офісі вперше прозвучало так добре знайоме всім нам слово «**апгрейд» (**upgrade – «модернізація»).

"Батько персонального комп’ютера" Генрі Едвард Робертс, винахід якого поклав початок створенню Біллом Гейтсом та Полом Алленом компанії Micro-Soft (тоді вона писалася саме так), помер у віці 68 років у лікарні від запалення легень 1 квітня 2010 р. (life.pravda.com.ua/private/). Один із засновників компанії Apple Стів Возняк заявив, що Робертс "зробив найважливіший крок, що привів нас до всього, що ми маємо зараз".

Перші "Альтаір-8800" керувались за допомогою вимикачів і не мали екрану. Проте зборка комп’ютера з комплектуючих, не дивлячись на його вельми обмежені можливості, була не простою справою. Навіть зібравши машину, її власник міг розраховувати тільки на власні сили, оскільки “Альтаїр” випускався без програмного забезпечення, і купити його було ніде. Програма для “Альтаїр” повинна була задовольняти двом вимогам. По-перше, бути досить короткою, оскільки об’єм оперативної пам’яті складав всього 256 байт. По-друге, склавши коротку двійкову програму, користувач міг ввести її в комп’ютер, маніпулюючи перемикачами.

Серед любителів, що не побоялися “викинути гроші” заради сумнівного задоволення, було немало тих, хто через десятиліття, опинився в рядах революціонерів комп’ютерної індустрії. **Джон Бекус** (один з творців Фортрану), говорив, що програміст повинен був стати майстерним винахідником, щоб пристосувати своє завдання до особливостей комп’ютера. Потрібно було зуміти розмістити програму та дані у мізерному об’ємі пам’яті і подолати труднощі, що були пов’язані з організацією введення та виведення інформації.

Для нового покоління програмістів, більшість яких складали студенти, нарешті можна було дістатися до комп'ютера та реалізувати свої можливості в області застосування комп'ютерів. Ці "комп'ютерні наркомани" не могли миритися з обмеженими можливостями системи пакетної обробки, тому вони шукали способи, які дозволили б їм дістатися до комп'ютера. Іноді студенти-любителі замовляли машинний час під ім'ям кого-небудь, хто мав доступ до комп'ютера, а то і таємно проникали в порожній машинний зал вночі.

3.1. Місце народження хакерів

"Розсадником" подібного неприборканого захоплення комп'ютерами став Массачусетський технологічний інститут (MTI), де програмісти-маніяки стали самі себе називати **хакерами (hacker)**. Спочатку цим прізвиськом нагороджували бешкетних студентів, а пізніше членів клубу залізничного моделювання. Члени клубу навіть створили власний словник. Члени клубу з фантастичним завзяттям просиджували нескінчені години за монтажем системи. Свій проект, що не мав практичного значення і який виконувався заради задоволення від самого процесу роботи, вони назвали "хэк" (hack, один з перекладів – людина, виснажена важкою роботою). Природно, ті, хто займався подібним проектом, почали називатися хакерами.

Залучення хакерів до комп'ютерів було лише справою часу, оскільки до інститутського могутнього комп'ютера фірми ІБМ вони мали дуже обмежений доступ. Все змінилося з появою у 1958 р. порівняно невеликої машини TX-0, в якій замість електронних ламп використовувалися транзистори. До того ж, машина була забезпечена звуковим виведенням та навіть світловим пером, за допомогою якого можна було малювати прямо на екрані монітора. Вперше в історії MTI комп'ютерні ентузіасти дістали у власні руки машину, в яку могли самостійно вводити перфострічку зі своїми програмами та одразу же отримувати результати.

Отримавши дозвіл використовувати машину TX-0, коли вона вільна, хакери почали творити дива. Один запрограмував машину таким чином, що вона виконувала музику Баха, інший створив програму, яка з величезною швидкістю перетворювала числа з арабської системи числення у римську. Ще один хакер (професор) створив прототип комп'ютерної гри.

Хакери вважали, що створене ними програмне забезпечення повинно бути доступним усім іншим хакерам (це назвали **етикою хакера**). Копії програм (зазвичай на перфострічках) циркулювали по неформальній мережі хакера та потрапляли в машинні зали університетів та корпорацій по всій території США. Питання про авторські права у той час не виникало. Предметом суперечок подібні випадкові запозичення стали лише тоді, коли парк машин значно виріс, і програми набули комерційної цінності.

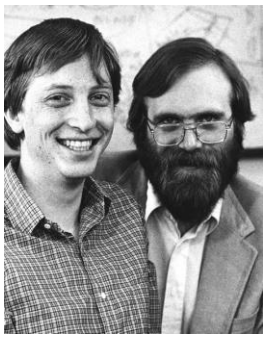
Відтепер володіння комп'ютерами перестало бути винятковою монополією великих та багатих організацій, що у свою чергу стимулювало створення великої кількості нових різноманітних компонентів програмного забезпечення.

Але, не дивлячись на досягнуті успіхи, подальший прогрес в області апаратних та програмних засобів був зобов'язаний не індустрії міні-комп'ютерів і не компаніям, а розробникам ПЗ. Новий етап настав з появою персонального комп'ютера, створення якого стало можливим завдяки винаходу мікропроцесора. Відтепер кожен хакер та навіть просто любитель міг стати володарем комп'ютера.

3.2. Два чародії програмування

Серед найгарячіших ентузіастів новинки машини "Альтаїр-8800" був **Пол Аллен** – молодий програміст фірми "Хоніуелл" (Honeywell) та його друг дитинства **Гейтс**, **Уїльям Генрі** (**Біл**, 1955 р.н.) студент 1-го курсу Гарвардського університету.

Ще у 1975 році вони продемонстрували MITS (Micro Instrumentation and Telermetry Systems) компілятор Бейсіка. Цей компілятор був написаний всього за вісім тижнів і ніколи не перевірявся на комп'ютері. Але все запрацювало з першого разу. Такий вдалий почин надихнув авторів кинути все і перемкнутися повністю на розвиток свого дітища. Вони вирішили, що машина "Альтаїр-8800" потребує програмного забезпечення. Те, що вони зробили після покупки машини, назавжди зберіглося в історії комп'ютерів. Вони подзвонили у фірму, що виготовляла машину "Альтаїр", і повідомили, що розробили програму, яка може переводити Бейсік у машинний код для "Альтаїра", і готові її продати фірмі. Власник фірми, не замислюючись, заявив, що готовий купити інтерпретатор Бейсіка для машини "Альтаїр".



Пол Аллен та Біл Гейтс

Біл Гейтс узяв в Гарварді академічну відпустку (він так ніколи і не закінчив навчання, Гейтс був відрахований з юридичного факультету Гарвардського університету) і разом з Аленом, який також був відрахований з Університету штату Вашингтон, взялися за роботу. Коли перший закінчений варіант інтерпретатора був готовий, друзі вивели його на перфострічку, і Аллен вилетів в Альбукерк, щоб показати програму. На той час одну з машин "Альтаїр" фірма обладнала додатковою платою пам'яті та пристроєм читання перфострічки. Але коли літак почав заходити на посадку, Аллен раптом згадав, що вони з Гейтсом забули маленьку, але досить важливу деталь – **програму завантаження**, без якої мікропроцесор просто не зможе ввести інтерпретатор у пам'ять. І Ален тут же у літаку на клаптику паперу накидав варіант завантажувача прямо у кодах машини.

На наступний ранок Аллен вніс останні доповнення, ввів стрічку з інтерпретатором Бейсіка у машину і почав чекати. Через декілька хвилин програма була завантажена. Аллен запустив її у роботу, і, перевівши дух, набрав елементарне завдання: PRINT 2+2. Машина вмить відповіла: 4. Робертс (голова фірми) і його колеги були схвильовані і здивовані, оскільки до цього

ніхто з них не бачив, щоб ця машина що-небудь робила. Робертс купив програму, а Аллен та Гейтс організували компанію **Microsoft**. Продовжуючи вдосконалювати інтерпретатор, вони написали також низку інших програм для малих машин, і **вперше створили індустрію з виробництва та продажу програмного забезпечення**.

Знання юриспруденції все-таки стали у нагоді Гейтсу. Складений ним контракт на використання Бейсика у комп'ютерах Altair увійшов згодом до всіх підручників по комп'ютерному праву. Він передбачав відрахування 500 доларів за кожен екземпляр проданої програми. Отримуючи відрахування авторського гонорару з кожного проданого екземпляра інтерпретатора, вони зрозуміли, що розробка та продаж програмного забезпечення обіцяла величезні бариші. Так і сталося, у 2003 році журнал Forbes назвав Біла Гейтса найбагатшою людиною на Землі, а Пола Аллена – №4 у списку найбагатших людей світу.

Гранична простота Бейсика дозволила вбудовувати його у постійну пам'ять найдешевших мікрокомп'ютерів, завдяки цьому в середині 80-х років на ньому працювали мільйони користувачів в усьому світі. Залишаючись незмінно популярним, Бейсик ріс і розвивався разом з персональними комп'ютерами. У 1984 році, до 20-ліття народження, Курц і Кемені у співдружності з Американським інститутом стандартів (ANSI) розробили нову стандартну версію мови (True Basic), а потім за його розвиток з усією ґрунтовністю узялася Microsoft, для якої він став фірмовою мовою та символом успіху.

3.3. Перші промислові стандарти

Поява на початку 1975 р. першого комерційно поширюваного персонального комп'ютера Альтаїр-8800 на основі 8-розрядного мікропроцесора Intel-8080 було тим поштовхом, що одразу декілька фірм оголосили про намір створити на основі мікропроцесора Intel-8080 свої персональні комп'ютери. Комп'ютер Альтаїр-8800 продавався за ціною близько 500 дол. І хоча можливості його були дуже обмежені (оперативна пам'ять складала всього 256 байт, клавіатура та екран були відсутні), його поява була зустрінута з великим ентузіазмом: у перші ж місяці було продано декілька тисяч комплектів машини. Покупці забезпечували цей комп'ютер додатковими пристроями: монітором для виведення інформації, клавіатурою, блоками розширення пам'яті тощо. Незабаром ці пристрої стали випускатися іншими фірмами. Створення Полом Алленом і Білом Гейтсом для комп'ютера «Альтаїр» інтерпретатора мови Basic дозволило користувачам досить просто спілкуватися з комп'ютером, і легко писати для нього програми. Це також сприяло популярності персональних комп'ютерів.

Ключову роль у змінах, що сталися в області апаратних та програмних засобів, зіграв **Стефан (Стів) Возняк** (Steve Wozniak, 1950 г.р, шт. Каліфорнію) разом зі своїм другом і партнером по роботі Стівеном **Джобсом** (Steve Jobs).

У Кремнієвій Долині (Silicon Valley – Селіконовая долина, географічної назви якої не існує, неправильний переклад silicon-кремній та silicone-селікон) і понині розповідають стару байку про те, як перший персональний комп'ютер був зібраний в гаражі двома молодими геніями (або одним генієм і одним пройдисвітом – залежно від того, хто розповідав). Здебільше ця байка дійсно правдива, хіба що дещо прикрашена.

С.Джобс, С.Возняк

Стів Возняк був студентом коледжу і жив у гуртожитку Каліфорнійського університету. І ось одного дня його мати принесла йому статтю з жовтневого номера журналу "Есквайр" (Esquire) за 1971 рік. Стаття називалася "Секрети блакитної коробочки".

У ній розповідалося про підпільну мережу "телефонних фанатів", попередників комп'ютерних зломщиків, що з'явилися через десятиліття. Такі блакитні коробочки могли перехитрити бухгалтерські машини телефонної компанії "Белл" (Bell) і давали можливість безкоштовно дзвонити у будь-яке місто світу. Далі – більше: за допомогою правильного чергування гудків та дзвінків телефонні спритні ділки зуміли проникнути в урядову комунікаційну мережу, яка обслуговувала правоохоронну систему США, Збройні сили і навіть ЦРУ.

Ця стаття настільки захопила Возняка, що він тут же кинувся дзвонити своєму другу Стіву Джобсу. Джобс, що був на чотири роки молодше Возняка, вчився на другому курсі Вищої школи і жив на околиці Сан-Хосе. Хлопці дружили вже багато років, їх пов'язували загальний інтерес до електроніки та загальна відсутність інтересу до молодіжних тусовок, що робило друзів відщепенцями серед однолітків.

Отже, два Стіва вирішили зайнятися виготовленням блакитних коробочок. Чотири наступних місяця вони працювали і нарешті зібрали сигнал-генератор за схемами з журналу "Популярна електроніка". Але добитися стабільності роботи генератора, аби ввести в оману апаратуру телефонних станцій, виявилося справою нелегкою. Тому Возняк сконструював невеликий цифровий пристрій, який працював від дев'ятивольтової батареї. Тепер генератор видавав стійкий сигнал, і два Стіва почали грати в телефонні ігри.

Але, як би зумовлюючи майбутнє партнерство, Джобс вирішив припинити ці технічні розваги, він мав намір робити гроші на винаході друга. Деталі для блакитної коробочки коштували 40 доларів, на збірку схеми у Возняка йшло не більше години часу. Джобс запропонував такий план: він оплачує комплектуючі, Возняк виконує монтаж, а прибуток вони ділять порівну. Возняк погодився. І незабаром друзі щосили торгували своїми хитромудрими коробочками, розносячи їх по кімнатах чоловічого гуртожитку. Потім коробочки стали продавати в Південній Каліфорнії завдяки одному приятелеві, який взявся бути їх представником в регіоні Беверлі Хиллз. Друзі продали

більше двохсот таких коробочок за ціною 150 доларів за штуку. Вони не попалися. А ось деяких з покупців притягнули до суду. Рік потому телефонна компанія вдосконалила свою комутаційну систему, і блакитні коробочки виявилися некорисними.

Але життя йшло своєю чередою. Возняк завалював один іспит за другим. Джобс взагалі став випадати з життя. Він оселився в авангардистському студентському кампусі в Орегоні, говорили, що він призвичаївся до маріхуани, захопився вивченням дзен-буддизму. Зрозуміло, що дуже скоро Джобса виключили з коледжу, і він повернувся додому до батьків. Заявивши, що колись він працював на фірмі "Хьюлетт-паккард" (Hewlett-Packard), Джобс зумів отримати роботу в компанії "Атарі" (Atari, при оплаті 5 доларів за годину) по створенню першої відеогри "Понго". Працюючи у фірмі, Джобс примудрився налаштувати проти себе майже всіх співробітників через те, що постійно сував ніс в їх роботу і обзивав всіх "тупоголовими". Сам же ходив на роботу брудним, таким, начесаним, загалом нечепурою, навіть для програміста.

Начальник Джобса, Ел Олкорн, перевів Джобса на роботу у нічну зміну, де він працював наодинці. При цьому начальник сказав, що колеги не люблять Джобса тому, що від нього погано пахне. Нічна зміна означала, що ніхто не заглядатиме Джобсу через плече, а це було важливо, бо Джобс особливою старанністю не відрізнявся. Йому належало розробити гру під назвою "Розбий стінку", яка, фактично, була модифікацією "Понго", з тією лише різницею, що в ній додавалася стінка, яка складалася з цеглинок і яка зникала, якщо в них потрапляли.

Джобс мав намір самостійно виконати всю роботу, але потім зрозумів, що йому не впоратися, тому звернувся за допомогою до Возняка. Довірливий друг погодився допомогти. Фірма "Атарі" виплачувала премію кожного разу, коли конструктору вдавалося скоротити кількість мікросхем у грі, оскільки при цьому знижувалася собівартість продукції. Возняк же був великим фахівцем з економії комплектуючих при одночасному поліпшенні якості.

Джобс погодився віддати другу половину преміальних, які, за його словами, складали 700 доларів. Возняк працював над грою, думаючи, що Джобс продасть її "Атарі" за 700 доларів, а він отримає свої 350. Лише декілька років потому він дізнався, що насправді Джобс отримував за неї не 700, а 7000 доларів. Крім того, Джобс вибив в "Атарі" кредит за цю гру і ніколи і нікому не говорив про участь у ній Возняка.

Отримавши гроші, Джобс виїхав у свою комуну на ферму Орегона. Тим часом Возняк, що жив з батьками після відрахування з Берклі, отримав роботу на фірмі "Хьюлетт-паккард" і зайнявся розробкою калькулятора. Але насправді його цікавили комп’ютери. Він почав відвідувати Комп’ютерний клуб електронщиків-любителів «Саморобний комп’ютер» в Менло Парку. Двічі на місяць у клубі збиралися інженери, програмісти та техніки. Вони обговорювали проблеми конструювання малогабаритних комп’ютерів.

Завданням Комп’ютерного клубу було вирвати монополію на комп’ютери з рук фінансової олігархії і зробити ці розумні машини доступними для простого

народу. У них були підстави розраховувати на успіх. Явно була зрозумілою перспектива швидкого та різкого здешевлення мікропроцесорів, зменшення їх розмірів, зниження складності та збільшення потужності.

Члени клубу почали обмінюватися схемними рішеннями, пропонувати один одному ідеї, ділитися або торгувати електронними деталями. Розробники клубу були на сьомому небі від щастя, коли у січні 1975 року журнал "Популярна електроніка" оповістив світ про випуск "**Альтаїра-8800**" (Altair) – першого комп'ютерного набору для електронщиків-любителів. Набор деталей коштував 375 доларів. Але навіть для комп'ютерних асів "Альтаїр" з'явився предметом глибокого розчарування. Набір продавали без програмного забезпечення, без операційної системи, тому програмувати машину був змушений сам покупець. Збирання та паяння теж належало виконати користувачеві, йому також пропонувалося самому підключити додаткові мікросхеми запам'ятовуючих пристроїв, телетайп або монітор, гнучкий диск (у 1971 компанія IBM представила гнучкий диск, який став стандартом для зберігання даних). І лише після цього комп'ютер ставав більш менш функціональним. Фактично, вартість зростала до 3000 доларів. Але тим не менше "Альтаїр" демонстрував доступність створення комп'ютера силами любителів.

Клуб витратив багато часу, обмірковуючи потенційні можливості малогабаритних комп'ютерів. Заводські виробники розраховували, що завжди контролюватимуть випуск обчислювальних машин та додаткових пристроїв до них, а молоді конструктори клубу вже переконалися, що персональні комп'ютери можуть управляти охоронною сигналізацією, автомобільним двигуном, системами опалювання, можуть складати музику, редагувати тексти, грати в карти та управляти діями роботів.

Возняк не міг дозволити собі купити "Альтаїр". Створювати ж свій комплект деталей він не хотів, тому почав конструювати власний мікрокомп'ютер з компонентів, здобутих ним у Джобса. Новий комп'ютер Возняка не був, строго кажучи, його першою обчислювальною машиною. Ще підлітком він з приятелем зібрав примітивний пристрій, який умів виконувати множення цілих чисел. Але коли Возняк став демонструвати своє дітище одному з репортерів, з калькулятора повалив дим і кімнату наповнив їдкий запах – калькулятор трагічно згорів.

Але тепер винахідника вже не підстерігала невдача. Комп'ютер працював саме так, як повинен був працювати. Проте, коли Возняк запропонував права на виробництво комп'ютера своїм босам на "Хьюллет-паккарде", ті, ввічливо вислухавши його, відмовили, заявивши, що "Хьюлетт-паккард" не зацікавлений в подібній продукції. А ось Джобс зацікавився. Він почав умовляти Возняка продавати плати комп'ютера членам клубу і іншим електронщикам-любителям. Джобс запропонував створити власну комп'ютерну компанію. Він і назву підібрав. По спогадах про фермерську комуну в Орегоні, а також аби виявитися в телефонному довіднику перед фірмою "Атарі", Джобс вирішив назвати компанію "Еппл Комп'ютер" (Apple).

Для виготовлення першої сотні плат були потрібні гроші. Возняк продав свій нагородний калькулятор за 500 доларів. Джобс, не бажаючи видавати, що у нього на банківському рахунку є 5000 доларів за гру "Розбий стінку" від фірми "Атарі", продав вантажівку "Фольксваген" і таким чином забезпечив собі половину прибутку підприємства.

Комп'ютерний клуб відреагував на все це гробовим мовчанням. Але один з членів клубу (Террелл) володів трьома магазинами з продажу електронних компонентів. Він сказав, що йому потрібні не печатні плати, які користувач ще повинен змонтувати, а повністю готові комп'ютери. Він мав намір придбати 25 комп'ютерів за умови постачання їх у готовому вигляді. Для двох Стівів таке замовлення було манною небесною, але у той же час і тяжким тягарем. Вони нашкрябали близько 25000 доларів (взяли кредит) та зайнялися збиранням комплектуючих. Гараж батьків Джобса друзі перетворили на складальний цех, а членів сім'ї умовили допомагати у роботі.

Коли друзі закінчили збирання, вони підрахували всі витрати. Із схильності до химер вони призначили таку ціну на свій виріб: 666 доларів та 66 центів. Проте Террелл був не в захопленні від комп'ютерів: машини були всього лише стопками печатних плат без корпусу, без клавіатури, без програмного забезпечення, без дисководів або іншого засобу для завантаження програм. Не було і монітора. Проте Террелл заплатив гроші, найняв фахівця з корпусів, і лише після цього комп'ютери набули гідний товарний вигляд.

Але перші персональні комп'ютери "Еппл" (**Apple-1**) ще довго порошилися на полицях магазина Террелла. А Возняк тим часом вже почав роботу над комп'ютером "Еппл-2" (**Apple-2**). Цього разу він вирішив створити такий комп'ютер, який приголомшить колег з Комп'ютерного клубу. І клавіатура у ньому буде, і вбудоване матзабезпечення Бейсик (BASIC), і звук, і колір, так що можна буде грати у "Розбий стінку". Крім того, в новій модифікації заради функціональності будуть передбачені вільні гнізда в панелі, а також усілякі інші новини для оновлення та коректування, для реалізації будь-яких спеціальних вимог замовника. Джобс висунув свою пропозицію – укласти комп'ютер в компактний вилитий пластмасовий корпус, який вигідно відрізнявся від незграбних ящиків з листового металу інших мікрокомп'ютерів.

1 квітня 1976 р. Джобс та Возняк заснували компанію Apple Computer, яка була зареєстрована у 1977 р. Її першою продукцією став покращений комп'ютер Apple-1. Цей комп'ютер відрізнявся простотою та компактністю і призначався, головним чином, для любителів та ентузіастів. Усього було продано 600 машин. Незабаром після Apple-2 з'явилися Apple Lisa, Apple Macintosh, які стали ще компактнішими та зручнішими у користуванні. Ось тепер Возняк і Джобс були нарешті готові штурмувати комп'ютерний світ. І вони вийшли переможцями. Успіх компанії виявився феноменальним, і у 1980 вона стала акціонерною компанією.

Apple-1

Lisa

Macintosh

На початку 1980-х років фірма приступила до виробництва принтерів та програмного забезпечення. У зв'язку з тим, що Возняк потрапив в авіакатастрофу на початку 1981 р., йому припало на деякий час залишити фірму, але вже у 1983 р. він повернувся і пропрацював у компанії до 1985 р., після чого створив нову компанію з розробки систем віддаленого доступу для використання у побутових комп'ютерах. Фірма, яка називалася CL - 9, проіснувала до кінця 1989 р. Після виходу з Apple у 1986 р. Возняк завершив освіту в Берклі із ступенем бакалавра в області комп'ютерних наук та електротехніки. Стів Джобс і до останього часу був головою компанії Apple.

Успіх Альтаїр-8800 та Apple змусив багато фірм також зайнятися виробництвом персональних комп'ютерів. Персональні комп'ютери стали продаватися вже у повній комплектації, з клавіатурою та монітором, попит на них склав десятки, а потім і сотні тисяч примірників на рік.

У червні 1976 р. у продажу з'явився комп'ютер Sol-20 від компанії Processor Technology за ціною $2129 (комплект для самосбора коштував $995). Коріння Sol-20 сходить до SOL-PC, що являла собою просту системну плату без блоку живлення та корпусу (з такої конфігурації починалася історія комп'ютерів Apple – для того часу це було нормою).

Компьютер Sol-20

Відмінною рисою Sol-20 була вбудована у корпус клавіатура, а також бічні панелі з натурального горіхового дерева. Комп'ютер використовував популярну тоді шину S-100 (що вперше з'явилася у комп'ютері Altair 8800 у 1975 р.) з п'ятьма слотами розширення. "Серцем" Sol-20 слугував процесор Intel 8080 з тактовою частотою 2 Мгц.

Об'єм оперативної пам'яті складав 1 Кб (з розширенням до 64 Кб). В якості накопичувачів могли використовуватися також популярні на той час дешеві касети. До Sol-20 також можна було підключати і дисковод для гнучких 8-дюймових дисків ємністю 384 Кб даних. Як операційне середовище Sol-20 використовував CP/M (Control Program for Microcomputers), а також Basic.

Неймовірний успіх мав комп'ютер TRS-80, що був випущений компанією Radio Shack у серпні 1977 р., про що говорять результати продаж – за один

місяць було реалізовано десять тисяч примірників TRS-80 (Model I). Усього за $599,95 користувач отримував і продуктивний комп'ютер (як CPU використовувався Zilog Z-80A, 1,77 МГц) та 12-дюймовий монохромний монітор і касетний привід.

Додатково можна було купити (за $299) спеціальний модуль розширення, що встановлювався під монітором, який містив порт для підключення принтера, контроллер флоппи-диска (до чотирьох), додаткову оперативну пам'ять (до 32 Кб).

Успіх першої моделі позначився на подальшому довгому житті всієї родини TRS-80. У жовтні 1979 р. з'явився ПК під назвою TRS-80 Model II, а в липні 1980 р. TRS-80 Model III.

TRS-80

У 1979 р компанія Atari випустила комп'ютер Atari 400, а також більш просунутий Atari 800. Перший коштував $549,95, другий – на $450 дорожче. Стандартно Atari 800 володів оперативною пам'яттю об'ємом 8 Кб (що розширювалась до 48 Кб), чотирма внутрішніми слотами розширення, виходами TV-video та RGB-video, а також контроллером для підключення зовнішнього флоппі-дисковода Atari 810, що був розрахований на встановклення 5-дюймових дискет ємністю 90 Кб.

Наявність співпроцесорів для обробки графіки та звуку робила Atari 800 чудовою гральною машиною. Нарешті для Atari 800 був матричний принтер Atari 820, який друкував із швидкістю приблизно 40 знаків за секунду. У той час флоппі-дисководи коштували дуже дорого (сотні доларів), тому у широкому вжитку були звичайні касети. З цієї причини було випущено багато касет не лише з програмами, але і з іграми, спеціально для запуску на Atari 400/800.

Atari 400

Atari 800

З появою персональних комп'ютерів з'явилося декілька журналів, які були присвячені персональним комп'ютерам. Але щоб куплений комп'ютер став корисним інструментом, їх власникам було потрібне програмне забезпечення, і у цьому їм готова була допомогти ціла армія програмістів та підприємців. Зростанню обсягу продаж дуже сприяли багаточисельні корисні програми, які

були розроблені для ділових застосувань. Програми зробили купівлю персональних комп'ютерів дуже вигідною для бізнесу: з їх допомогою стало можливим виконувати бухгалтерські розрахунки, складати документи і таке інше. Використання ж великих комп'ютерів для цих цілей було ще дуже дорогим.

Особливу популярність отримали текстові процесори. На початку 1975 р. кінорежисер **Шрейер** з Нью-Йорка у пошуках успіху, став одним з володарів набору "Альтаїр". Незабаром він удосконалив свою машину, підключивши до неї клавіатуру та телевізор, і почав продавати невеликі програми. По мірі ускладнення програм ставало все важчим керівництво до них. І одного разу Шрейера осяяло, що було б набагато зручніше готувати це керівництво за допомогою комп'ютера, а не друкувати на машинці. Він працював над своєю програмою по 16-20 годин на добу протягом року. Закінчивши роботу, він пустив свою програму в продаж під назвою "**Електронний олівець"** і повністю присвятив себе програмуванню.

У перші два роки після появи у світ "Електронний олівець" був фактично єдиним текстовим процесором, доступним масовому споживачеві. Проте по мірі розширення ринку персональних комп'ютерів та зростання його прибутковості інші підприємці послідували прикладу Шрейера. Одним з них був житель Нью-Йорка **Сеймур Рубінштейн**. Серед багатьох його задумок був і текстовий процесор, і спільно з програмістом **Джоном Барнебі** вони почали розробляти відповідну програму під назвою "**Уордстар" (WordStar).** Швидко завоювавши ринок, кількість користувачів цієї програми на протязі 6 років перевищило кількість усіх інших текстових процесорів разом узятих.

Поки Рубінштейн готував випуск свого текстового процесора, **Денієл Бріклін**, інженер-програміст фірми ДЕК (DEC, Digital Equipment Corporation), що створив популярну серію машин **PDP**, почав розробляти своє дітище.

Займаючись виснажливими обчисленнями, пов'язаними з визначенням зміни суми податку в результаті зростання ставки на витрати та прибуток підприємства, він раптом зрозумів, що є інший, зручніший спосіб виконання таких розрахунків. Основна проблема полягала у тому, що при зміні якийсь однієї цифри доводилося перераховувати усі похідні від неї величини.

Досвід програміста наштовхнув Брікліна на думку, що всі ці нудні розрахунки непогано б доручити комп'ютеру. У кінці 1978 р. він написав програму під назвою "**Візікалк" (VisiCalc – Visible Calculator).**

Спочатку програма була написана для Apple-2. Протягом першого року після початку продажу "Візікалк" збут персональних комп'ютерів різко зріс, сама програма розійшлася тиражем близько 100 тис. примірників, і залишалася "бестселером" цілий рік.

У 1981 році інженер американського космічного відомства NASA **Уейн Ретліфф (Ratliff, Wayne)** у вільний від роботи час почав створювати простеньку систему керування базою даних (СКБД) для ведення футбольної статистики. У її основу була покладена реляційна модель, але реалізація була найпримітивнішою, потужності персонального комп'ютера вистачало лише на

обробку кількох сотень записів. Ні про яке промислове використання такого виробу не могло бути й мови, але для побутових потреб цього вистачало. Ретліфф почав продавати програму під назвою «Вулкан», втім, без особливого успіху.

Реклама «Вулкана» попалася на очі Джорджу Тейту – підприємцеві без спеціальної освіти, що займається програмним бізнесом. Його фірма Ashton-Tate уклала з Ретліффом договір на виняткове право продавати програму. Оскільки до цього часу марка «Вулкан» була зайнята іншою фірмою, то потрібно було придумати іншу назву. Торговий агент запропонував назвати програму **dBase II**, це виглядало солідно і містило натяк на нібито існуючу першу версію.

Настільна СКБД dBase дуже швидко завоювала ринок, встановивши промисловий стандарт на системи подібного класу, і дуже швидко стала новим "бестселером". Її мова маніпулювання даними породила безліч успадкувань, а формат dbf став класичним для подання реляційних таблиць. Фірма Ashton-Tate вибилася у лідери комп'ютерного бізнесу, вона продовжувала вдосконалювати систему, випускаючи все нові версії dBase III, III+, IV і супроводжуючи свої дії галасливою рекламою. І майже так само стрімко її автори поповнили все зростаючі ряди мільйонерів, що нажили багатство на програмному забезпеченні.

Проте навички утриматися на верхівці ринку настільних СКБД Ashton-Tate не змогла. У кінці 80-х років утворився клон конкуруючих dBase-подібних систем (FoxBase, FoxPro, Clipper та інші), сумісних за мовою та форматам даних, які володіли кращими операційними характеристиками. Поступово слава dBase зійшла нанівець, а фірма Ashton-Tate була куплена компанією Borland, що вирішила освоїти нову для себе нішу. Забігаючи наперед, зауважімо, що ця покупка виявилася невдалою і слави Borland не принесла.

Сучасний ринок настільних СКБД досить багатий та різноманітний. Окрім FoxPro, що й досі залишається популярною, на ньому представлені системи Borland Paradox, Clarion та інші. Більш того, сучасні персональні комп'ютери досягли такої потужності, що на них можна запускати обмежені версії промислових СКБД типу Oracle, DB2, Informix тощо.

Природно, що не обійшлося у цьому секторі і без Microsoft. Вона придбала права на FoxPro, а також у складі професійної версії MS Office пропонує достатньо потужну MS Access. Не дивлячись на те, що система позиціонується як настільна і продається за відповідною ціною, в ній присутньо багато рис «справжніх» СКБД, зокрема, мова запитів SQL.

Незабаром ряди мільйонерів поповнив ще один розробник програмного забезпечення – **Мітчел Кепор**. Його програма "**Lotus 1-2-3"** завоювала широке визнання, де вперше були закладені елементи інтеграції, оскільки вона поєднувала у собі кращі якості системи "Візікалк" з графічними можливостями та засобами інформаційного пошуку.

Закінчивши у 1971 р. університет, Кепор два роки пропрацював диск-жокеєм, після чого перебрався до Кембріджа, де продовжив свою кар'єру вже

як програміст. Продавши свої авторські права на програму за 1.2 млн. дол., він пішов далі, заснувавши власну компанію по виробництву та продажу його універсальної системи “Lotus 1-2-3”. Своє підприємство Кепор відкрив дуже вчасно.

Наприкінці 70-х років розповсюдження персональних комп’ютерів навіть призвело до деякого зниження попиту на великі комп’ютери та міні-комп’ютери (міні-ЕОМ). Це стало предметом серйозного занепокоєння фірми IBM – провідної компанії з виробництва великих комп’ютерів, і у 1979 р. фірма IBM вирішила спробувати свої сили на ринку персональних комп’ютерів. Проте керівництво фірми недооцінило майбутню важливість цього ринку і розглядало створення персонального комп’ютера всього лише як дрібний експеримент – щось подібне до одного з десятків робіт, які проводилися у фірмі. Аби не витрачати на цей експеримент надто багато грошей, керівництво фірми надало підрозділу, відповідальному за даний проект, небачену у фірмі свободу. Зокрема, йому було дозволено не конструювати персональний комп’ютер «з нуля», а використовувати блоки, які виготовлялись іншими фірмами. І цей підрозділ сповна використовав наданий шанс.

Перш за все, як основний мікропроцесор комп’ютера був обраний новітній тоді 16-розрядний мікропроцесор Intel-8088. Його використання дозволило значно збільшити потенційні можливості комп’ютера, оскільки новий мікропроцесор дозволяв працювати з 1 Мбайтом пам’яті, а тоді комп’ютери все ще були обмежені 64 Кбайтамі.

У серпні 1981 р. фірма IBM, представивши свій комп’ютер IBM 5150 PC, офіційно підтвердила, що вона приступає до випуску **персональних комп’ютерів**, які стали самими купівельними комп’ютерами в історії. Ключовим нововведенням IBM PC стала системна шина ISA із стандартними слотами, що дозволяло вставляти у ПК різноманітні плати розширення (відео, звукові, мережні та інші адаптери).

Але компанія IBM зробила одну річ, про яку вона пізніше пошкодувала. Замість того щоб тримати проект машини у таємниці, як вона зазвичай робила, компанія опублікувала повні проекти, включаючи всі електронні схеми, в книзі вартістю $49. Ця книга була опублікована для того, щоб інші компанії могли проводити змінні плати для IBM PC, що підвищило б сумісність і популярність цього комп’ютера. На нещастя для IBM, як тільки проект IBM PC став широко відомий, оскільки всі складові частини комп’ютера можна було легко придбати, багато компаній почали робити **клони** PC і часто продавали їх набагато дешевше, ніж IBM. Хоча деякі компанії (такі як Commodore, Apple, Amiga, Atari) застосовували персональні комп’ютери з використанням своїх процесорів, а не Intel, потенціал виробництва IBM PC був настільки великим, що іншим компаніям доводилося пробиватися силою.

Вижити вдалося тільки деяким з них, та й то через те, що вони спеціалізувалися на вузьких областях, наприклад на виробництві робочих станцій або суперкомп’ютерів. Через один-два роки комп’ютер IBM PC зайняв провідне місце на ринку, витіснивши моделі 8-бітових комп’ютерів.

У 1982 р. Кепор випустив нову систему “Лотос 1-2-3” для 16-розрядних процесорів фірми IBM, витративши тільки на її рекламу 1 млн. дол. Витрати окупилися дуже скоро. Система “Лотос 1-2-3” зайняла перший рядок у списку найпопулярніших програм.

IBM 5150 PC

Своїми успіхами компанія Кепора була зобов’язана фірмі IBM, яка заполонила ринок своїми комп’ютерами. Іншим важливим наслідком вторгнення фірми IBM з’явився масовий перехід виробників комп’ютерів на нову операційну систему (ОС), що стала промисловим стандартом.

У 1974 р. компанія **Intel** (від INtegrated ELectronics) випустила Intel 8080 – перший універсальний 8-розрядний процесор, – для нього було потрібно ОС, за допомогою якої можна було б протестувати новинку. Компанія Intel залучила до розробок і написання потрібної ОС одного із своїх консультантів **Гері Килдолла** (Gary Kildall). Спочатку Килдолл з другом сконструювали контролер для 8-дюймового гнучкого диску, який був незабаром випущений компанією Shugart Associates, і підключили цей диск до процесора Intel 8080. Таким чином, з’явився перший мікрокомп’ютер з диском. Потім Килдолл створив дискову операційну систему, названу **CP/M (Control Program for Microcomputers**).

У 1977 р. компанія Digital Research переробила CP/M, щоб зробити цю систему придатною для роботи на мікрокомп’ютерах не тільки з процесорами Intel 8080, а також з іншими процесорами. Було написано безліч прикладних програм, які працювали у CP/M, що дозволило CP/M займати високу позицію у світі мікрокомп’ютерів впродовж 5 років.

На початку 80-х корпорація IBM розробила IBM PC і почала шукати для нього програмне забезпечення. Фірма IBM почала вести переговори з Килдоллом про розробку нової ОС, але коли він відлетів у двотижневу відпустку, співробітники IBM зв’язалися з Білом Гейтсом (Bill Gates), щоб отримати ліцензію на право використання інтерпретатора мови Бейсік. Вони також поцікавилися, чи не знає він ОС, яка працювала б на PC. Гейтс порадив звернутися до Digital Research. Фірма IBM почала вести переговори з Килдоллом про розробку нової ОС, але Килдолл запропонував перенести переговори після його відпустки (за іншими відомостями – він відмовився підписати строгі договори про нерозголошування, як того вимагала IBM). Корпорація IBM знову звернулася до Гейтса з проханням забезпечити їх ОС.

Після повторного запиту IBM Гейтс з’ясував, що у місцевого виробника комп’ютерів, Seattle Computer Products (SCP), є відповідна ОС **86-DOS**. У 1980 р. компанії SCP була потрібна ОС для їх модуля пам’яті з орієнтацією на новий 16-розрядний процесор Intel 8086. Вона попросила **Тіма Патерсона** (Tim Paterson) написати нову ОС, яку назвали **QDOS** (Quick and Dirty Operating System – в буквальному перекладі це звучить, як “швидка та брудна ОС”). У повній відповідності з назвою, вона була написана всього за кілька місяців, але

не була повністю протестована та відлагоджена. До кінця року Патерсон покращив свою ОС, яка була випущена під назвою 86-DOS.

При створенні ОС 86-DOS переслідувалися кілька цілей. Вона повинна була бути сумісною з програмами, які були написані для ОС CP/M, щоб дати можливість використовувати безліч прикладних програм. Тому 86-DOS повинна була мати той же API (Application Programming Interface – прикладний програмний інтерфейс). Щоб зробити 86-DOS ефективнішою, ніж CP/M, Патерсон написав її на мові Асемблер і вбудував у неї модуль роботи з накопичувачами FAT, який заснований на автономній системі роботи з накопичувачами з інтерпретатора мови BASIC фірми MicroSoft.

Білл Гейтс запропонував компанії викупити 86-DOS за 50 тис. доларів. Коли корпорація IBM захотіла деяких удосконалень у програмі, Біл Гейтс запросив для цієї роботи **Тіма Патерсона**, який написав DOS. Фірма IBM уклала контракт з компанією "**Microsoft"**, очолюваною Білом Гейтсом, на розробку ОС.

Нова видозмінена система, перейменована у **MS-DOS (Microsoft Disk Operation System)**, була добре сприйнята багатьма виробниками персональних комп'ютерів, що були сумісні з IBM PC. Система MS-DOS незабаром узурпувала місце, яке раніше належало CP/M, завоювавши до того ж ширший ринок. Коли у 1983 р. з'явився IBM PC/AT з центральним процесором Intel 80286, система MS-DOS вже міцно стояла на ногах. Пізніше система MS-DOS широко використовувалася на комп'ютерах з процесорами 80386 та 80486. Хоча первинна версія MS-DOS була досить примітивна, подальші версії системи виходили з усе краще розробленими властивостями, включаючи багато того, що було запозичене від CP/M та UNIX.

У 1984 р. Microsoft вийшла на ринок офісних додатків, випустивши Word та Excel для Apple Macintosh. Наступного року побачила світ операційна система з графічним інтерфейсом Windows 1.0 для IBM-сумісних ПК.

3.4. Дружній інтерфейс

В епоху «великих» ЕОМ та перших двох поколінь мікрокомп'ютерів, тобто до середини 80-х років правила гри задавав комп'ютер, а людина була вимушена пристосовуватися до його можливостей. Основним засобом введення інформації в ЕОМ була клавіатура, а виведення – алфавітно-цифровий дисплей. Спілкування людини з комп'ютером (точніше кажучи, з операційною системою рівня CP/M, MS-DOS) здійснювалося на жахливо далекому від природньої командній мові, засвоїти яку могли тільки професіонали. Поки ЕОМ було відносно небагато, таке становище здавалося прийнятним, але як тільки комп'ютери стали товаром масового попиту і потрапили на стіл до школярів та домогосподарок, проблема простого та інтуїтивно зрозумілого способу спілкування людини з комп'ютером вийшла на перший план. Без її розв'язання комп'ютер ніколи б не став дійсно персональним.

Пізніше, завдяки дослідженням зі створення дружнього інтерфейсу «людина – ЕОМ», що був проведений у 60-і роки **Дагом Енгелбартом** (Doug Engelbart), ця властивість ОС змінилася. У кінці 50-х рр за підтримки Міністерства оборони США Енгелбарт заснував в містечку Менло-парк недалеко від Стенфорда у відомій нам Кремнієвій Долині самостійний підрозділ інституту під назвою «Augmentation Research Center - ARC».

З електронікою Енгельбарт познайомився у роки другої світової війни, коли служив на флоті техніком на радіолокаційній установці. У 1951 році Енгелбарт, залишивши службу, поступив до аспірантури у Каліфорнійському університеті Берклі, де спеціалізувався в області електроніки.

У той час, коли студенти технічних коледжів працювали на комп'ютерах, що займали цілі зали, Енгелбарт роздумував про такі нечувані речі, як застосування обчислювальних машин у навчальному процесі, та нові способи взаємодії людини з машиною. Протягом декількох років ARC не мав можливості найняти співробітників, і Енгелбарт працював по суті наодинці.

Більш ніж за 15 років до того, як системи обробки текстів набули широкого поширення, Енгелбарт вже думав про друкарську машинку, яка, автоматизуючи роботу на нижчому рівні ієрархії, дозволить письменникові ефективніше використовувати верхні рівні свого інтелекту. "Наприклад, – писав він, – нарис майбутньої роботи можна швидко скласти з перебудованих фрагментів старих нарисів, поповнивши їх новими словами або абзацами, які легко вставити, користуючись клавіатурою".

Енгелбарт проаналізував також теоретичну основу маніпулювання символами і зображеннями. Але наукова громадськість, яка була пов'язана з обчислювальною технікою, залишила без уваги роботи і вислови Енгелбарта. Одним з небагатьох, хто розділяв його погляди, був **Роберт Тейлор,** фінансовий директор відділу методів обробки інформації ARPA (Advanced Research Projects Agency – Агентство перспективних досліджень). У 1964 р. Тейлор виділив Стенфордському інституту та його центру ARC річний бюджет у півмільйона доларів для цих досліджень. У Енгелбарта з'явилася можливість спробувати втілити свою теорію на практиці.

У 1964 році Енгелбарт винайшов маніпулятор "**миша"**, без якого важко уявити собі сучасний комп'ютер. Цей пристрій з технічною назвою "**Покажчик х-у-позиції для дисплея"** пересувався на прихованих роликах, мав зверху пару кнопок і з'єднувався кабелем з консоллю. Енгелбарт охрестив його "мишею" – через маленький розмір та кабель, що був схожий на хвіст. Протягом наступних чотирьох років з колективом обдарованих фахівців він фантазував та експериментував, створюючи і випробовуючи нові системи. Результатом цих експериментів став **графічний інтерфейс користувача** (**GUI**, Graphical User Interface), що складається з вікон, значків, різних меню та миші.

У 1968 р. Енгелбарта запросили виступити на представницькій конференції з обчислювальної техніки у Сан-Франциско. Енгелбарт вирішив скористатися цією пропозицією, щоб продемонструвати плоди своїх майже двадцятирічних праць і роздумів.

Один з піонерів комп'ютерної графіки Ендріє ван Дам охарактеризував доповідь Енгелбарта як "щось приголомшливе". Глядачі, можливо, навіть не усвідомлюючи того, спостерігали сцени з майбутнього. Хоча демонстраційна апаратура була кустарною і недосконалою, уявлення Енгелбарта про взаємодію людини з машиною того дня як би ожили перед аудиторією. Люди побачили, як легко та природно можна маніпулювати зображеннями, змінювати текст, швидко переходити від одного файлу до іншого. Роками Енгелбарт намагався викласти свої ідеї словами. Того дня у Сан-Франциско він нічого не пояснював, а просто показав все на сцені.

Після закінчення півторагодинної демонстрації публіка влаштувала Енгелбарту справжню овацію. Проте, як це відбулося практично з усіма ідеями Енгелбарта, впровадження продемонстрованих пристроїв у комерційне виробництво відбулося набагато пізніше, і плоди винаходів талановитого вченого пожинали вже інші.

Коли на початку 70-х років військові припинили фінансувати ARC, багато співробітників Енгелбарта перейшли на роботу в дослідницький центр фірми Xerox, що знаходиться поряд із Стенфордським університетом у містечку Пало-альто (Palo Alto Research Center – PARC). Розбагатівши на продажі копіювальної техніки, ця фірма вклала дуже великі гроші у наукові дослідження, причому пріоритетними вважалися не сьогохвилинні вдосконалення, а принципово нові високі технології.

Центр PARC у 1970-1980 роки зібрав багато видатних вчених і, безумовно, лідирував у багатьох областях обчислювальної техніки та інформатики. Окрім комп'ютера «Alto», тут була розроблена локальна комп'ютерна мережа «Ethernet», створений на ті часи кращий видавничий пакет програм «Ventura Publisher» і таке інше.

У 1971 році у відділі обчислювальної техніки PARC, стартував проект «Alto», керівником та натхненником якого був **Алан Кей** (Alan Kay), що тільки-но захистив докторську дисертацію у Стенфордському університеті. Як і Енгелбарт, Кей прагнув того, щоб комп'ютерні програми виконувалися за допомогою таких простих дій, як вибір команд або символів на екрані. Кей виконав теоретичну роботу з проблеми взаємодії людини з комп'ютером майбутнього. За його задумом, ідеальний комп'ютер формою та розмірами повинен нагадувати книгу (було придумано навіть назва «Dynabook» – динамічна книга), мати плоский кольоровий графічний монітор високої розподільчої здатності, за допомогою засобів зв'язку підключатися до будь-яких бібліотек світу і бути, до того ж, дешевим (не дорожче 500 доларів). Крім того, гіпотетичний комп'ютер повинен підтримувати таку просту та інтуїтивно зрозумілу мову спілкування, щоб з ним могла працювати навіть дитина: «Якщо людині для роботи з машиною доведеться перегортати керівництво з експлуатації, то ця система приречена». Навіть за сьогоднішніми мірками проект «Dynabook» виглядає фантастично, що ж можна говорити про технічну базу 70-х років? Проте, фірма Xerox надала Кею можливість спробувати хоча б частково реалізувати свою ідею у

чудових умовах PARC. Проект «Alto» якраз і був спробою створити «проміжний Dynabook».

У 1973 році комп'ютер «Alto» був побудований (за два роки раніше до офіційної демонстрації у 1975 р. "Альтаїра"). Хоча його габарити та ціна були далекі від заявлених у проекті «Dynabook», він є видатним досягненням обчислювальної техніки тих часів. Принципи, які були закладені у його основу, виявилися настільки прогресивними, що випередили свій час на ціле десятиліття і стали базовими для наступного покоління персональних комп'ютерів.

Комп'ютер мав графічний дисплей, звичайну та фортепіанну клавіатуру, синтезатор звуків (Кей був прекрасним музикантом і не міг залишити своє творіння «глухонімим»), керувався «мишею», яку запозичив у Енгелбарта. Проте, головним у новому комп'ютері було його абсолютно оригінальне програмне забезпечення. Кей відмовився від стандартних операційних систем та мов програмування (Basic, Fortran та ін.) і розробив принципово нову мову Smalltalk.

Для взаємодії з користувачем в системі Smalltalk вперше з'явилася ідея «**робочого столу**» з «**вікнами**», в яких відображалися тексти та малюнки. Вікна за допомогою миші могли переміщуватися по столу, відкриватися та закриватися. Таким чином, в проекті «Alto» вперше була запропонована ідея віконного, графічного, призначеного для користувача інтерфейсу – GUI. Цей комп'ютер, який був створений за два роки до «Altair-8800» та за 8 років до IBM PC, може вважатися першим дійсно персональним комп'ютером. Проте корпорація Xerox так і не пустила машину у масове виробництво, побоюючись малого попиту на неї. Було виготовлено всього 2000 машин, в основному для внутрішнього користування.

Алан Кей займався вдосконаленням комп'ютера «Alto» та системи Smalltalk аж до 1980 р., проте фірма Xerox так і не зуміла налагодити їх масове виробництво та отримати комерційну вигоду зі свого видатного досягнення. Цьому заважали об'єктивні труднощі, оскільки 8- та й 16-розрядна мікропроцесорна база 70-х років не дозволяла створити дешевий комп'ютер такого класу. Розроблений у 1981 р. на основі «Alto» комп'ютер «Star-8010» призначався для комп'ютеризації офісів і був дуже зручним для застосування, оскільки на його екрані моделювалася обстановка контори з документами, картковими скринями, сміттєвою корзиною і т.і., проте його ціна не опускалася нижче 16 000 доларів і продаж був невеликим. Поступово весь проект дружнього комп'ютера зазнав занепаду, і, можливо, став би тільки історичним курйозом, якби не його друге народження у фірмі Apple.

Фірма Apple Computer ще у грудні 1979 р., почувши про фінансові проблеми центру PARC, домовилася про те, щоб групі провідних інженерів Apple на чолі зі Стівом Джобсом (Steve Jobs, розробник комп'ютера Apple), продемонстрували всі секрети комп'ютерів «Alto» і «Star» в обмін на продаж фірмі Xerox 100 000 своїх акцій за один мільйон доларів.

Побачене справило на Джобса дуже глибоке враження. Він чітко усвідомив, який скарб залишається без попиту і зробив все, щоб дружні комп'ютери нового покоління вперше з'явилися під його фірмовою маркою. З цією метою він переманив у фірму Apple 15 з 25 провідних інженерів PARC і дав їм завдання створити на основі проекту «Star» власну ЕОМ під назвою «Lisa». Протягом 1982 р. на конструювання самої «Лізи» було витрачено 50 млн доларів, а на розробку її програмного забезпечення – 100 млн доларів.

У січні 1983 р. комп'ютер «Lisa» був пущений у продаж, проте, всупереч розрахункам фірми продати 50 000 примірників, за рік було реалізовано всього 15 000. Причина та сама, що і у «Xerox Star» – висока ціна. Невдача не бентежила керівництво компанії, адже свого часу «Apple» теж не вдався з першої спроби, до того ж нинішнє фінансове становище фірми дозволяло їй експериментувати – у 1983 р. доходи компанії впритул наблизилися до мільярда доларів.

У 1984 р. друга спроба Джобса, Apple **Macintosh**, мала величезний успіх, не тільки через дешевизну, але ще і тому, що на ньому працював дружній інтерфейс, тобто призначений для користувачів, які нічого не знають про комп'ютери і, більш того, зовсім не охочих будь-чому навчатися. Тільки за перших 9 місяців було продано 275 тис. машин. На 1989 рік операційна система Macintosh MACOS не мала собі рівних за простотою використання.

3.5. Прообраз сучасного «ноутбука»

Технологічна революція, на хвилі якої персональні комп'ютери зайняли лідируюче становище серед обчислювальної техніки другої половини двадцятого сторіччя, змусила розробників прагнути не тільки до збільшення потужностей ПК, але й до зменшення їхніх габаритів.

Американський підприємець **Адам Осборн** (1940 р.н.) закінчив два університети у Бірмінгемі та Делавері, у 1968 році став доктором наук, захистивши дисертацію по застосуванню хімічних технологій в електроніці. Приблизно у той самий час Осборн вирішив все таки пов'язати свою долю з комп'ютерами, влаштувавшись програмістом-консультантом у каліфорнійську компанію Shell Development. Проте, бажання мати власний бізнес неминуче підштовхувало Осборна до ідеї створення нової компанії, яка була б здатна приносити стабільний дохід. Першим проектом Адама стало засноване у 1970 році видавництво Osborne & Associates, проте після десяти років ця фірма, що так і не виправдала надій її власника, була продана.

Все глибше і глибше ознайомлюючись з електронними технологіями того часу, Осборн поставив на меті створити перший у світі переносний персональний комп'ютер, який міг би вміщуватися під сидінням пасажирського авіалайнера. На думку Осборна, цей винахід корінним чином змінив би американський діловий світ, оскільки за допомогою такого інструменту бізнесмени могли б не відриватися від роботи навіть під час подорожей та ділових поїздок, передаючи результати своєї роботи за призначенням або за

допомогою дисків, або за допомогою комутаційних пристроїв. Завдяки природному оптимізму та рідкісній заповзятливості Осборн вирішив не відкладати справу у довгу скриню і зробив те, чого не зробили гіганти комп'ютерної індустрії через побоювання втратити кошти через непередбачуваність ринку.

Вклавши у новий почин накопичені на видавничому бізнесі особисті заощадження у розмірі 250 тисяч доларів, Осборн заснував компанію Osborne Computer Corporation, яка вже на початку 1980 року взялася за розробку прообразу сучасного «**ноутбука».** Готовий виріб, що отримав назву Osborne I, з'явився на ринку рівно через рік, поступивши у роздрібний продаж за ціною 1795 доларів.

В основу архітектури Osborne I був закладений універсальний процесор Z80A виробництва компанії Zylog з тактовою частотою 3,5 Мгц. Комп'ютер був оснащений 64 Кбайт оперативної пам'яті, двома дисководами 5,25", монохромним дисплеєм з діагоналлю 5". У складі "персоналки" був паралельний та послідовний порти, порт для підключення модему, а також спеціальний розйом для з'єднання машини із зовнішнім монітором. Даний портативний комп'ютер важив трохи більше 10 кілограм і розмірами був схожий на невелику, але містку валізу. У якості програмної платформи Osborne I використовував подібну до MS-DOS операційну систему з командним рядком, адаптований для Osborne текстовий редактор WordStar та редактор таблиць.

Ледве з'явившись на ринку, Osborne I придбав справді нечувану популярність: протягом місяця компанія зуміла виконати річний план по реалізації своєї продукції, продавши більше 10 тисяч примірників Osborne I. Попит на переносні комп'ютери помітно перевищував пропозицію та виробничих потужностей Osborne Computer просто не вистачало для повноцінного покриття ринку.

Відчувши комерційний успіх свого проекту, Адам Осборн розпорядився розпочати проектування нової, ще досконалішої та могутнішої моделі портативного комп'ютера – Osborne Executive. Але у даному випадку оптимізм та енергійність Адама зіграли негативну роль: повідомивши пресу про безперечні переваги своєї нової розробки ще задовго до того, як вона була готова до продажу, Осборн, сам того не підозрюючи, знищив своє єдине джерело доходу – в очікуванні сучаснішого комп'ютера споживачі практично перестали купувати Osborne I.

Проте з Osborne Executive справи пішли зовсім не так гладко, як планував Осборн: нова машина, була розроблена поспіхом, мала безліч «слабких місць», які вимагали негайного виправлення, але як тільки чергова «ахіллесова п'ята» виявлялася ліквідованою, в системі комп'ютера негайно виявлялася нова. Зірвавши партнерам всі терміни поставок Osborne Executive, не виконавши умови контракту та виплативши значні неустойки, у 1982 році компанія оголосила себе банкрутом.

Таким чином, комерційна ініціатива на ринку портативних персональних комп'ютерів була втрачена і прапор, що впав, тут же підхопила корпорація

IBM, анонсувавши 29 лютого 1984 року новий вироб під маркою IBM Personal Portable Computer 5155. Зовні ця машина практично нічим не відрізнялася від своєї попередниці, хіба що в цій моделі переносного комп'ютера використовувався досконаліший дисплей з діагоналлю 9". IBM PPC 5155 був зібраний на базі процесора Intel 8088 з тактовою частотою 4,77 Мгц, мав два 5,25" дисководи ємністю 360 Кбайт, 256 Кбайт оперативної пам'яті, об'єм якої міг бути розширений до 640 Кбайт.

Після краху надій, що був викликаний банкрутством Osborne Computer, невгамовне бажання діяльності та любов до комп'ютерних технологій змусили Адама Осборна створити нову компанію з розробки програмного забезпечення – Paperback Software. Але й ця фірма благополучно звалилася після судового розгляду з корпорацією Lotus Development, предметом якого стала суперечка про авторські права на постачання програмних продуктів Paperback Software.

Прагнення зробити комп'ютер переносним виникло і в інших фірм. У 1982 році фірма Compaq випустила перший портативний персональний комп'ютер, який за свої розміри отримав назву – **laptop (**«наколінний»,**)**. Надалі розміри портативних комп'ютерів ще зменшилися, і вони стали порівнюватися за розмірами з великим зошитом – **notebook**. Наступний крок – спробувати покласти його в кишеню, що уже й зроблено.

У боротьбу за перспективний сектор ринку включилися декілька фірм. Apple Computers, яка досягла успіху на ринку персональних комп'ютерів, випустила у 1993 році модель Newton, що мала на ті часи цілком пристойні характеристики: 640 Кбайт оперативної та 3 Мбайт постійної пам'яті, 32-бітовий процесор з тактовою частотою 20 Мгц, рідкокристалічний екран з розподільчою здатністю 240 х 336 точок. Вельми оригінально була вирішена проблема введення даних. При такому розмірі комп'ютера на ньому важко розмістити клавіатуру, тому була реалізована технологія «електронного олівця», у якій текст, що вводиться, пишеться від руки спеціальним стрижнем на екрані, а потім відбувається розпізнавання символів.

Навколо проекту Newton фірма розгорнула широку рекламну кампанію і навіть у свій час заявляла, що ці мініатюрні пристрої, названі Personal Digital Assistant (PDA) – «персональний цифровий помічник» – найважливіший стратегічний напрям у виробників комп'ютерів. Проте, не дивлячись на всі зусилля, Newton не повторив долю легендарних Macintosh – всього було продано 80 000 примірників, після чого проект тихо помер.

Невдача Apple не бентежила решту виробників мініатюрних комп'ютерів – фірми Psion, Hewlett-Packard та ін., які під загальною назвою «**palmtop» (**на долоні) випустили на ринок декілька моделей кишенькових комп'ютерів. Вони цілком успішно продаються і вдало доповнюють звичайні настільні (desktop) персональні комп'ютери. Більш того, щоб користувач міг працювати у звичному середовищі, фірма Microsoft розробила полегшену версію операційної системи Windows з назвою Windows CE (Compact Edition), яка спеціально призначена для кишенькових комп'ютерів.

4. Болісний шлях розвитку програмування

Влітку 1985 р. у водах Північної Атлантики на глибині 3 км. просувався підводний апарат “Арго”, який досліджував гористий рельєф океанського дна. А нагорі, на поверхні океану, у рубці дослідницького судна ВМС США “Кнорр” перед відеоекраном розташувалися члени франко-американскої наукової експедиції. Після 16 днів пошуку був виявлений океанський лайнер “Титанік”, який затонув у 1912 р. і поніс в океанські глибини більше 1500 чоловік з тих 2200, що знаходились на борту.

Пошуки залишків “Титаніка” на величезній океанській глибині символізували видатні досягнення комп’ютерної технології та гнучкість сучасних засобів програмування. Апарат “Арго” керувався декількома комп’ютерами, кожен з яких був запрограмований на своїй мові. Комп’ютер підводного апарату використовував компактну, але гнучку мову **Форт** (FORTH, не плутати з Фортраном), яка була розроблена спочатку для керування рухом телескопів. Комп’ютер на борту судна ”Кнорр” був запрограмований на могутній, але малозрозумілій мові **C**. Телеметрична система на обох кінцях коаксіального кабелю, що зв’язував підводний та надводний човни, використовувала ще одну мову – мову асемблера.

Кожна мова має свою граматику та синтаксис. Мови програмування, що імітують природні мови та здатні на підставі одного речення будувати кілька команд комп’ютера, прийнято вважати **мовами “високого рівня**”. Але у порівнянні з природними мовами, вони, як правило, більш строгі та точніші.

Зараз налічується декілька сотень таких мов, якщо ж ураховувати всі їх варіанти, так звані **діалекти**, то можливо, і більше тисячі. Мови програмування призначені для найрізноманітніших цілей – від розв’язку складних математичних завдань до створення музичної партитури, машинної графіки та ігор. Проте, не існує жодної машинної мови, яка б у рівній мірі була придатна для всіх випадків. Вибір мови зазвичай визначається трьома чинниками: мова повинна бути зручною для програміста, придатною для даного комп’ютера і для вирішення поставленого завдання.

4.1. Плануюче обчислення

Навесні 1945 р. світ навколо Конрада Цузе почав руйнуватись: кільце військ союзників стискалося навколо Берліна, де жив молодий німецький інженер. Ще перед війною він зайнявся розробкою серії порівняно невеликих універсальних комп’ютерів, перетворивши на лабораторію одну з кімнат своїх батьків. Зусилля Цузе – це дивовижний приклад незалежної наукової творчості одинака: він не мав ані найменшого уявлення про досягнення у цій області в інших країнах, а уряд фашистської Німеччини мало цікавився його роботою над комп’ютером. Незадовго до падіння Берліна Цузе поклав свій єдиний комп’ютер Z4, що залишився, на воза та відправився у маленьке містечко у Баварських Альпах. Щоб не потрапити у полон в останні дні війни, він

приєднався до групи вчених, що розробляли ракети у гітлерівській Німеччині, які намагалися сховатися в Альпах Баварії. Але американцям все ж таки вдалося захопити цю групу у полон. Одразу ж вони запропонували роботу одному з її членів, конструктору ракет Вернеру фон Брауну (творцеві ракет Фау-2), що надалі добре попрацював на "американський космос". На Цузе, що встиг приховати свою машину у підвалі селянського будинку, американці не звернули особливої уваги.

У важкі післявоєнні роки, не маючи ані засобів, ані можливості продовжувати роботу над комп'ютером, він спрямував усю свою енергію на розвиток теорії. Ще з 1942 року Цузе виношував ідею алгоритмічної мови програмування. Він вигадав ефектний спосіб програмування комп'ютерів, причому придатний не тільки для Z4, але і для будь-якої машини. Він вирішив, що потрібна система числових та символьних позначок, яка побудована на основі логіки, – по суті, система обрахунку кроків вирішення задачі.

Працюючи наодинці він створив систему програмування, яку назвав "**Планкалкюль"** (Plancalkul, плануюче обчислення). Він написав брошуру, де розповів про своє творіння та можливість його використання для вирішення різноманітних завдань, включаючи сортування чисел та виконання арифметичних дій у двійковому записі (інші комп'ютери того часу працювали у десятковій системі). Навчившись грати у шахи, Цузе написав 49 сторінок фрагментів програм на Планкалкюле, які дозволяли комп'ютеру оцінювати шахові позиції. На жаль, багато ідей систематичної мови, як і решта його робіт, залишилися невідомі цілому поколінню фахівців з комп'ютерної лінгвістики. Тільки у 1972 р. робота Цузе була опублікована. Ця публікація змусила фахівців задуматися над тим, який вплив міг би зробити Планкалкюль, будь він широко відомий раніше.

В Планкалкюль вводилось поняття об'єкту, він дозволяв працювати з підмасивами даних та підпрограмами. Цузе винайшов оператор присвоєння та визначив для нього окремий знак.

4.2. Внесок Великої Британії

Програмісти вже були знайомі з поняттям підпрограми: Грейс Хоппер та її колеги застосовували підпрограми на гарвардському "Марку-1", проте кожна підпрограма вирішувала свою специфічну задачу. Перші програмісти майже завжди мали блокноти із записом найспоживаніших підпрограм, щоб у разі потреби не складати їх наново. Проблема полягала у тому, що адреси розташування команд та змінних мінялися залежно від їх розміщення у пам'яті.

Налаштування підпрограми на певне місце у пам'яті, вочевидь, потребувало автоматизації, і вперше це було зроблено на новому комп'ютері EDSAC (1949 р), який був створений Морісом Уїлксом у Кембриджі та заснований на ідеї комп'ютера EDVAC.

Моріс Уїлкс назвав мнемонічну схему для EDSAC та бібліотеки підпрограм **збираючою системою (assembly system –** звідси слово

“**асемблер”**), оскільки вона збирала послідовності підпрограм. Ідея виявилася настільки продуктивною, що всі подальші покоління програмістів на всіх ЕОМ відмовилися від абсолютного кодування, тобто програмування в абсолютних адресах. Мови програмування низького рівня, в яких коди операцій були замінені мнемонічними позначками, почали називатися мовами Асемблера або Автокодами (мнемокодами), а перетворюючі програми – асемблерами.

Мови Асемблера використовуються й дотепер, оскільки тісно пов’язані з машиною. Дійсно, мова асемблера – це машинно-залежна мова, яка спроектована так, щоб вона відповідала набору машинних команд конкретного комп’ютера. Тому мову асемблера люблять ті програмісти, які прагнуть стиснути свої програми до мінімального розміру, що дозволяє виконати їх максимально швидко та ефективно. Саме з цих причин асемблер ідеально підійшов для програмування телеметричної системи, яка була використана в експедиції з пошуку “Титаніка”.

Мова Асемблера – це перший крок до позбавлення програміста від дрібниць двійкового кодування, завдяки використанню символів (слів), які легко зрозуміти та запам’ятати.

Влітку 1952 р. у Манчестері **Алік Гленн** розробив першу мову високого рівня під назвою “Автокод”, і став одним з перших творців комп’ютерних мов. “Автокод” дозволяв програмістові користуватися у програмах математичними формулами. Після введення програм у “Марк-1” “Автокод” транслював їх у програми на машинному коді. На відміну від інтерпретаторів, програми у машинному коді, що були згенеровані “Автокодом”, можна було зберігати та виконувати у будь-який час.

4.3. Програмування англійською мовою

Грейс Хоппер вирішила піти далі. Спираючись на власний досвід роботи, вона зі своєю групою зайнялася розробкою системи, яка могла б транслювати програми, що були написані на мові високого рівня, у програми на машинному коді. Транслююча програма повинна за мілісекунди виконувати велику роботу, що пов’язана з програмуванням: організація підпрограм, виділення пам’яті комп’ютера, перетворення операторів мови високого рівня у низку машинних команд. Хопер назвала цю транслюючу програму компілятором (**compiler – компонувальник).**

Таким чином, транслюючі програми почалися ділитися на дві категорії: **інтерпретатори** та **компілятори**. Інтерпретатор перетворить невеликий фрагмент початкової програми у внутрішні команди і, лише дочекавшись, коли комп’ютер їх виконає, переходитиме до обробки наступного фрагменту. Компілятор навпаки, транслює всю програму, яка написана на мові високого рівня, а спеціальна програма-завантажувач операційної системи розміщує коди команд у пам’яті комп’ютера та передає управління на їх виконання. Після компіляції програму можна зберігати, щоб у подальшому використовувати. Кожен з цих способів має свої переваги та недоліки.

Хоча “Автокод” Гленна мав аналогічні властивості, компілятор Хопер став першою транслюючою системою, що здобула широку популярність. Таким чином, на той час до вжитку увійшли два типи мов високого рівня: такі, що компілюються та такі, що інтерпретуються. Відмінності між ними залишаються важливими і дотепер. Хопер назвала свій перший компілятор А-0, маючи на увазі, що це лише початок. Впродовж років вийшли кращі версії А-1, А-2, А-3. Треба зазначити, що сучасні системи програмування включають обидві компоненти – інтерпретуючу та компілюючу. Перша призначена для відлагодження програм і має можливість перегляду щодо коректного виконання програм, а друга – для остаточного формування коду програм для багаторазового використання. З іншого боку, мови програмування для Internet-програм суто інтерпретуючі, що робить їх мобільними для різних програмних платформ. Мобільність, тобто переносимість на інші програмні платформи, забезпечується відповідними інтерпретаторами, які однаково інтерпретують програми при їх виконанні.

Тим часом, Грейс Хоппер поставила перед собою нову мету: створити мову, за допомогою якої можна було б розв’язувати будь-які завдання, а не тільки математичні.

Виходячи з цього, Хопер та її група взялися за розробку компілятора, який дозволяв би особам, які зайняті у бізнесі, програмувати на мові, яка за можливістю наближається до звичайної англійської мови. Через декілька місяців роботи був виділений список приблизно з 30 слів, які стали “робочими конячками” мови.

У 1956 р. компілятор, що отримав назву “Флоу-метік” (FLOW-MATIC), надійшов до ринку. Таким чином, в середині 50-х років завдяки стрімкому розширенню використання компіляторів розвиток програмування отримав міцний фундамент. Золоте століття мов програмування було не за горами.

5. Три комерційні гіганти

У квітні 1957 р. в обчислювальному центрі атомної лабораторії “Вестінгауз-беттіс” (Пітсбург), що була оснащена новенькими машинами IBM-704, листоноша доставила загадкову посилку. Програміст **Херб Брайт** та двоє його колег відкрили скриню і виявили там стопку приблизно з 2 тис. перфокарт, але без будь-яких інструкцій.

Розглядаючи карти, Брайт пригадав, що у фірмі IBM вже впродовж декількох місяців йде відлагодженняка мови високого рівня, яка призначена для використання на IBM-704. Ця мова, названа **Фортраном,** була автоматичною кодуючою системою, що бере на себе найважчу частину роботи програміста. У передчутті, що це довгоочікуваний компілятор, Брайт та його друзі вирішили завантажити загадкові карти у комп’ютер.

Після завантаження компілятора зі стрічки магнітної пам’яті завертілись та зупинилися. Брайт вклав у зчитуючий пристрій тестову програму, яка була написана на Фортрані, і натиснув кнопку пуску. Затріщав принтер. Новий компілятор видав повідомлення: на карті № 25 виявлена помилка – в операторі пропущена кома. Програмісти, звиклі до повідомлень у вигляді числових кодів, що важко розуміються, були вражені ясністю цієї інформації. Після виправлення неправильного оператора комп’ютер видав стопку перфокарт програми. Після завантаження програми комп’ютер видав на друк 28 сторінок правильних розрахункових даних.

Так Брайт та його колеги опинилися у числі перших користувачів Фортрану – першої широко поширеної мови програмування високого рівня.

5.1. Перша комерційна мова програмування

Пройшов час, з’явилося безліч мов високого рівня, і вже наступне покоління програмістів почало вважати Фортран ”копалиною” серед машинних мов, називаючи його “динозавром”. Едсгер Дейкстра, авторитет в області програмного забезпечення, відмітив, що навчання студентів Фортрану повинно каратися як злочин. Проте і через десятиліття Фортран не втратив своєї популярності (наприклад, Фортран-77, 90). Ця мова, що надає великі можливості по обробці числової інформації, отримала широке визнання серед науковців та інженерів. Основна причина такого довголіття проста – **Фортран був першим**.

Дослідники фірми IBM, що створили Фортран, і не підозрювали, що ця мова набуде такого великого значення. Роками пізніше керівник групи розробників **Джон Бекус** відзначав, що завдяки Фортрану виникла нова порода людей – програмістів та фахівців з інформатики.

До коледжу Бекус був посереднім учнем (“Я пройшов більше шкіл, ніж можу пригадати”). Після служби в армії під час другої світової війни він поступив у школу радіотехніки (Нью-Йорк). Проте викладач з ремонту

телеапаратури збудив у Бекуса інтерес до математики та переконав його продовжити навчання у Колумбійському університеті. Бекусу було 25 років, коли він, отримавши ступінь магістра математики, поступив у 1959 р. програмістом до фірми IBM. Бекус, який неодноразово заявляв, що "завжди був ледарем", звернувся до свого шефа з новаторською пропозицією. Це була складна мова і компілятор до неї, що спрощували програмування моделі IBM-704. Система дозволяла записувати програми у формі алгебри, а компілятор повинен був автоматично переводити її у машинні коди.

Створити ефективний та надійний компілятор для мови Фортран виявилося важчим завданням, ніж передбачалося. Групі довелося мати справу з введенням, виведенням, циклами та безліччю інших заплутаних речей, які необхідні для перетворення команд на мові високого рівня у машинний код.

Бекус передбачав, що успіх Фортрану підштовхне інших виробників до розробки аналогічних мов для своїх машин, що дійсно і відбулося. Проте Фортран швидко став нормою, і його адаптували для багатьох машин інших марок. Завдяки зусиллям розробників усього через рік з'явився Фортран II, однією з переваг якого була можливість під'єднувати частини програми на асемблері. Пізніші версії (Фортран III – 1958 р., Фортран IV – 1962 р.) ще більше розширили можливості Фортрану.

Сильною стороною мови завжди був високий ступінь переносимості початкового коду між різними платформами, що забезпечувало довговічність програм. В основі такої сумісності лежить жорсткий стандарт. Фортран випереджав конкурентів і за швидкодією виконання програм, і за їх компактністю завдяки високій ефективності виконуваного коду. Це пояснюється як багаторічним відпрацюванням алгоритмів компілятора, так і застосуванням простіших конструкцій мови. Гордістю Фортрану завжди була багата колекція різноманітних бібліотек, перш за все математичних. Одна з найпопулярніших бібліотек (IMSL) включає понад тисячу процедур математичної та статистичної обробки даних і фактично є стандартом для найрізноманітніших комп'ютерних платформ.

Фортран і дотепер (Фортран-90) залишається популярним засобом програмування, особливо серед науковців. Проте головна перевага Фортрану полягає у тому, що він був першою широко поширеною комерційною мовою – ефективним засобом написання програм, простою у навчанні та такою, що не вимагала знання машини. З тих пір науковці, інженери та студенти отримали можливість спілкуватися з комп'ютером без допомоги професійних програмістів, що створювали програми на асемблері.

5.2. Обчислювальна техніка приходить у бізнес

Поки Фортран набував прихильників у традиційній області застосування комп'ютерів – науці та техніці, – у світі бізнесу все сильніше відчувалася потреба автоматизації процесів обробки інформації. Було розроблено декілька мов, але жодна з них не підходила для широкого застосування у сфері бізнесу.

Як і передбачав Бекус, виникли безліч імітацій Фортрану, проте більшість з них були прийнятні лише для того комп'ютера, для якого призначалися спочатку. Проблема перенесення програмного забезпечення з комп'ютера на комп'ютер ставала все гострішою. Усі спроби якось модифікувати Фортран під завдання економічних, комерційних та бухгалтерських розрахунків були не дуже вдалими, до того ж він був власністю IBM, а інші фірми побоювалися засилля «блакитного гіганта».

У квітні 1959 р. зацікавлені споживачі, виробники комп'ютерів та академічні кола провели нараду, яка зібралася у Пенсільванському університеті. Група прийшла до висновку, що вона (і решта світу) потребує стандартизовану машинно-незалежну мову для ділових застосувань.

Було зрозуміло, що Фортран – не те, що потрібно. У бізнесі була потрібна швидка робота з накладними, створення різних списків споживачів, а не розв'язання рівнянь. Потрібна була мова, яка припускала б не тільки такі важливі операції, як злиття файлів, сортування елементів файлу та генерації звітів, але і ґрунтувалися на реченнях, подібно до фраз англійської мови.

Мало сподіваючись у такому питанні на компанії-виробники, група почала пошуки нейтрального спонсора проекту. Невелика делегація на чолі з Грейс Хоппер звернулася у Міністерство оборони США з проханням підтримки проекту.

5.3. Народження CODASYL

Відгукнувшись з незвичайною жвавістю на пропозицію делегації, Міністерство оборони організувало спеціальну конференцію з мов програмування у травні 1959 р. Пізніше ці збори отримали назву **CODASYL (Conference on Data Systems Languages – конференція з мов систем обробки даних)**. Всі найважливіші постачальники комп'ютерів направили на конференцію своїх представників. Їм було повідомлено, що Пентагон бажає мати єдину мову програмування. Було вирішено утворити три групи, або комітети: короткотривалий, проміжний та довготривалий. Перший повинен був розробити основу проекту, другий, спираючись на ці розробки, – створити діючу систему, третій – за кілька років розробити те, що повинне стати останнім словом у мовах програмування.

До середини осені 1959 р. мова знайшла певний вигляд та ім'я **Кобол** (**COBOL**, Common Business Oriented Language – універсальна мова, що призначена для бізнесу). На відміну від обчислювальних завдань, в яких головне – це алгоритм, а прості неструктуровані дані просто вводяться та виводяться, в економічних завданнях дані мають пріоритет. Вони можуть мати дуже складну структуру та зберігатися у вигляді файлів на зовнішніх носіях самі по собі, незалежно від програм обробки. Тому у Коболі з'явився спосіб опису ієрархічних структур даних, який потім перекочував в усі подальші універсальні мови програмування.

Концепція систем керування базами даних (СКБД), що з'явилася на початку 70-х років, не могла не вплинути на мови, які призначені для економічних завдань. Наприклад, група CODASYL продовжила свою роботу та у 1968 році опублікувала розширення Коболу для роботи з базами даних. Фірми-виробники СКБД також пропонували різні мови маніпулювання даними. Ці мови погано поєднувалися одна з одною, тому що самі системи ґрунтувалися на різних моделях даних. Проте, коли у кінці 70-х років почав переважати реляційний підхід, який був запропонований співробітником IBM **Едгаром Коддом** (1923 р.н., на даний час він є класичним), з'явилася можливість створити уніфіковану мову реляційних баз даних, яка заснована на теорії відношень та математичній логіці. Такою мовою стала мова запитів SEQUEL, яка була розроблена у 1974 році Дональдом Чемберленом та Реєм Бойсом у дослідницькому центрі IBM Сан-Хосе. Згодом вона була доопрацьована та перейменована у SQL (Structured Query Language – структурована мова запитів). Зараз SQL є загальновизнаним стандартом мови баз даних, вона реалізована в усіх промислових СКБД різних виробників.

Не дивлячись на теоретичну опрацьованість, реляційні системи далеко не одразу набули широкого розповсюдження, довгий час вважалося за неможливе добитися ефективної реалізації таких систем. Перше масштабне втілення реляційної моделі вдалася все тій же IBM в експериментальній СКБД System R, що розроблялася у 1975-1979 роках. Це був гігантський успіх, реляційні СКБД стали реальністю. Після успішного завершення робіт із створення прототипу системи та отримання експериментальних результатів при її використанні, фірмою IBM був запущений повномасштабний проект промислової реляційної СКБД. У 1983 році система, яка отримала назву DB2, поступила у продаж, з тих пір протягом багатьох років вона успішно функціонує на всіх фірмових апаратних платформах: S/390, AS/400, POWERPC, є реалізація і для платформи WIntel.

Проект System R, який був опублікований Чемберленом із співавторами у листопаді 1976 року в журналі «IBM Journal of Research and Development», надихнув одразу кілька груп розробників. Одна з них, яку очолював Ларрі Еллісон (Ellison, Larry), вирішила створити власну фірму для розробки реляційної СКБД, характерними рисами якої були б переносимість на різні платформи та сумісність з мовою SQL. Фірму назвали Relation Software Incorporated, а систему – Oracle. Версія 2 (перша у продаж не поставлялася) була створена у 1979 році для ЕОМ PDP-10. Згодом система постійно вдосконалювалася та адаптувалася для різних платформ, і у 90-і роки зайняла лідируюче становище на ринку реляційних СКБД. Фірма була перейменована в Oracle і до кінця століття перетворилася на одну з найбільших софтверних компаній світу, а її президент Ларрі Еллісон зайняв почесне 30-е місце у списку найбагатших людей планети (9,5 млрд доларів).

До середини 80-х років вартість програм, що були написані на Коболі, оцінювалася у 50 млрд. доларів. Такі вкладення гарантують довге життя мові, оскільки на Коболі, який періодично оновлюється, та як і раніше є ефективним

у тій області, для якої він створювався (обробка ділової інформації), і зараз створюється безліч програм.

У Радянський Союз Кобол потрапив на початку 70-х років, в епоху ЄС ЕОМ та загальної захопленості АСУ, проте великої популярності він не набув, англомовні тексти дратували вітчизняних програмістів. Русифікований Кобол, розроблений для ЕОМ Мінськ-32, також не користувався успіхом, оскільки тексти через меншу лаконічність російської мови виходили дуже громіздкими.

Ще до появи Коболу намітилася спроба розробити третю універсальну мову. **Алгол (ALGOrithmic Language – алгоритмічна мова)** подібно до Фортрану, який був мовою алгебри і був розроблений переважно для вирішення чисельних завдань. Як і Кобол, він народився у кімнаті засідань комітету – дитя конфліктів та компромісів. Але на цьому його схожість закінчується. Після багаторічних суперечок комітет з Алголу дав світу мову, яка швидше викликала інтерес, ніж привернула споживачів. Це, мабуть, перша мова, яку можна було б без перебільшення назвати "витонченою", – термін, який визначила Грейс Хоппер, як просту та зрозумілу з погляду математики, але не зовсім практичну.

5.4. Конференція в Цюріху

Громадськість програмістів Європи турбував диктат IBM та інших агресивних американських фірм на європейському ринку комп'ютерів. Заклопотаність щодо створення єдиної мови для наукового програмування прийняла міжнародний характер. Перше обговорення відбулося у Федеральному технічному університеті в Цюріху (Швейцарія) 27 травня 1959 р. Через тиждень зустріч перервалася, мова, що отримала назву **Алгол-58**, вже виникла. У Цюріху зібралися вісім найповажніших авторитетів: четверо з Європи та четверо зі США. На зустрічі був присутній творець Фортрану Джон Бекус.

Мова, яка була створена на конференції в Цюріху, багато у чому успадкувала від Фортрану, але в Алголі основні поняття Фортрану були зібрані у логічну витончену структуру. Проте через нестачу часу та компромісів були припущені помилки. Одна з них полягала у відсутності процедур введення та виведення даних. Після закінчення конференції частина делегатів виїхала з думкою, що мова ще далека від досконалості, інша частина – з надією, що Алгол знаходиться на шляху, щоб на довгі роки стати міжнародним стандартом.

Одним з таких оптимістів був Джон Бекус. Його доповідь на фірмі IBM була прийнята з ентузіазмом, і фірма пообіцяла розробити транслятор. Оскільки фірма IBM, світовий лідер з продажу комп'ютерів, всюди впроваджувала Фортран, було наївно сподіватися, що вона стане на бік конкуруючої мови. По мірі того як капіталовкладення у Фортран зростали, ентузіазм відносно Алголу згасав. Але це не зупинило Бекуса. Повернувшись до Цюріха, він зацікавився **синтаксисом мови**, тобто правильним розташуванням слів та знаків у мові. Бекус розробив строгу та точну систему визначення кожної структури мови

логічним чином. Наприклад, визначаючи елемент “цифра”, він писав: <цифра>:= 0|1|2|...|9. Незабаром програмісти почали називати цей підхід щодо опису синтаксису **Нормальною Формою Бекуса**, або **БНФ**. Подальше уточнення цієї роботи данським астрономом **Пітером Науром** призвело до того, що такий підхід почали називати “**форма Бекуса-Наура**”.

Оскільки почалися проблеми з реалізацією мови, автори Алголу прийшли до висновку, що їх мова зразка 1958 р. – всього лише ескіз. У січні 1960 р. 13 представників країн Європи та США, включаючи 7 осіб з минулої цюріхської групи, зустрілися у Парижі, щоб “залатати дірки”, що залишилися у першому варіанті Алголу. Одним з новоприбулих був Наур, який запропонував варіант зміненого Алголу, що був описаний за допомогою БНФ. Група ухвалила рішення щодо мови одноголосно і покинула Париж у піднесеному настрої. Але ледве американські учасники конференції зійшли з трапа літака, на **Алгол-60** обрушився вогонь критики. Виявилось, що у мову знову так і не включені засоби введення-виведення.

5.5. На шляху до сумісності комп’ютерів

7 квітня 1964 р. представники IBM, гіганта комп’ютерної індустрії, провели 77 прес-конференцій у 15 країнах світу, зробивши, як висловився глава фірми Томас Уотсон-молодший, “найважливіше оголошення за всю історію компанії”. Подальший розвиток подій підтвердив такий прогноз.

Компанія, яка відома зараз під ім'ям IBM, була заснована 15 червня 1911 і називалася C-T-R (Computing-Tabulating-Recording). У травні 1914 генеральним керуючим компанії C-T-R став Томас Уотсон (Thomas Watson, 17.02.1874 - 19.06.1956).

Томас Уотсон-молодший, змінив батька на посту глави IBM, ознаменувавши початок ери комп'ютерів для життя. Під його керівництвом доходи компанії зросли до 8 млрд дол, а число співробітників – до 270 тис. осіб.

Поширене прізвисько компанії – Big Blue, що можна перекласти з англійської як «блакитний гігант». Існує кілька версій щодо цього прізвиська. За однією з них назва походить від мейнфреймів, що поставляються компанією в 1950-1960-х рр. Ці мейнфрейми були розміром з кімнату і мали блакитне забарвлення. За іншою теорією прізвисько просто посилається на логотип компанії (у 1972 р. був представлений оновлений логотип – літери з синіх смужок компанії, який використовується до теперішнього часу).

Дійсно, того дня IBM оголосила про створення не однієї якої-небудь машини, а цілої їх родини. **S/360 (“Система-360”)** – так була названа серія – дебютувала одразу шістьма моделями, що розрізнялися за потужністю та вартістю. За оцінками фахівців, на дослідження, розробку та впровадження у виробництво одночасно шести машин компанія витратила близько 5 млрд. доларів, що вдвічі перевищувало витрати США під час другої світової війни на Манхеттенський проект, метою якого було створення атомної бомби.

Це рішення зустріло наполегливий опір частині співробітників фірми, і було скептично сприйнято її конкурентами. По суті IBM поставила на карту не більше, не менше як саме своє існування, свій капітал і честь, зробивши ставку на проект, здійснення якого зробило б решту всіх комп'ютерів фірми безнадійно застарілими. Розробка "Системи-360" означала рішучий відхід від технічних стандартів, що склалися за багато років.

У кінці 1961 року Уотсон та Лірсон, віце-президент фірми, призвали на допомогу комітет, що складався з представників усіх основних відділів фірми: системного програмування, досліджень, конструювання, впровадження тощо. Уотсон та Лірсон "замкнули" 13 членів комітету у мотелі недалеко від Гринвіча, штат Коннектикут, заявивши, що не випустять їх, поки не вдасться у загальних рисах намітити, яким повинно бути наступне покоління машин фірми IBM. Під головуванням Боба Еванса, керівника з питань планування та розробок у відділі систем даних, комітет безперервно працював протягом 60 діб.

Пристрасті особливо розгорілися при обговоренні двох принципово нових підходів щодо розробки майбутніх комп'ютерів. Згідно першому з них, будь-яка нова машина повинна бути універсальною, тобто справлятися з широким спектром проблем – від розв'язання логічних і обчислювальних завдань наукового характеру до обробки даних у сфері управління та бізнесу. Назва серії "360" вказувала саме на здатність машин працювати на всіх напрямках – в межах 360°. До початку 60-х років комп'ютери конструювалися із розрахунку або на наукові, або на управлінські застосування. Проте, до цього часу комп'ютери, що були призначені для наукових досліджень, все частіше почали застосовуватися у сфері бізнесу та навпаки. Ця тенденція не залишилася непоміченою в IBM.

Другий підхід припускав, що нові комп'ютери повинні бути сумісні один з одним. У ті роки несумісність машин різних моделей була характерна для комп'ютерної індустрії – кожна модель мала свій власний центральний процесор та систему введення-вивення. Програми, які були написані відповідно до внутрішніх інструкцій процесора однієї машини, були непридатні для інших центральних процесорів. Периферійні пристрої, що були призначені для якогось одного типу, не можна було підключати до інших машин.

Одним із найпереконливіших доводів на користь створення серії сумісних комп'ютерів зводився до того, що таким чином фірма IBM полегшить важкий тягар витрат, які пов'язані з розробкою та обслуговуванням великої кількості несумісних одна з одною моделей комп'ютерів. У ситуації, що склалася, компанія витрачала велику частину капіталовкладень на розробку різних моделей центральних процесорів у збиток програмному забезпеченню та периферійним пристроям.

На початку січня 1962 р. комітет представив свої висновки адміністрації фірми. Так народилася "Система-360".

Інші компанії, виробники обчислювальної техніки, швидко зрозуміли, що в модульній комп'ютерній системі, компоненти якої відповідають загальним

стандартам, не обов'язково всі модулі повинні виготовлятися однією і тією ж фірмою. Отримуючи великі прибутки, вони кинулися робити пристрої, які були придатні для "стикування" з комп'ютерами серії "Системи-360", тобто повністю сумісні з машинами фірми IBM.

Створивши "Систему-360", IBM ввела обчислювальну техніку у нову еру – еру розквіту технології пристроїв введення-виведення. У 60-70-х роках виробництво іншими фірмами пристроїв, які були сумісними з машинами IBM, настільки зросло, що IBM поступилася цим конкурентам близько 13% виробництва периферійних пристроїв до своїх же комп'ютерів. Та все ж успіх "Системи-360" був настількі великий, що доходи фірми IBM продовжували зростати швидкими темпами, а її лідируюче становище у комп'ютерній промисловості стало ще міцнішим.

5.6. Розбіжності Нового Світу

В Америці Алгол-60 зустріли прохолодно. Могутня організація SHARE споживачів комп'ютерів фірми IBM перекинулася на сторону Фортрану. Виявилося, що Алгол важко реалізувати.

Європейці одразу прийняли Алгол. Подібно до того, як міністерство оборони США надало рішучу підтримку Коболу, незалежна дослідницька рада ФРН та інші урядові організації західноєвропейських країн підтримали Алгол. Завдяки ним Алгол розповсюдився від Великої Британії до Радянського Союзу. Деякі із спроб реалізації Алголу закінчилися невдачею, але багато компіляторів виявилися вдалими, і незабаром Алгол був визнаний мовою європейських програмістів.

У Радянському Союзі одразу ж після перших публікацій з Алголу-60 почалися роботи зі створення компіляторів для найпоширенішої та наймогутнішої на той час ЕОМ М-20. Оскільки ці роботи мали дуже велике значення для становлення вітчизняних шкіл системного програмування, про них слід розповісти докладніше.

На початку 60-х років в країні були три колективи програмістів, які були здатні вирішити непросту задачу написання компілятора з мови високого рівня. Перший розташовувався на батьківщині машини М-20 – в ІТМ і ОТ Академії Наук і очолювався Святославом Сергійовичем Лавровим. Враховуючи обмежені можливості пам'яті ЕОМ (всього 4096 слів) автори компілятора вирішили спростити завдання, відмовившись від рекурсії та деяких інших, важких для реалізації, властивостей Алголу. Одначе їх транслятор ТА-1, випущений вже у 1962 році, вийшов достатньо ефективним та швидкодіючим, він достатньо широко використовувався у багатьох організаціях європейської частини Союзу.

Другий колектив працював в Інституті прикладної математики АН СРСР, туди був розподілений майже весь перший випуск учнів А.А.Ляпунова, що закінчили у 1954 році механіко-математичний факультет МГУ за фахом «Програмування». Під керівництвом М.Р. Шура-Бури та Едуарда Зіновьєвіча

Любімського вони зважилися на реалізацію повної версії Алголу. Весною 1963 року компілятор був готовий, він справлявся з найскладнішими прикладами рекурсивних алгоритмів, але платою за це були гірша швидкодія компілятора та невисока ефективність коду.

Третій колектив сформувався у новосибірському Академмістечку, куди у 1960 році вслід за А.А.Ляпуновим переїхав 28-річний А.П. Ершов, який організував відділ програмування в Обчислювальному центрі Сибірського відділення АН СРСР. Молодіжна команда Єршова зважилася на розробку першого у світі оптимізуючого компілятора із запропонованої ними мови Альфа, яка являла собою Алгол без рекурсій, але проте була розширена комплексними числами та матричними операціями. В результаті закінчений до 1965 року Альфа-транслятор вийшов вельми зручним та ефективним, він широко використовувався при написанні різноманітних програм на М-20, особливо в сибірських ВНЗ та НДІ.

У США справи йшли інакше. Зрозуміло, що Алгол, врешті-решт, пробився до багатьох американських комп'ютерів, але так і не здолав Фортран. Проте теоретичний вплив Алголу був у США так само сильним, як і в Європі.

Багато мов, що виникли пізніше, базувалися (принаймні, частково) на Алголі, і його внесок в їх розвиток був пов'язаний з декількома основними ідеями. Одна з найбільш важливих – **блокова структура**, що дозволяє ділити програми на замкнені одиниці, що можуть розроблятися незалежно. Друга ідея – **рекурсія**, тобто здатність програм повторно звертатися до самої себе. Третя ідея – **БНФ**, формалізоване визначення синтаксису. Гігант серед виробників комп'ютерів, фірма IBM, приступивши у кінці 1963 р. (одночасно з новою лінією комп'ютерів "Система-360") до здійснення нового проекту універсальної мови, вельми уважно поставилися до цієї спадщини. Машини лінії "Система-360" проектувалися так, щоб задовольняти потреби як в аналітичних обчислення, так і в обробці даних у сфері бізнесу та спеціальних застосувань.

Природно, що комп'ютер з таким широким спектром можливостей вимагав могутньої універсальної мови високого рівня. Асоціація SHARE, оцінивши перспективність цієї справи, допомогла створити "комітет з розробки передової мови". До комітету увійшли представники найбільших фірм, а також кращі фахівці з відділів програмування та розробки мов самої IBM.

Не дивлячись на адміністративний тиск (мова повинна була бути розроблена до терміну закінчення робіт по "Системі-360", квітень 1964 р.), специфікації мови були завершені лише до лютого 1964 р. – нова мова явно запізнювалася до дебюту "Системи-360". Про результати зусиль, нову мову програмування **PL/1 (Programming Language One – мова програмування, перша)**, комітет доповів асоціації SHARE у березні. Деякі фахівці характеризували мову PL/1 як дуже складну.

Сподіваючись, що більшість проблем, що були пов'язані з мовою, проясняться у процесі реалізації, фірма IBM дала завдання своїй лабораторії в Харслі (Англія) приступити до роботи над компілятором. Співробітники лабораторії так сильно відредагували мову (точне визначення мови займало

близько 200 сторінок), що розробники навряд чи дізналися б, що вийшло в результаті. Проте уточнення перетворили мову на практичний інструмент програмування, який вітали багато програмістів.

Багато того, що було краще у Фортрані, Коболі та Алголі знайшло відображалення у PL/1. Але, як висловився Джон Бекус, перші комерційні мови зробили негативний вплив на розвиток PL/1 та подальших мов програмування. “Оскільки перші мови дуже глибоко проникли в середовище програмістів, – казав Бекус, – вони до цих пір залишаються безвихіддю на шляху до більш досконалих мов”.

Не дивлячись на теоретичну недосконаліть, мова PL/1 зіграла виключно важливу роль у практиці програмування для машин класу IBM 360/370. В СРСР в епоху АСУ (70-80-і роки) вона була основною мовою програмування економічних та науково-технічних завдань на ЄС ЕОМ, а також стандартною мовою для навчання програмістів. До того ж наявні у PL/1 механізми низькорівневого програмування (покажчики, засоби зв’язку з обчислювальним середовищем) дозволяли писати вельми ефективні програми. Проте після появи персональних комп’ютерів популярність мови різко впала. Перші покоління мікрокомп’ютерів були дуже слабкими для реалізації компілятора з PL/1, а коли комп’ютери стали достатньо потужніми, було вже пізно – нішу PL/1 зайняли Паскаль та С.

6. Десятиліття динамічного розвитку

В середині 60-х років у програмуванні панували три мови: Fortran та Cobol володіли американською наукою та бізнесом, Algol-60 був поширений в Європі. Нова мова PL/1 претендувала на те, щоб з'єднавши у собі кращі риси цих трьох мов, стати універсальним мовним засобом. Але у коледжах та дослідницьких лабораторіях фірм зароджувалися нові мови, що були призначені для програмування сучасних комп'ютерів. До кінця 60-х років мови почали розмножуватися в такому темпі, що, за словами одного фахівця з інформатики, розробникові "слід було знати не менше десятка мов, щоб заслужити громадську думку". З'явилися численні варіанти мов, що називались **підмножинами, розширеннями та діалектами**. Діалекти містять незначні зміни, які сприяють або налаштуванню мови на спеціальне застосування, або виявленню сильних сторін конкретного комп'ютера. Діалекти несумісні як з початковою мовою, так і з іншими діалектами тієї ж мови.

6.1. Перші кроки непроцедурної мови

Такі мови як Fortran, Algol, PL/1 відносяться до **процедурних мов**, що задають комп'ютеру явну послідовність дій, яку потрібно виконати для вирішення завдання. На відміну від цього **непроцедурні мови** дозволяють програмістові просто описувати бажаний результат, не задаючи точного способу його отримання. Прикладом непроцедурної мови може бути **Пролог (PROLOG, PROgramming in LOGic – програмування у логіці)**, який був розроблений у 1972 р. **Аланом Колмарі** з університету в Луміні (Марсель).

Витриманий у традиціях формальної логіки, Пролог придбав багато гарячих прихильників в європейських дослідницьких центрах зі штучного інтелекту. При роботі з ним програмістові не потрібно розписувати крок за кроком процедури – досить просто визначити безліч фактів та встановити відносини між ними. Наприклад, що "автомобіль" та "віз" кваліфікуються як таке, що називається "Засіб пересування", і що воно має щось, що називається "колесами". За допомогою цих співвідношень процедури, які вже вбудовані у мову, отримують логічний висновок про те, що, скажімо, "автомобіль" має "колеса". Це вже були перші зародки **об'єктно-орієнтованого програмування (ООП).** Ця особливість робить Пролог дуже зручним для написання різновиду програм, які отримали назву "**Експертні системи**". Будуючи логічні висновки на основі людських знань, які вводяться у програму, експертні системи досягають успіхів у таких областях діяльності, як діагностика захворювань, пошук родовищ корисних копалин тощо.

Одним із спеціалізованих розширень Алголу-60 була розроблена у 1964 році в Норвегії під керівництвом Дала і Нюгорда мова **Simula**. Ця мова призначалася для запису алгоритмів моделювання динамічних процесів (simulation – моделювання) і набула дуже великої популярності серед фахівців

аж до наших днів, вона є живою мовою імітаційного моделювання. Проте історичне значення Simula виходить далеко за рамки спеціалізованого застосування, в ній вперше з'явилося революційне для програмування поняття об'єкту. У мові Simula об'єкти були ще спеціалізованими, а в новій версії мови Simula-67 поняття об'єкту було узагальнене і йому був наданий універсальний характер.

Ідея об'єкту далеко не одразу завоювала комп'ютерний світ. Першим масштабним проектом, що втілив принцип об'єктно-орієнтованого програмування, був реалізований у 1973 році під керівництвом **Алана Кея** в Xerox PARC проект Alto, про який ми розповідали вище. Програмну частину проекту Alto склали мова та система програмування Smalltalk – «розмовонька», що розвивають принципи Simula-67, та докорінно відрізняються від усього, що було раніше.

Відомі мови (Фортран, Алгол, Кобол, PL/1) ґрунтувалися на концепції «дані-процедури», тобто були процедурно-орієнтованими. Smalltalk була першою мовою, яка була заснована на концепції «об'єкти-повідомлення», тобто об'єктно-орієнтованою. З погляду мови комп'ютер є середовищем населеним об'єктами, які посилають один одному повідомлення. Кожен об'єкт – це нібито мікроскопічний комп'ютер зі своєю пам'яттю та своєю системою команд (змінні та методи об'єкту).

Для спілкування з користувачем у системі були вперше застосовані особливі графічні об'єкти – вікна, а для управління ними використовувався створений Дугласом (Дагом) Енгелбартом у 1968 році маніпулятор «миша». Мова шліфувалася протягом кількох років, і у 1980 році з'явилася версія Smalltalk/V, яка й на сьогоднішній день є недосяжним ідеалом виключно об'єктного програмування.

Проект Smalltalk був проривом у майбутнє, що випередив свій час на 10-20 років. Хоча фірма не зуміла отримати з нього комерційну вигоду, могутні ідеї проекту мали далекі наслідки. Після відвідин керівництвом Apple у 1979 році центру PARC у світ ПК були вкладені принципи графічного віконного інтерфейсу, а ідеї об'єктності почали завойовувати усі сучасні мови програмування.

У 1980 році з'явилася мова програмування «Сі з об'єктами», пізніше перейменованна в C++. Починаючи з версії 5.5, об'єкти увійшли до Паскаля, об'єктною мовою став Visual Basic. У 1995 році із швидкістю епідемії розповсюдилася 100% об'єктна мова Java.

Разом з тим, можливості оригінальної мови Smalltalk до кінця не вичерпані. Дивовижна стрункість, логічність, витонченість цієї системи програмування, як нам здається, ще довго слугуватиме джерелом ідей для інших систем. На Smalltalk звернула увагу фірма IBM, вона включила її в список базових мов програмування.

6.3. Алфавітне хрещення

У роки дебюту мови Пролог з'явилася ще одна мова – **C**. Вона народилася у "Белл телефон лабораториз" (науково-дослідній фірмі, яка належала корпорації "Американ телефон энд телеграф", що розташована у Мюррей-хил, шт. Нью-Джерсі). Ніхто з керівників не замовляв нової мови. Мова С виникла лише як результат дружнього змагання всередині невеликої групи програмістів лабораторії. Не було ані проектів, ані специфікацій, ані вимог, все народилося просто без обговорень. Назва "Сі" (С) з'явилася також випадково, як і сама мова. Вона виявилася наступником раніше створеної мови для внутрішнього використання, що отримала назву Бі (B, C – друга і третя букви латинського алфавіту). Розробником мови **B** був Кенет (Кен) Томпсон, який брав участь ще раніше у розробці проекту Multics.

Мову С розробив у 1972 р. **Денніс Річи**, 31-річний фахівець з системного програмування, який, отримавши ступінь бакалавра з прикладної математики у Гарвардському університеті, поступив у 1968 р. на службу в лабораторію. Він наділив мову B додатковими можливостями, а також змінив синтаксис для полегшення роботи з нею.

Річи сподівався, що його нова мова стане у нагоді для програмування нової операційної системи "**Юнікс" (UNIX)**, робота над якою також не входила в офіційні плани компанії. Автор системи "Юнікс" **Кен Томпсон** вважався серед колег кращим програмістом у світі. Заради підвищення швидкості роботи операційної системи вони традиційно писалися на мові низького рівня – Асемблері, але мова С настільки добре зарекомендувала себе, що, на ній Томпсон впродовж 70-х років продовжував розробляти UNIX. Так, на С було написано більше 90% всього коду **центральної програми (ядра)** системи "Юнікс".

Після того, як ця ОС була завершена, про мову С заговорили знов у 1980 році завдяки комп'ютеру Belle. Цей комп'ютер, який виграв світову першість з шахів серед комп'ютерів, був розроблений Томпсоном та Джо Кондоном. Томпсон продовжував працювати у компанії Bell Labs до пенсії, на яку вийшов у 2000 році. Річи дотепер працює у компанії Bell Labs і продовжує свою діяльність над розробкою операційних систем.

Як тільки система "Юнікс" отримала визнання, мова С знайшла популярність як мова середнього рівня, в якій зручність, стисліть та мобільність мов високого рівня поєднуються з можливістю безпосереднього доступу до апаратури, що традиційно забезпечувалося Асемблером. Програмісти, що займаються машинною графікою, виявили, що мова С зручна при вирішенні таких завдань, як створення спеціальних ефектів (особливо у фільмах) або обробка зображень, що передаються зі дна океану в експедиції "Титанік".

«Чистий» С дотепер дуже широко використовується системними програмістами для створення додатків, що порівнюються за своєю ефективністю з програмами, які реалізовані на мові Асемблер, проте дійсну

славу цій витонченій та компактній мові принесли об'єктно-орієнтовані нащадки C++, Java та C#.

Еволюційний розвиток мови С перетворив її на принципово могутніший інструмент, ніж первинна версія. Влітку 1983 р. програміст **Ріки Масситті** запропонував нову версію мови – С++. Назва вказує на еволюційну природу мови (++ – це операція приросту в Сі). Втілив ідею у життя інший співробітник Bell Labs Бьярн Страуструп (Stroustrup, Bjarne; 1950 р.н.). На основі синтаксису С він розробив універсальну об'єктно-орієнтовану мову програмування С++. Вона є розширенням мови Сі, тому програми, які написані на Сі, можуть оброблятися компіляторами С++.

Мова програмування С була збережена як підмножина, тому зробленого у С акценту на засоби низького рівня достатньо, щоб вирішувати найнасущніші завдання системного програмування. Іншим джерелом нової версії слугувала мова Simula-67, з якої була запозичена концепція класу (разом з похідними класами та функціями-членами). Це було зроблено, щоб сприяти модульності завдяки використанню віртуальних функцій.

Можливості С++ з перевантаження операцій та свобода у розташуванні описів скрізь, де може зустрічатися оператор, схожі на Алгол-68. Мова С++ дозволяє розумним чином структурувати великі програми та приховувати їх за елегантними та надійними інтерфейсами. Крім того, вона володіє можливостями для безпосередньої та ефективної роботи з апаратними засобами. Мова С++ вважається однією з перших (якщо не першою) по-справжньому мовою об'єктно-орієнтованого програмування.

До кінця XX століття мова С++ стала стандартною мовою професійного програмування. А на самому початку нового XXI століття з'явився ще один наступник С під назвою C# (читається «сі шарп»; музичний знак дієза натякає на підвищення можливостей базової мови). Мова C# була запропонована фірмою Microsoft як зброя у конкурентній боротьбі проти переможної ходи мови Java, про яку йтиметься нижче.

За словами розробників, мова C# увібрала в себе все краще з таких популярних мов як С++, Visual Basic, Java та Object Pascal. Вона забезпечує швидку розробку і у той же час дозволяє створювати ефективний код. Більш того, фірма заявила, що C# – перша мова, в якій реалізована принципово нова концепція компонентного програмування, де програмною одиницею стає не об'єкт, а компонент, що є готовим додатком, який поширюється по комп'ютерній мережі. Питання про те, чи дійсно компонентне програмування є зміною парадигми, або це – деяка еволюція об'єктного програмування, є дискусійним. Відповідь на це дасть час.

6.4. Успіх та суперечки

У Європі з 60-х років йшли дискусії навколо спірних спроб щодо вдосконалення мови програмування Алгол-60. На початку 1962 р. Міжнародна федерація з обробки інформації IFIP сформувала новий комітет з Алголу. **К.**

Хоар з Оксфордського університету, член комітету від Великої Британії, лауреат Тьюрінговської премії, вищої нагороди за науковий внесок в області інформатики, та швейцарський учасник **Никлаус Вірт** (Wirth, Niklaus; 1934 р.н.), що викладав у Стенфордському університеті, запропонували помірно модифіковану версію Алголу. Ця версія отримала назву Алгол-w на честь Вірта (Wirth). Вона набула популярності в деяких університетах, але була знехтувана комітетом.

Нарешті, у грудні 1968 р. в Мюнхені, після численних суперечок, народився новий переношений монстр Алгол-68. Хоар, Вірт та інші відомі вчені висловлювалися проти нової мови. Як і передбачалося Алгол-68 виявився дуже громіздким, незрозумілим та складним, і на відміну від свого попередника Алголу-60 не отримав визнання навіть в Європі. Проте в стороні від суперечок виникла ще одна нова елегантна мова, яка у подальші роки зробила вплив на програмістів в усьому світі. Член комітету з Алголу-68 Н. Вірт був проти ухвалення складного стандарту Алгол-68. Коли з ним не порахувалися, він на знак протесту у 1971 році розробив нову алголоподібну мову, просту та зрозумілу, призначену, перш за все, для навчання студентів.

Мова називалася **Pascal (Паскаль)** – на честь французького філософа та математика 17 ст., винахідника першої обчислювальної машини Блеза Паскаля. Вірт почав писати її у 1968 р., будучи професором інформатики у Федеральному технічному університеті в Швейцарії, йому потрібен був інструмент для навчання студентів навичкам програмування. Вірта не задовольняв не тільки новий Алгол, але й усі інші мови програмування, властивості та конструкції яких часто не можна пояснити логічно та переконливо.

6.5. Інженерний підхід

У 1963 р. Вірт отримав ступінь бакалавра з електротехніки у Каліфорнійському університеті. Він підійшов до розробки мови, як інженер при конструюванні машини. “Мистецтво інженера, – говорив він, – полягає у тому, щоб робити складні завдання простими”.

В окремому розділі на початку програми на Паскалі програміст повинен визначити всі змінні та задати явно типи їх значень, а також описати процедури та функції разом з параметрами та їх типами і результатами виконання процедур. Паскаль також заохочує використання логічної структури, яка ділить програму на невеликі прості підзадачі. Вводячи такий розділ, Паскаль обмежує свободу програміста. Але одночасно мова сприяє більш строгому стилю програмування, який скорочує кількість помилок. Структура Паскаля робить програми такими, що легко читаються, дозволяючи навіть людям, що не писали ту чи іншу програму, виявляти та виправляти у ній помилки та вносити зміни.

Завдяки цьому Паскаль став особливо зручним для вивчення теорії та практики програмування, але не для реальних застосувань. Успіх Паскаля набагато перевищив скромні очікування Вірта. Колеги в усьому світі прийняли

його як засіб навчання програмуванню майбутніх фахівців з інформатики. Можливо, найважливіше полягало у тому, що Паскаль став путівною зіркою руху до структурного **програмування,** яке зароджується у той час і набирає сили у кінці 70-х років. Воно ставило на меті перегляд способу розробки програм і почалося з публікації у 1972 р. книги англійця К. Хоара, норвежця Дж. Дала та видатного голландського фахівця з інформатики Едсгера Дейкстри.

6.6. Структурний підхід

Хоча цю фразу різні програмісти розуміли по-різному, проте по суті вона характеризує систематичний математичний підхід до створення програмного забезпечення, зокрема вимагає розділення програми на невеликі логічно пов'язані завдання. Одна із специфічних цілей структурного програмування – зменшення використання так званих безумовних переходів, або операторів GOTO. Хоча оператор GOTO і зручний при написанні програм, але зазвичай ускладнює їх читання, а, отже, збільшує ймовірність пропуску можливої помилки.

Підкреслюючи необхідність суворої дисципліни при написанні програм, прихильники структурного програмування сподівалися обмежити все зростаючу складність розробки програмного забезпечення. Системи управління авіаційним рухом, тривимірні завдання математичної фізики, а пізніше системи управління космічними супутниками та кораблями та інші, вимагали таких великих програм, що на їх розробку витрачались роки. Причому писалися вони по частинах різними групами програмістів, які, у більшості випадків, і не знали інших частин програми. Дуже часто в результаті таких робіт з'являлися програмні системи вартістю у мільйони доларів, які були створені із запізненням на багато місяців та з тисячами помилок. Проблема набувала серйозного характеру, переростаючи у "**кризу програмного забезпечення**".

Ніде криза не переживалася так гостро, як у військовому відомстві США, найбільшому споживачеві комп'ютерів та програм. Тому у 1973 р. Пентагон виділив близько половини з 7.5 млрд. доларів свого комп'ютерного бюджету на розвиток та супровід програмного забезпечення. Асигнування на апаратне забезпечення комп'ютерів, навпаки, було зменшене, не дивлячись на суттєве збільшення потужності та пам'яті комп'ютерів.

Значною мірою проблема породжувалася неймовірною мішаниною мов, на яких писалися програми особливо для вбудованих систем крихітних комп'ютерів (мікропроцесорів) балістичних ракет або для великих за розміром комп'ютерів, що передавали сигнали на кораблі або літаки. Як випливає з оглядів [2], на початку 70-х років **існувало близько 450 мов високого рівня та їхніх діалектів**, що використовували для кодування таких програм. А якщо врахувати мови Асемблера, то ця цифра доходила до 1.5 тис. Багато мов були розроблені лише для конкретних робіт, оскільки жодна із універсальних мов програмування не містила засобів, які були необхідні при вирішенні конкретної задачі.

Практично всі основні концепції – процедурне, логічне, об'єктно-орієнтоване програмування були запропоновані у це бурхливе десятиліття. У наступні роки прогрес в автоматизації програмування йшов не у бік створення нових мов, а, навпаки, шляхом природного відбору. Мови програмування народжувалися та вмирали, але тільки деякі з них – найбільш стійкі та життєздатні – дожили до кінця XX століття і стали стандартними у міжнародному співтоваристві програмістів.

У січні 1975 р. Пентагон вирішив навести лад в "лінгвістичному хаосі". Він заснував великий комітет HOLWG (High Order Language Working Group) – робочу групу з мови високого рівня, до якої увійшли представники всіх родів військ США, а також трьох союзників США по НАТО (Франція, Західна Німеччина, Великобританія). Комітету було доручено знайти або розробити мову, яка була б придатна для програмування всіх відомих вбудованих систем.

6.7. Поява мови "Ада"

Головою комітету та основним натхненником у розробці програм був призначений підполковник ВПС **Уїльям Уайтекер**. Уайтекер досяг найвищих академічних ступенів за всю історію льотної школи, але керувати літаком так і не навчився. Проте, він залишився у ВПС і за 16 років роботи над ядерною зброєю став висококваліфікованим програмістом.

Замість того, щоб займатися нескінченними дебатами (як це було раніше в інших комітетах), а потім ухвалювати рішення про мову, комітет (за наполяганням Уайтекера) почав збирати побажання користувачів комп'ютерів як серед військових, так і серед інших фахівців з програмування.

Їх попросили допомогти у формулюванні необхідних вимог до універсальної мови. Завдання вироблення проекту загальних специфікацій було покладене на **Девіда Фішера** – цивільного співробітника Інституту оборонних досліджень США. Фішер вніс у цю роботу власний солідний теоретичний та практичний багаж. Він закінчив два університети і свого часу розробляв військове програмне забезпечення у фірмі "Барроуз".

У квітні 1975 р., через три місяці після утворення комітету HOLWG, перший проект вимог до універсальної мови був переданий для ознайомлення військовим, у промислові та академічні кола під кодовою назвою "**Солом'яний".** Вибір назви був не випадковий: Фішер і Уайтекер розраховували, що рецензенти як слід пошарпають документ і запропонують свої покращення.

"Биття" не змусило себе довго чекати, і з урахуванням критичних зауважень "Солом'яний" був переглянутий. Цикл рецензування та переробки протягом подальших трьох років повторювався ще п'ять разів. Кожен подальший документ отримував назву, яка начебто відзначала прогрес у зміцненні вимог: "**Дерев'яний", "Олов'яний", "Залізний"**, "**Переглянутий залізний"** і, нарешті – "**Сталевий".**

Список вимог подовжився, досягнувши приблизно сотні на етапі “Олов’яного”, що жодна з існуючих мов задовольнити їм не може. Комітет вирішив, що вимоги можна задовольнити, тільки створивши абсолютно нову мову.

У травні 1977 р. комітет запитав пропозиції у найвідоміших проектувальників мов з різних країн, причому пропозиції повинні були базуватися на одній з трьох мов: Пл/1, Алгол-68 та Паскаль.

Було відібрано чотири пропозиції (всі засновано на Паскалі) для доопрацювання протягом 6-місячного етапу. Всі команди розробників отримали закодовані колірні імена, щоб зберегти анонімність при рецензуванні.

У 1978 р. на основі оцінок більше 400 рецензентів два з чотирьох проектів (Червоний – компанія “Інтерметрікс” (Каліфорнія) та Зелений – “Сі Хоннуелл Буль” (паризька філія американської компанії “Хоннуелл корпорейшн”)) були відібрані для остаточного розгляду.

Переможця оголосили у травні 1979 р. Ним виявилася фірма “Сі Хоннуелл Буль”. Мова, що перемогла, а мова була охрещена **Адою**, на честь **Августи Ади Байрон (графині Лавлейс)**, математика та письменниці 19 ст., яку називають першим у світі програмістом – за популярний опис Аналітичної машини Чарльза Беббіджа та цикл програм для цієї машини.

Ця перемога була особистим тріумфом **Жана Іхбіа**, який очолював групу “Зеленого” проекту. Іхбіа народився у Парижі, навчався в престижній Політехнічній школі, володар “коричневого поясу” з дзюдо. Пізніше був направлений на навчання у США. Незабаром після захисту кандидатської дисертації з розміщення систем метро поступив на службу у нову французьку компанію “Сі” (пізніше – “Сі Хоннуелл Буль”), де і захопився програмуванням. Результатом роботи групи була мова, яка була розроблена не комітетом, а невеликою групою (10 осіб) з сильним лідером.

Найхарактерніша риса мови Ада – головний акцент на структурне програмування. Мова дозволяє писати програми у вигляді **пакетів**, тобто самостійних **модулів**, які розробляються окремими програмістами, а потім збираються разом. Пакет може бути розроблений, відлагоджений, протестований та збережений у бібліотеці модулів для подальшого використання у програмі, начебто це була частина стандартного програмного забезпечення. Така модульна схема дозволила створювати надійні, легкі для читаня та зручні у супроводі програми.

Проте “шанувальники” Ади припускали, що за всі названі переваги доведеться платити. Оскільки цей проект створювався відповідно до урядового документа ”Сталевий”, у ньому виявилося багато особливостей, що ускладнюють його вивчення. Крім того, компілятор Ади займав набагато більший об’єм пам’яті, ніж компілятор її попередника, Паскаля.

Розміри та складність мови Ада викликали критику фахівців, зокрема автора Паскаля Н. Вірта та його колеги К. Хоара, які брали участь у групі “СРІ Інтернешил”, яка вибула із змагання у півфіналі.

Зокрема, Вірт казав: “Дуже багато всього вивалюється на програміста. Я не думаю, що, вивчивши третину мови Ади, можна нормально працювати. Якщо не засвоїти всіх деталей мови, то надалі можна спіткнутися об них і це призведе до неприємних наслідків”.

Захищаючи свою мову, головний “архітектор” Ади Жан Іхбіа, заперечивши Вірту своє “захоплення і пошану”, додав: “Іноді Вірт вірить в малі вирішення великих проблем. Я не вірю в такі міражі. Крупні проблеми вимагають крупних рішень”.

Як і слід було чекати, створення компілятора, що дозволяв ефективно виконувати програми на Аді, виявилося нелегким завданням. Воно посилювалося вимогами Пентагону, щоб нова мова не засмічувалася діалектами, розширеннями або підмножинами. Пентагон вимагав також, щоб кожен пропонований компілятор піддавався строгій процедурі перевірки, яка складалася з серії (близько 2000) спеціальних тестів.

Не дивлячись на всі труднощі, вдалі компілятори врешті-решт з’явилися, і Ада почала заявляти про себе. У 1983 р. міністерство оборони США розпорядилося, щоб усі нові програми для “**критичних”** додатків писалися тільки на Аді. Під “критичними” розумілися комп’ютеризовані системи зв’язку та системи озброєнь. Пентагон передбачив, що до кінця десятиліття 85% нового “критичного” програмного забезпечення (вартістю 5 млрд. дол.) буде написано на Аді.

Таким чином, Ада у 1980 р. впритул підійшла до гарантованої мобільності: програму, яка була написана для одного комп’ютера, майже напевно можна було перекомпілювати та правильно виконати на іншій машині. Одне це робить Аду важливим програмним засобом при розробці великих проектів.

7. Програмування приходить у наші домівки

Після написання інтерпретатора мови Бейсік для комп'ютера "Альтаїр" першокурсником Гарвардського університету Білом Гейтсом та його другом Поллом Алленом, молодим програмістом з фірми "Хоніуелл", фірма MITC почала продавати власну версію Бейсіка за 500 дол., відраховуючи відсотки з кожної проданої копії Гейтсу і Аллену. Приблизно у той же час у 1975 р. **Боб Албрехт** почав свій проект **Малюк-Бейсік**. На початку 70-х років Албрехт організував видавничу фірму "Дімакс", яка випустила серію книг-посібників з Бейсіку (серед них була і його книга), метою яких було прагнення впровадити комп'ютери у повсякденне життя.

Албрехт також організував загальнодоступний обчислювальний центр в Менло-парку неподалік від Сан-Франциско. Центр мав у своєму розпорядженні міні-комп'ютер PDP-8, поставлений фірмою ДЕК (DEC, від Digital Equipment Corporation – корпорація з виробництва цифрового устаткування) в обмін на обіцянку Албрехта написати посібник з програмування на Бейсіку для машини PDP-8. За 50 центів за годину кожен міг зайти у центр і потішитися з комп'ютером. Центр також організував доступні всім термінальні класи для роботи з Бейсіком, які особливо привертали підлітків і навіть бізнесменів. Останні сподівалися завоювати ринок, розробивши програми, що прогнозують зміни цін.

Спільно з **Томом Пітманом**, членом комп'ютерного клубу, Албрехт написав інтерпретатор для Малюка-Бейсіка. Колеги Пітмана по комп'ютерному клубу обурилися високою ціною Бейсіка для "Альтаїра", і наполягали на безкоштовному розповсюдженні програмного забезпечення. Різні варіанти Малюка-Бейсіка були надруковані у журналах. Албрехт та його друзі тоді вважали, що все програмне забезпечення повинне бути вільним.

7.1. Розквіт Бейсіка

Історія "Альтаїра" та Малюка-Бейсіка завершилася у кінці 70-х років. Мікрокомп'ютери тепер випускалися у повністю зібраному вигляді, з великою пам'яттю, здатною вміщувати великі діалекти Бейсіка. Деякі виробники вбудовували інтерпретатори власних версій Бейсіка безпосередньо в ПЗП (Постійний Запам'ятовуючий Пристрій) комп'ютера.

Багато у чому завдяки заповзятливості Гейтса та Аллена, невтомній діяльності Боба Албрехта та ентузіазму маловідомих розробників, ця мова завоювала провідне місце у програмуванні мікрокомп'ютерів. В середині 80-х років вже мільйони людей у США та інших країнах використовували Бейсік, причому головним чином на персональних комп'ютерах.

Два перші розробники Бейсіка, Джон Кемені та Томас Курц, переживали стосовно феноменального успіху своєї мови змішані почуття. Вони були безмірно горді тією роллю, яку ця мова зіграла "відкриттю комп'ютерів масам",

але одночасно збентежені тим, на що перетворилося їх творіння – безліч клонів та варіантів в основному через обмежену пам'ять перших мікрокомп'ютерів.

В результаті багато версій Бейсіка виявилося неможливим переносити з машини на машину. Виявилися, також, інші проблеми. Багато фахівців з інформатики відносилися до Бейсіка досить зневажливо. Наприклад, англієць Хоар порівнював програмування на Бейсіку з грою на піаніно двома пальцями: граючи прості мелодії, новачки швидко роблять успіхи, але потім стикаються з величезними труднощами. Бейсік зручний для написання простих коротких програм, але через відсутність у мові структури не придатний для складних програм. Це стало причиною рішення деяких коледжів та шкіл відмовитися від використання Бейсіка для навчання програмуванню, та замінити його більш структурованою мовою Паскаль, яка була створена Никлаусом Віртом.

7.2. Поява мови Модула-2

Кемені та Курц виявилися друзями по нещастю з Никлаусом Віртом. Створена ним структурована мова Паскаль також переживала величезний успіх та у кінці 70-х років набула широкого поширення в університетах. Але розробники програмного забезпечення, що спробували пристосувати Паскаль для мікрокомп'ютерів і використовувати його у комерційних цілях, внесли до мови значні зміни. З цієї причини в компіляторах Паскаля з'являлися все нові розширення, які виводять мову за межі чистих академічних застосувань.

Вірт не визнав нові діалекти, проголосивши у 1977 р. свою позицію: "Якщо мова не підходить для застосування, на яке його творець явно не розраховував, то слід набратися сміливості і створити нову мову, а не ускладнювати ту, що існує".

Саме це Вірт і зробив. У 1981 р. він розробив мову Модула-2, яка повинна була замінити Паскаль в універсальних застосуваннях. Подібно до Ади мова Модула-2 була розширенням Паскаля у декількох напрямах. Найпомітнішим було введення засобів написання великих та складних програм, які можна тепер створювати у вигляді окремих самостійних модулів, а потім збирати їх разом. Проте у кінці 80-х, поки компанії з розробки програмного забезпечення працювали над компіляторами для Модула-2, які дозволили б використовувати мову на мікрокомп'ютерах, з Паскалем трапилося диво.

Донині різні компілятори Паскаля для мікрокомп'ютерів не мали особливого комерційного успіху, оскільки були дорогими, займали великий об'єм пам'яті на диску і працювали з черепашачою швидкістю. Але тут на арену вийшов колишній студент Вірта зі своїм компілятором для мікрокомп'ютерів, який він назвав Турбо-Паскаль. Це був компактний, такий, що миттєво спрацьовує та напрочуд дешевий компілятор.

7.3. Музикальний француз

Розробником Турбо-Паскаля став “божевільний француз” на ім’я **Пилип Кан**. Родом з Парижа, Кан якийсь час вчився у Цюріху, де прослуховував вступний курс Вірта з Паскаля. Повернувшись до Франції, він захистив дисертацію на ступінь кандидата наук з математики. У цей період він іноді підробляв грою на саксофоні у нічних клубах.

Кан займався математикою, а комп’ютерами цікавився лише тому, що це допомагало йому у вирішенні деяких завдань. Потім він придбав свій перший мікрокомп’ютер “Епл-2”. “Епл-2” приніс йому такий же успіх, який він приніс творцям цієї машини Стівену Джобсу та Стефану Возняку. Перша версія компілятора Паскаль для “Епл-2” була написана данцем **Андерсом Хейльсбергом** і продавалася в Європі під назвою **Compass Pascal**. Пилип Кан придбав цей компілятор на правах продажу компілятора безпосередньо користувачам. Навіть ця перша версія мала інтегрований в середовище редактор коду та компілятор. Редактор був простим у користуванні з набором команд популярного тоді редактора WordStar. Компілятор дозволяв створювати програми, які використовували до 64К оперативної пам’яті для збереження коду програми та даних, з якими вона працювала, та додатково генерував швидкий код. Незадоволеність існуючими компіляторами Паскаля спонукала Кана взятися за розробку Турбо-Паскаля. У 1982 р., маючи на руках цю програму та 2000 дол. у кишені, він покинув Францію і виїхав до Каліфорнії шукати успіху.

Спочатку Кан не міг отримати там навіть роботу. Подорожуючи по туристичній візі, він не мав права на так звану зелену картку – урядовий дозвіл на отримання роботи у США. У відчаї він вирішив зробити ставку на Турбо-Паскаль і створив нову фірму з розробки програмного забезпечення. Він назвав її “Борланд интернэшнл, нині – Inprise” – назва, що була підказана телепередачею про колишнього космонавта з схожим ім’ям, Франк Борман. Кан вирішив, що ім’я, яке має “всеамериканське звучання”, послужить відповідним прикриттям для чужака у світі бізнесу.

Не дивлячись на гучне ім’я фірми, Кану не вдалося зацікавити навіть представників ризикового капіталу. Кан залучив до роботи у своєму новому підприємстві інших освічених новачків: колишнього власника японського ресторану та офіціантку коктейль-бару, що мала ступінь кандидата наук з психології. Кан зібрав невелику суму (20 тис. дол.) у членів своєї родини, та у березні 1984 р. організував контору у двокімнатному офісі над авторемонтним гаражем недалеко від Сан-Хосе. Приступивши до справи, Кан умовив службовця одного популярного комп’ютерного журналу провести широку рекламу Турбо-Паскаля у кредит.

Кан розумів, що рекламне оголошення – єдино можливий спосіб пробитися. Прочитавши книгу про продаж товарів поштою, де рекомендувалося використовувати яскраві кольори для залучення уваги читачів, він оформив рекламне оголошення в різких зелених, синіх і рожевих

тонах. У книзі також йшлося, що для швидкого успіху вартість товару повинна бути менше 50 доларів. Тому він оцінив свій компілятор і редактор програм у 49 дол. 95 центів. Це була гранична нижня межа – решта систем продавалася в 10 разів дорожче.

Реклама виявилася геніальною знахідкою для завоювання ринку. Тільки за перший місяць вона принесла Кану замовлень на 150 тис. доларів, так що місцеві банки навіть запідозрили його у поштовому шахрайстві і спочатку відмовлялися оплачувати чеки. Проте замовлення продовжували поступати.

У перші два роки було продано не менше 300 тис. копій Турбо-Паскаля, що перевершило об'єм продажу всіх інших компіляторів мов для мікрокомп'ютерів. Подібно до того, як десятьма роками раніше нова версія Бейсіка "вивела на орбіту" фірму "Microsoft", так Турбо-Паскаль майже за одну ніч перевів "Борланд" у розряд основних виробників програмного забезпечення.

У 1985 р. Кан з легкістю викинув 45 тис. доларів на організацію прийому, запросивши на нього 600 клієнтів фірми "Борланд" та їх гостей. На честь свого успіху, який був досягнутий у 33 роки, він зіграв декілька речей на саксофоні.

Потім спільно з Андерсом Хейльсбергом була випущена покращена версія Турбо Паскаля. З кожною новою версією Паскаль ставав все могутнішим, вбираючи, подібно Бейсику, нові ідеї у програмуванні. На початку 90-х років у ньому з'явилися об'єкти, що прийшли з мов Simula та Smalltalk, а в середині десятиліття фірма Borland, об'єднавши в одному продукті всі досягнення свого часу, зокрема концепцію модульного програмування з Modula-2, вийшла на ринок IBM PC з новим візуальним середовищем програмування Delphi. Цей пакет придбав дуже велику популярність серед прикладних програмістів, зокрема у СНД.

Згодом Андерс Хейльсберг став провідним конструктором мови Турбо-Паскаль і автором оболонки Delphi фірми Borland. За час його роботи у корпорації Microsoft з 1996 р. головною його заслугою вважається створення мови Cool, пізніше перейменованого в C#. На даний момент Андерс Хейльсберг займається розробками нових мов програмування корпорації Microsoft.

7.4. Довгожитель Lisp – інструмент функціонального програмування

Lisp (LISt Processing – обробка списків) – ще один довгожитель, разом з Фортраном та Коболом, у родині мов програмування. Мова створена у 1960 році **Джоном Маккарті** (McCarthy, John) у Массачусетському технологічному інституті на теоретичному фундаменті так званого лямбда-числення, яке було запропоноване ще у 1930 році відомим американським логіком Алонзо Черчем.

Як випливає з назви, оброблюваним об'єктом у мові є список, а програма на Lisp – послідовність викликів функцій, немає нічого, окрім викликів функцій. Тому програмування на такій мові називається ***функціональним***.

У Lisp дуже широко використовується рекурсія, у цілому це неймовірно елегантна з математичної точки зору мова, яку вдосконалювати майже

неможливо, проте програма на ній має незвичайний зовнішній вигляд через застосування великої кількості дужок.

Оідразу ж після появи, Lisp став дуже популярним, перш за все як мова для розв'язку логічних завдань. Вона дуже легко реалізується на апаратному рівні, на даний час спеціалізовані Lisp-процесори широко використовуються в системах штучного інтелекту для військових потреб, наприклад, у системах самонаведення крилатих ракет. Більш того, Lisp, не дивлячись на деяку громіздкість, широко застосовується у США як обчислювальна мова, поступаючись за поширеністю тільки Фортрану. Зокрема, у відомій системі автоматизованого проектування AUTOCAD використовується діалект Lisp під назвою AutoLisp.

7.5. Prolog – нездійснена мрія ЕОМ V покоління

Мова **Prolog** (PROgramming for LOGic – програмування для логіки) була створена в Європі на початку 70-х років. Теорія була розроблена в Едінбурзі (Великобританія) Робертом Ковальським та Маартеном ван Емденом, перша реалізація здійснена Аленом Кольмарі (Colmerauer, Alain) з Марселя (Франція).

На відміну від усіх попередніх мов, включаючи Лісп, Пролог не є алгоритмічною мовою, тобто він не описує **як** треба отримати результат. Він є ***декларативною*** мовою, тобто показує **що** дано та що треба отримати.

Програма на Пролозі складається з ***фактів*** деякої наочної області та ***правил***. Інтерпретуюча система сама робить логічний висновок, тобто застосовує правила до фактів, для цього використовується апарат математичної логіки (логіка предикатів першого порядку).

Варто пригадати те, які надії декілька десятиліть тому покладалися на штучний інтелект, якими розумними уявлялися майбутні машини і те, яким провалом все зрештою скінчилося. Фон нейманівські машини (фактично – послідовні цифрові процесори) здатні сприймати єдиний вхідний потік одиниць і нулів та обробляти дані, що складаються з тих же одиниць і нулів. Тому, можливості таких комп'ютеров з часом виявляться вичерпаними, і ні про яке реальне використання сучасних комп'ютерів для вирішення інтелектуальних завдань говорити не доводиться. Рано чи пізно, окрім вирішення рутинних завдань, сучасні комп'ютери повинні стати інтелектуальними асистентами людини так, як це їх собі уявляли Ванневар Буш та Даг Енгельбарт.

Тому саме на Пролог – могутню та красиву мову у 80-і роки були покладені дуже великі надії. Мова йде про вельми амбітний проект ЕОМ п'ятого покоління, який був заявлений Японією у 1981 році та розпропагандований у пресі під назвою «Японський виклик». Йшлося про комп'ютер з паралельними процесорами, що працює з даними, які зберігаються в обширній базі даних, а не у файловій системі. При цьому доступ до даних повинен був здійснюватися за допомогою мови логічного програмування.

Метою програми було створення «епохального комп'ютера» з продуктивністю суперкомп'ютера і потужними функціями штучного інтелекту.

Початок розробок – 1982, кінець розробок – 1992, вартість розробок – 57 млрд ієн (близько 500 млн дол).

Цей термін повинен був підкреслити, що Японія планує здійснити новий якісний стрибок у розвитку обчислювальної техніки. У той час як попередні покоління вдосконалювалися за рахунок збільшення кількості елементів на одиницю площі (мініатюризації), комп'ютери п'ятого покоління повинні були для досягнення надпродуктивності інтегрувати величезну кількість процесорів.

Передбачалося, що у 90-х роках будуть створені комп'ютери, які мають принципові відмінності від усіх попередніх:

– нова технологія виробництва мікросхем, що знаменує перехід від кремнію до арсеніду галію, яка дає можливість на порядок підвищити швидкодію основних логічних елементів;

– прототип машини володітиме продуктивністю між 100 млн та 1 млрд LIPS, де LIPS – це логічний висновок за секунду (на той час типові робочі станції були здатні на продуктивність близько 100 тисяч LIPS);

– нова архітектура (не фон нейманівська);

– нові способи введення-виведення інформації – розпізнавання та синтез мови та образів;

– відмова від традиційних алгоритмічних мов програмування (Фортран, Алгол і т. п.) на користь декларативних;

– створення робочих станцій з високою продуктивністю та розподіленими функціями;

– орієнтація на завдання штучного інтелекту з автоматичним пошуком розв'язання на основі логічного висновку;

– комп'ютери повинні оперувати величезними базами даних та базами знань;

– підвищення продуктивності за рахунок реалізації програмних рішень на апаратному рівні;

– продуктивність таких машин повинна сягати одного мільярда логічних висновків за секунду.

Йшлася мова про комп'ютер з паралельними процесорами, що працюють з даними, які зберігаються в обширній базі даних, а не в файловій системі. При цьому доступ до даних повинен був здійснюватися за допомогою мови логічного програмування. Передбачалося, що прототип машини буде володіти продуктивністю між 100 млн і 1 млрд LIPS, де LIPS – це логічний висновок в секунду. До того часу типові робочі станції були здатні на продуктивність близько 100 тис. LIPS.

Хід розробок представлявся так, що комп'ютерний інтелект, набираючи потужність, починає змінювати сам себе, і метою було створити таке комп'ютерне середовище, яке само почне виробляти наступне, причому принципи, на яких буде побудований остаточний комп'ютер, були заздалегідь невідомі, ці принципи належало виробити в процесі експлуатації початкових комп'ютерів.

Далі, для різкого збільшення продуктивності, пропонувалося поступово замінювати програмні рішення апаратними, тому не робилося різкого поділу між завданнями для програмної та апаратної бази.

Очікувалося домогтися істотного прориву в області вирішення прикладних задач штучного інтелекту. Зокрема, повинні були бути вирішені наступні завдання:

- автоматичний портативний перекладач з мови на мову (зрозуміло, безпосередньо з голосу), який відразу би усунув мовний бар'єр японських підприємців на міжнародній арені;
- автоматичне реферування статей, пошук сенсу;
- інші завдання розпізнавання образів – дешифрування, аналіз дефектів тощо.

В основу програмного забезпечення майбутніх ЕОМ передбачалося покласти мову Пролог, а всю архітектуру орієнтувати на реалізацію алгоритмів логічного висновку.

Цю програму передбачалося реалізувати за 10 років, три роки для початкових досліджень і розробок, чотири роки для побудови окремих підсистем, і останні чотири роки для завершення всієї системи.

Віра в майбутнє паралельних обчислень була на той час настільки глибокою, що проект «комп'ютерів п'ятого покоління» був прийнятий в комп'ютерному світі дуже серйозно. Після того, як Японія в 70-і рр. зайняла передові позиції в побутовій електроніці, і в 80-ті стала виходити в лідери в автомобільній промисловості, японці придбали репутацію непереможних.

На жаль, цьому проекту не судилося збутися. Основна причина – не вдалося досягти ефективної апаратної інтерпретації Прологу. Пролог також не підтримував паралельні обчислення. Його робота у мультипроцесорному середовищі виявилася неефективною, не дивлячись на всі спроби модернізувати його. Було прийнято рішення по створенню нових типів мов програмування.

Був зроблений помилковий вибір мов типу Лісп і Пролог для створення бази знань та маніпулювання даними. У 1980-і рр. ці системи програмування користувалися популярністю для САПР та експертних систем, проте експлуатація показала, що додатки виявляються мало надійними і погано відлагоджувались в порівнянні з системами, які були розроблені звичайними технологіями, від чого від цих ідей довелося відмовитися. Корпораціями, що займалися розробкою програмного забезпечення, були запропоновані нові мови, але кожна з них володіла істотними недоліками, що не дозволяло у повній мірі використовувати паралельні обчислення.

Інша проблема виникла з продуктивністю процесорів. Запаралелювання багатьох процесорів не викликало очікуваного різкого стрибка продуктивності. Вийшло так, що робочі станції, які були створені в рамках проекту, успішно досягли і навіть перевершили необхідні потужності. Розроблені у лабораторіях

машини, швидко застарівали. З'являлися комерційні комп'ютери, які за швидкістю вже перевершували їх.

План заміни програмних засобів їх апаратними аналогами був невдалий. Таке бачення у інженерів було у попередньому поколінні комп'ютерів. Але на сьогоднішній день ситуація змінилася докорінно. Процес розвитку інформаційних технологій пішов зворотнім шляхом. Апаратне забезпечення спрощувалося, надаючи універсальності, а всі завдання перекладалися на програмне забезпечення.

Ідея саморозвитку системи, за якою система сама повинна змінювати свої внутрішні правила і параметри, виявилася непродуктивною – система, переходячи через певну точку, скочувалася в стан втрати надійності та втрати цілісності, різко «дурніла» і ставала неадекватною.

З будь-яких точок зору проект можна вважати абсолютним провалом. За десять років на розробки було витрачено понад 50 млрд ієн, і програма завершилася, не досягнувши мети. Робочі станції так і не вийшли на ринок, тому що однопроцесорні системи інших фірм перевершували їх за параметрами, програмні системи так і не запрацювали, поява Інтернету зробило всі ідеї проекту безнадійно застарілими. Проект був обмежений категоріями мислення 1970-х рр. і не зміг провести чіткого розмежування функцій програмної та апаратної частини комп'ютерів. Завдання створення нового, революційного покоління комп'ютерів явно відкладається на XXI століття.

7.6. Революція на ім'я Java

Починаючи з кінця 1995 року, весь комп'ютерний світ був захлеснутий лавиною рекламних повідомлень, серйозних публікацій та просто чуток навколо модної новинки – мови програмування Java, яка була запропонована компанією **Sun Microsystems**. Згідно із запевненням розробників, через декілька років Java повинна була витіснити всі існуючі мови програмування на всіх типах комп'ютерів – від вбудованих у побутові прилади мікропроцесорів до суперсистем. Пройшло декілька років, пил улігся, тепер можна спокійно обговорити феномен Java, його переваги та недоліки.

Мова Java зародилася як частина проекту створення передового програмного забезпечення для різних побутових приладів. Реалізація проекту була почата на мові C++, але незабаром виникла низка проблем, найкращим засобом боротьби з якими була зміна самого інструменту – мови програмування. Звідси й існує помилкова думка, що мова Java відбулася від мови програмування C++. Стало зрозумілим, що необхідна незалежна мова програмування, яка дозволить створювати програми, які не доводилося б компілювати окремо для кожної архітектури і можна було б використовувати на різних процесорах під різними операційними системами.

Народженню мови Java передувала досить цікава історія. У 1990 році розробник ПЗ компанії Sun Microsystems **Патрік Ноутон** (**Patrick Naughton**) зрозумів, що йому набридло підтримувати сотні різних інтерфейсів програм, які

використовувались у компанії, і повідомив виконавчого директора Sun Microsystems та свого друга **Скотта Макнілі** (**Scott McNealy**) про свій намір перейти працювати у компанію NEXT. Макнілі, у свою чергу, попросив Ноутона скласти список причин своєї незадоволеності.

Ноутон, хоча і не розраховував на те, що хтось зверне увагу на його лист, все ж таки виклав свої претензії, нещадно розкритикувавши недоліки Sun Microsystems, зокрема, архітектуру програмного забезпечення NEWS, що розроблялося на той момент. На здивування Ноутона, його лист здобув успіх: він був розісланий усім провідним інженерам Sun Microsystems, які не забарилися відгукнутися та висловити гарячу підтримку своєму колезі і схвалити його погляди на ситуацію в Sun Microsystems. Звернення викликало схвалення й у вищого керівництва компанії, а саме у **Біла Джоя** (**Bill Joy**), засновника Sun Microsystems, та **Джеймса Гослінга** (**James Gosling**), начальника Ноутона.

Того дня, коли Ноутон повинен був піти з компанії, було ухвалено рішення про створення команди провідних розробників з тим, щоб вони робили що завгодно, але створили щось незвичайне. Команда з шести осіб приступила до розробки нової об'єктно-орієнтованої мови програмування, яка була названа **Oak** (дуб), на честь дерева, що росло під вікном Гослінга.

Незабаром компанія Sun Microsystems перетворила команду Green, що працювала над секретним проектом під назвою "Green", у компанію First Person. Цей невеликий проект мав на меті створення абсолютно нової платформи для розробки програмного забезпечення, яка вирішувала б багато з існуючих проблем. Після низки невдач компанія анонсувала броузер Mosaic – так народився **World Wide Web**, з якого почався бурхливий розвиток Internet.

Ноутон запропонував використовувати Oak у створенні Internet-додатків. Так Oak став самостійним продуктом. Незабаром був написаний Oak-компілятор та Oak-браузер "**WebRunner**". У 1995 році компанія Sun Microsystems ухвалила рішення оголосити про новий продукт, перейменувавши його в **Java** (єдине розумне пояснення назві – любов програмістів до кави). Коли Java опинилася у руках Internet, стало необхідним запускати Java-аплети – невеликі програми, що завантажуються через Internet. WebRunner був перейменований у **HotJava,** а компанія Netscape встала на підтримку Java-продуктів.

Мова повинна була втілювати наступні якості: простоту та потужність, безпеку, об'єктну орієнтованість, надійність, інтерактивність, архітектурну незалежність, можливість інтерпретації, високу продуктивність та легкість у вивченні. Навіть якщо ви ніколи не напишете жодного рядка на мові Java, знати про її можливості вельми корисно, оскільки саме перераховані вище властивості мови додають динаміку сторінкам Всесвітньої павутини.

Створення мови Java – це дійсно один з найзначніших кроків уперед в області розробки середовищ програмування за останні 20 років. Мова HTML (Hypertext Markup Language – мова розмітки гіпертексту) була необхідна для статичного розміщення сторінок у "Всесвітній павутині" WWW. Мова Java

була потрібна для якісного стрибка при створенні інтерактивних продуктів для мережі Internet.

Вивчаючи Java, ви будете приємно здивовані тим, що її синтаксис близький до синтаксису мови C++. Успадкувавши найкраще від мови програмування C++, мова Java при цьому позбавилася від деяких недоліків C++, внаслідок чого на ній стало простіше програмувати. У цій мові немає, наприклад, покажчиків, які складні у використанні та потенційно можуть бути причиною доступу програми до області пам'яті, яка їй не належить. Немає множинного успадкування та шаблонів, хоча функціональні можливості мови Java від цього не постраждали. Якщо ви вмієте програмувати на C++, для вас не складе особливих труднощів вивчити мову Java.

Чому треба вивчати нову мову програмування Java? Якщо відповісти на це питання стисло, то тому, що вона спеціально орієнтована на найсучасніші технології, які пов'язані з мережею Internet. Популярність Internet, яка постійно зростає, а особливо, серверів Web, створює для програмістів нові можливості для реалізації своїх здібностей.

Величезна перевага Java полягає у тому, що на цій мові можна створювати додатки, які здатні працювати на різних платформах. До мережі Internet підключені комп'ютери найрізноманітніших типів – Pentium PC, Macintosh, робочі станції Sun і таке інше. Навіть у рамках комп'ютерів, що створені на базі процесорів Intel, існує декілька платформ, наприклад, Microsoft Windows версії 3.1, Windows 98, Windows NT, Windows XP. Тим часом, створюючи сервер Web у мережі Internet, ви напевно б хотіли, щоб ним могли користуватися якомога більша кількість людей. ***Сервером*** називають потужний («головний») постійно працюючий комп'ютер разом із програмним забезпеченням, який надає користувачам певні Internet-послуги. У цьому випадку допоможуть додатки Java, які призначені для роботи на різних платформах та не залежні від конкретного типу процесора та операційної системи.

Програми, які складені на мові програмування Java, можна розділити за їхнім призначенням на дві великі групи.

До першої групи відносяться додатки Java, призначені для автономної роботи під управлінням спеціальної інтерпретуючої машини Java (**віртуальна машина Java**). Реалізації цієї машини створені для всіх основних комп'ютерних платформ.

Друга група – це так звані аплети (applets). Аплети є різновидом додатків Java, які інтерпретуються віртуальною машиною Java, вбудованою практично в усі сучасні браузери.

Додатки, що відносяться до першої групи (ми будемо називати їх просто додатками Java), – це звичайні автономні програми. Оскільки вони не містять машинного коду і працюють під управлінням спеціального інтерпретатора, їх продуктивність помітно нижче, ніж у звичайних програм, які створені, наприклад, на мові програмування C++. Проте не слід забувати, що програми Java без перетрансляції здатні працювати на будь-якій платформі, що само по собі має велике значення у плані розробок для Internet.

Аплети Java вбудовуються у документи HTML, що зберігаються на сервері Web. За допомогою аплетів можна зробити сторінки сервера Web динамічними та інтерактивними, тобто забезпечувати діалог між користувачем та Java-додатком за допомогою елементів управління (меню, вікон діалогу тощо). Аплети дозволяють виконувати складну локальну обробку даних, які надходять від сервера Web або введених користувачем з клавіатури. З міркувань безпеки аплети (на відміну від звичайних додатків Java) не мають ніякого доступу до файлової системи локального комп'ютера. Всі дані для обробки вони можуть отримати тільки від сервера Web.

Для підвищення продуктивності додатків Java у сучасних браузерах використовується так звана JIT-компіляція (Just-In-Time compilation – компіляція "по ходу справи", "на льоту"). Коли будь-яка інструкція (або набір інструкцій) Java-процессора виконується вперше, відбувається компіляція відповідних байт-кодів із збереженням скомпільованих кодів у спеціальному буфері. При подальшому виклику тієї ж інструкції замість її інтерпретації відбувається виклик з буфера скомпільованих кодів. Тому інтерпретація відбувається лише при першому виклику інструкції. В результаті загальна швидкість виконання Java-аплета збільшується у кілька разів.

Складні оптимізуючі JIT-компілятори діють ще більш вишукано. Оскільки зазвичай компіляція інструкції йде набагато довше у порівнянні з інтерпретацією цієї інструкції, час її виконання вперше за наявності JIT-компіляції може помітно відрізнятися у гірший бік у порівнянні з чистою інтерпретацією. Тому буває вигіднішим спочатку запустити процес інтерпретації, а паралельно йому у фоновому режимі компілювати інструкцію. Лише після закінчення процесу компіляції при подальших викликах інструкції виконуватиметься її скомпільований код. До цього усі її виклики інтерпретуватимуться. Розроблена Sun віртуальна машина HotSpot здійснює JIT-компіляцію лише тих ділянок байт-кодів, які критичні до часу виконання програми. При цьому по ходу роботи програми відбувається оптимізація скомпільованого коду. Кодом у даному випадку є внутрішнє уявлення програми у вигляді внутрішніх інструкцій.

Мова Java є об'єктно-орієнтованою і постачається разом з достатньо об'ємною бібліотекою класів. Так само як і бібліотеки класів систем розробки додатків на мові C++, бібліотеки класів Java значно спрощують розробку додатків, представляючи у розпорядження програміста могутні засоби вирішення поширених завдань. Тому програміст може більше уваги приділяти вирішенню прикладних завдань, а не таких, як, наприклад, організація динамічних масивів, взаємодія з операційною системою або реалізація елементів призначеного для користувача інтерфейсу.

Загальновідомо, що мова програмування Java є мобільною. Це треба розуміти у тому сенсі, що існує принципова можливість перенесення програм Java на різні платформи.

Проте слід зазначити, що створення додатків, що дійсно працюють на різних платформах, – непросте завдання. На жаль, справа не обмежується

необхідністю перекомпіляції початкового тексту програми для роботи в іншому середовищі. Багато проблем виникає через несумісність програмних інтерфейсів різних операційних систем та графічних оболонок, які реалізовують призначений для користувача інтерфейс.

Програма на мові Java компілюється у спеціальний модуль, що складається з команд віртуального процесора Java. Такий модуль містить **байт-код** (інструкції для абстрактного Java-процессора у вигляді байтових послідовностей команд цього процесора та даних до них) та призначений для виконання Java-інтерпретатором. На сьогодні вже створені перші моделі фізичного процесора, здатного виконувати цей байт-код, проте інтерпретатори Java є на всіх основних комп'ютерних платформах. Зрозуміло, що на кожній платформі використовується свій інтерпретатор, або, точніше кажучи, свій віртуальний процесор Java.

Ще одна проблема, що виникає при перенесенні програм, які створені, наприклад, на мові програмування Сі, полягає у тому, що розмір області пам'яті, яка потрібна для змінних стандартних типів, який є різний на різних платформах. Наприклад, в середовищі операційної системи Windows версії 3.1 змінна типу int у програмі, яка складена на мові С, займає 16 бітів. У середовищі Windows NT цей розмір складає 32 біти. Очевидно, що важко складати програму, не знаючи точно, скільки є біт у слові. При перенесенні програм на платформи з іншою розрядністю можуть виникати помилки, які важко виявити. У мові Java всі базові типи даних мають фіксовану розрядність, яка не залежить від платформи. Тому програмісти завжди знають розміри змінних у своїй програмі.

На даний час технологія Java ще не зовсім стала, йдуть суперечки про єдиний стандарт, при цьому суперечності доходять до рівня судового розгляду. Протягом кількох років тягнулася судова суперечка між корпорацією Microsoft, що спробувала «поліпшити» мову стосовно свого браузеру Internet Explorer, та фірмою Sun, що відстоює свої права на «чисту» Java. Microsoft цю суперечку програла, проте з поразкою не змирилася і як хід у відповідь виступила у 2000 році за ініціативою .NET (читається «дот нет» ", DotNet – у перекладі з англійського "точка Net"). Кінцева мета цієї ініціативи така ж, як у технології Java – забезпечити єдине середовище для компонентного мережного програмування, коли готові модулі (компоненти), реалізовані в різних місцях, можуть завантажуватися по мережі і виконуватися на різних апаратних платформах. При цьому .NET йде далі: передбачається можливість використання не єдиної мови Java, а й інших мов, зокрема розробленої і активно просувної Microsoft перспективної мови C#.

Мабуть, найближчими роками ми зможемо стати свідками «битви гігантів» на полі технологій мережного програмування. Хто б в ній не переміг, у виграші опиняться програмісти, одержуючи інструмент зручної, швидкої та надійної розробки своїх додатків.

8. Історія і шляхи розвитку супер-ЕОМ

8.1. Усе починалося з менфреймов

Історія супер-ЕОМ бере свій початок з мейнфреймов – “великих машин”. **Мейнфрейм** (mainframe) термін походить від двох англійських слів. Main – основний або головний. Frame – рамка, скелет, тобто основа чогось. Мейнфрейм це головний комп’ютер обчислювального центру з великим об’ємом внутрішньої та зовнішньої пам’яті. Він зазвичай призначався для завдань, які вимагають складні обчислювальні операції. Але спочатку під даним терміном розуміли стійку, де знаходився процесор.

Перші покоління комп’ютерів, що займали цілі зали, складалися з величезних стійок, кожна з яких реалізовувала певний компонент комп’ютера. Процесор – займав цілу стійку. Сьогодні весь процесор розміщується на одній інтегральній схемі. Але у той час стійку з процесором називали основною, тобто мейнфреймом.

Розвиток мейнфреймів почався у 60-х роках. На той час корпорація IBM, замисливши над створенням потужних та універсальних ЕОМ, запропонувала ринку комп’ютер System/360. До сьогоднішнього дня безперечним лідером у виробництві мейнфреймов є та ж компанія IBM. Новітні її серії – zSeries (800, 890, 900, 990 та Z9 – 2005 р).

На початку 90-х інтерес до мейнфреймів значно знизився. Їх поступово стали витісняти кластерні системи. Вони дозволяють розподілено зберігати та обробляти дані, що підвищує їх надійність перед мейнфреймами. Підвищити продуктивність розподілених систем, значно дешевше, ніж централізованих. Ці ключові чинники призвели до значного зменшення кількості мейнфреймов в усьому світі. Проте для певного класу завдань, побудова централізованого сховища даних, виправдовує себе як з точки зору продуктивності, так і з економічних міркувань. Тому деякі підприємства свої обчислювальні центри будують на базі мейнфреймів.

Але все ж таки потужнішими на сьогоднішній день залишаються суперкомп'ютери. Їх відрізняє швидкодія, складність конструкції та висока вартість. Вони можуть виконувати мільярди операцій за секунду та зберігати величезні об'єми інформації.

Для незалежних компаній, що не пішли у фарватері IBM, на ринку ЕОМ залишалися тільки дві незаповнені ніші: дуже великі і, навпаки, дуже малі комп’ютери, туди і попрямували конкуренти. В результаті до кінця 60-х років ринок розшарувався на три нерівні сектори:

– **СУПЕР-ЕОМ (supercomputers)**, спеціально спроектовані для особливо складних наукових розрахунків;

– **ЕОМ загального призначення (mainframes)**, призначені для економічних та інженерних розрахунків. Цей сектор ринку був найбільш потужним і заповнений в основному IBM-сумістними моделями;

– **МІНІ-ЕОМ (minicomputers) –** гранично прості та дешеві комп'ютери вартістю менше 100 000 доларів.

Оскільки про ЕОМ загального призначення та МІНІ-ЕОМ було детально розглянуто раніше, у цьому параграфі ми зупинимося на історії створення суперкомп'ютерів.

Створення обчислювальних машин найвищої можливої на даний історичний момент продуктивності завжди було престижною справою для комп'ютерних фірм. Але, крім престижу, для виробництва СУПЕР-ЕОМ були і вагомі економічні причини: в наукових та військових колах постійно з'являються завдання, що пред'являють екстремальні вимоги до параметрів комп'ютерів. Це – складні розрахунки в ядерній фізиці, метеорології, космічній техніці і таке інше. Хоча ринкова ніша для суперкомп'ютерів і є невеликою, але є постійною та прибутковою.

Вперше термін "СУПЕР-ЕОМ" був використаний на початку 60-х років, коли група фахівців університету Ілінойса (США) під керівництвом доктора Д. Слотника запропонувала ідею реалізації першої у світі паралельної обчислювальної системи. Проект, що отримав назву SOLOMON, базувався на принципі векторної обробки, який був сформульований ще Дж. фон Нейманом, з використанням матричної паралельної архітектури, яку запропонував С. Унгер на початку 50-х років.

Річ у тім, що більшість суперкомп'ютерів демонструють продуктивність завдяки саме цьому (векторному) виду паралелізму. Будь-який програміст, розробляючи програми на звичних мовах високого рівня, напевно неодноразово стикався з так званими циклами DO. Але мало хто замислювався, який потенціал збільшення продуктивності полягає в цих операторах, які часто використовуються. Відомий фахівець в області систем програмування Д. Кнут показав, що цикли DO займають менше 4% коду програм на мові FORTRAN, але вимагають більше половини рахункового часу завдання.

Ідея векторної обробки циклів такого роду полягає у тому, що в систему команд комп'ютера вводиться векторна операція, яка працює з усіма елементами векторів-операндів. При цьому реалізуються одразу дві можливості прискорення обчислень. По-перше, скорочується кількість виконуваних процесором команд, оскільки відпадає необхідність у перерахунку індексів та організації умовного переходу, а, по-друге, всі операції складання елементів векторів-операндів можуть бути виконані одночасно через паралелізм обробки. Така можливість реалізована у так званих SIMD-командах, які виконують одну дію над кількома даними.

Лідером з виробництва суперкомп'ютерів в середині 60-х років стала фірма Control Data Corporation (CDC). Відмовившись від конкуренції з IBM в ніші мэйнфремов, CDC під керівництвом талановитого інженера **Сеймура Крея** (Seimour Cray, 1925 – 1996) розробила серію CDC-5000, потім CDC-6000. Машина CDC-6600 цієї серії продуктивністю 3 млн оп./с (або 3 MFLOPS, МегаФлопс). **FLOPS** – акронім від англ. Floating point Operations Per Second

(виголошується як флопс) – величина, яка використовується для виміру продуктивності комп'ютерів, показує, скільки операцій над данними з плаваючою крапкою за секунду виконує така обчислювальна система.

Довгий час вважалася найшвидшою у світі, поки не поступилася пальмою першості CDC-8000 продуктивністю більше 10 млн. оп./с. Для забезпечення віддаленого доступу клієнтів до своїх суперкомп'ютерів фірма створила обчислювальну мережу «Cybernet» з комутацією каналів.

Разом з Control Data на ринку суперкомп'ютерів у 70-і роки фігурували багатопроцесорні моделі STAR-100 (1970 р.) продуктивністю 100 млн. оп./с, а також ЕОМ серії Cyber (моделі 70, 72 та ін.), Берроуз (Burruoghs B-5000) та ін.

Першою СУПЕР-ЕОМ, що використовувала переваги векторної обробки, була ILLIAC IV. На початку 60-х років група все того ж Слотника, об'єднана у Центр передових обчислювальних технологій при університеті Ілінойса, приступила до практичної реалізації проекту векторної СУПЕР-ЕОМ з матричною структурою. Виготовлення машини взяла на себе фірма Burroughs Corp.

Система повинна була складатися з чотирьох квадрантів, кожен з яких включав 64 процесорних елементів та 64 модулів пам'яті, об'єднаних комутатором на базі мережі типа гіперкуб. Один квадрант ILLIAC IV здатний одночасно обробляти 64 елементи вектора, а вся система з чотирьох квадрантів – 256 елементів. У 1972 р. перша система ILLIAC IV була встановлена у дослідницькому центрі NASA в Еймсе. Результати її експлуатації у цій організації отримали неоднозначну оцінку. З одного боку, використання суперкомп'ютера дозволило вирішити низку складних завдань аеродинаміки, з якими не могли впоратися інші ЕОМ. Навіть найшвидша ЕОМ для наукових досліджень того часу – Control Data CDC 7600, яку спроектував "патріарх СУПЕР-ЕОМ" Сеймур Крей, могла забезпечити продуктивність не більше 5 MFLOPS, тоді як ILLIAC IV демонструвала середню продуктивність приблизно в 150-200 MFLOPS. З іншого боку, ILLIAC IV так і не була доведена до повної конфігурації з 256 процесорних елементів; практично розробники обмежилися лише одним квадрантом. Демонтований ILLIAC IV був лише у 1983 році.

У 1976 р. була випущена модель **Cray-1** з продуктивністю 180 мільйонів операцій за секунду над числами з плаваючою крапкою, у 1985 – чотирьохпроцесорна **Cray-2** з швидкодією близько 1 млрд. оп./с. (GFLOPS, ГігаФлопс), на початку 90-х років в моделі **Cray-3** був перевищений поріг продуктивності 10 млрд. оп./с. Впродовж 80-х і першої половини 90-х років фірма Cray, не дивлячись на запеклу конкуренцію з боку молодих фірм, продовжувала лідирувати, але врешті-решт вона зіткнулася з великими фінансовими проблемами після гибелі С. Крея (авіакатастрофа), і була куплена Silicon Graphics Incorporated (SGI).

Комп'ютер Cray-1 (збірка)

Комп'ютер Cray-2

Гонка фірм за престижне право створити наймогутніший комп'ютер продовжується й дотепер, ведуться розробки багатопроцесорних суперкомп'ютерних систем з продуктивністю понад трильйон (10^{12}) операцій з плаваючою крапкою за секунду (TFLOPs, ТераФлопс). Традиційним способом перевірки граничних можливостей комп'ютерів є гра у шахи з людиною. Раніше в таких змаганнях комп'ютер незмінно програвав, якщо його супротивником був сильний шахіст, проте навесні 1997 року відбулася подія, значення якої важко переоцінити: у матчі з шести партій з чемпіоном світу з шахів Гарі Каспаровим суперкомп'ютер "Deep Blue", спеціально побудований для цього фірмою IBM, зумів отримати перемогу. Журналісти із цього приводу написали: "Комп'ютер поставив мат людству".

Суперкомп'ютер IBM Blue Gene

У 2005 році компанія IBM розробила суперкомп'ютер Blue Gene продуктивністю понад 30 трильйонів операцій за секунду. Він містить 12000 процесорів і володіє у тисячу разів більшою потужністю, ніж знаменитий Deep Blue.

Хоча принципове значення СУПЕР-ЕОМ для розвитку інформатики дуже велике, але з економічної точки зору їх присутність на ринку дуже мала. Виробництво суперкомп'ютерів завжди було поштучним, кількість працюючих машин цього класу ніколи не перевищувала декількох сотень (у 1985 р. в світі їх налічувалося 165), ціна кожного перевищувала 10 млн. доларів.

8.2. Напрями розвитку обчислювальної техніки

За шістдесят з лишком років історії обчислювальна техніка зробила гігантський стрибок у своєму розвитку, жодна інша галузь промисловості не розвивалася такими бурхливими темпами. За образним виразом одного з журналістів «якби літаки прогресували з такою ж швидкістю, як ЕОМ, то ми могли б зараз облетіти земну кулю за декілька хвилин, витративши півсклянки пального, а коштував би такий літак всього декілька доларів».

Як приклад можна навести дані про відносну продуктивність основних ліній процесорів, які були випущені фірмою Intel. За зовсім короткий час (27 років, починаючи з 1983) обчислювальна потужність мікропроцесорів збільшилася у 10 тисяч разів!

Приблизно такі ж темпи зростання характеристик інших основних вузлів комп'ютера (оперативної пам'яті, дискових запам'ятовуючих пристроїв). Порівняємо: у першої IBM PC (1981 р.) ємкість ОЗУ складала 64 Кбайт, у сучасного персонального комп'ютера вона дорівнює 2-4 Гбайт та вище, перші покоління "вінчестерських" дисків мали ємкість 5 -10 Мбайт, а сучасні – 400-600 Гбайт. Таким чином, навіть без принципової зміни фізичних принципів функціонування ЕОМ, а тільки за рахунок конструктивних та технологічних удосконалень, відбувається постійне покращення експлуатаційних характеристик комп'ютерів.

Як довго продовжуватиметься цей поступальний розвиток, сказати важко, оскільки довгострокові прогнози у світі високих технологій – заняття ненадійне та небезпечне.

Якщо зупинитися на найскладнішому елементі комп'ютера, що визначає технічний рівень всього виробу, – центральному процесорі – то можна відмітити, що прогрес тут йде двома паралельними шляхами: розвиток елементної бази та вдосконалення архітектури.

8.3. Розвиток елементної бази. Закон Мура

Технологія виготовлення мікропроцесорів розвивається у напрямі подальшої мініатюризації електронних схем і, як наслідок, підвищення ступеня їхньої інтеграції. Рівень мініатюризації прийнято оцінювати кроком конструкційної сітки мікросхеми, що визначає ширину друкованих провідників та розміри інших елементів на кристалі. Передові фірми вже давно засвоїли 0.5-мікронну технологію (це означає, що розміри дискретних елементів складають півмікрона, для порівняння – товщина людського волосу в середньому дорівнює 50 мікронам, а довжина хвилі видимого світла складає 0.4-0.7 мікрона), і наполегливо рухаються до 0.1-мікронної. Елементи такої мікросхеми неможливо побачити в оптичний мікроскоп – тільки в електронний.

Усім цим неймовірними звершеннями комп'ютерної революції ми зобов'язані кільком фізичним принципам [15]. По-перше, комп'ютери рахують з такою разючою швидкістю тому, що електричні сигнали рухаються по дротах зі швидкістю, близькою до швидкості світла.

По-друге, кількість інформації, яку можна передати за допомогою лазерного променя, практично нічим не обмежена. Світлові хвилі переміщуються на багато швидше звукових і можуть нести на собі набагато більше інформації, ніж звук. Уявіть, наприклад, довгу натягнуту мотузку, по якій з одного кінця пускають хвилі. Чим швидше рухається цей кінець, тим більше сигналів вміщується на мотузці. Тому кількість інформації, яку можна

передати при допомозі хвилі, тим більше, чим швидше вона коливається, тобто чим більше її частота.

По-третє – і це найголовніше, – основою комп'ютерної революції є мініатюризація транзисторів. Транзистор – це електронний ключ, або керуючий елемент, контролюючий потік електрики. Якщо порівняти електричний контур з водопроводом, то транзистор – це кран, керуючий потоком води.

Зменшення розмірів дозволяє «упакувати» на одному чипі більшу кількість елементів та ускладнити мікросхему. Ще у 1968 році, на зорі мікроелектроніки, один із засновників фірми Intel **Гордон Мур** сформулював емпіричний «**закон Мура**», за яким кількість елементів на одному кристалі повинна подвоюватися кожні півтора року. Дивно, але факт – пройшло 45 років, невпізнанно перетворилися технології, проте закон продовжує діяти і зараз. Отже відзначимо, що вперше така закономірність була відмічена не Гордоном Муром, як думає багато хто, а великим Дагом Енгельбартом. Саме він зробив прогноз про періодичне подвоєння щільності у 1960 році, тобто на багато раніше появи знаменитої статті Мура.

Спочатку закон Мура сприймався як деякий казус. Він більшою мірою дивував, ніж слугував керівництвом до дії, але з часом коректність закону перестала викликати сумнів, і зараз є впевненість у його дієвості на довгі роки наперед, з ним пов’язують надії на майбутнє. Показово, що про закон Мура широко заговорили лише років десять тому. Як не дивно, до цього він був мало кому відомий, а вже його точне формулювання тим більше. Довгий час його помилково інтерпретували як закон періодичного подвоєння продуктивності, і лише коли продуктивність стала падати, пригадали про початкове формулювання, яке говорить про щільність розміщення **транзисторів** на кристалі.

З переходом на транзистори комп'ютери стрімко зменшилися в розмірах: «обчислювач», який раніше вимагав приміщення розміром з заводський цех, став уміщатися в невеликій кімнаті! Так, в 1955 р. Фірма Bell Laboratories створює перший транзисторний комп'ютер другого покоління TRADIC, що містить 800 транзисторів і 10 000 діодів, а в 1960 р. компанія DEC випустила легендарний «міні»-комп'ютер PDP-1, який уміщався в кутку невеликої кімнати.

Але еволюція комп'ютерів на цьому не зупинилася: розміри транзисторів постійно зменшувалися. Та 12 вересня 1958 р. у електронному світі відбулася нова революція: співробітник компанії Texas Instruments Джек Кілбі створив елементи, що об'єднують безліч транзисторів – **інтегральні схеми**. Перша з них містила всього п'ять транзисторів, пізніше їх кількість збільшувалася і збільшувалася.

Поява інтегральних мікросхем породило новий вид обчислювальних машин Duo комп'ютери третього покоління, ЕОМ, які здатні виконувати 300 млн операцій за секунду.

Але і цьому поколінню комп'ютерів залишалося жити недовго: вже в 1971 р. компанія Intel представила нову розробку, об'єднала на одному кристалі

більше двох тисяч транзисторів. Так народився мікропроцесор – а разом з ним народилося четверте покоління комп'ютерів – те саме, з яким ми і працюємо сьогодні.

У повній відповідності із законом Мура, сучасні мікропроцесори є неймовірно складними пристроями. Наприклад, якщо в першому процесорі Intel 4004 містилося всього лише 2300 транзисторів, то кристал P5 фірми Intel, випущений у 1993 р., який отримав торговельну марку «Pentium», містить близько 3 млн транзисторів, P6 – «Pentium Pro» (1996 р.) – 7,5 млн, а процесор P7 (під час розробки він називався «Merced», а в продаж пішов під ім'ям «Itanium»), випуск якого почався у 2000 р., мав уже близько 20 млн транзисторів, а в сучасних Core 2 Duo (65 нм технологія, 2006 р.) – Core i7-3820 Duo (32 нм технологія, 2012 р.) їх вже від 290 млн до 2.7 млрд. Для наочності пояснимо, що на кінчику людської волосини може помістится більше 1000 транзисторів.

Зменшення розмірів деталей та довжин провідників, що з'єднують їх, дозволяє поліпшити ще одну характеристику мікропроцесора, яка пропорційно впливає на його продуктивність – тактову частоту. Якщо у чипа i4004 вона дорівнювала 108 кГц, то сучасні схеми припускають збільшення тактової частоти до 2000-3000 Мгц, тобто 3 Ггц, і це ще не межа.

До тих пір, поки складність процесорів не почала стримувати подальший розвиток, закон Мура сприймався виключно як гарант прогресу, проте сьогодні складність стала серйозною проблемою. Ставлення до закону Мура та його місця у комп'ютерній історії потребує переосмислення [Леонид Черняк. Архитектура фон Неймана, реконфигурируемые компьютерные системы и антимашина. Открытые системы, №06, 2008]. Загалом в еволюційному процесі закон Мура грає подвійну роль, будучи як гальмом, так і стимул-реакцією одночасно. Гальмом – або, швидше, консервуючим чинником – закон Мура можна рахувати тому, що він впродовж багатьох років дозволяв зберігати динаміку розвитку, обходячись без радикальних інноваційних кроків. Працював своєрідний позитивний зворотний зв'язок – із збільшенням кількості транзисторів розробники процесорів компенсували зниження якості архітектури. Як результат продуктивність продовжувала зростати, але повільніше, ніж число транзисторів. Подальший розвиток напівпровідникових технологій за законом Мура не може компенсувати неефективність сучасних процесорів. Але з іншого боку, кількісне зростання, яке постулюється законом Мура, зумовлює неминучі зміни, і в якийсь момент повинен спрацювати діалектичний перехід кількості в якість.

Закон Мура так невпинно забезпечував нам експоненціальне зростання потужності комп'ютерів, зокрема, тому, що виробники мікросхем, відпрацьовуючи технологію, поступово зменшували довжину хвилі УФ-випромінювання, що дозволяло їм витравлювати на кремнієвих пластинках все більш і більш крихітні транзистори. Довжину хвилі УФ-випромінювання можна довести до 10 нм (нанометр – це одна мільярдна частина метра), і найменший

транзистор, який можна витравити на підкладинці таким способом, буде близько тридцяти атомів у поперечнику [15].

Але цей процес не може тривати до нескінченності. Можна навіть прикинути, коли приблизно завалиться закон Мура: в той момент, коли подальша мініатюризація зажадає робити транзистори розміром з окремий атом. Таким чином, ми зіткнемося з тим, що витравити таким способом транзистор розміром з один атом фізично неможливо.

Близько до 2020 року або трохи пізніше закон Мура поступово перестане діяти; якщо не буде знайдена нова технологія, здатна замінити нинішню і забезпечити подальший прогрес [15]. Згідно законам природи з часом Кремнієва ера закінчиться і почнеться Посткремніева. Транзистори стануть такими маленькими, що на сцену вийдуть квантова теорія або атомна фізика.

8.4. Вдосконалення архітектури

На жаль, нескінченно зменшувати розміри елементарних схем перемикачів неможливо, оскільки вони обмежені знизу розмірами кристалічних решіток. Так само не можна безмежно підвищувати тактову частоту, оскільки швидкість розповсюдження електричного струму кінцева.

Мабуть, найближчим часом елементні можливості мікросхем підійдуть до теоретичної межі, подальше підвищення продуктивності комп'ютерів досягатиметься тільки за рахунок вдосконалення архітектури, яке розвивається у чотирьох основних напрямах.

1. Збільшення розрядності. Тенденція до підвищення розрядності виразно простежується в історії мікропроцесорів. Сучасні кристали в основному 32-розрядні, проте просунуті мікросхеми, наприклад, POWERPC, а також перспективні масові моделі, наприклад, "Merced" – 64-розрядні. Мабуть, у майбутньому можна чекати і появи 128-бітових чипів.

2. Рух убік RISC. Абревіатура **RISC** означає Reduced Instruction Set Computer (комп'ютер із скороченим набором команд). Вважається неправильним Reduced Instruction Set Computing – обчислення із скороченим набором команд. Для того щоб зрозуміти сенс цього явища, потрібно повернутися до ранньої історії ЕОМ. У ті часи алгоритмічні мови і компілятори ще не були відомі, і все програмування велося вручну, у командах процесора. Тому розробники комп'ютерів прагнули зробити систему команд зручною для ручного програмування, наситивши її складними та місткими командами. Наприклад, однією машинною командою можна було обчислити функцію log або sin, або перетворити число в іншу систему числення. Репертуар машинних команд виходив достатньо складним, наприклад, в IBM-360 були реалізовані 144 команди центрального процесора. Така організація системи команд отримала назву **CISC –** Complex Instruction Set Computing, тобто обчислення з складним набором команд.

Засновником CISC-архітектури можна вважати компанію IBM з її базовою архітектурою /360, ядро якої використовується з 1964 року і дійшло до наших

днів, наприклад, в таких сучасних мейнфреймах як IBM ES/9000. Лідером у розробці мікропроцесорів на повний набір команд CISC вважається компанія Intel зі своєю серією x86 та Pentium (виключаючи сучасні Intel Pentium 4, Pentium D, Core, AMD Athlon, Phenom, які є гібридними), а також процесори Motorola MC680x0. Ця архітектура є практичним стандартом для ринку мікрокомп'ютерів.

Для CISC-процессоров характерне:

- порівняльно невелика кількість регістрів загального призначення;
- велика кількість машинних команд, деякі з яких навантажені семантично аналогічно операторам високорівневих мов програмування і виконуються за багато тактів;
- велика кількість методів адресації;
- велика кількість форматів команд різної розрядності;
- використання переважно двоадресного формату команд;
- наявність команд обробки типа регістр-пам'ять.

Стандартний набір команд чипа i8086 та всіх подальших поколінь процесорів Intel містить близько ста інструкцій найрізноманітнішого призначення та формату. Оскільки формат команди змінний, то вона може бути коректно вибрана з пам'яті тільки після розшифровки коду операції. В результаті кожна інструкція вимагає для свого виконання кілька тактів процесора. Програма, яка реалізовує деякий алгоритм, може бути короткою, проте час виконання цієї програми у комп'ютері виявляється значним.

Ідея створити RISC-процесор прийшла IBM. Ще у 70-х роках вони відмітили, що багато архітектурних та функціональних особливостей не використовуються розробниками програмного забезпечення. До того ж, одні з перших компіляторів мов високого рівня не використали багато інструкцій стандартних процесорів. Як наслідок з цього, програми, які були написані на таких компіляторах, нераціонально використовували процесорні ресурси.

Процесори з **RISC**-архітектурою працюють по-іншому. У цих процесорах набір команд сильно обмежений, всі інструкції максимально спрощені, вони мають однаковий формат і, в ідеалі, можуть виконуватися за один машинний такт. Програма, що виконує твкий самий алгоритм примітивними командами, виходить довшою, проте за рахунок високої швидкодії процесора спостерігається значний виграш у продуктивності. Зрозуміло, що програмувати вручну для такої машини було б незручно, проте цього ніхто і не робить, оскільки техніка компіляції досягла великих висот. Швидкодіючі оптимізуючі компілятори дозволяють створити такий код, який використовує всі особливості набору команд і дозволяє досягнути найвищої обчислювальної потужності.

Паростки цієї RISC-архітектури беруть початок своїм корінням від комп'ютерів CDC6600, розробники яких (Торнтон, Крей та ін.) усвідомили важливість спрощення набору команд для побудови швидких обчислювальних машин. Цю традицію спрощення архітектури С. Крей з успіхом застосував при

створенні широко відомої серії суперкомп'ютерів компанії Cray Research. Проте остаточне поняття RISC-процессоров у сучасному його розумінні сформувалося на початку 1980-х років на базі трьох дослідницьких проектів університету Берклі, Стенфордському та Каліфорнійському університетах США. Процесори 801 компанії IBM, процесори RISC університету Берклі та процесори MIPS Стенфордського університету виконували невеликий (50-100) набір команд, тоді як звичайні CISC-процессори виконували 100-200 команд.

Прихильники RISC-архітектури на ділі довели силу своїх аргументів – найпродуктивніши сервери і робочі станції сьогодні використовують RISC-процесори, проте і прихильники CISC-технології не здаються, на їх боці гігантський об'єм накопиченого програмного забезпечення у кодах ix86. В останніх моделях мікропроцесорів Intel спеціально для мультимедійних додатків введені ще складніші «векторні» команди додаткового набору MMX (MultiMedia eXtention – мультимедійне розширення) та SSE (Streaming SIMD Extensions – потокове SIMD-розширення), що виконують у наддовгих, – до 128 розрядів – регістрах кілька паралельних операцій додавання або множення. SIMD (Single Instruction Multiply Data) – архітектура обчислювальної системи, яка одним потоком команд обробляє кілька потоків операндів або пар операндів.

Характерні особливості RISC-процесорів.

1. Фіксована довжина машинних інструкцій (наприклад, 32 біта) та простий формат команди.

2. Спеціалізовані команди для операцій з пам'яттю – читання або записи. Операції типу «прочитати-змінити-записати» відсутні. Будь-які операції «змінити» виконуються лише над вмістом регістрів.

3. Велика кількість регістрів загального призначення (32 і більш у порівнянні з 8–16 регістрами у CISC-архитектурах), що дозволяє більшій кількості даних зберігатися у регістрах на процесорному кристалі більший час і спрощує роботу компілятора щодо розподілу регістрів під змінні.

4. Відсутність підтримки операцій типу «змінити» над скороченими типами даних – байт, 16-бітове слово. Так, наприклад, система команд DEC Alpha містила лише операції над 64-бітовими словами, і вимагала розробки та подальшого виклику процедур для виконання операцій над байтами, 16- та 32-бітовими словами.

5. Відсутність мікропрограм всередині самого процесора. Те, що у CISC-процесорі виконується мікропрограмами, у RISC процесорі виконується як звичайний (хоча і розміщений в спеціальне сховище) машинний код, що не відрізняється принципово від коду ядра ОС та додатків.

Для того щоб об'єднати переваги обох підходів, в останніх розробках компанії Intel (маються на увазі Pentium та Pentium Pro), а також її послідовників-конкурентів (AMD R5, Cyrix M1, NexGen Nx586 і ін.) широко використовуються ідеї, які реалізовані у RISC-микропроцессорах, на ходу перетворюючи CISC команди у набір RISC-команд, і виконуючи їх на своєму RISC-ядрі (**гібридна архітектура**). На зовнішньому рівні мікропроцесор

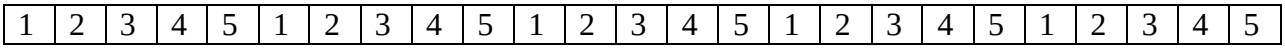
виконує стандартний CISC-набір команд, а на внутрішньому – деякий спрощений RISC. Вбудований мікропрограмний емулятор перетворить кожну зовнішню команду у низку внутрішніх, і виконує її з усією можливою продуктивністю RISC-рахівника.

3. Ускладнення архітектури процесора. Ще один резерв підвищення продуктивності криється у розпаралелюванні обчислень всередині одного кристала, при цьому розробники мікросхем намагаються реалізувати в конструкції принципи, які є типовими для організації промислового виробництва.

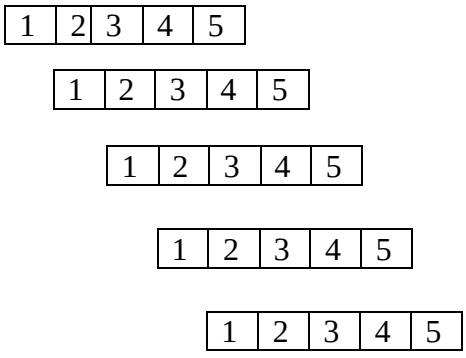
Як відомо, кожна машинна операція складається з кількох фаз: вибірка команди, розшифровка її, читання операндів, безпосередньо виконання операції, запис результату. У старих моделях процесора ці фази виконувалися для кожної операції строго послідовно подібно до того, як в кустарних майстернях йшла колись зборка автомобілів – спочатку збирали одну машину, потім другу, при цьому частина робітників постійно простоювала.

Сучасний мікропроцесор влаштований значно складніше, він схожий на підприємство, в якому безліч робітників збирають на конвеєрі потік автомобілів. Конвеєрний процесор суміщає за часом виконання кількох команд: для однієї відбувається читання операції, для другої – декодування та вибірка регістрів, для третьої – виконання команди обчислювальним блоком і так далі В результаті при тій самій тактовій частоті істотно підвищується загальна продуктивність.

На малюнку нижче показано звичайну, послідовну обробку команд та конвеєрну обробку. Для спрощення показано п’ять машинних команд, у яких є п’ять машинних циклів, що пронумеровані



Звичайна послідовн обробка



Конвеєрна обробка

Як бачимо з цього прикладу час обробки п’яти команд скорочується майже втричі (за дев’ять машинних циклів). Кількість таких конвеєрів визначає глибину конвеєризації. Сучесні комп’ютери мають глибину конвеєризації у кілька десятків.

Більш того, у найдосконаліших конструкціях в чіп мікропроцесора вбудовується декілька самостійних (до 6-8) обчислювальних блоків з фіксованою та плаваючою арифметикою, надшвидка внутрішня пам'ять (кеш) та дивовижний пристрій управління. Цей пристрій пророчить можливий розвиток обчислювального процесу на декілька кроків уперед, і з урахуванням цього організує паралельну роботу кількох конвеєрів. Неминуча плата за таку організацію – значне підвищення складності та вартості схеми.

4. Багатопроцесорні конфігурації. Неважко прийти до висновку, що, якщо не можна використовувати можливості закону Мура на одному ядрі через виняткову складність такого ядра, то слід піти шляхом збільшення числа ядер. Саме так поступили в Sun Microsystems, випустивши 8-ядерний процесор Niagara. Під ядром розуміється процесорний елемент разом із власною кеш-памяттю.

Тобто, коли можливості одного кристала вичерпані, продуктивність комп'ютера у цілому може бути збільшена за рахунок багатопроцесорної організації. Аналіз реальних додатків показує, що довгі ланцюжки машинних команд, які повинні виконуватися строго послідовно, зустрічаються відносно рідко, в основному в наукових розрахунках. Як правило, обчислювальний процес можна розбити на декілька паралельних гілок, кожну з яких доцільно доручити самостійному мікропроцесору.

Зараз можна говорити про дві тенденції, що помітно різняться між собою у процесі збільшення кількості ядер. Одну зараз називають *мультиядерність* (multi-core), у цьому випадку передбачається, що ядра є високопродуктивними та їх відносно небагато. Зараз їхня кількість – два-чотири, і згідно закону Мура вона періодично має подвоюватися. Цей шлях має два основні колишні недоліки: перший – високе енергоспоживання, другий – висока складність чіпа і, як наслідок, низький відсоток виходу готової продукції. При виробництві 8-ядерного процесора IBM Cell лише 20% кристалів, які виробляються, є придатними. Інший шлях – *багатоядерність* (many-core). У такому випадку на кристалі збирається на порядок більша кількість, але при цьому простих ядер, які споживають мілівати. В осяжному майбутньому з'являться процесори з тисячами та десятками тисяч ядер.

В принципі кількість процесорів у комп'ютері нічим не обмежена. Відомі конструкції з сотнями і навіть тисячами процесорів, проте сумарна продуктивність багатопроцесорної системи зростає далеко не лінійно з кількістю процесорів, оскільки у кожній програмі є деяка межа розпаралелювання, до того ж у багатопроцесорних системах різко зростають накладні витрати на диспетчеризацію обчислювального процесу. Практика показала, що на стандартних комерційних завданнях продуктивність системи зростає як sqrt(n), тобто чотирипроцесорна конфігурація лише у два рази продуктивніше однопроцесорної, проте на спеціальних завданнях, що припускають багаторазове розпаралелювання (наприклад, аналіз варіантів ходу у шаховій партії), багатопроцесорні комп'ютери можуть показувати рекорди продуктивності.

8.5. Топ-500 – рейтинг супер-ЕОМ

На закінчення цього розділу наведемо деякі дані по розробках нових супер-ЕОМ. На першу половину 2008 року найшвидшим суперкомп'ютером на землі був BlueGene L з піковою продуктивністю в 478,2 Терафлоп (478,2 трильйона операцій за секунду). Він очолював рейтинг **Топ-500** (Top-500). Топ-500 (www.top500.org) – рейтинг п'ятисот найпотужніших супер-ЕОМ, що оновлюється кожні півроку.

Перші петафлопові (1 петафлопс – квадриллион (10^{15}), або тисяча трильйонів, операцій за секунду) суперкомп'ютери з'явилися у цьому списку у червні 2008 року. Згідно з рейтингом суперкомп'ютерів за станом на листопад 2008 року лідирував суперкомп'ютер Roadrunner з Аламосской національної лабораторії у Нью-Мексіко (США) з продуктивністю, яка дорівнювала 1105 Терафлоп.

Табл. 8.1. Десятка найпотужніших комп'ютерів з TOP-500

Топ	**Rmax (Tflops)**	**Назва**	**Комп'ютер Число процесорних ядер**	**Виробник**	**Країна, рік**
1	1759.00	Jaguar	Cray XT5 224162 (Opteron)	Cray	США, 2009
2	1271.00	Nebulae	Dawning TC3600 Blade System 120640 (Xeon)	Dawning	Китай, 2010
3	1042.00	Roadrunner	BladeCenter QS22/LS21 122400 (Cell/Opteron)	IBM	США, 2009
4	831.70	Kraken	Cray XT5 98928 (Opteron)	Cray	США, 2009
5	825.50	JUGENE	Blue Gene/P Solution 294912 (POWER)	IBM	Німеччина, 2009
6	772.70	Pleiades	Altix ICE 81920 (Xeon)	SGI	США, 2010
7	563.10	Tianhe-I	NUDT TH-1 71680 (Xeon)	NUDT	Китай, 2009
8	478.20	Blue Gene/L	eServer Blue Gene Solution 212992 (POWER)	IBM	США, 2007
9	458.61	Intrepid	Blue Gene/P Solution 163840 (POWER)	IBM	США, 2007
10	433.50	Red Sky	Sun Constellation System 42440 (Xeon)	Sun	США, 2009

На 22 жовтня в списку Топ-500 першу десятку місць займали найбільш «круті» у світі машини петафлопсного діапазону. 291 машини з цього топового

списку працювали у США, причому американські машини займали перші два місця рейтингу. Третій за потужністю суперкомп'ютер у світі належав Об'єднаній Європі, яка на той час мала 140 топових супер-ЕОМ. З четвертого по дев'яте місця – знов у США. У Китаї – 21 машина (найпотужніша зійшла з конвеєра 16 вересня 2008 – «Шугуан 5000А», продуктивністю більше 100 трильйонів операцій за секунду), в Японії було 15 суперобчислювачів, найпотужніший з них – на 22-му місці. Росія вище за 36-е місце в цьому списку ніколи не піднімалася. У неї на той час було лише п'ять машин з Топ-500. Найпотужніший суперкомп'ютер в Росії – МВС-100К з Міжвідомчого суперкомп'ютерного центру РАН – займав 36-й рядок світового рейтингу (71 Терафлоп).

В наведеній таблиці представлена перша десятка 35-ої за рахунком редакції списку Топ-500 за станом на червень 2010.

Новини про китайську розробку з'явилися у жовтні 2010 року і, як результат створення цієї системи, попередній лідер списку Топ-500 – Cray XT5 "Jaguar", який був створений для міністерства енергетики США і має продуктивність 1,75 петафлоп, тепер займає друге місце. Третє місце займає китайський суперкомп'ютер Nebulae.

Географічно суперкомп'ютери розподілилися таким чином: 274 суперкомп'ютерів знаходяться у США (282 у попередньому 35-му списку Топ-500 за червень), 124 в Європі (144 раніше), 84 (збільшення з 57) в Азії. У порівнянні з минулим випуском кількість суперкомп'ютерів в Росії не змінилося і як раніше складає 11 систем. Загальний розподіл по кількості суперкомп'ютерів в різних країнах виглядає таким чином (у дужках зміна у порівнянні з минулою редакцією рейтингу) :

- США: 274 (- 8);
- Китай: 41 (+17, Китай вирвався з 5 на 2 місце);
- Франція: 26 (- 1);
- Німеччина: 26 (+2);
- Японія: 26 (+8);
- Велика Британія: 25 (- 13);
- Росія: 11;
- Канада: 6 (- 1);
- Італія: 6 (- 1);
- Польща: 6 (+1);
- Саудівська Аравія: 6 (+2);
- Швеція: 6 (- 2).

Представники Top-500 оголосили про вихід 36-го (листопад 2010 г) видання списку 500 найшвидших суперкомп'ютерів планети, який підтвердив інформацію про те, що Китай побудував найпотужнішу на сьогодні систему Tianhe-1A, працюючу на основі шестиядерних центральних процесорів Intel X5670 та графічних прискорювачів NVIDIA Tesla. Продуктивність цього суперкомп'ютера сягає **2.57 петафлоп**, він оснащений **229 Терабайт ОЗУ**.

Новини про китайську розробку з’явилися у жовтні 2010 року і, як результат створення цієї системи, попередній лідер списку Top-500 – Cray XT5 " Jaguar", створений для міністерства енергетики США, і який має продуктивність 1,75 петафлоп, тепер займає друге місце. Третє місце займає китайський суперкомп’ютер Nebulae.

У червні 2011 року опублікована 37-а редакція Top500 – список 500 найпотужніших комп’ютерів світу. У гонці суперкомп’ютерних потужностей знову змінився лідер. Японський суперкомп’ютер **K** став найпотужнішим у світі, обігнавши своїх найближчих переслідувачів з Китаю та США. Він досяг швидкості роботи у **8,162 квадрильйона (петафлоп)** операцій за секунду, повідомляє його виробник Fujitsu.

Зараз комп’ютер складається з 672 спеціальних комп’ютерних корпусів, в яких знаходяться 68 544 процесори. Зрештою кількість процесорів збільшиться до 80 тис., а продуктивність – до 10 петафлопс.

Випробування, що проводились 7-8 жовтня 2011 року, підтвердили передбачувані результати, що і повідомляється на сайті компанії Fujitsu, розробника K комп’ютера. Тест Linpack, який традиційно застосовується для оцінки продуктивності суперкомп’ютерів, дав результат в **10,51 петафлопс**. Таким чином, **K** комп’ютер побив власний рекорд продуктивності.

K комп’ютер у його нинішній конфігурації об’єднує 864 обчислювальних стойки та 88128 процесорів. Теоретична продуктивність системи складає 11,28 петафлопс.

Та все ж, найпотужніший суперкомп’ютер почав свою роботу у 2012 році. За повідомленням Оукриджской національної лабораторії (ORNL) Міністерства енергетики США (DOE) у 2012 році був введений у дію новий суперкомп’ютер – система **Titan**.

Проектуванням обчислювального комплексу займалася компанія Cray. Установка суперкомп’ютера здійснювалася у два етапи: складання першої черги (кінець 2011 року), а друга черги – 2012 рік.

Очікувалося, що швидкодія "Титану" складе до **20 квадриллионов операцій з плаваючою крапкою за секунду (20 петафлоп)**. Розрахункова вартість комплексу – біля $100 млн; використати його планується для проведення складного моделювання енергетичних систем.

Далі ми представимо ТОП-10 найпотужніших суперкомп'ютерів за версією 2013 року.

1. На першому місці, як і очікувалося, знаходиться суперкомп'ютер Titan. Цей мощній і дорогий суперкомп'ютер був створений за участю компаній Cray і Nvidia. Суперкомп'ютер знаходиться в Національній лабораторії Оук-Рідж, що в Теннесі. За одну секунду цей суперкомп'ютер може виконати до 17.58 квадрильйона операцій з плаваючою крапкою. Його продуктивність орієнтовно еквівалентна 17.59 петафлопс.

2. На другому місці розташувався суперкомп'ютер під назвою **Sequoia**. Він був створений співробітниками Ліверморської національної лабораторії (США),

де він, власне, і знаходиться. Потенціал цього комп'ютера використовується для симуляції роботи ядерної зброї, адже його створення дозволило проводити ядерні випробування не на місцевості, а у віртуальному просторі. Його приблизна продуктивність дорівнює 16.32 петафлопс.

3. Трійку лідерів замикає японський суперкомп'ютер – «**K computer**», який був зібраний компанією Fujitsu. Комп'ютер знаходиться в будівлі Інституту фізико-хімічних досліджень (місто Коба). Потужності цього суперкомп'ютера використовуються для моделювання катастроф і прорахунків гіпотетично можливих землетрусів. Його продуктивність складає 10.51 петафлопс.

4. На четвертому місці суперкомп'ютер, який був зібраний в США компанією IBM. Комп'ютер був зібраний в 2011 році і знаходиться в Арагонській національній лабораторії. Його потенціал використовується вченими для оцінки кліматичних змін, а також моделювання електрокарів і оцінки розвитку нашого Всесвіту. Приблизна продуктивність 8.16 петафлопс.

5. На п'ятому місці закріпився німецький суперкомп'ютер, який був зібраний в Німеччині в 2012 році. На даний час комп'ютер перебуває в Дослідницькому центрі Юліха. Головним завданням комп'ютером є моделювання різних процесів, у число яких входять і кліматологія, і фізика. За версією TOP-500 даний суперкомп'ютер є найпотужнішим суперкомп'ютером в Європі.

6. У золоту середину вписався **SuperMUC**, зібраний фахівцями з IBM, продуктивністю на рівні 3 петафлопс. Цей суперкомп'ютер містить 324 терабайта DDR3 пам'яті, а в якості ОС на комп'ютер встановлена Linux.

7. На сьомому місці розташувався суперкомп'ютер Техаського університету (США) продуктивністю на рівні 2-3 петафлопс, 280 терабайт оперативної пам'яті і близько 20 петабайт дискового простору.

8. На восьмому місці знаходиться китайський суперкомп'ютер **Tianhe-1A**. Комп'ютер складається з 103 стійок і важить близько 160 тонн. Комп'ютер знаходиться в Національному університеті оборонних технологій Китаю. До складу суперкомп'ютера увійшли 7168 графічних процесорів від компанії Nvidia і 14556 центральних процесорів від компанії Intel. Продуктивність його – 2.507 петафлопс.

9. На дев'ятому місці ще одне творіння компанії IBM – суперкомп'ютер **Fermi**. Цей суперкомп'ютер розташований в Італії. Він вільно доступний для потреб 54 італійських університетів. Його продуктивність складає 2.1 петафлопс.

10. На десятому місці французький суперкомп'ютер розроблений і зібраний компанією Bull. Загальна кількість ядер – 80 000. Пікова продуктивність цього суперкомп'ютера наближається до показника в 2 петафлопса.

У Росії будується найпотужніший суперкомп'ютер в Європі **МВС-10П**, який повинен у 3 рази збільшить продуктивність обчислювального комплексу Міжвідомчого суперкомп'ютерного центру Російської академії наук (МСЦ РАН).

На квітень 2013 р. в МСЦ створений прототип майбутнього комп'ютера потужністю 523,8 Тфлопс. Він складається з 208 обчислювальних вузлів, кожен з яких включає по два 8-ядерних процесора (всього 3,3 тис. ядер). У 2013 р. планується довести продуктивність системи до 2 петафлопс.

Найпотужнішим на 2013 р. російським суперкомп'ютером є «Ломоносов», розташований в МДУ. В останній редакції рейтингу він посідає 26-місце. Його максимальна продуктивність складає близько 920 Тфлопс. У комп'ютері стоїть 78,6 тис. ядер.

Згідно списку рейтингу 2013 р. в США встановлено п'ять з десяти найбільш продуктивних обчислювальних систем в світі. На Європу припадає 105, Азію – 124, обидві Америки – 264 суперкомп'ютера зі списку.

Час іде і все змінюється. Вже на кінець 2013 р. (42-й випуск рейтингу був оприлюднений 18 листопада 2013 р.) ТОП-10 найпотужніших суперкомп'ютерів очолив ктайский суперкомп'ютер **Tianhe-2** (Тяньхэ – Чумацький шлях), розроблений і виготовлений китайським Національним університетом оборонних технологій. Спочатку планувалося закінчити проект в 2015 році, але його вдалося запустити достроково до кінця 2013 року.

Tianhe-2 состоит из 16 тысяч узлов, каждый из которых включает в себя 2 процессора Intel Xeon E5-2692 12 ядрами каждый (частота 2,2 ГГц) и 3 специализированных сопроцессора Intel Xeon Phi31S1P по 57 ядер на ускоритель (частота 1,1 ГГц), который в тесте Linpack продемонстрировал производительность **33.86 петафлоп** за секунду (квадрильонов операций з плаваючею крапкой за секунду).

Японські вчені також планують в 2020 році створити новий суперкомп'ютер, здатний виконувати один квінтильйон (мільйон трильйонів) операцій з плаваючою крапкою за секунду (http://www.unn.com.ua).

Новий суперкомп'ютер буде в 30 разів перевищувати потужність самого потужного на даний момент суперкомп'ютера, китайської системи Tianhe-2. Такі суперкомп'ютери знайдуть своє місце в областях розробки лікарських препаратів, дослідження в галузі астрофізики, моделювання мозку та інших сферах, де потрібні дуже складні математичні обчислення.

9. Сучасний стан та перспективи розвитку програмування

В середині 70-х років у розвитку обчислювальної техніки почався якісно новий етап, який пов'язаний із справді революційними науково-технічними досягненнями в області фізики твердого тіла. В результаті цих досягнень були створені та технологічно засвоєні великі та надвеликі (у надмалих вимірюваннях) інтегральні схеми, що забезпечують масове виробництво надійних мікромініатюрних вузлів, блоків та окремих пристроїв, зокрема мікропроцесорів як технічної бази нового покоління ЕОМ.

Сьогодні застосування мікропроцесорної техніки у багато разів понизило вартість та зменшило розміри обчислювальних пристроїв, підвищило надійність та спростило умови експлуатації ЕОМ. Значно вдосконалилися технічні засоби спілкування людини з машиною, подальший розвиток отримали мови програмування, з'явилася безліч пакетів прикладних програм, які були розроблені програмістами-професіоналами для використання їх як іншими програмістами, так і непрофесійними користувачами ЕОМ.

З того, що вже нам відомо, повинно бути зрозумілим, що інформатика є дуже могутнім знаряддям, яке кидає виклик нашому інтелекту. Велика складність цієї дисципліни полягає у тому, що обчислювальна машина дає засіб не боятися завдань таких розмірів, які перевершують все що раніше розглядалося, а, отже, такої складності, до засвоєння якої людський розум погано підготовлений. Потреба у такому точному описі крок за кроком послідовності перетворень інформації з такою строгістю, яка **не потрібна ні при якому виді людської діяльності (навіть у математиці)**, викликала у видатних програмістів думки про складність програмування, що вимагає не стільки засвоєння конкретної мови, скільки нового способу мислення.

Сьогодні нескладно простежити новий технологічний виток, в рамках якого спостерігається наплив візуальних середовищ розробки. У більшості випадків вони фактично не вимагають навичок програмування, за рахунок чого залучають величезні армії користувачів різного рівня підготовки.

Для роботів і DSP (*Digital signal processing – Цифрова обробка сигналів*) вже фактично брендовою є середовище DSP Robotics Flowstone, в області веба ми можемо спостерігати величезну кількість CMS (*Content Management System – Система керування вмістом*), а якщо говорити про конструктори, то кращий і популярний – британський Serif Web Plus, у звукових технологіях дуже багато продукції створюється в Synth Edit і Synth Maker, розробка ігор зараз переживає чергову революцію, при цьому інтерактивні мультимедійні додатки можна легко і швидко створювати в Scirra Construc, Multimedia Fusion і так далі. При цьому зверніть увагу на те, що ми назвали ніші, в яких десять років тому були потрібні серйозні програмісти.

9.1. Криза у програмуванні

Розвиток програмного забезпечення на даний час характеризується кризовим станом, який спостерігається вже декілька років. Витрати на розробку програмного забезпечення складають основну частину вартості обчислювальних систем. Для багатьох систем ця цифра сягає 90% і, як очікується, надалі може зрости ще більше [2].

Існує серйозний дефіцит програмістів. Враховуючи, що кількість комп'ютерів щорічно зростає, і, якщо нічого не робити для підвищення продуктивності програмування, то програмістів буде потрібно все більше і більше. Кількість додатків у сучасних центрах електронної обробки даних щорічно зростає на 45%, а це означає, що продуктивність праці програмістів необхідно підвищити на два порядки [2].

Продуктивність роботи програмістів, які писали на таких мовах як КОБОЛ, ФОРТРАН, PL/1, Pascal, **C**, дійсно підвищувалася, але дуже повільно. Зрозуміло, що ключовим завданням зараз є забезпечення програміста такими інструментами, які зроблять ефективнішим процес розробки програм. Стійке зростання витрат на програмування при відносному скороченні їх на технічні засоби породило на Заході так звану "софтверну кризу" (тобто програмну кризу), сутність якої полягає у зростаючому відставанні продуктивності праці програмістів від продуктивності праці в індустрії ЕОМ.

Одним з основних інструментів, що дозволяють підвищити продуктивність праці програмістів, є мови програмування високого рівня. Сучасні мови Pascal, **C**, **C++**, які є найпоширенішими, здійснюють сильний вплив на мислення програміста та можливості програмування, і в цьому сенсі не придатні для **кінцевого користувача** (непрограміста). Щоб полегшити процес програмування, довелося створити нову мову, складнішу у порівнянні з його попередником (**C++).** Очевидно, що застосування **C++** само по собі не гарантує кращого або простішого ПЗ. Щоб скористатися перевагами даної мови, необхідно прийняти нову методологію програмування, так зване **об'єктно-орієнтоване програмування** (ООП).

9.2. Методологія процедурно-орієнтованого програмування

Якщо спробувати охарактеризувати сучасний рівень розвитку комп'ютерних та інформаційних технологій, то перше, на що треба звернути увагу – це зростаюча складність не тільки окремих фізичних та програмних компонентів, але і тих, що лежать в основі цих технологій, концепцій та ідей. Здається, ще зовсім нещодавно професійному програмістові було достатньо досконало оволодіти однією-двома мовами програмування, щоб розробляти серйозні програмні додатки. Вибір платформи та операційної системи, як правило, не був серйозною проблемою. А супровід програми, хоча і був

пов’язаний з об’єктивними труднощами, міг бути реалізований простим додаванням або зміною коду початкової програми.

Поява перших електронних обчислювальних машин або комп’ютерів ознаменувала новий етап у розвитку техніки обчислень. Здавалося, що досить розробити послідовність елементарних дій, кожну з яких перетворити у зрозумілі комп’ютеру інструкції, і будь-яке обчислювальне завдання може бути вирішене. Ця ідея виявилася настільки життєздатною, що довгий час домінувала над усім процесом розробки програм. З’явилися спеціальні мови програмування, які дозволили перетворювати окремі обчислювальні операції у відповідний програмний код.

Основою даної методології розробки програм була **процедурна** або алгоритмічна організація структури програмного коду. Це було настільки природно для вирішення обчислювальних завдань, що ні у кого не викликала сумнівів доцільність такого підходу. Початковим поняттям цієї методології було поняття ***алгоритму,*** під яким, у загальному випадку, розуміється деяке розпорядження виконати точно певну послідовність дій, які спрямовані для досягнення заданої мети або вирішення поставленої задачі.

З часом розробка великих програм перетворилася на серйозну проблему і зажадала їх розбиття на дрібніші фрагменти. Основою для такого розбиття якраз і стала процедурна ***декомпозиція,*** *при* якій окремі частини програми або модулі були сукупністю процедур для вирішення деякої сукупності завдань. Розглянуті ідеї сприяли становленню деякої системи поглядів на процес розробки програм та написання програмних кодів, яка отримала назву методології ***структурного програмування.*** Період найбільшої популярності ідей структурного програмування припадає на кінець 70-х – початок 80-х років.

9.3. Методологія об’єктно-орієнтованого програмування

З часом ситуація почала істотно змінюватися. Стало очевидним, що традиційні методи процедурного програмування не здатні впоратися ані зі складністю програм та їх розробки, ані з необхідністю підвищення їх надійності. У другій половині 80-х років виникла нагальна потреба у новій методології програмування, яка була б здатна розв’язати весь цей комплекс проблем. Такою методологією стало ***об’єктно-орієнтоване програмування*** (ООП)**.**

Фундаментальними поняттями ООП є поняття ***класу*** та ***об’єкту***. При цьому під класом розуміють деяку абстракцію сукупності об’єктів (шаблон для створення об’єкта), які мають загальний набір властивостей і володіють однаковою поведінкою. Кожен об’єкт у цьому випадку розглядається як екземпляр відповідного класу. Об’єкти, які не мають повністю однакових властивостей або не володіють однаковою поведінкою, за визначенням, не можуть бути віднесені до одного класу.

Таким чином, ***клас – це програмна заготовка, яку програміст проектує і описує у своїй програмі, а об'єкт – це програмна одиниця, яка генерується і «працює» в процесі виконання програми.***

Технологія ООП була розроблена для того, щоб точніше змоделювати програму, яку необхідно написати так само, як ми дивимося на навколишній світ. Ми починаємо з найбільш абстрактного, загального класу, властивості і поведінка якого є характерними для всіх членів класу і які повністю описують набір окремих об'єктів, що складають проблему.

Однією з головних переваг ООП є широке повторення використання коду. Як тільки поведінка (метод) запрограмована для більш загального класу, не має потріби повторно її переписувати для похідних класів, де її не треба змінювати.

Найбільш істотною обставиною у розвитку методології ООП з'явилося усвідомлення того факту, що процес написання програмного коду може бути відокремлений від процесу проектування структури програми. Дійсно, перед тим як почати програмування класів, їх властивостей та методів, необхідно визначити, чим же є самі ці класи. Більш того, треба дати відповіді на такі питання, як: скільки та які класи потрібно визначити для розв'язання поставленого завдання, які властивості та методи потрібні для надання класам необхідної поведінки, а також встановити взаємозв'язки між класами.

Ця сукупність завдань не стільки пов'язана з написанням коду, скільки із загальним аналізом вимог до майбутньої програми, а також з аналізом конкретної наочної області, для якої розробляється програма. Всі ці обставини привели до появи спеціальної методології, що отримала назву методології об'єктно-орієнтованого аналізу та проектування (ООАП).

9.4. Методологія об'єктно-орієнтованого аналізу та проектування

Необхідність аналізу наочної області до початку написання програми була усвідомлена давно при розробці масштабних проектів. Процес розробки баз даних істотно відрізняється від написання програмного коду для вирішення обчислювального завдання. Головна відмінність полягає у тому, що при проектуванні бази даних виникає необхідність у попередній розробці ***концептуальної схеми,*** яка відображала б загальні взаємозв'язки наочної області та особливості організації відповідної інформації. При цьому під ***наочною областю*** прийнято розуміти ту частину реального світу, яка має істотне значення або безпосереднє відношення до процесу функціонування програми. Іншими словами, предметна область включає тільки ті об'єкти та взаємозв'язки між ними, які необхідні для опису вимог та умов розв'язання деякої задачі.

Виділення початкових або базових компонентів наочної області, які необхідні для вирішення того чи іншого завдання, являє собою, у загальному випадку, нетривіальну проблему. Складність даної проблеми проявляється у неформальному характері процедур або правил, які можна застосовувати для

цієї мети. Більш того, така робота повинна виконуватися спільно з фахівцями або експертами, які обізнані з предметною областю.

Поява методології ООАП зажадала, з одного боку, розробки різних засобів концептуалізації наочної області, а з іншого – відповідних фахівців, які володіли б цією методологією. На даному етапі з'являється відносно новий тип фахівця, який отримав назву ***аналітика*** або ***архітектора***. Разом з фахівцями предметної області аналітик бере участь у побудові концептуальної схеми майбутньої програми, яка потім перетвориться програмістами у код. При цьому окремі компоненти обираються так, щоб при подальшій розробці їх було зручно представити у формі класів та об'єктів.

Розподіл процесу розробки складних програмних додатків (прикладних програм) на окремі етапи сприяло становленню концепції життєвого циклу програми. Під ***життєвим циклом*** (ЖЦ) програми розуміють сукупність взаємопов'язаних та спрямованих у часі етапів програми, починаючи від розробки вимог до неї і закінчуючи повною відмовою від її використання. Методологія ООАП тісно пов'язана з концепцією автоматизованої ***розробки програмного забезпечення (***Computer Aided Software Engineering, CASE) та появою інструментів ***швидкої розробки додатків*** (Rapid Application Development, RAD).

9.5. Технології програмування

Свого часу розробка мов високого рівня була значним кроком уперед. Програми, які були на них написані, містили менше помилок, і праця програмістів стала продуктивнішою. Але статистика показує, що **кількість рядків програмного коду, що створюється програмістом за один робочий день, залишається в середньому постійною, незалежно від мови програмування**. Ось дані з американської статистики:

"Якщо розміру великої програми поставити у відповідність час, який необхідний для її реалізації, та кількість зайнятих у ній програмістів, то виявиться, що все відбувається так, наче кожен програміст створює 5 команд за день, а решта часу пише команди, які потім визнаються не потрібними або неправильними" [1].

Таким чином, для розв'язання кризи програмування, яка обумовлена багато у чому можливостями існуючих мов програмування, необхідний перехід до мов вищого рівня, які забезпечать такий стрибок у підвищенні продуктивності праці програмістів, який був отриманий при переході від машино-орієнтованих до алгоритмічних мов.

На думку багатьох фахівців у цій області, досягнути цього можна тільки шляхом розробки так званих **непроцедурних мов**.

Процедурна мова вказує, **як виконується дія**. **Непроцедурна мова** вказує, **яка дія виконується, але без деталізації** – **як це робиться**. Отже, мови типу PL/1, FORTRAN, Pascal, **C** є процедурними.

На даний час більшість програмістів розробляють ПЗ, використовуючи ті ж методи, що і 25 років тому. Застосування непроцедурних мов високого рівня дозволить принципово переглянути погляди на створення призначених для користувача додатків, життєвий цикл створення яких порівняний з очікуваним часом їх експлуатації. Життєвий цикл ПЗ – це безперервний процес, який починається з моменту ухвалення рішення про необхідність його створення і закінчується у момент його повного вилучення з експлуатації. Іншими словами, в сучасних умовах компанії (підприємства) перебудовують свої бізнес-процеси приблизно раз на два (три) роки (не відстаючи від традиційних технологій), стільки ж потрібно і для створення досить пристойних додатків. Інакше може виявитись, що розроблене ПЗ вже частково або повністю застаріло, оскільки підприємство перейшло на нову технологію. Отже, для створення призначених для користувача додатків є потреба в інструментальному засобі, що значно (у декілька разів) зменшує час розробки професійних призначених для користувача додатків.

За останнє десятиліття сформувався новий напрям у програмуванні – **CASE** (Computer-Aided Software System Engineering – автоматизоване проектування програмного забезпечення) та системи **RAD-программирования** (Rapid Application Development – швидка розробка додатків). Дуже грубо, CASE–технологія являє сукупність методологій аналізу, проектування, розробки та супроводу складних систем програмного забезпечення, яка підтримується комплексом взаємопов’язаних засобів автоматизації. CASE – це інструментарій для системних аналітиків, розробників та програмістів для автоматизації процесу проектування та розробки всіх етапів ПЗ.

CASE- та RAD-технології дозволяють не тільки створювати “**правильні”** продукти, але і забезпечити “правильний” процес їх створення. Основна їх мета полягає у тому, щоб відокремити проектування ПЗ від його кодування та подальших етапів розробки, а також приховати від розробників всі деталі середовища розробки і функціонування ПЗ.

У більшості сучасних CASE-систем застосовуються методології структурного аналізу та проектування, які засновані на наочній діаграмній техніці, при цьому для написання моделі проектованої системи використовуються графи, діаграми, таблиці та схеми. Таким чином, CASE-технології не можуть вважатися самостійними методологіями, вони тільки розвивають структурні методології і здійснюють їх застосування за рахунок автоматизації.

На початку в існуючих однорідних інформаційних системах кожен додаток був єдиним цілим. Для розробки такого типу додатків застосовувався **каскадний спосіб**. Його основною характеристикою є розбиття всієї розробки на етапи, причому перехід з одного етапу на наступний відбувається тільки після того, як буде повністю завершена робота на поточному. Кожен етап завершується випуском повного комплекту документації, достатньої для того, щоб розробка могла бути продовжена іншою командою розробників.

Каскадний підхід добре зарекомендував себе при побудові ПЗ, для якого на самому початку розробки можна достатньо точно та повно сформулювати всі вимоги з тим, щоб надати розробникам свободу реалізувати їх якнайкраще з технічної точки зору. Проте, у процесі використання цього підходу виявилася низка його недоліків, які були викликані, перш за все тим, що реальний процес створення ПЗ ніколи повністю не вкладався у таку жорстку схему. В процесі створення ПЗ постійно виникала потреба у поверненні до попередніх етапів та уточненні або перегляді раніше ухвалених рішень.

Основним недоліком каскадного підходу є істотне запізнення з отриманням результатів. Узгодження результатів з користувачами проводиться тільки у точках, що плануються після завершення кожного етапу робіт, вимоги до ПЗ "заморожені" у вигляді технічного завдання на весь час його створення. Таким чином, користувачі можуть внести свої зауваження тільки після того, як робота над системою буде повністю завершена. У разі неточного викладу вимог або їх зміни протягом тривалого періоду створення ПЗ користувачі отримують систему, яка не задовольняє їх потребам.

Для подолання перерахованих проблем була запропонована спіральна модель ЖЦ, що робить акцент на початкові етапи ЖЦ: аналіз та проектування. На цих етапах реалізація технічних рішень перевіряється шляхом створення прототипів. Кожен виток спіралі відповідає створенню фрагмента або версії ПЗ, на ньому уточнюються цілі та характеристики проекту, визначається його якість і плануються роботи наступного витка спіралі. Таким чином, поглиблюються та послідовно конкретизуються деталі проекту, і в результаті обирається обґрунтований варіант, який доводиться до реалізації.

Розробка ітераціями відображає об’єктивно існуючий спіральний цикл створення системи. Неповне завершення робіт на кожному етапі дозволяє переходити на наступний етап, не чекаючи повного завершення роботи на поточному етапі. При ітеративному способі розробки необхідну роботу можна буде виконати на наступній ітерації. Головне ж завдання – якнайшвидше показати користувачам системи працездатний продукт, тим самим активізуючи процес уточнення та доповнення вимог.

9.6. CASE –засоби

Тенденції розвитку сучасних інформаційних технологій призводять до постійного зростання складності програмного забезпечення, що створюється в різних областях економіки. Накопичений до теперішнього часу досвід проектування ПЗ показує, що це логічно складна, трудомістка та тривала у часі робота, яка вимагає високої кваліфікації фахівців, що беруть участь у ній. Проте до недавнього часу проектування ПЗ виконувалося в основному на інтуїтивному рівні із застосуванням неформалізованих методів, що засновані на мистецтві, практичному досвіді, експертних оцінках та дорогих експериментальних перевірках якості функціонування ПЗ. Крім того, у процесі створення та функціонування ПЗ інформаційні потреби користувачів

можуть змінюватися або уточнюватися, що ще більше ускладнює розробку та супровід таких систем.

Перераховані чинники сприяли появі програмно-технологічних засобів спеціального класу – **CASE-засобів**, що реалізовують CASE-технологію створення та супроводу ПЗ, включаючи аналіз та формулювання вимог, проектування прикладного ПЗ (додатків) та баз даних, генерацію коду, тестування, документування, управління проектом та інші процеси.

Згідно з огляду передових технологій (Survey of Advanced Technology), яка була запропонована фірмою Systems Development Inc. у 1996 р. за наслідками анкетування більше 1000 американських фірм, CASE-технологія на даний час потрапила у розряд найстабільніших інформаційних технологій. Її використовувала половина усіх опитаних користувачів більш ніж у третині своїх проектів, з них 85% завершилися успішно. Проте не дивлячись на всі потенційні можливості CASE-засобів, існує безліч прикладів їх невдалого впровадження, в результаті яких CASE-засоби стають "поличними" ПЗ (shelfware).

9.7. Методологія RAD

Якщо людина має справу з комп'ютером, то рано чи пізно у неї виникне бажання, а іноді і необхідність, навчитися програмувати. Серед користувачів персональних комп'ютерів зараз найпопулярніша родина операційних систем Windows і, природно, що той, хто збирається програмувати, прагне писати програми, які працюватимуть у цих системах.

Бурхливий розвиток обчислювальної техніки, потреба в ефективних засобах розробки ПЗ призвели до появи засобів програмування, що орієнтовані на так звану "швидку розробку" RAD (Rapid Application Development). Сутність технології RAD полягає у тому, що середовище розробки бере на себе велику частину генерації коду програми, залишаючи програмістові роботу суто з написання логічних ділянок програми та функцій обробки подій. Продуктивність програміста при використанні деяких RAD-систем фантастична!

Технологія RAD в основному базується на спіральній моделі ЖЦ ПЗ, тобто циклі, що повторюється, при якому розробники, по мірі того, як додаток починає знаходити форму, реалізують у продукті вимоги, які отримуютьсяі через взаємодію із замовником.

Життєвий цикл ПЗ за методологією RAD складається з чотирьох фаз:

- аналізу та планування вимог;
- проектування;
- побудови;
- впровадження.

На фазі аналізу та планування вимог користувачі системи визначають функції, які вона повинна виконувати, виділяють найпріоритетніши з них, такі, що вимагають опрацювання у першу чергу, описують інформаційні потреби.

Визначення вимог виконується в основному силами користувачів під керівництвом фахівців-розробників. Результатом даної фази повинні бути список та пріоритетність функцій майбутньої системи.

На фазі проектування частина користувачів бере участь у технічному проектуванні системи під керівництвом фахівців-розробників. Користувачі, безпосередньо взаємодіючи з ними, уточнюють та доповнюють вимоги до системи, які не були виявлені на попередній фазі. Детальніше розглядаються процеси системи. Аналізується та, при необхідності, коректується функціональна модель. Після детального визначення складу процесів оцінюється кількість функціональних елементів системи, що розробляється, та ухвалюється рішення про розбиття ПЗ на підсистеми.

На фазі побудови виконується безпосередньо сама швидка розробка додатку. На даній фазі розробники проводять ітеративну побудову реальної системи на основі отриманих на попередній фазі моделей, а також вимог нефункціонального характеру. Програмний код частково формується за допомогою автоматичних генераторів, які отримують інформацію безпосередньо з CASE-засобів. Кінцеві користувачі на цій фазі оцінюють отримані результати та вносять корективи, якщо у процесі розробки система перестає задовільняти визначеним раніше вимогам. Тестування системи здійснюється безпосередньо у процесі розробки.

Після закінчення робіт кожної окремої команди розробників проводиться поступова інтеграція даної частини системи з останніми, формується повний програмний код, виконується тестування спільної роботи даної частини додатка з рештою частин, а потім тестування системи у цілому.

На фазі впровадження проводиться навчання користувачів, організаційні зміни та паралельно з впровадженням нової системи здійснюється робота з існуючою системою (до повного впровадження нової). Оскільки фаза побудови достатньо нетривала, планування та підготовка до впровадження повинні починатися заздалегідь, як правило, на етапі проектування системи.

Наведена схема розробки ПЗ не є абсолютною. Можливі різні варіанти, які залежать, наприклад, від початкових умов, в яких ведеться розробка: розробляється абсолютно нова система; вже було проведено обстеження підприємства та існує модель його діяльності; на підприємстві вже існує деяке ПЗ, яке може використовуватись як початковий прототип або повинно бути інтегровано з тим, що розробляється.

Проте, слід відзначити, що методологія RAD, як і будь-яка інша, не може претендувати на універсальність, вона гарна у першу чергу для відносно невеликих проектів, що розробляються для конкретного замовника.

Оцінка розміру додатків проводиться на основі так званих функціональних елементів (екрани, повідомлення, звіти, файли тощо). Подібна метрика не залежить від мови програмування, на якій ведеться розробка.

Серед засобів **RAD-програмування** особливо хотілося б відзначити середовище розробки ПЗ **Microsoft Visual Basic** (версії 6 .NET)**, Borland Delphi** (версії 6, 7 .NET) та **Clarion** фірми TopSpeed (нині SoftVelocity) [5].

9.8. Microsoft Visual Basic

Ми живемо у дивовижний час, і він ще ніколи не був найбільш сприятливим для програмістів, ніж зараз. Поява мов візуального проектування ПЗ дозволила спростити процес написання програм для системи Windows. Піонером в цьому напрямі була мова та середовище розробки Microsoft Visual Basic. Visual Basic завжди дозволяв спростити процес написання програм, але для його перетворення на справжню мову професійної розробки та створення середовища розробки були потрібні роки. Visual Basic є продуктом багаторічної еволюції мови Basic.

В основному ця мова призначалася для створення програм, які працюють у режимі діалогу з користувачем, тобто у **візуальному режимі**. Тому її і назвали Visual Basic. Під час розвитку комп'ютерної технології Visual Basic поступово перетворювався з простої мови для початківців у могутній інструмент для швидкого створення працездатних додатків. Використовуючи інтегроване середовище розробки IDE (Integrated Development Environment) Visual Basic дозволяє значно спростити такий, на перший погляд, складний процес, як розробка призначеного для користувача інтерфейсу. **Призначений для користувача інтерфейс** – це "обличчя" вашого додатку, на створення якого раніше "вручну" було потрібно дуже багато часу програміста. У ряді випадків весь цей сервіс починав займати до 80-90% об'єму програмних кодів. За своєю складністю та можливостями Visual Basic можна поставити в один ряд з Delphi або C++.

За допомогою Visual Basic можна як і раніше швидко писати програми для системи Windows, але, крім цього, починаючи з версії 6, тепер можна також створювати додатки типу клієнт/сервер та надійні додатки керування базами даних.

Пакет Visual Basic відіграє основну роль у стратегії доступу до даних. За даними компанії Microsoft у більш ніж 90% всіх додатків Visual Basic використовуються з'єднання з системами керування реляційними базами даних (СКРБД). У більшості цих програм використовуються СКРБД Jet (Access) та Microsoft SQL.

Використовуючи Visual Basic, можна без зусиль створювати документи та елементи управління ACTIVEX. Ви вже знайомі з різноманітними елементами управління середовища Windows, такими як перемикачі, прапорці, кнопки, списки тощо. Багато розробників разом з ними використовують власні елементи управління, відомі як **елементи управління ACTIVEX**. По мірі того як система Windows еволюціонувала від версії 3.x до версій 95, 98, ME, NT, 2000 і XP, WINDOWS 7 та WINDOWS 8, те ж відбувалося і з елементами управління. Вперше вони з'явилися у мові Visual Basic під назвою VBX-елементів (таким було розширення відповідних файлів). По суті цими елементами управління були невеликі динамічні бібліотеки (DLL). Можливості сучасних 32-розрядних елементів управління значно розширені, а їх файли

тепер мають розширення .OCX, а самі елементи отримали назву елементів управління ACTIVEX.

Фірма Microsoft інтегрувала також спеціальну версію Visual Basic, яка відома під ім'ям Visual Basic for Application (VBA) в усі компоненти пакету Microsoft Office та деякі інші програми. VBA – це основний компонент мови Visual Basic.

Оскільки програмісти часто використовували Visual Basic для розробки власних елементів управління, які вони включали у додаток, що були написані на C/C++, то стало зрозумілим, що мови C/C++ слід забезпечити засобами розробки елементів управління. Але з появою нової версії компілятора Microsoft Visual Studio.Net, що включає мови програмування розробки ПЗ C#, Visual Basic, Visual C++ та Visual FoxPro, а також спеціального майстра розробки елементів управління, така потреба відпала.

Разом з іншими компонентами пакет Visual Studio .Net є новим підходом компанії Microsoft при розробці додатків баз даних для кінцевого користувача. Основним методом доступу до файлів, що розділяються, та баз даних клієнт/сервер стає використання високорівневих технологій **ACTIVEX Data Objects** (**ADO** – елементи (об'єкти) управління даними ACTIVEX, так званий **MDAC –** Microsoft Data Access Components). А також новий низькорівневий інтерфейс **OLE DB** (**OLE** – Object Linking and Embedding, зв'язування та впровадження об'єктів), що прийшли на зміну старому прикладному інтерфейсу програмування Open DataBase Connectivity (ODBC).

9.9. Borland Delphi

Delphi – це середовище швидкої розробки, в якому мовою програмування є Object Pascal, який побудований на міцному фундаменті ООП. В основі ідеології Delphi лежить технологія візуального проектування та методологія об'єктно-орієнтованого програмування за подіями. Виконання програми у середовищі Windows відбувається під впливом подій. Подія виникає, наприклад, тоді, коли користувач клацнув мишею на полі екрану, або натиснув клавішу на клавіатурі. Кожна дія користувача у процесі виконання програми ініціює посилку повідомлення від Windows до програми з інформацією про те, що користувач зробив. Посилка повідомлення відіграє роль події, і у програми з'являється можливість обробити цю подію відповідним чином. Тепер стає очевидним, що концепція програмування Windows прямо протилежна концепції DOS – операційна система (Windows) наказує програмі, що останній потрібно робити, а не навпаки (DOS).

Якщо немає "під рукою" такої мови як Clarion або Delphi, то написання Windows-програм буде вельми складним заняттям, оскільки необхідно явним чином кодувати обробку усіх повідомлень, які посилаються Windows, та розробляти графічний інтерфейс користувача.

Вихід з цієї ситуації позначився завдяки двом підходам. Перший з них – стандартизація багатьох функцій інтерфейсу, завдяки чому з'явилася

можливість бібліотеки, яка існує, наприклад, у Windows. Зокрема з'явився **Windows API – інтерфейс**. Windows API (Application Programming Interface, інтерфейс прикладного програмування) – це набір функцій, які надає програмістові ОС Windows. Причому, при зміні стилю графічного інтерфейсу (наприклад, при переході від Windows 98 до Windows XP) форма виклику функцій не змінюється. Тому на цьому шляху створилися чудові умови для вирішення одного з найважливіших завдань удосконалення техніки програмування – **повторного використання кодів**.

Іншим революційним кроком, що кардинально полегшив життя програмістів, з'явилася поява візуального програмування, яке виникло у Visual Basic та знайшло блискуче втілення у Delphi, C++ Builder (фірми Borland) та Clarion. Візуальне програмування дозволило звести програмування призначеного для користувача інтерфейсу (і не тільки) до простих та наочних процедур, які дають можливість за хвилини або години зробити те, на що раніше витрачались місяці роботи. Але переваги візуального програмування зводяться навіть не до цього. Найголовніше полягає у тому, що під час проектування форми та розміщення на ній компонентів Delphi автоматично формує коди програми, включаючи у неї відповідні фрагменти, які описують даний компонент. Тобто проектування зводиться, фактично, до розміщення компонентів на формі, завданню деяких їх властивостей та написанню, при необхідності, обробників подій.

9.10. Clarion

Clarion – це орієнтоване на обробку даних середовище швидкої розробки додатків, яке включає кодогенерацію та **метадані** (дані про дани), які багаторазово використовуються для швидкого створення додатків рівня управління підприємством для обробки бізнес-даних та бізнес-процесів. Основою Clarion є об'єктно-орієнтована **мова програмування 4-го покоління** (**4GL – Fourth Generation Languages**).

Якщо для організації деякої операції з даними на мові 3-го покоління необхідно буде написати сотні рядків коду, то для реалізації цієї ж операції на мові 4-го покоління буде достатньо 10-15 рядків.

Тоді як мови 3-го покоління є процедурними, мови 4GL виступають як непроцедурні, оскільки користувач визначає, ***що*** повинно бути зроблене, але не говорить, ***як*** саме бажаний результат має бути досягнутий. Реалізація мов 4-го покоління значною мірою заснована на використанні компонентів високого рівня, які часто називають "**інструментами четвертого покоління**". Користувачеві не потрібно визначати всі етапи визначення програми, які необхідні для вирішення поставленого завдання, а достатньо буде лише визначити потрібні параметри, на підставі яких згадані вище інструменти автоматично здійснять генерацію прикладного додатку. За оцінкою фахівців мови 4-го покоління дозволяють підвищити продуктивність роботи на порядок,

але за рахунок обмеження типів завдань, які можна вирішувати з їх допомогою. Виділяють наступні типи мов 4-го покоління:

- мови подання інформації, наприклад мови запитів або генератори звітів;
- спеціалізовані мови, наприклад мови електронних таблиць та баз даних;
- генератори додатків, які при створенні додатків забезпечують визначення, вставку, оновлення або вибірку відомостей з баз даних;
- мови дуже високого рівня, які призначені для генерації коду додатків.

Як приклади перших трьох типів мов 4-го покоління можна навести наступні.

Мови запитів. Реляційні СКБД у тій чи іншій формі включають підтримку непроцедурних мов маніпулювання даними – найчастіше це мова структурованих запитів SQL (Structured Query Language) або мова запитів за зразком
QBE (Query-by-Example). Мову запитів можна визначити як високорівневу вузькоспеціалізовану мову, яка призначена для вибірки інформації з Бази Даних.

Генератори звітів – є інструментом створення звітів на основі тієї інформації, яка зберігається у Базі Даних. Вони схожі на мову запитів у тому сенсі, що користувачеві надаються засоби створення запитів до Бази Даних та вибірки з неї інформації, яка використовується для подання у звіті.

Генератори форм – є інтерактивним інструментом, що призначений для швидкого створення шаблонів введення та відображення даних в екранних формах. Генератор форм дозволяє користувачеві визначити зовнішній вигляд екранної форми, її вміст та місце розміщення на екрані. Досконаліші генератори форм дозволяють створювати обчислювані атрибути з використанням арифметичних операторів або узагальнених функцій, а також задавати правила перевірки даних, що вводяться.

Генератори графічного представлення даних – інструменти для отримання інформації з Бази Даних та відображення її у вигляді гістограм, колових, лінійних, точкових діаграм і таке інше.

Генератори додатків – є інструментом для створення програм, які взаємодіють з Базою Даних. Генератори додатків зазвичай складаються із заздалегідь створених модулів (або шаблонів), що містять фундаментальні функції, які потрібні для більшості програм. Ці модулі, які зазвичай створюються на мовах високого рівня, утворюють “бібліотеку” доступних функцій (шаблонів), і генератор додатків визначає які з них застосувати, і як їх слід виконувати. Єдиним поки що представником **Генераторів повністю закінчених додатків** є Clarion, який призначений для розробки (генерації) значної кількості комерційних додатків без написання рядків програмного коду, – і при цьому з високою якістю програм та їх гарантованою надійністю.

Яким же чином Clarion дозволяє досягти такого ефекту? Перш за все, Clarion є системою **RAD-програмування** (Rapid Application Development – швидка розробка додатків). Але таких систем вже багато.

Крім того, при роботі з базами даних розробник повинен усього лише описати структуру бази (**Словник Даних**) – усе решту зробить Clarion. Якщо ж бази даних вже є (наприклад, створені програмами на FoxPro), то для їх використання у Clarion досить вказати на файли таких баз даних, як на джерело структур записів та перевести бази даних у більш місткий та зручний у використанні формат TopSpeed. При всьому цьому Clarion забезпечує роботу з базами даних практично у будь-якому з використовуваних на даний час форматів від BASIC та ASCII-файлів до SQL, Betreave, SYBASE або Oracle.

Як доповнення до 4GL Clarion-мови в продукт Clarion включені компілятори C++ та Modula-2. Усі мови використовують один і той самий загальний оптимізатор коду і можуть одночасно застосовуватися у рамках одного додатку.

Завдяки застосуванню словникових метаданих кодогенерації, Clarion допомагає розробникам задовольняти вимогам стандартів, скорочувати час кодування, уникати помилок та доводити проекти до завершення у рекордно короткі терміни. Він дозволяє команді розробників використовувати та розвивати узгоджений набір стандартів кодування. Він також допомагає новачкам легко та швидко адаптуватися до специфічного стилю кодування, який прийнятий в організації.

Метабази Clarion дозволяють новачкам генерувати проектний код, використовуючи **Шаблони** (Templates) та **ABC** (Application Builder Classes – механізм, що дозволяє на підставі готових кодів програм, які зберігаються у бібліотеках, створювати високоефективні додатки, які розроблені головними програмістами різних фірм). Метабази містять велику частину інформації, яка потрібна для створення повнофункціонального додатку, що повністю відповідає своїм проектним вимогам. Розробникам це дає можливість миттєво створювати високоякісні корпоративні додатки, які забезпечують ведення даних та звітність, а також доопрацьовувати їх з використанням додаткового інструментарію (призначені для користувача вставки – **Embeds**) для вирішення специфічних завдань.

У Clarion використовується вбудована підготовка документів (звітів) і для їх формування не потрібно ніяких зовнішніх продуктів, що спільно використовувались, додатково постачались та підключались. Документи просто компілювалися у додаток, що розроблявся, та виконувався із швидкістю машинного коду. Можна генерувати документи у форматах **PDF, HTML, XML** та **TEXT**.

9.11. Microsoft Visual Studio LightSwitch 2012

Visual Studio LightSwitch – це засіб розробки, за допомогою якого можна швидко створювати бізнес-додатки. LightSwitch надає спрощене середовище розробки, що дозволяє зосередитися на бізнес-логіці, а не на інфраструктурі додатків.

9.11.1. Знайомство с LightSwitch

Сучасні тенденції розвитку інтернет-додатків показують, що в майбутньому більшість бізнес-додатків будуть віртуальними і доступними через Інтернет з будь-якого мобільного пристрою. Причому це відбудеться в достатньо осяжному проміжку часу. І вже зараз затребувані інструменти для створення такого типу додатків. Нові інструменти повинні бути орієнтовані на користувачів, що добре розбираються у своїй предметній області, але не професіоналів в програмуванні.

Більшість бізнес-додатків мають архітектуру форм на основі даних, тобто являють собою користувацький інтерфейс для перегляду, додавання та зміни даних. При створенні додатків з архітектурою форм на основі даних при допомозі інших засобів розробки велика частина часу витрачається на виконання завдань, що повторюються. Доводиться писати код для взаємодії з базою даних, а також для реалізації інтерфейсу користувача і бізнес-логіки. Тому, природно, що для більшості задач, що повторюються хотілося б виконувати ці функції автоматично. Фактично додатки можна створити без написання будь-якого коду! Для більшості додатків єдиним доступним і необхідним для написання кодом є код бізнес-логіки.

Компанія Microsoft досить довгий час розробляла спеціальні інструменти, що спрощують створення додатків, які засновані на базах даних, наприклад, Visual FoxPro, Visual Basic та інші. Створення платформи .Net також підтверджує орієнтованість компанії на випуск простих та зручних інструментів для розробників. Розвиток платформи. Net від Windows Forms до Windows Presentation Foundation, і від ASP.Net (технологія створення веб-додатків і веб-сервісів) до Silverlight (об'єднує мультимедіа, графіку, анімацію та інтерактивність в одній програмній платформі) показує, що велика увага приділяється створенню зручних механізмів доступу та обробки даних.

26 липня 2011 року компанія Microsoft випустила новий продукт – **Microsoft Visual Studio LightSwitch 2011**. Офіційне визначення цього продукту звучить наступним чином "гнучкий інструмент розробки, який використовується для швидкого створення і розгортання настільних бізнес-додатків і бізнес-додатків для хмарних технологій (Облака)".

Зараз LightSwitch є доповненням для Visual Studio 2012 на основі дизайнера, який допомагає в роботі з сервісами та додатками, що орієнтованиі на дані. Visual Studio IDE при роботі з проектом LightSwitch перетворюється на середовище розробки всього з трьома основними редакторами (у так званому логічному режимі): Entity Designer, Query Designer і Screen Designer. Ці редактори в основному призначені для швидкого отримання результатів за рахунок інтуїтивно зрозумілого інтерфейсу та простоти у використанні. Це дає деякі очевидні переваги розробнику LightSwitch.

Microsoft Visual Studio LightSwitch – гнучкий інструмент розробки, за допомогою якого розробники з будь-яким рівнем підготовки можуть швидко

створювати та розгортати привабливі настільні бізнес-додатки, а також і бізнес-додатки з використанням хмарних технологій.

Пропонуючи зручні інструменти та шаблони, а також гранично просту і зрозуміле середовище розробки, Microsoft Visual Studio LightSwitch допомагає скоротити терміни розробки і спростити всі процеси, від дизайну користувацького інтерфейсу до розгортання програми в хмарі Windows Azure. Тепер з появою Microsoft Visual Studio LightSwitch нарешті з'явилася реальна можливість створювати масштабовані користувацькі програмні рішення помірної вартості, що є сполучною ланкою між існуючими системами і даними і забезпечують всебічні представлення бізнес-даних, з якими користувачам зручно працювати.

Основоположною цільовою аудиторією Visual Studio LightSwitch є так звані кінцеві користувачі-розробники. Аналітична агенція Forrester визначає цей тип розробників як «користувачі – розробники бізнес-додатків» (End-User Business Developers). До них відносяться знавці, які не є професійними розробниками ПЗ, але формують бізнес-додатки для забезпечення нових бізнес-функцій або розширюють можливості вже використовуваних програмних рішень.

Новий продукт проектування Microsoft споруджений на базі технології Visual Studio. Коли можливостей LightSwitch буде недостатньо, проекти LightSwitch можна відкривати і доопрацьовувати в Visual Studio. Професійним розробникам LightSwitch сподобається в якості елементарного у використанні та швидкісного створення прототипів додатків. Можна швидко створити в LightSwitch прототип майбутнього додатка, продемонструвати його майбутнім користувачам, а слідом за тим розширювати його функціональністю в Visual Studio.

Користувачі-розробники мають всі шанси підключати свої програми до існуючих джерел даних, включаючи Microsoft SQL Server, Microsoft SQL Azure, SharePoint, Microsoft Office Access і бази даних сторонніх виробників. Це полегшує створення, збір, аналіз і повторне застосування відомостей з різноманітних джерел даних. Додатки, що формуються за допомогою LightSwitch, ще можуть експортувати дані в Microsoft Office Excel, що забезпечує більш швидку і просту організацію спільного вживання та складання звітів.

9.11.2. Архитектура LightSwitch

Багатьом цікаво, що ж ховається під LightSwitch, на основі яких технологій і архітектурних рішень він побудований. Середовище виконання LightSwitch побудовано за класичною тришаровою архітектурою. **Презентаційний шар** вирішує завдання взаємодії з кінцевим користувачем і основні функції які він виконує – візуалізація, редагування і введення даних. **Логічний шар** обробляє запити від клієнта по редагуванню, вибірці даних та іншими операціями. Головна мета цього шару – захистити дані від прямої зміни. Це дає можливість

бути впевненим у цілісності та безпеці даних. І нарешті, **шар даних**, призначений для зберігання структурованої інформації.

Дизайн таких тришарових додатків може бути досить складним завданням. Кожен шар може бути створений з величезною кількістю технологічних рішень і прийомів. Кожен шар так само може вимагати різних підходів з розгортання та управління під час експлуатації. І нарешті, кожен шар повинен вміти "розмовляти" один з одним з урахуванням забезпечення безпеки.

У LightSwitch багато перераховані завдання вже вирішені, пропонуючи програмісту готові механізми. Технологічною базою для цього служать технології, які добре себе зарекомендували. У першу чергу це Microsoft Silverlight 4.0, який призначений для побудови шару подання. Логічний шар створений на основі WCF RIA DomainServices (домен послуг Windows Communication Foundation – WCF, які инкапсулируются в бізнес-логіку програми), що працюють під управлінням ASP.NET 4.0. При цьому дизайн цього шару надає можливість запуску як у локальному режимі (на комп'ютері кінцевого користувача) так і на сервері IIS (Internet Information Services, до версії 5.1 – Internet Information Server) або навіть в Windows Azure (WebRole). В якості системи зберігання даних використовується SQL Server, або SQL Azure, крім цього є можливість так само працювати з даними списків Microsoft Sharepoint 2010 (колекція програмних продуктів і компонентів).

9.11.3. Створення проекту в Microsoft Visual Studio LightSwitch

Створити проект в LightSwitch просто. Незважаючи на те, що в багатьох випадках Вам навіть не доведеться писати код, все одно заздалегідь потрібно буде вибрати мову програмування за замовченням. Вибір мови програмування, Visual C# або Visual Basic, є єдиним рішенням, яке має бути прийнято заздалегідь при створенні нового проекту LightSwitch. Логічно проекти LightSwitch завжди розділені на три рівні і використовують кращі практики та шаблони n-рівневих додатків, а також такі популярні технології .NET, як Entity Framework, Silverlight. LightSwitch включає найрізноманітніші шаблони, крім того, безліч шаблонів доступні для скачування в Інтернеті.

Спрощення розробки. Система LightSwitch є гранично простою і зрозумілою, але при цьому потужним середовищем розробки, яке дозволяє розробникам незалежно від їх рівня підготовки створювати бездоганні зручні у використанні бізнес-додатки. Використовуючи можливості проектування користувальницьких інтерфейсів та інструменти розробки Visual Studio LightSwitch, можна дуже швидко створювати користувацькі додатки, які за якістю та зовнішнім виглядом нічим не відрізняються від готових комерційних рішень.

Скорочення часу отримання готового рішення. LightSwitch істотно скорочує час створення користувацького додатка, в роботі якого враховується взаємодія з існуючими додатками, застарілими системами та веб-сервісами.

LightSwitch автоматично формує весь типовий код, дозволяючи зосередитися на розробці спеціальної логіки, яка і відрізняє даний конкретний додаток.

Гнучкість сьогодні і завтра. LightSwitch забезпечує створення працездатних масштабованих додатків, що повністю відповідають поточним вимогам, і при цьому залишає можливість вибору технологій і зміни бізнес-опцій згодом. Використання LightSwitch дозволяє вільно взаємодіяти з існуючими системами управління даними та веб-сервісами, вибирати з широкого діапазону варіантів розміщення і розгортання –навіть додавати плагіни сторонніх виробників у разі потреби при зміні вимог.

Проблеми, які допомагає вирішити Visual Studio LightSwitch. Організації хочуть вирішувати бізнес-завдання за допомогою додатків, що володіють достатньою гнучкістю і масштабованістю, щоб виконувати свої функції в умовах змінн вимог. Але не завжди для створення таких додатків є необхідні час і ресурси. LightSwitch відповідає цим вимогами, у чому можна переконатися в наступному розділі.

9.11.4. Дванадцять основних переваг LightSwitch

1. Створення бізнес-додатків, дизайн який ні в чому не поступається додаткам професійних дизайнерів. Visual Studio LightSwitch поставляється з розширеним набором оболонок додатків, які забезпечать створювати користувачем додатки, що мають звичний зовнішній вигляд популярного ПЗ Майкрософт.

2. Створення більш якісних додатків в більш стислі терміни. LightSwitch включає шаблони для самих популярних типів бізнес-додатків, також пропонуються вбудовані типи даних для часто використовуваних полів (атрибутів), які часто використовуються, таких як номери телефонів та адреси електронної пошти.

3. Спрощення розробки. Гранично просте та зрозуміле середовище розробки LightSwitch спрощує процес розробки і забезпечує допомогу, коли це потрібно, відображаючи інструменти відповідно поточному завданню. У цьому середовищі необхідні інструменти доступні тоді, коли вони потрібні.

У сучасних бізнес-додатках повинні підтримуватися такі можливості, як пошук, сортування, упорядкування сітки і експорт даних. Додатки LightSwitch створюються з уже вбудованими цими та багатьма іншими функціями. Крім того, є вбудованими типові операції з даними, такі як додавання, оновлення, збереження та вилучення, а також базова логіка перевірки даних.

4. Сутності даних та екрани. LightSwitch спрощує розробку бізнес-додатків за допомогою сутностей даних та екранів. Сутності даних, або таблиці, є способом представлення даних у LightSwitch. Сутності даних створюються за допомогою вбудованої бази даних програми або за допомогою імпорту даних із зовнішньої бази даних, списку SharePoint або іншого джерела даних. Між сутностями можна створювати відносини, навіть якщо їх джерела даних різні.

Можна також створювати запити до даних з використанням графічного конструктора і при необхідності доопрацьовувати їх, змінюючи код.

Екрани, або форми, є способом відображення даних у LightSwitch. Екрани створюються на основі попередньо визначених шаблонів. Щоб прив'язати дані до екрану, достатньо вказати сутності або запити, які слід відобразити. Після створення екрану можна змінити його зовнішній вигляд в конструкторі; при цьому написання коду не потрібно.

5. Можливість зміни програми відповідно мінливих бізнес-вимог. Вбудовані шаблони і компоненти LightSwitch є абсолютно розширюваними, завдяки чому вводити додаткові можливості у додатки швидко і просто. Всі проекти LightSwitch створюються із застосуванням Silverlight і .NET Framework, та рекомендованих шаблонів проектування, охоплюючи шаблон model-view-controller. Це забезпечує вертикальну масштабованість програми LightSwitch і можливість його доопрацювання за допомогою Visual Basic або ж Visual C# (C++, Fortran 90 – LightSwitch 2012).

6. Швидке введення можливостей, що полегшують роботу користувачів. Додатки LightSwitch підтримують експорт даних в Microsoft Office Excel для спрощення їх спільного використання та формування звітів, не вимагаючи ніяких додаткових зусиль з боку розробника. Використання асинхронних процедур завантаження даних, що надаються LightSwitch, дозволить створювати додатки, які зберігають працездатність під час завантаження даних.

7. Простота додавання потужних можливостей аутентифікації. Вбудовані моделі аутентифікації спрощують завдання з надання різних рівнів доступу та авторизації різним користувачам. LightSwitch автоматично створює консоль адміністрування, забезпечуючи адміністратору простий і інтуїтивно зрозумілий інструмент завдання ролей і прав доступу користувачів.

8. Створення користувацького додатка відповідно стилю ведення бізнесу. Створюваний додаток повинен робити саме те, для чого він призначений. LightSwitch дозволяє створювати для користувача бізнес-логіку та правила, унікальні для конкретного застосування і користувацької аудиторії. LightSwitch спрощує завдання по збору, аналізу та повторному використанню відомостей з найрізноманітніших джерел даних, включаючи Microsoft SQL Server, Microsoft SQL Azure, Microsoft SharePoint, Oracle та інших баз даних сторонніх виробників, допомагаючи використовувати бізнес-відомості з максимальною ефективністю.

9. Налаштування за допомогою розширень. У LightSwitch включені користувацькі розширення, такі як бізнес-орієнтовані типи даних Money і Phone Number. Розширення компоновки Picture та Text забезпечують нові можливості подання даних на екрані. За допомогою розширень оболонки і розширень теми можна змінювати зовнішній вигляд і поведінку програми, змінюючи лише один параметр.

У LightSwitch за допомогою розширень оболонки і розширень теми можна змінювати зовнішній вигляд програм, застосовуючи такі можливості

розширень, як теми та елементи керування, що налаштовуються, або змінюючи макет за допомогою розширень оболонки. Щоб зменшити обсяг створюваного коду та спростити форматування в інтерфейсі, можна використовувати бізнес-типи, що налаштовуються.

10. Перевірка даних, тестування та розгортання. Базову перевірку в IDE можна виконувати, використовуючи обов'язкові поля та довжину рядків. Для більш складних перевірок, заснованих на бізнес-логіці, потрібно написання коду. Під час виконання на екранах використовується вбудований інтерфейс для виконання перевірки.

Щоб протестувати додаток, треба просто запустити його. Зміни користувацького інтерфейсу можна виконувати безпосередньо при виконанні додатків. За допомогою уособлення ролі при наявності дозволів налагодження можна перевірити параметри перевірки автентичності та авторизації. Готовий додаток можна розгорнути за допомогою моделі ClickOnce, яка дозволяє надалі виконувати автоматичне оновлення.

11. LightSwitch повністю сумісний з Visual Studio. LightSwitch вільно інтегрується з Visual Studio Professional, Premium або Ultimate, забезпечуючи кращі можливості цих двох середовищ розробки. У LightSwitch використовуються ті самі шаблони проектів, що і в Visual Studio Professional, що спрощує перенесення додатків у цю IDE. Якщо додаток досяг рівня, коли потрібно використовувати інструменти і можливості більш потужного середовища розробки, треба просто перейти в це більш потужне середовище.

12. Прискорення розгортання та оновлень. З Visual Studio LightSwitch можна створити один додаток, який може бути розгорнутий на настільних клієнтах, в Веб або в Хмарі. Вибір найбільш відповідного методу розгортання грунтується на тому, як і де користувачі будуть працювати з додатком.

9.12. Огляд WinDev

9.12.1. ПризначенняWinDev

WinDev являє собою інтегроване середовище розробки (IDE) з мовою п'ятого покоління (5GL), вперше опубліковане французькою компаніею PC SOFT в 1993 році. Це середовище використовує 5GL мову програмування, відому як **WLanguage**. Інструмент дозволяє використовувати заздалегідь певні набори стандартних форм і алгоритмів, які будуть використовуватися в автоматичному режимі для створення додатків (RAD).

Вихідний код попередньо прекомпілюється та інтерпретується під час виконання, як Visual Basic, Java або C#. Це робить виконувану програму незалежною від операційної системи і дозволяє динамічну компіляцію, а також миттєве тестування в процесі розробки. WinDev був спочатку задуманий як CASE інструмент для написання програм для Windows. Починаючи з версії 16 (нині 17, 18 версії, 2012 р.), WinDev може бути використаний для написання споріднених для Linux і Mac додатків.

WinDev це середовище розробки, яке спрямоване на швидку розробку (в 10 разів швидше, ніж традиційні середовища), орієнтованих на дані додатків. Його інтегрований набір редакторів дозволяє розробляти прототипи, які потім можуть бути перетворені на повноцінні додатки. WinDev може генерувати програми в Java поряд зі стандартними додатками або додатки для .NET платформи. Він підтримує процедурні, а також об'єктно-орієнтовані парадигми програмування.

Продукція компанії PC SOFT, WebDev і WinDev Mobile, використовує одні й ті самі концепції, а також базується на одній мові програмування (WLanguage). Вони відповідно використовуються для створення додатків, веб-сайтів і додатків для мобільних пристроїв (смартфони, КПК, таблетки, промислові пристрої тощо). Той же вихідний код може бути використаний для створення додатків, які будуть працювати на різних платформах (Windows, Linux, Android, IOS, Windows Mobile, Windows Phone тощо).

WinDev пропонує ряд контрольних центрів, де розробники можуть керувати вихідним кодом (check-in/checkout, управління версіями), відстеженням помилок, встановленням QA тестів, і так далі.

З WinDev моделювання баз даних дуже легке. Ви можете створити "MERISE" аналіз (LDM та CDM), не знаючи MERISE (універсальна методологія моделювання в області розробки інформаційних систем) і "UML" модель, не знаючи UML! WinDev технологія дійсно революційна: він наочно представляє організацію даних і процесів. Завдяки WinDev, будь-який розробник може автоматично створювати MERISE або UML аналіз.

WinDev постачається з власною інтегрованою СУБД, HyperFileSQL, але також підтримує більшість комерційних баз даних через ODBC, OLE DB або через рідний доступ. Найбільш поширеними є:

1. AS/400, Access, Sybase, Informix, ...
2. DB2, Oracle, SQL Server, MySQL, Xbase, ...
3. Будь-які бази даних, доступні в SQL мовою в Windows.
4. Текстовий (ASCII файли).

Потужний механізм бази даних HyperFileSQL доступний в клієнтській, мережний версії і в Клієнт-серверної версії.

Зв'язок зі сторонніми базами даних або файловими системами (XML, Oracle, SQL Server, MySQL, Access, DB2, Informix та інші) набагато простіший. При описі бази даних треба просто перемістити цей опис від сервера до даних і в Редакторі моделі вказати обрані таблиці. Якщо структура імпортованих баз даних розвивається, то WinDev виявляє це і пропонує модифікувати програму та аналіз даних!

WinDev підтримує .NET технологію. WinDev створює файли до виконання, а також класи, які можуть бути використаними іншими .NET мовами (наприклад, asp.Net). WinDev знає, як використовувати класи, які створені іншими мовами, і може включати .Net управління в WinDev вікна.

WinDev автоматично підтримує зображення, звук, відео з дійсно дивовижною силою і простотою. Підтримка всіх стандартних форматів без необхідності чогось кодувати.

Як правило, якість і безпека програм залежить від багатьох параметрів: Складність або простота у використанні мови, складність або інтуїтивність навколишнього середовища, потрібно чи не потрібно використовувати стороннє програмне забезпечення тощо. Автоматизована фаза тесту тільки одна з фаз, щоб забезпечити високу якість вашого програмного забезпечення. WebDev також пропонує регресивне тестування, юніт-тести, тести інтеграції і стрес-тести. Просто велика економія ваших дій, тестові сценарії створюються і можуть бути відтворені пізніше.

WinDev включає в себе Аналізатор і Генератор програмної документації як частина продукту. Повна документація створюється в один клік по реверсивному аналізу вашого проекту і може бути змінена в будь-який час, і завжди буде сучасною. Цей генератор документації допоможе вам заощадити багато днів.

"Software Infrastructure Supervisor" ("Супервізор Програмного Забезпечення Інфраструктури", SIS) дозволяє відслідковувати стан вашої Інформаційної системи. SIS тримає вас у курсі про використання користувачами ваших додатків. SIS автоматично виконує аудити безпеки, щоб попередити вас про потенційні проблеми: різні версії однієї програми у використанні, не оптимізовані дані на сервері, доступ до даних, розташованих на незвичайних серверах, тощо. Інфраструктура Supervisor дозволяє збільшити розмір і швидкість вашої мережі.

Створення звітів з WinDev є легким або через кодування або за допомогою Генератора звітів. Підтримується PDF стандарт, а також у якості фону сторінки: штрих-, бар-, QR-коди, етикетки, і все, що вам потрібно! І можна поширювати сам Генератор звітів безкоштовно з додатками.

WinDev поставляється с генератором допомоги в "HLP" і "CHM" форматах. Цей генератор дозволяє забезпечити професійну допомогу з додатком, яка покращить користувачам ефективність роботи. За запитом WinDev також вставляє "?" в меню вашої програми, яке містить посилання на довідкові системи, покажчики до списку всіх нових функцій, параметри резервного копіювання тощо.

WinDev пропонує інноваційні та потужні рішення для підтримки іноземних мов (Західні набори символів, nonWestern набори символів-Unicode UTF8-). Для програми, яка орієнтована на Північно-Американський континент, англійська, іспанська та французька мови, а там вже нові країни такі як: Китай, Росія та інші.

МАКСИМАЛЬНИЙ ПОТЕНЦІАЛ БД	
Columns per table Index per table Rows per table Size of a table Size of a row Size of a column	65,535 65,535 329 millions of billions 4 millions of tera 2 Go 65,535 bytes

9.12.2. Деякі характеристики WLanguage

1. WLanguage являє собою інтегровану мову програмування 5GL. Функції WLanguage дозволяють створювати процеси для кожного проекту, вікна, сторінки та контроли. (***Контрол*** – це компонент, який представляє частину інтерфейсу користувача (GUI). Аналогом англомовного терміну *control* в українській мові є «елемент управління», проте такий переклад довше, тому частіше використовувати, може і не дуже вдало, калька *контрол*). WLanguage функції можуть бути написані малими або великими літерами, або в будь-якій комбінації прописних-малих літер. Редактор коду визнає і те й інше. Редактор коду відображає різні WLanguage ключові слова у різних кольорах. Ця функція дуже корисна при перевірці на помилки, які були зроблені на стадії введення імен функцій.

2. Для кожного об'єкту, що був створений WinDev, або WebDev, WinDev Mobile, існує можливість зв'язати один або декілька процесів, які реагуватимуть на Дії даної події. Не треба нічим керувати, потрібно тільки ввести процес. У WebDev процеси можуть бути введені:

- в коді сервера: процес буде запущений на сервері;
- в коді браузера: процес буде виконуватися в браузері, а відображатися на комп'ютері веб-користувача (WLanguage код, введений в браузер, автоматично буде переведений на JavaScript).

Для управління редагуванням, існує можливість визначити:

- процес, який запускається при ініціалізації контрола;
- процес, який запускається при закритті (завершенні) контрола;
- процес, який запускається при виході з управління;
- процес, який виконується при виході, коли управління буде змінено;
- інші процеси, пов'язані з іншими подіями, які можуть бути безпосередньо додані в редакторі коду.

3. WLanguage складається з ключових слів, які являють собою:

- функції;
- властивості;
- заданих констант;
- визначення для оголошення змінних;

- визначення для оголошення функцій та процедур;
- ключові слова;
- змінні стану тощо.

4. WLanguage підтримує об'єктну парадигму програмування.

5. Всі WLanguage ключові слова (функції, задані константи, властивості тощо) існують також французькою мовою.

6. Використання інших мов: Java, C++, C#, VB, PASCAL, COBOL, FORTRAN. Можна використовувати сторонні мови в WinDev. З сторонньіх мов можна викликати WinDev об'єкти (вихідний код, вікна тощо). Ваш успадкований код зберігається. З WinDev можна також викликати програми, які написані на інших мовах (скомпільовані DLL-бібліотеки або збірки).

7. Візуальний редактор запитів дозволяє створювати SQL запити, які будуть використані у ваших програмах. Запит буде створений на природній мові, так що можна переконатися, що він відповідає тому, що ви хотіли, без ризику на помилки.

9.13. Технологія MODEL CHECKING

Розробка паралельних та розподілених багатопоточних програм вимагає особливої ретельності та синхрогвзації роботи окремих частин програми. Це обумовлено великою кількістю помилок, які виникають при розробці таких програм. Особливо це актуально для бортових систем управління літаком або космічним кораблем. Паралельні та розподілені системи можуть працювати коректно довгий час і роками зберігати 'тонкі' помилки, які можуть проявитись у виключних ситуаціях. Такі помилки призводили до крупних аварій і катастроф. Нагадаємо найбільш гучні з них. Так 4 червня 1966 р. на 39 секунді при першому запуску вибухнула ракета «Аріан 5» – аналог російської ракети «Протон». Бортова система ракети «Аріан 5» використовувала ПЗ попередньої версії «Аріан 4», але архітектура апаратної частини була змінена. Після проведеного ретельного розслідування було виявлено помилку в навігаційній програмі бортового комп'ютера. Втрати були величезними. Це ракета, вартість якої перевищувала 7 млрд доларів, встановлене на борту наукове обладнання вартістю більш ніж півмільярда доларів, До цього треба додати втрати від недоотриманої вигоди, а також втрати репутації ракети-носія. Непрямі втрати склали біля 2 млрд. доларів.

25 грудня 2004 р. після семирічного польоту космічний зонд «Гюйгенс» відділився від автоматичної міжнародної станції «Кассіні» і почав спуск на поверхню Титану – супутника Сатурну. 15 січня 2005 р. зонд вперше в історії почав передавати інформацію з поверхні Титану. Помилка в бортовій системі призвела до того, що половина переданої інформації була втрачена. Загальні затрати на розробку складали 2,5 млрд. доларів.

20 грудня 1995 р. зазнав катастрофи літак «Боінг» 757, загинуло 159 людей. Розслідування виявило помилку в одному символі в програмній системі

управління польотом. Компанії-виготовники бортового комп'ютера та ПЗ були визнані винними в загибелі людей, родичам жертв було виплачено 300 млн. доларів.

23 березня 2003 р. система «Patriot» помилково ідентифікувала британський бомбардувальник «Tornado» як наближаючу ворожу ракету, та автоматично виконала залп. Загинули 2 пілоти. 2 квітня 2003 р. системою «Patriot» був знищений американський винищувач F-16, пілот загинув. В обох випадках причиною були помилки в бортовій системі автоматичного виявлення цілі та наведення.

Наведені приклади показують, що помилки в складних програмних та апартних системах регулярно виявляються при розробці складних систем.

В наш час, коли все частіше людське життя залежить від роботи автоматичних систем, проблема гарантії коректності програмних та апаратних частин цих систем набуває першочергового значення. Надійність та передбачуваність у поведінці таких систем є більш важливим чинником, ніж усе інше.

Правильність таких програм неможливо довести простим тестуванням, майже на всіх наборах початкових даних вони можуть працювати правильно. Тільки строгий формальний аналіз та верифікація програми можуть виявити помилку.

Формальна верифікація програми являє собою сукупність прийомів та методів формального доведення того, що модель програмної системи задовільняє заданій формальній специфікації. Оскільки будь-що формально довести можна тільки відносно формальної моделі, то аналізуєма система для верифікації та наступної реалізації повинна бути представлена формальною моделлю. Вимоги до кінцевого продукту частіше за все розмиті. Для здійснення верифікації специфікація програми повинна складатись з набору формальних тверджень щодо бажаних властивосткй поведінки системи. Для формальної специфікації застосовується мови логики, кожне твердження якої може бути істиним або хибним для програмної системи, яка перевіряється. Верифікація передбачає перевірку виконання заданих формальних тверджень на данній формальній моделі. В області верифікації програм впродовж десятиліть робились серйозні зусилля по розробці теорії, алгоритмів та техніки верифікації. На даний час ці методи розробляються у трьох основних напрямках:

- дедуктивна верифікація;
- перевірка еквівалентності;
- переваірка моделі (model checking).

Дедуктивна верифікація – це перевірка правильності програми, яка зводиться до доведення теорем у підходящій логічній системі. Цей напрямок розробляється вже більше 40 років, з тих часів, коли Роберт Флойд та Ентоні Хоар вперше поставили проблему доведення правильності програм. Верифікація програм при цьому виконується дедукцією на основі аксиом та

правил виводу. Це досить складна процедура не може бути повністю аітоматизована, вона вимагає участі людини, яка діє на основі здогадок та передбачень, та використовує інтуіцію при побудові інваріантів та нетривіальному виборі альтернатив.

Перевірка еквівалентності пов'язана з розробкою формальних моделей взаємодіючих процесів, поданням в рамках таких моделей, як специфікації, так і реалізації та перевіркою еквівалентності поведінок формально визначених моделей поведінки. Цей напрямок був розпочатий працями Робіна Міллера розробкою ним алгебри процесів *Обчислення взаємодіючих систем* (Calculus of Communicating systems – CCS) біля 30 років тому.

Метод model checking [16] пов'язаний з формальною перевіркою виконання на моделі реалізації властивостей поведінки, які специфіковані на мові формальної логики. Цей метод розроблявся біля 20 років. Метод model checking може бути повністю автоматизований. Model checking – це метод перевірки того, що на даній формальній моделі системи задана логічна формула виконується (приймає істинне значення). Ця техніка була розроблена для моделей систем переходів і так званих «темпоральних» логик. Для реалізації за цим методом реальна система подається простою моделлю у вигляді кінцевого автомату з системою переходів та кінцевою кількістю станів. При цьому кожен стан описується так званими структурами Крипке, які враховують всі стани моделі, початковий стан, правила переходу в інші стани із застосуванням темпоральної логики.

На даний час ця методика розробки складного ПЗ має промислове впровадження та підвищений рівень довіри і широко застосовується при розробці складних систем. Вчені Е. Кларк, А.Емерсон та Дж. Сифакис були були нагороджені у 2007 р. премією імені А. Т'юринга за їх внесок в доведення метода model checking до виробничої технології, яка дозволяє здійснювати верифікацію моделей реальних програмних та апаратних систем. В багатьох роботах наводяться результати використання цієї технології для верифікації програмно-апаратних систем у критичних застосуваннях.

9.14. NeoBook – программирование для непрограммистов

Якщо вам потрібно написати невелику програму, або швидко автоматизувати якесь підручне завдання, або необхідно записати свій DVD-диск з колекцією програм (фотографій) і хотіли б гідно оформити заставку до нього, то звичайно, це просто зробити, якщо ви програміст. Але як бути, якщо програмування не ваше покликання, а написати додаток потрібно швидко і якісно?

Велике коло подібних завдань дозволяє вирішити NeoBook – програма-конструктор, що дозволяє створювати мультимедіа-програми і не тільки. При цьому врахуйте: у програмуванні з її допомогою зможе розібратися будь-хто –

в лічені години. Як констатує відомий серед програмістів афоризм: «як тільки ви зрозуміли, як писати програми, змусьте робити це кого-небудь іншого».

І якщо тільки ви не великий начальник, то цим «кимось іншим» для вас зможе стати NeoBook.

9.14.1. Вступ для секретарок

Всю безліч програмних продуктів сьогодні можна умовно поділити на дві великі категорії Перша – це користувальницький сегмент: програми, створені для звичайних користувачів, число яких не злічити – у кожному випадку це спроба створювати готовий універсальний інструмент для всього різноманіття можливих (і неможливих, часом теж) завдань нашого життя. Сильний мінус такого поширеного підходу – низька індивідуалізація та облік особистих потреб кожного конкретного користувача, що часто приводить його (користувача) до нескінченного пошуку неіснуючого ідеалу.

Другий, протилежний клас програмних продуктів, – це набір професійних інструментів для розробки своїх власних програм, який традиційно використовують у своїй повсякденній роботі IT-професіонали, яких в народі прийнято називати програмістами. Так, програміст вольний створити те, що він задумав і що потрібно саме йому, враховуючи всі нюанси і тонкощі особистого досвіду і його індивідуальних потреб. Але – професія програміста вимагає тривалого навчання і серйозної попередньої підготовки, та й для якісного результату не обійтися без дорогоцінного досвіду, який здобувається не за один місяць, і не під час прочитання навіть самих просунутих книг і мануалів.

Як же бути: вибрати готовий напівфабрикат, який часто в чомусь не влаштовує, або засукавши рукава почати гризти граніт «програмістських наук», витрачаючи значні сили і час, для створення чергового «велосипеда»?

Зараз ж мова піде про інше, про більш просте, легке та доступне рішення, що дозволяє буквально в потоковому режимі створювати свої найпростіші програми, наприклад, мультимедійні заставки-презентації своіх DVD-дисків. Для його вивчення знадобиться від одного дня, до максимум тижня – в залежності від ваших стартових знань про програмування і кмітливості взагалі. Отже, наш огляд присвячений досить відомому програмному конструктору такого роду – NeoBook.

9.14.2. NeoBook

NeoBook 5 – це професійний інструмент-конструктор для швидкого і легкого створення невеликих мультимедіа-програм практично професійної якості. NeoBook дозволяє легко об'єднати воєдино в рамках однієї Windows-програми: текст, зображення, різні звуки, музику, мультиплікації, забезпечити управління усім цим багатством за допомогою діалогових засобів Windows, та інших стандартних графічних елементів Windows, щоб у результаті отримати свій унікальний мультимедіа-шедевр.

NeoBook також можна застосовувати в створенні сучасних електронних журналів (e-zines), які в подальшому можна буде поширювати по мережі Інтернет, в створенні інтерактивних і освітніх навчальних матеріалів. Також в програмі є засіб пошуку друкарських помилок (спеллер), яке почасти сприяє тому, щоб ваші публікації були без помилок. І що особливо важливо: закінчений проект можна скомпілювати в повноцінний і автономний додаток Windows (exe-файл), також тут же є вбудована можливість створити програму – інсталятор для вашого проекту.

9.14.3. Можливості та області застосування

Давайте перерахуємо всі ці «корисності» одним списком, для більшої читабельності і наочності, бо можливостей у цієї програми є незліченна кількість. NeoBook 5 можна використовувати для створення різних типів додатків, таких, як:

1. Електронні книги (e-books), можливо з мультимедіа-начинкою.
2. Інтерактивні тести (освітні лінгвістичні або психологічні тощо).
3. Інформаційні оболонки (заставки) для CD/DVD-дисків.
4. Сберігачі екранів з потрібною вам функціональністю (Screen Savers).
5. Невеликі утиліти різного призначення.
6. Демонстраційні програми.
7. Навчальні матеріали.
8. Мультимедійні проекти.
9. Електронні рекламні буклети.
10. Каталоги.
11. Електронні резюме.
12. Персональні електронні кастомізовані щоденники.
13. Термінали.
14. Слайд-шоу, презентації.
15. Простенькі ігри.
16. Створення невеликих і середніх баз даних.
17. І багато, багато іншого!

Тепер розглянемо, що ж такого є в цій програмі для ефективного вирішення всіх перелічених вище завдань і напрямків. Отже, самі найзагальніші риси характерні для розробки будь-яких додатків в NeoBook.

1. Для розробки можна використовувати просте перетягування (drag&drop) програмних і графічних елементів (контролов), щоб з них, подібно мозаїці, створювати складні додатки. Цей підхід аналогічний для стандартних RAD-середовищ типу «народного» Delphi, але реалізований він тут ще простіше – тому для цього не потрібно практично ніяких знань, або вірніше – самий мінімум.

2. Розширені можливості по імпорту зображень та ілюстрацій.

3. Імпорт форматованого тексту, створеного зовнішнім редактором або вбудованим редактором в NeoBook.

4. Зручне додавання навігації по файловій системі і система управління вкладеними меню (наприклад, для фото-браузера або відео-галереї), для чого спочатку грунтовно заточений користувальницький інтерфейс, включаючи: кнопки, прапорці, радіо-кнопки, текстові області входу (з перевіркою правильності), списки, поля зі списком, медіа плеєри, таймери.

5. Зручний запуск і керування анімаційними GIF-зображеннями та Flash-заставками.

6. Динамічне відображення сайтів і живого інтернет-змісту у Вашому додатку. Тобто, інакше кажучи, – дуже просто реалізована можливість безшовної інтеграції сайтів (web-сторінок) і інтернет-сервісів в вашу програму.

7. Є можливість створити і відправляти поштові повідомлення безпосередньо з програми.

8. Досить потужний вбудований редактор тексту, який дозволяє підготувати і відформатувати текст з таблицями, закладками і гіпертекстовими командами.

9. Є власна інтегрована мова для створення сценарію. Включає умовні затвердження, прокрутку, операції по введенню-виводу файлу, обробку тексту тощо. Що цікаво, навіть тут є вибір: команди сценарію можуть бути введені вручну досвідченими користувачами або вибрані зі списку і додані в інтерактивному режимі, проходячи через серію меню-опитувань за типом візарду.

10. Красиві графічні ефекти переходу між екранами: змиття, розпадання, переплетення тощо.

11. Вбудовані можливості по створенню своїх власних мультиплікацій.

12. Запуск зовнішніх Windows або DOS програм, включаючи інші зібрані додатки NeoBook.

13. Програмування натиснення клавіш і клацань миші в інших (сторонніх) додатках Windows, що дає можливість не тільки запускати якісь зовнішні додатки-супутники, а й дистанційно керувати ними. Типовий приклад: ви можете реалізувати автоматизовану покрокову установку якогось додатка, попутно навчаючи і пояснюючи процес його установки. Якщо раніше подібні речі було прийнято просто знімати на відео, то тепер розгортання будь-кого додатки можна проводити буквально на очах користувача в реальних робочих умовах, при цьому забезпечуючи його гідний інформаційно-методичний супровід.

14. Можливість використання свого улюбленого текстового редактора, редактора мультиплікації або можливість створити мультик безпосередньо в NeoBook, щоб редагувати текст, зображення, аудіо та кліпи мультиплікації в єдиному інтерфейсі.

15. Величезний плюс програми – це можливість збільшити функціональні можливості NeoBook через зовнішні плагіни (plug-in – підключати. Додаток, незалежно компільований програмний модуль, що динамічно підключається до основної програми, призначений для розширення або використання її можливостей), яких написано вже досить багато.

16. Забезпечуються можливості тесту, налагодження та перевірки орфографії проектів, не виходячи з середовища NeoBook.

17. Повноцінний компілятор NeoBook для складання проектів у 32-бітові додатки Windows (.Exe), екранні заставки Windows (.Scr), додатки системи Windows або управління web-браузера (Plug-In/ActiveX (.Ocx)). Зібрані додатки після цього стають повністю самостійними і повноцінними додатками.

18. Можливість створення професійної установки/інсталяції для додатків, що підтримують стиснення і безліч налаштувань.

В якості хороших стартових ресурсів для знайомства з можливостями програми і перших кроків у її освоєнні рекомендуємо два російськомовних форуми: NeoBooker.ru Русский Форум і Уроки NeoBook.

9.15. Файлові системи найближчого майбутнього

У травні 2011 року, в доповіді Майкла Рубіна (Michael Rubin), що займається системами зберігання даних в Google, пролунав докладний порівняльний огляд сучасних файлових систем, в якому, окрім перерахування їх переваг і недоліків, були дані і деякі прогнози про майбутній розвиток і потребах інформаційної індустрії в цій сфері. Зокрема, була розглянута здатність існуючих файлових систем (ФС) адаптуватися до викликів вже найближчого майбутнього, у зв'язку з чим, із усіх були виділені чотири провідні файлові системи «нової школи», які були спроектовані з урахуванням задоволення найвимогливіших вимог і запитів.

Серед усіх розглянутих варіантів були ретельно відібрані кращі претенденти, які вже сьогодні (або в найближчому майбутньому) зможуть стати успішним рішенням для великих інформаційних сховищ. Сьогодні ми розглянемо і порівняємо основні плюси і мінуси кожного з цих чотирьох претендентів, щоб допомогти зробити вибір файлової системи майбутнього.

Але перед тим як ми ознайомимося з меню найбільш перспективних файлових систем найближчого майбутнього за версією експертів Google, спробуємо визначитися і розповісти, що має на увазі Google, коли говорить, що ми входимо в нову стадію розвитку IT – «Епоху Великих Даних».

9.15.1. Зетта-повінь настає

Комп'ютерний світ проходить болісний процес еволюційного переходу від обчислювальних систем в їх, якщо так можна сказати, «чистому вигляді» через системи обробки даних і далі до систем управління інформацією. Болісність цього процесу полягає в тому, що на кожному з його етапів ускладнюються не тільки кошти, а й предмет обробки.

З часів клинопису і понині, роблячи короткі записи або створюючи багатотомні романи, люди й не підозрювали, що вони створюють зразки неструктурованих даних. Для того щоб дані стали структурованими, потрібно було винайти комп'ютер.

Коли з'явилися перші комп'ютери, вони використовувалися безпосередньо, відповідаючи своїй назві – як засіб автоматизації розрахунків. Аж до 70-х років застосування комп'ютерів обмежувалося вирішенням математичних завдань, тому достатньо було представляти дані у вигляді масивів чисел, що не представляло особливих труднощів. Потім необхідність оперування даними з більш складною структурою призвела до появи систем управління базами даних. Цей етап розпочався в 70-ті роки і протягнувся до сучасності. На перших двох етапах участь людини обмежувалося підготовкою даних та аналізом кінцевих результатів.

Щоб далі оперувати великими обсягами інформації, наведемо одиниці виміру для зберігання, обробки і передачі даних у різних Міжнародних стандартах, де рекомедуется використовувати спеціальні префікси для десяткових та бінарних ступенів (російською мовою). За пропозицією Міжнародної електротехнічної комісії (МЕК), назва «Эксабайт» загальноприйнята, але невірно, так як приставка *экса-* означає множення на 10^{18}. Правильною для 2^{60} є двійкова приставка *эксби-*.

Вимірювання в байтах

ГОСТ 8.417-2002			**Приставки СИ**		**Приставки МЭК**			
Назва	**Символ**	**Ступінь**	**Назва**	**Ступінь**	**Назва**	**Символ**		**Ступінь**
байт	Б	10^0	-	10^0	байт	В	Б	2^0
килобайт	кБ	10^3	кило-	10^3	кибібайт	KiB	КиБ	2^{10}
мегабайт	МБ	10^6	мега-	10^6	мебібайт	MiB	МиБ	2^{20}
гігабайт	ГБ	10^9	гига-	10^9	гібібайт	GiB	ГиБ	2^{30}
терабайт	ТБ	10^{12}	тера-	10^{12}	тебібайт	TiB	ТиБ	2^{40}
петабайт	ПБ	10^{15}	пета-	10^{15}	пебібайт	PiB	ПиБ	2^{50}
ексабайт	ЕБ	10^{18}	экса-	10^{18}	ексбібайт	EiB	ЭиБ	2^{60}
зеттабайт	ЗБ	10^{21}	зетта-	10^{21}	зебібайт	ZiB	ЗиБ	2^{70}
йоттабайт	ЙБ	10^{24}	йотта-	10^{24}	йобібайт	YiB	ЙиБ	2^{80}

Дослідження Університету Південної Кароліни встановило, що загальний обсяг збережених даних до 2007 року склав 295 ексабайт (295 міліардів гігабайтів).

Дослідники прийшли до таких цифр грунтуючись на даних, що зберігаються в 60 аналогових або цифрових форматах, починаючи від паперу або книг і закінчуючи даними на серверах і кредитних картах. Це оцінка обсягу зберігання даних з 1986 до 2007 років, яка склала 295 ексабайт.

"Якби ми зібрали всю інформацію і помістили її на сторінки книг, то ми б покрили території США або Китаю трьома шарами книг", – сказав Мартін Гілберт з Університету Південної Каліформії в передачі BBC Science in Action.

У доповіді говориться, що відбулася революція, яка перенесла дані з аналогових на цифрові формати. У 2000 році 3/4 всіх даних у світі були в аналогових форматах, але всього лише 7 років потому 94% інформації зберігалось вже в цифрі.

На думку вчених, загальний обсяг інформації, вироблений людьми до 2007 року, будучи записаним на компакт-диски, може утворити стопку висотою більше орбіти Місяця. До 2020 року, за оцінкою дослідницької компанії IDC, "цифровий всесвіт" буде в 44 рази більше, ніж у 2009-му, тобто стопка дисків до цього часу дорівнювала б половині відстані до Марса.

Бурхливе зростання кількості цифрової інформації директор однієї з дослідницьких лабораторій компанії Intel професор Джим Хелд пояснює збільшенням кількості інтернет-сервісів, поширенням мобільних телефонів з якісними відеокамерами і збільшеними вимогами корпоративних інформаційних систем. На доказ своєї думки Хелд навів такі приклади: "Walmart (найбільша торговельна мережа США) додає по мільярду записів на хвилину в свої бази даних. Відеосервіс YouTube зберігає більше даних, ніж всі комерційні телеканали передають за рік".

У 2014 р. обсяг глобального Інтернет-трафіку досягне 767 ексабайт, збільшившись вчетверо. Такі дані дослідження «Індексу розвитку візуальних мережних технологій за 2009-2014 рр..», проведеного компанією Cisco. У ньому зокрема повідомляється, що до 2014 р. обсяг глобального Інтернет-трафіку зросте більш ніж у чотири рази і досягне 767 ексабайт, тобто перевищить 3/4 зеттабайт. Це на 100 ексабайт більше рівня, прогнозованого на 2013 р., і в 10 разів перевищує загальний обсяг трафіку в IP-мережах в 2008 р.

Більшою мірою вся ця лавина інформації – плід невгамовної тяги людей до мультимедіа, особливо, до відео. До 2015 року понад 90% даних у "всесвітній павутині" будуть припадати на відеоконтент. Це створить величезне навантаження на мережі та потребують оптимізації архітектури безпеки, а також підвищення якості послуг передачі даних.

Предбачаючи такий наплив «океана» інформації можна сказати, що ми стоїмо на порозі приходу «Епохи Великих Даних». І якщо говорити про файлові системи (яким і присвячений наш невеликий огляд) – забудьте про NTFS або ext3 в цьому контексті, вони фізично не здатні ефективно обслуговувати сучасні сховища на десятки і сотні петабайт.

9.15.2. Файлова система ZFS

ZFS (Zettabyte File System) – одна з найвідоміших файлових систем, спочатку створена в Sun Microsystems для операційної системи Solaris, перенесення якої на інші платформи викликало сплеск діаметрально різних емоцій у розробників.

Ознайомимося в загальних рисах з її особливостями і властивостями:

1. 128-бітна файлова система, що дає можливість зберігання практично необмежених обсягів інформації. На практиці це означає, що ZFS теоретично може зберігати обсяги інформації, які перевищують сьогоднішні технологічні можливості, за умови використання поточного підходу до організації зберігання даних.
2. Дуже велика увага приділяється цілісності та надійності зберігання, як для користувача даних, так і метаданих ФС, для цього використовуються просунуті алгоритми хешування.
3. Підтримка снапшотів (snapshot – Знімок файлової системи; моментальний знімок, копія файлів і директорій файлової системи на певний момент часу) і пулів зберігання (storage pools), завдяки чому ZFS поєднує в собі можливості файлової системи і системи управління томами (нова концепція storage-пулів).
4. Підтримка автоматичного розпізнавання і об'єднання (виключення) файлів-дублікатів.
5. Можливості для виборчого стиснення і/або шифрування окремих файлів або файлових систем.
6. Традиційно вважається, що ZFS – це досить продуктивна файлова система. Зокрема, конкретні цифри дуже сильно залежать від типу задачі, на якій проводиться подібне порівняльне тестування продуктивності.
7. І багато-багато іншого, оскількі, – ZFS надзвичайно велика у своїх можливостях та особливостях ФС, і перерахувати всі їх тут просто не представляється можливим.

Звичайно, якщо дивитися на ці можливості окремо, то вони багато в чому не нові і зустрічаються в тому чи іншому вигляді в інших файлових системах, але такий єдиний комплекс з наведених можливостей вперше представлений тільки в ZFS, що і робить її настільки унікальною і цікавою на даний момент.

9.15.3. Файлова система Btrfs

Btrfs – це новітня файлова система, яка в даний момент дуже активно розвивається, і вже інтегрована в ядро ОС Linux. У цілому, говорячи гранично коротко, Btrfs – це файлова система, створена спеціально для Linux, заснована на структурах B-дерев і екстентах, і працює за принципом «копіювання при записі» (copy-on-write).

Перш ніж привести список основних технічних можливостей цієї файлової системи, слід підкреслити, що розробка цієї ФС ще в самому розпалі, і, не дивлячись на те, що багато чого вже зроблено – багато можливостей ще далекі від досконалості і логічного завершення.

Ось її основні особливості:

1. Динамічне виділення індексних дескрипторів (динамічні иноди, dynamic inodes). Інакше кажучи, в цій ФС немає максимальної кількості обслуговуваних файлів.
2. Знімки файлової системи (snapshots), а також можливості робити знімки знімків і записувані знімки (writeable snapshots).
3. Для просунутого контролю цілісності, застосовуються хеши на всі дані та метадані (на даний момент в Btrfs застосовується CRC-32C, що істотно скромніше, ніж в ZFS).
4. Дуже корисна можливість міграції з традиційних файлових систем ext3/ext4, що дає можливість дуже прозорого переходу на цю нову ФС.
5. Перевірка файлової системи в робочому режимі (online) і дуже швидка перевірка в неробочому режимі (offline).
6. Прозорі можливості виборчого стиснення файлів.
7. Підтримка режиму роботи як RAID-масиву (одна файлова система на декількох томах).
8. І багато іншого.

І хоча ця ФС поки активно розвивається, в якості ілюстрації точок її майбутнього зростання і масштабності загальної задумки, як приклад можна привести рішення компанії Oracle реалізувати мережевий протокол нової розподіленої файлової системи CRFS (фактично заміняє застарілі NFS і CIFS), який буде спроектований і оптимізований повністю під btrfs.

9.15.4. Файлова система Hammer

Hammer – це 64-бітна кластерна файлова система побудована на B-деревах, створена спеціально для свого проекту DragonFly BSD відомим гуру з FreeBSD Project, – Меттью Діллоном (Matthew Dillon).

Перелічимо основні можливості Hammer, які доступні вже на даний момент (або реалізація яких близька до завершення):

1. Hammer – це файлова система доступна негайно навіть після падіння і перезавантаження системи.
2. Розмір ФС Hammer може досягати розміру до 1 екзабайта (1 мільярд гігабайтів), і може при цьому вміщати в себе до 256 томів, кожен з яких може досягати розміру до 4 петабайт (4096 терабайт).
3. Можливість відкату будь-якої дискової операції і повернення стану ФС в певну точку.
4. Метод грубозернистої історії реалізується через миттєві знімки ФС (снапшоти) . За замовчуванням, системний крон (cron – демон-

планувальник завдань у UNIX-подібних операційних системах) генерує один снапшот в день, який зберігається в протягом 60 днів. Кількість і частота снапшотов необмежена. Всі збережені снапшоти індексуються також за допомогою B-дерева таким чином, щоб зробити їх зберігання на носіях максимально ефективним.

5. Можливості для створення псевдо-файлової системи (PFS) всередині файлової системи Hammer. Можна створити до 65535 таких файлових систем.

Не зайвим буде ще раз підкреслити, що Hammer у своїй стабільній версії доступній на даний момент лише на своїй рідній DragonFlyBSD (також є експериментальний FUSE-модуль для Linux, який дозволяє працювати з цією ФС в режимі read-only).

Ось такий загальний план огляду по «великим» файловим системам, які готові до прийому так званих Зетта-даних.

10. Історія операційних систем

Операційна система (ОС) – найважливіша частина системного програмного забезпечення. ОС – це сукупність програмних засобів, яка забезпечує зручний інтерфейс користувача з ЕОМ та організує і обслуговує обчислювальний процес. Без ОС неможлива робота комп'ютера. Власно кажучи, сучасний користувач не бачить саму машину, він спілкується тільки з ОС. Проте так було не завжди. Операційні системи пройшли тривалий та складний шлях еволюції, аналіз цього шляху виключно корисний для розуміння сучасного стану та тенденцій розвитку ОС.

Історія розвитку операційних систем (ОС) тісно пов'язана з історією розвитку комп'ютерів, оскільки ОС з'явилися та розвивалися у процесі конструювання комп'ютерів. Тому, щоб уявити, як виглядали ОС, ми ще раз стисло обговоримо послідовні покоління деяких моделей комп'ютерів, для яких, в основному, створювалися ОС. Така схема взаємозв'язку поколінь ОС та комп'ютерів забезпечує певну структуру, без якої нічого не було б зрозуміло.

10.1. Послідовна обробка даних

Перше покоління комп'ютерів (1943-1955): механічні реле, електронні лампи та комутаційні панелі. Про операційні системи ніхто і не чув, програми безпосередньо взаємодіяли з апаратним забезпеченням машини. Ці комп'ютери керувалися безпосередньо з пульта управління (панелі), що складався з сигнальних ламп та тумблерів. Звичайний режим роботи програміста був такий: записатися на певний час, потім спуститися у машинну кімнату (або залу), вставити свою комутаційну панель у комп'ютер (або пізніше, на загальну панель комп'ютера) та проводити різні маніпуляції на панелі для розв'язку свого завдання.

На початок 50-х, з випуском перфокарт, ситуація дещо покращала. Стало можливо замість використання комутаційних панелей записувати та прочитувати програми з карт, але в усьому іншому процедура обчислень залишалася незмінною. Фактично тоді на комп'ютерах займалися тільки прямими числовими обрахунками.

У перші роки, коли машини тільки вчилися обчислювати, ними управляли самі програмісти. Кожному користувачеві відводився певний час. Програміст сідав за пульт і залишався з машиною наодинці. Ніяких операційних систем та файлів на диску, – «все своє ношу із собою». І дійсно, програмісти носили з собою стягнуті гумками колоди перфокарт, а якщо їх було багато, то в скриньках, які були зроблені найчастіше з фанери. Носили з собою і всі інші «причандали»: паперову «тирсу» від пробивки отворів на перфокартах, аби у будь-який час можна було заклеїти (звичайно ж нігтем) «зайвий» отвір для отримання іншого коду; ножик або шило для проколювання отворів на перфокарті і таке інше.

Перша дія програміста за пультом – натиснення кнопки «Стирання ОЗУ», потім у порожню машину вводилася колода перфокарт, і управління передавалося у комірку за адресою 0001. Коли програма закінчувала роботу, машина зупинялася і спалахувала лампочка «Останов», а якщо виникала особлива ситуація – ділення на нуль або переповнення, то спалахувала лампочка аварійної зупинки «Авост». Ось і все управління обчислювальним процесом.

За таких умов відлагодження програм було дуже повільною та малопродуктивною справою. Поки програміст думав, чому відбувся авост, машина простоювала, дорогоцінний час витрачався даремно. На самому початку 50-х років це було ще стерпно, але незабаром політика відкритих дверей себе вичерпала. Машин було ще мало, а програмістів стало дуже багато, буквально кожна хвилина машинного часу була на рахунку. Думати за пультом ЕОМ стало неприпустимою розкішшю. Двері машинних залів наглухо закрили, туди допускалися тільки люди з екзотичною професією «оператор ЕОМ».

Програміст заздалегідь розмірковував та планував усі дії з відлагодження програми, писав розлогу інструкцію операторові і разом з колодою перфокарт здавав її диспетчерові обчислювального центру. Диспетчер відповідно до пріоритетів формував «живу чергу» завдань, вкладав колоди одну за одною у спеціальні довгі та вузькі скрині і відносив їх у машинну залу операторові.

Робота оператора була неймовірно напруженою. Підходячи до ящика із завданнями, він брав чергову колоду, вставлював її у зчитуючий пристрій. Потім, відповідно до інструкції, набирав на пульті команди, записував покази лампочок, змотував в рулон “видачу” з пристрою друку та разом з відпрацьованою колодою складав усе у скриню для оброблених завдань, яка поверталась назад у диспетчерську обчислювального центру.

Не дивлячись на всі старання операторів, уникнути втрат машинного часу все одно не вдавалося. Поки оператор бігав від введення до виведення, машина простоювала. До того ж він часто припускався помилок, тому природною та необхідною стала постановка завдання про автоматизацію праці операторів.

Такий режим роботи можна назвати **послідовною обробкою даних**. Цю назву відображає той факт, що призначені для користувача програми виконувалися на комп’ютері послідовно. Через деякий час у спробі підвищити ефективність послідовної обробки були розроблені різні системні інструменти. До них належать бібліотеки функцій, редактори зв’язків або компоновники, завантажувачі та драйвери введення-виведення, що існують у вигляді програмного забезпечення, яке є загальнодоступним для всіх користувачів.

10.2. Прості пакетні системи

Друге покоління (1955-1965): транзистори та системи пакетної обробки. В середині 50-х винахід та застосування транзисторів радикально змінили всю картину. Комп’ютери стали достатньо надійними, з’явилася висока ймовірність того, що машина працюватиме досить довго, виконуючи

при цьому корисні функції. Саме у цей період відбулося розділення персоналу на програмістів та операторів, експлуатаційників та розробників обчислювальних машин.

Машини, так звані **мейнфрейми**, розміщувалися у спеціальних кімнатах, де ними керував цілий штат професійних операторів. Тільки великі корпорації могли дозволити собі техніку, ціна якої обчислювалася мільйонами доларів. Щоб виконати завдання, програміст спочатку повинен був записати його на спеціальні бланки (на Фортрані або Асемблері), а потім перенести на перфокарти. Після цього принести колоду перфокарт у кімнату введення даних та йти чекати результатів. Оператор брав одну з колод перфокарт і вводив у комп'ютер.

Якщо враховувати високу вартість устаткування, то не дивно, що люди досить швидко зайнялися пошуком способу підвищення ефективності використання машинного часу. Простої, які відбувалися через неузгодженість розкладу, а також час, який був витрачений на підготовку та розв'язання задачі, – все це обходилось дуже дорого. Загальноприйнятим рішенням стала **система пакетної обробки**. Задум полягав у тому, щоб зібрати декілька завдань (колод перфокарт), переписати їх на магнітну стрічку (барабан), потім оператор завантажував спеціальну програму (прообраз сьогоднішньої ОС), яка прочитувала перше завдання із стрічки і запускала його. Вихідні дані видавалися або протягом роботи, або також записувалися на стрічку, а потім окремо роздруковувалися на дешевшому комп'ютері або спеціальному пристрої.

У ході реалізації систем пакетної обробки була створена та формалізована **мова управління завданнями**, за допомогою якої програміст повідомляв систему, яку роботу він хоче виконати на обчислювальній машині. Сукупність кількох завдань, як правило, у вигляді колоди перфокарт, отримала назву пакету завдань.

Головна ідея, яка була покладена в основі простих пакетних схем обробки, полягала у використанні спеціальної програми, відомої під назвою **монітор** або **супервізор** (на вітчизняних машинах – **диспетчер, автооператор**). Використовуючи ОС такого типу, користувач не мав безпосереднього доступу до машини.

Робота з погляду монітора виглядала таким чином. Монітор керував послідовністю подій. Щоб це було можливо, основна його частина розміщувалась в основній пам'яті і завжди була готова до роботи. Цю частину називали **резидентним монітором**. Решта частини складала утиліти та загальні функції, які завантажувалися у вигляді підпрограм (зазвичай з барабанів або магнітних стрічок), які призначались для користувача програмами та монітором. Монітор прочитував з пристрою введення-виведення поточне завдання і розміщував його в області пам'яті, яка призначалась для програм користувача, та передавав йому управління. Після завершення завдання управління поверталось монітору, який одразу ж починав читати наступне завдання.

З погляду процесора він у деякий момент виконував команди монітора. Монітор після читання завдання віддавав процесору команду переходу, за якою процесор повинен був почати виконувати програму користувача. Процесор переходив до обробки програми користувача і виконував її команди доти, доки не доходив до кінця або поки не виникала аварійна ситуація.

Таким чином, монітор, або пакетна ОС, – це звичайна комп'ютерна програма. Її робота була заснована на здатності процесора обирати команди з різних областей пам'яті, при цьому відбувалась передача управління програмі та повернення управління у монітор.

10.3. Багатозадачні пакетні системи

При використанні монітора бажано було б застосовувати також й інші можливості апаратного забезпечення, а саме:

- **захист пам'яті**. Під час роботи програми користувача вона не повинна вносити зміни в область пам'яті монітора. Якщо це трапиться, то апаратне забезпечення процесора повинно виявити помилку та передати управління монітору;
- **таймер**. Таймер використовується для того, щоб запобігти ситуації, коли одне завдання захопить безмежний контроль над системою. Таймер виставляється на початку кожного завдання, і після закінчення певного проміжку часу програма користувача зупиняється, а управління передається монітору;
- **привілейовані команди**. Деякі команди повинні мати підвищені привілеї і виконуватися тільки монітором. Це, наприклад, команди введення-виведення. Якщо програмі користувача потрібно здійснити введення-виведення, вона повинна запитати для виконання цих операцій монітор;
- **переривання**. Ця можливість надає ОС велику гнучкість при передачі управління програмі користувача та її відновленні.

Постачальники комп'ютерів швидко зрозуміли, що без таких можливостей апаратного забезпечення навіть прості пакетні ОС можуть привести до хаосу.

Третє покоління (1965-1980): інтегральні схеми та багатозадачність. На початку 60-х років у світі переважали комп'ютери типу IBM-7094 для числових розрахунків в науці та техніці, та IBM-1401, які широко використовувались банками для сортування та друку даних. Розвиток та підтримка цих двох ліній для виробників були достатньо дорогим задоволенням. Тому фірма IBM спробувала вирішити ці проблеми разом, випустивши єдину серію універсальних машин IBM/360, які повинні задовільняти потребам усіх покупців. Машини мали однакову структуру та набір команд, розрізнялися вони тільки ціною та продуктивністю (об'ємом пам'яті, швидкодією процесора, кількістю пристроїв введення-виведення і т.і.), і були програмно-сумісними знизу догори, тобто від простих моделей до більш складних.

Корпорація IBM досягла миттєвого успіху, а ідею родини сумісних комп'ютерів прийняла і решта всіх основних виробників. Програмна сумісність вимагала і сумісності операційних систем. Такі операційні системи повинні були б працювати і на великих, і на малих обчислювальних системах, з великою та з малою кількістю різноманітної периферії, у комерційній області та в області наукових досліджень. Тому, щоб управляти всією цією родиною комп'ютерів, була створена єдина операційна система OS/360, яка повинна була добре працювати на всіх цих моделях.

Ні IBM, ні інша фірма не могли написати ПЗ, яке задовольняло б багатьом суперечливим вимогам. В результаті з'явилася величезна та надзвичайно складна ОС. Вона складалась з мільйонів рядків, які були написані на Асемблері тисячами програмістів, і містила тисячі помилок. Не дивлячись на свої величезні розміри та недоліки, OS/360 та подібні до неї ОС 3-го покоління насправді непогано задовільняли вимогам клієнтів.

Іншим важливим плюсом ОС 3-го покоління стала здатність читати завдання з перфокарт на диск по мірі того, як їх приносили до машинної зали (або зі стрічки), як тільки закінчувалося поточне завдання. Цей технічний прийом називається "**підкачкою"** даних або **спулингом** (**spooling**, від Simultaneous Peripheral Operation On Line – миттєва периферійна операція в інтерактивному режимі). З появою підкачки стали більше не потрібні спеціальні пристрої та багаторазові перенесення магнітних стрічок.

Ефективність роботи процесора в ОС 3-го покоління значно підвищилася. Тепер у пам'яті комп'ютера було достатньо місця для операційної системи та двом або більше програмам користувачів. Тепер, коли одне із завдань чекає завершення операцій введення-виведення, процесор може перемикатися на інше завдання і так далі. Такий режим відомий як **багатозадачність** (multiprogramming) і є основною рисою в сучасних ОС.

Робота багатозадачної пакетної системи та деяких простих пакетних систем базувалася на певних апаратних особливостях комп'ютера. Найзначнішим доповненням, яке стало корисним для багатозадачності, з'явилося апаратне забезпечення, що підтримує переривання введення-виведення. Використовуючи цю можливість, процесор генерує команду введення-виведення для одного завдання та переходить до іншого на той час, поки контролером пристрою здійснюється введення-виведення. Після завершення операції введення-виведення процесор отримує переривання, і управління передається програмі обробки переривань зі складу ОС. Потім управління передається іншому завданню.

Багатозадачні ОС складніші за системи послідовної обробки завдань, хоча б через те, що, для того, щоб обробити декілька завдань одночасно, вони повинні знаходитись в основній пам'яті, а для цього потрібна система **управління пам'яттю**.

10.4. Системи з режимом розподілу часу

При системах 3-го покоління по суті ще залишався різновид пакетної обробки. Багато програмістів сумували по першому поколінню машин, коли двері машинних залів були для них відкриті, і коли вони могли годинами розпоряджатися всією машиною і досить швидко відлагоджувати свої програми.

Бажання скоротити час очікування відповіді призвело до розробки режиму розділення часу – варіанту багатозадачності, при якому у кожного користувача є свій термінал. Якщо двадцять користувачів зареєстровано в системі, яка працює в режимі розподілу часу, а 17 з них замислюються, розмовляють або п'ють каву, то центральний процесор (ЦП) по черзі надається трьом користувачам, які працюють у цей час на машині, і забезпечує швидке інтерактивне обслуговування цих користувачів. При цьому він мав можливість працювати над великими пакетами у фоновому режимі, коли ЦП не зайнятий іншими завданнями.

Перша серйозна система з режимом розподілу часу **CTSS** (Compatible Time Sharing System – Сумісна система розподілу часу) для управління обчислювальними ресурсами комп'ютера IBM 7090 була розроблена на початку 60-х років групою програмістів під керівництвом професора **Фернандо Корбато** в Массачусетському технологічному інституті (MTI). Головним розробником (ідеологом) був математик **Джон Маккарті (**ця ідея була висловлена ним ще у 1959 році).

Після успіху системи CTSS та ж група з MTI, дослідницька лабораторія Bell Labs та корпорація General Electric у 1965 вирішили почати розробку машини, яка повинна була підтримувати одночасно сотні користувачів у режимі розподілу часу. Проектувальники цієї системи, які відомі як **MULTICS** (MULTiplexed Information and Computing Service – мультіплексна інформаційна та обчислювальна служба), уявляли собі одну величезну машину, скористатися послугами якої могла кожна людина в районі Бостона. Думка про те, що набагато могутніші машини, ніж їх менфрейм (мейнфрейм) GE-645, продаватимуться мільйонами за ціною тисяча доларів та менше всього лише через тридцять років, здавалася чистісінькою науковою фантастикою.

Існувало багато причин, через які система MULTICS не захопила весь світ. Не останню роль зіграв той факт, що система була написана на мові високого рівня PL/1, а нормальний компілятор мови PL/1 з'явився лише через декілька років. В результаті система MULTICS подала багато конструктивних ідей комп'ютерним теоретикам, але перетворити її на серйозний продукт та досягнути комерційного успіху виявилося набагато важчим, ніж очікувалося. Саме розробники MULTICS почали застосовувати термін "процес" у його сучасному значенні в контексті ОС.

За свій внесок у розробку CTSS та MULTICS Ф. Корбато у 1990 році був нагороджений престижною премією Тюрінга.

Незабаром Bell Labs вибула з проекту, а корпорація General Electric зовсім залишила комп'ютерний бізнес. Проте Массачусетський технологічний інститут (MTI) проявив завзятість і з часом отримав працюючу систему. Вона була продана як комерційний продукт компанії Honeywell, яка купила комп'ютерний бізнес у General Electric, та встановлена приблизно у вісімдесяти великих компаніях та університетах по всьому світу. Не дивлячись на свою нечисленність, користувачі системи були відчайдушно віддані їй. Так, наприклад, компанії General Motors, Ford залишили свої системи у кінці 90-х років, через 30 років після виходу системи. Останній комп'ютер під управлінням MULTICS працював до 2000 року.

Не дивлячись на невдачу з погляду комерції, система MULTICS значно вплинула на подальші операційні системи. Ще одним важливим моментом розвитку за часів 3-го покоління було феноменальне зростання міні-комп'ютерів, починаючи з випуску PDP-1 корпорацією DEC у 1961 р. Ці комп'ютери володіли оперативною пам'яттю, що складалась всього лише з 4 k 18-бітових слів, але коштували вони по 120 тисяч доларів (близько 5% від ціни IBM-7094) і тому розхоплювалися як гарячі пиріжки. За цією машиною з'явилась ціла серія інших, аж до – PDP-10.

Кен Томпсон (Ken Thompson), один з розробників MULTICS з Bell Labs, згодом знайшов міні-комп'ютер PDP-7, яким ніхто не користувався, і вирішив написати урізану однокористувацьку версію системи MULTICS, яка пізніше розвинулася до ОС **UNIX** і яка стала популярною в академічному світі та в урядових закладах. Надалі ще один дослідник лабораторії Bell Labs, **Брайан Керніган**, якось жартома назвав цю систему **UNICS** (UNiplexed Information and Computing Service – примітивна інформаційна та обчислювальна служба). Призвисько, яке було надане Керніганом, міцно закріпилося за новою системою, хоча написання цього слова стало трохи коротше при вимовлянні, перетворившись на **UNIX.**

Робота Томпсона справила на його колег з Bell Labs таке сильне враження, що незабаром до нього приєднався **Денніс Рітчи**, а трохи пізніше і весь його відділ. Незабаром система UNIX була перенесена на могутніші комп'ютери PDP-11/20,45,70. У 1974 р. Рітчи та Томпсон опублікували статтю про ОС UNIX, яка стала важливою віхою в історії комп'ютерів. За цю роботу асоціацією з обчислювальній техніки (ACM) їм була присвоєна премія Тюрінга.

10.5. Основні досягнення

Четверте покоління (з 1980 року): персональні комп'ютери. Наступний період в еволюції ОС пов'язаний з появою Великих Інтегральних Схем (ВІС або LSI – Large Scale Integration) – кремнієвих мікросхем. З погляду архітектури персональні комп'ютери (спочатку звались **мікрокомп'ютерами**) були багато у чому схожі на міні-комп'ютери класу PDP-11, але відрізнялися за ціною, що дозволило відділам компаній та факультетам університетів мати власні комп'ютери.

У 1974 р. компанія **Intel** випустила Intel 8080 – перший універсальний 8-розрядний процесор. Для нього була потрібна ОС, за допомогою якої можна було б протестувати новинку. Компанія Intel залучила до розробок по створенню потрібної ОС одного зі своїх консультантів **Гері Килдолла** (Gary Kildall). Спочатку Килдолл з другом сконструювали контролер для 8-дюймового гнучкого диска, який був нещодавно випущений компанією Shugart Associates, та підключили цей диск до процесора Intel 8080. Таким чином, з'явився перший мікрокомп'ютер з диском. Потім Килдолл створив дискову операційну систему, яку назвав **CP/M (Control Program for Microcomputers**).

У 1977 р. компанія Digital Research переробила CP/M, щоб зробити цю систему придатною для роботи на мікрокомп'ютерах не тільки з процесорами Intel 8080, а також з іншими процесорами. Потім було створено безліч прикладних програм, які працювали у CP/M, що дозволило CP/M займати високу позицію у світі мікрокомп'ютерів впродовж 5 років.

На початку 80-х корпорація IBM розробила IBM PC і почала шукати для нього програмне забезпечення. Фірма IBM почала вести переговори з Килдоллом про розробку нової ОС, але коли він відлетів у двотижневу відпустку, співробітники IBM зв'язалися з Білом Гейтсом (Bill Gates), щоб отримати ліцензію на право використання інтерпретатора мови Бейсік. Вони також поцікавилися, чи не знає він ОС, яка працювала б на PC. Гейтс порадив звернутися до Digital Research. Фірма IBM почала вести переговори з Килдоллом про розробку нової ОС, але Килдолл запропонував продовжити переговори після його відпустки. Корпорація IBM знову звернулася до Біла Гейтса з проханням забезпечити її ОС.

Після повторного запиту IBM Гейтс з'ясував, що у місцевого виробника комп'ютерів, Seattle Products, є відповідна ОС **DOS (Disk Operation System)**. Він запропонував компанії викупити DOS за 50 тис. доларів. Коли корпорація IBM захотіла деяких удосконалень у програмі, Біл Гейтс запросив для цієї роботи **Тіма Патерсона** (Tim Paterson), який написав DOS. Фірма IBM уклала контракт з компанією – "**Microsoft"**, яку очолював Білл Гейтс, на розробку ОС.

Нова видозмінена система, перейменована на **MS-DOS (Microsoft Disk Operation System)**, була добре сприйнята багатьма виробниками персональних комп'ютерів сумісних з IBM PC. Система MS-DOS, створена компанією Microsoft, узурпувала місце, яке раніше належало CP/M, завоювавши до того ж ширший ринок. Коли у 1983 р. з'явився IBM PC/AT з центральним процесором Intel 80286, система MS-DOS вже міцно стояла на ногах. Пізніше система MS-DOS широко використовувалася на комп'ютерах з процесорами 80386 та 80486. Хоча первинна версія MS-DOS була досить примітивна, подальші версії системи виходили з усе краще розробленими властивостями, включаючи багато чого, що було запозичено від CP/M та UNIX.

На 1989 рік операційна система Macintosh MacOS не мала собі рівних за простотою використання. Такий величезний успіх компанія мала через те, що на MACOS працював дружній інтерфейс, тобто призначений для користувачів, які нічого не знали про комп'ютери і, більш того, зовсім були не охочі чому-

небудь навчатися. Правда, до цього часу фірмі IBM вдалося власними зусиллями колективом з 5000 програмістів за 5 років розробити ОС OS/2, що містить біля 1млн рядків коду (згодом OS/2 Warp, як і Windows NT, буде сучасною багатозадачною, багатопотоковою ОС).

Коли корпорація Microsoft вирішила створити наступника MS-DOS, вона знаходилася повністю під впливом успіхів компанії Macintosh. Історія нової ОС бере свій початок у 1985 році, коли з'явилася перша версія системи під назвою Windows 1.0, базою для якої послужив GUI. Система Windows 1.0 спочатку працювала поверх MS-DOS (це була скоріше оболонка, ніж справжня ОС). Наприклад, у Windows 1.0 вікна не могли перекриватися, що було вже усунуто для Windows 2.0.

Найважливішою особливістю Windows 2.0 стала поява підтримки **захищеного режиму** (protected mode) для програм DOS, який забороняв програмам записувати дані у простір пам'яті інших програм, завдяки чому була підвищена стабільність роботи системи.

Проте у 1995 році вийшла автономна версія Windows 95, яка стала новим етапом в історії Windows. Лише через десять років ця ОС зрівнялася за своїми можливостям з MACOS, які використовувалися в Apple Macintosh ще у 1984 році. Windows 95 включала безліч особливостей ОС MS-DOS, але тільки для завантаження та виконання старих програм.

У порівнянні з Windows 3.1 значно змінився інтерфейс, зросла швидкість роботи програм. Однією з нових можливостей Windows 95 була можливість автоматичного налагодження додаткового устаткування комп'ютера для роботи без конфліктів один з одним. Іншою важливою особливістю системи стала можливість роботи з Internetом без використання додаткових програм. Продовженням розвитку Windows 95 стала трохи змінена версія цієї системи – Windows 98, що з'явилася у 1998 році. Проте, і Windows 95, і Windows 98 все ще містили велику кількість програм 16-розрядного асемблера Intel.

Паралельно з розробкою Windows компанія Microsoft у 1988 році почала роботу над новою операційною системою, яку назвали **Windows NT** (NT означає New Technology – нова технологія), яка на певному рівні сумісна з Windows 95, але її ядро було написане повністю заново. Під впливом ОС OS/2 корпорацією Microsoft спочатку була зроблена невдала спроба розробити цю систему спільно з фірмою IBM, але згодом ця повністю 32-розрядна система була розроблена самостійно.

Девід Кетлер (David Catler), головний розробник Windows NT, був також одним з творців ОС VMS для комп'ютерів PDP-11 та VAX (у 1988 році він прийняв від Біла Гейтса пропозицію стати одним з керівників проекту розробки наступника OS/2 – Windows NT), тому деякі ідеї VMS присутні і в Windows NT. Кетлер наполіг на прийняті 20 розробників з корпорації DEC, де розроблялася ОС VAX, щоб сформувати нову команду.

Система Windows NT використовувала всі можливості сучасних мікропроцесорів і забезпечувала багатозадачність в однокористувацькому або багатокористувацькому середовищі. В ній вперше була реалізована підтримка

інтерфейсу Win32, проте Кетлеру довелося частково займатися і підтримкою API DOS, POSIX та OS/2 на додаток до забезпечення виконання програм для Windows 3.0. Натхненні успіхом ОС Mach на основі мікроядра, команда Кетлера розробила нову ОС з достатньо компактним ядром та багаторівневою модульною структурою для обробки різних інтерфейсів.

Корпорація Microsoft чекала, що вже перша версія NT витіснить MS-DOS та всі інші версії Windows, оскільки це була система, яка набагато перевершувала попередні, але надія не виправдалася. І лише системі Windows NT 4.0 нарешті вдалося відносно широке розповсюдження, особливо в корпоративних мережах.

Нова версія Windows NT 5.0 у 1999 році була перейменована в Windows 2000 (скорочений варіант – **W2K**, 2K – це, власне, і є 2000: «два кіло»). Єдина річ, якої немає у Windows 2000, – це MS-DOS. Її просто немає тут у жодному вигляді (як не було й у NT). Є інтерфейс командного рядка, але це нова 32-розрядна програма, що включає функціональність старої системи MS-DOS. 16-розрядного коду в NT дійсно немає, але це не заважає запускати в NT та Windows 2000 більшість старих 16-розрядних програм. Для цього в системі міститься спеціальна система емуляції 16-розрядної машини. Windows 2000 успадкувала від NT високу надійність та захищеність інформації від стороннього втручання.

Windows 2000 повинна була стати наступником і Windows 98, і Windows NT 4.0. Але цьому також не судилося статись, тому корпорація Microsoft випустила ще одну версію Windows 98, яку назвала **Windows Me** (Millenium edition – випуск тисячоліття). У порівнянні з Windows 98 **Me** набула багато нових можливостей. Перш за все, це краща робота із засобами мультимедіа, можливість записувати не тільки аудіо, але й відеоінформацію, могутні засоби відновлення інформації після збоїв та багато ще чого.

Поступово різниця між різними системами Windows стирається, і на арену виходить нова операційна система версії NT 5.1, більш відома як **Windows XP**, яка призначена для заміни як Windows 2000, так і Windows Me. Букви **XP** у назві нової версії популярної ОС Windows є частиною англійського слова **eXPerience**, яке перекладається як **життєвий досвід**, **знання**. При створенні операційної системи Windows XP був використаний багаторічний досвід розробників найпопулярніших комп'ютерних програм та систем, а також знання, які були накопичені в результаті спілкування з численними користувачами. Без сумніву, нова версія Windows є значним кроком уперед, у порівнянні з попередніми версіями.

ОС Microsoft Windows XP створена на основі технології NT і є прямою спадкоємицею систем Windows 2000 та Me. При збереженні високих показників надійності, безпеки та швидкодії, вона стала простою в опануванні, у ній з'явилося безліч засобів, які призначені для індивідуальних домашніх користувачів. Windows XP має вдосконалений GUI та розширену підтримку мультимедіа. Вперше від цих нововведень виграли як корпоративні, так і

домашні користувачі. Кетлер і до теперішнього часу залишається провідним розробником архітектури ОС Windows.

У своїй новій операційній системі (під кодовою назвою **Longhorn**, нині клієнтська частина – **Vista**, перспектива) корпорація Microsoft планувала об'єднати формати різних типів файлів. Це, наприклад, дозволило користувачам здійснювати у своїй системі пошук усіх файлів (документів, електронних таблиць, електронних листів тощо), які містять певні ключові слова. ОС Longhorn також включає вдосконалений тривимірний інтерфейс користувача, надійнішу систему безпеки та підтримку універсальних цифрових дисків (DVD), у тому числі і для запису.

Але пройшло вже 2 роки з моменту випуску Windows Vista, а система так і не прижилася. Тому на зміну їй вже з'явилася бета-версія Windows 7 (весна-літо 2009). Чи варто розглядати її як заміну доступним на ринку системам, включаючи Windows XP та Windows Vista? На це питання кожен користувач повинен відповісти самостійно, проте, ми вирішили взяти на себе завдання трохи розповісти про нові функції Windows 7.

Нова ОС створена на базі технологій, які впроваджувались у Windows Vista. Не будь Windows Vista, ми навряд чи отримали б Windows 7 такою, яка вона є сьогодні. По суті, все, що ми бачимо у Windows 7, є логічним продовженням того, що було представлене у Windows Vista. Це ні в якому разі не повинно лякати користувачів, що начулися про проблеми, які пов'язані з Windows Vista. Мабуть, навпаки – архітектурні зміни, що викликали бурю претензій в тих, хто спробував Windows Vista, у Windows 7 були розумно вирішені і створили у користувачів враження, що перед ними готова ОС, хоча вона знаходиться на стадії бета-тестування. І не варто забувати про те, що за 2 роки, які пройшли з моменту релізу Windows Vista, виросли об'єми оперативної пам'яті та жорстких дисків, кількість ядер у процесорах. Це зовсім не означає, що зросли системні вимоги Windows 7 – вони залишилися незмінними з моменту релізу Windows Vista, а десь стали нижчими (відносно займаного дискового простору). Більш того, у Windows 7 є всі шанси стати успішною на ринку, на якому у Windows Vista нічого не вийшло, – на ринку ноутбуків, недорогих мобільних комп'ютерів. Ті, хто встиг завантажити бета-версію Windows 7, рапортують про відмінний рівень продуктивності. Хороший початок для бета-версії.

Ми виділимо лише декілька з найпомітніших нововведень Windows 7. Насправді, їх значно більше. Також у Windows 7 багато нововведень, які носять характер дрібних удосконалень, і які навряд чи зможуть спонукати користувачів до розгляду питання про міграцію на нову ОС.

Нова панель задач. Мабуть, найбільш помітною зміною у Windows 7 є нова панель задач, яку користувачі одразу ж після її появи охрестили "супербаром". Панель задач Windows не піддавалася будь-яким помітним змінам з моменту виходу Windows 95. Правда, для Windows Vista вона була дещо оновлена. Проте, у Windows 7 панель піддалася серйозному перетворенню. Зокрема, Microsoft відмовилася від панелі швидкого запуску

(Quick Launch). Необхідність в панелі швидкого запуску відпала сама по собі, зважаючи на можливість закріплення на новій панелі задач будь-якого додатка та навіть файла/папки/веб-сайта.

Нова панель задач стала не лише більш функціональною, але ще і привабливою. Щоб порадувати користувачів, які знають толк в естетиці, на панелі завдань при наведенні на ікону запущеного додатка, Windows 7 динамічно підсвітлює ікону на базі переважаючого RGB-компонента.

Функція AeroSnap. Одним із завдань Microsoft у Windows 7 було спростити організацію вікон на робочому столі. Не дивлячись на те, що у контекстному меню панелі завдань завжди були пункти "Cascade windows", "Show windows stacked" та "Show windows side by side", користувачі часто не розуміли їх призначення, тому і користувалися ними досить-таки рідко. В результаті користувачам доводилося самостійно міняти розміри вікон і проводити вирівнювання вікон одне відносно одного.

Функція AeroSnap дає користувачеві можливість швидко та без зусиль розміщувати вікна на робочому столі. Припустимо, вам потрібно скопіювати документ з однієї папки в іншу. У такому разі було б необхідно відкрити одну папку, скопіювати документ, перейти у цільову папку та вставити документ. У Windows 7 можна поступити аналогічним чином, хоча можна і простіше. Досить відкрити два вікна Windows Explorer, підтягнути одне вікно до лівого кордону робочого столу, друге – до правої, і перед вами деяка подібність інтерфейсу файлового менеджера. Можна копіювати, переміщувати, порівнювати вміст папок.

Libraries (віртуальні папки або теки). Бібліотеки – це віртуальні теки, які об’єднують декілька фізичних тек схожої тематики. Сенс цієї функції простий**:** якщо на жорсткому диску комп’ютера є декілька тек зі схожим вмістом (наприклад, із зображеннями), то їх можна додати у бібліотеку. Потім, відкривши цю віртуальну теку, ви зможете дістати доступ до зображень, які фізично знаходиться в різних куточках вашого жорсткого диску. Це істотно спрощує організацію інформації на жорсткому диску комп’ютера.

Підтримка біометричних пристроїв. На сьогоднішній день найбезпечнішим методом аутентифікації є біометричний метод, при якому аналізується відбиток пальця, сітківка, ДНК або інша фізіологічна характеристика, яка дозволяє однозначно визначити людину. На жаль, Windows не володіє можливістю забору та аналізу проб ДНК, проте для Windows є підтримка сканерів відбитків пальців. Windows і раніше підтримувала аутентифікацію за відбитком пальцю і багато комп’ютерів, що працюють під управлінням Windows Vista, оснащувалися сканерами відбитків пальців. Проте для їх роботи була завжди потрібна установка сторонніх додатків.

Підтримка multi-touch. Компанія протягом довгих років працює над реалізацією так званих натуральних (або природних) призначених для користувача інтерфейсів, аби полегшити взаємодію користувача з комп’ютером. Microsoft вирішила стати першою у світі компанією, що випустила ОС з повною підтримкою технології одночасного натиснення

(інакше multi-touch). На даний момент на ринку присутньо всього два комп'ютери з підтримкою multi-touch: Dell Latitude XT та HP TouchSmart tx2. У березні компанія Albatron, яка відома своїми графічними картами, має намір почати продажі LCD-мониторов з повноцінною підтримкою одночасного натиснення – так званих оптичних сенсорних панелей. Та і Dell анонсувала Dell Latitude XT2. До моменту релізу Windows 7 подібних пристроїв буде значно більше.

Точковим натисненням, яке розуміють резистивні екрани, вважається лише натиснення *стилусом*. З іншого боку, екрани з підтримкою multi-touch абсолютно не розуміють пір'яного введення. Точніше так – до цього часу не могли розуміти, а тепер можуть. І все завдяки старанням Microsoft та ізраїльській компанії N-Trig, яка розробила технологію DuoSense. Таким чином, у Windows 7 ви зможете використовувати як стилус, так і власні пальці.

Device Stage. Підключення пристрою у попередніх версіях Windows могло призвести іноді до несподіваних результатів: наприклад, пристрій, що поєднує можливості принтера/сканера/копіювальника, міг відображатися у вигляді кількох незалежних пристроїв. У Windows 7 розробники Microsoft вирішили цю проблему за допомогою спеціального інтерфейсу управління пристроями, що отримав назву "Device Stage".

За допомогою Device Stage виробники пристроїв дістануть можливість приєднати до інтерфейсу свій бренд, налаштувавши його під кожен продукт окремо та додавши до нього функції, які властиві кожному конкретному продукту.

Наприклад, при підключенні плеєра до комп'ютера, що підтримує режим Device Stage, виникає вікно Device Stage, в якому відображуються рівень зарядки акумуляторів плеєра, індикатор наявності вільного місця, посилання на сайт виробника та інструкцію з експлуатації. Через це вікно можна відкрити вміст флэш-диску у Windows Explorer та запустити відтворення у Windows Media Player. Вікно Device Stage завжди присутнє на панелі. При наведенні на ікону пристрою у панелі задач виникає вікно з точною копією пристрою з кількома іконами – кнопкою для управління синхронізацією, а також індикаторами зарядки акумулятора. Дуже зручно.

DIRECTX 10. Здається, що зовсім недавно Microsoft почала говорити про те, які новини DIRECTX 10 привнесе у процес розробки ігор, хоча було це більше двох років тому. З тих пір вийшло вже не одне покоління відеокарт як від AMD, так і від Nvidia. Згодом з'явився DIRECTX 10.1, який увійшов до складу Vista SP1 і який підтримується виключно графічними рішеннями від AMD. На черзі Windows 7 та DIRECTX 10.

Не дивлячись на те, що DX11 створений на базі DX10, у нову версію API додана маса нових функцій, які оптимізують роботу на сучасному поколінні комплектуючих, зокрема, на багатоядерних процесорах, а це гарантує збільшення продуктивності в іграх.

Пошук та усунення проблем. Якщо родичі і друзі часто просять вас допомогти розібратися з проблемами на комп'ютері, то ви напевно зрадієте

Windows 7. Засоби діагностики та усунення проблем тепер не обмежуються самою Microsoft, тобто будь-який сторонній розробник може створити свої функції для таких цілей. Отже проблем у Windows стане менше, а вирішувати їх стане набагато простіше. Наприклад, якщо після встановлення системи неможливо виконати безпровідне з'єднання, Windows 7 не лише дасть пораду, у чому може бути причина, але і дозволить її усунути, перезапустивши, наприклад, деяку службу. Це позбавить ваших друзів та родичів від необхідності непокоїти вас по дурницях, оскільки за допомогою Windows вони зможуть самостійно усунути практично будь-яку проблему.

Запис ISO-образів. Відбулося те, чого чекали багато користувачів! На доповнення до можливості записувати звичайні диски Windows 7 отримала функцію запису й ISO-образів! Особливим набором опцій ця функція похвалитися не може, та чи потрібні вони при записі звичайного образу? У Windows 7 також стало можливим записувати диски Blu Ray.

Ribbon-інтерфейс у стандартних додатках. Вбудовані додатки Windows та, зокрема, Wordpad та Paint роками не піддавалися будь-яким значущим змінам. Проте у Windows 7 обидва додатки перетворилися. По-перше, оновився інтерфейс. Тепер це знайомий по Office 2007 рядковий (або Ribbon-інтерфейс). По-друге, набір функцій додатків став розширений.

Багато хто скаже, що вбудовані додатки Windows ніколи не порівняються за функціональними можливостями з додатками сторонніх розробників. Це, може бути і так, але у них-то і мети такої ніколи не було. Всі вони пропонують базові можливості, яких вистачає більшості користувачів. Щодо доцільності появи Ribbon-интерфейса у вбудованих додатках, то він повністю відповідає загальній концепції дизайну системи та орієнтованості на сенсорне введення.

Багато фахівців вважають, що Windows 7 – навіть у своєму нинішньому стані – є відмінною операційною системою. Настільки вдалою, що навіть сьогодні багато користувачів використовують її як основну систему. Мабуть, кращу з будь-коли випущених ОС. Продуктивність, функціональність – все на вищому рівні.

Розробка чергової ОС Windows 8 почалася ще до випуску її попередниці, Windows 7, у 2009. Система була анонсована на CES 2011, потім були випущені три підготовчі версії з вересня 2011 по травень 2012. На друк операційна система була направлена 1 серпня 2012, і остаточно випущена для широкого загалу 26 жовтня 2012. **Windows 8** – операційна система від компанії Microsoft для персональних комп'ютерів, включаючи домашні та офісні системи, ноутбуків, планшетів і домашніх театрів.

Windows 8 ввела значні зміни у платформу операційної системи та користувацький інтерфейс для покращення роботи з планшетами, де Windows зараз конкурує з такими мобільними операційними системами як Android та Apple iOS. Зокрема, ці зміни включають оптимізовану під сенсорні екрани оболонку Microsoft "Metro", стартовий екран (що показує програми і динамічно оновлює вміст решітки плиток), нову платформу для

розробки додатків з акцентом на сенсорному вводі, інтеграцію онлайн служб (включаючи можливість синхронізувати додатки і налаштування між пристроями), і онлайновий магазин для звантаження та купівлі нових програм. Windows 8 також додав підтримку таких нових технологій як USB 3.0, Advanced Format твердих дисків і хмарні обчислення. Були додані нові додаткові можливості безпеки, такі як вбудовані антивіруси, інтеграція з фільтрацією Microsoft SmartScreen.

26 березня 2013 року в Microsoft офіційно підтвердили, що працюють над оновленням під кодовим ім'ям Windows Blue. 14 травня оновлення отримало офіційну назву **Windows 8.1**. Публічна попередня версія Windows 8.1 з'явилася 26 червня 2013 року, а фінальна версія з 18 жовтня 2013 року доступна для офіційних версій Windows 8.

Схоже, що Microsoft скоріше хоче позбутися свого бренду Windows 8, тому нова версія черговий операційної системи вийде вже через рік і буде носити іншу назву – **Windows 9**. Про це пише відомий оглядач Пол Таррот в своєму блозі, присвяченому Microsoft.

Сама Windows 8.1 отримає ряд оновлень ще в 2014 році, але вже в квітні 2015 вийде те, що зараз носить кодове ім'я Threshold («поріг», «переддень»). Це, за даними Пола Таррот, повинні оголосити на конференції Build в квітні 2014 року.

Що зміниться в Windows 9? Приміром, в новій ОС напевно буде кнопка «Пуск» і залишиться знаменитий плитковий інтерфейс, який буде якимось чином змінений і доопрацьований. Десктопна і плиточная частини можуть бути якось інакше інтегровані один з одним – наприклад, можна буде запускати Metro-програми з десктопного режиму.

Microsoft підготує три проміжні версії ОС (дві бети і реліз-кандидат), але невідомо, скільки з них будуть доступні широкому загалу.

Build 2014 буде використана як відправна точка для фіналізації ідей і початку роботи над операційною системою. Windows 9, тобто наступна версія, обов'язково стане саме мажорним оновленням, щоб дати людям зрозуміти різницю між Windows 9 і Windows 8, яку постараються зробити прохідною (як Windows Vista). Якщо врахувати відому напівжарт про те, що у Microsoft виходить нормально робити тільки кожну другу операційну систему, то будемо чекати і нормальну ОС.

Один з найбільш розгромних відгуків на нову операційну ОС від Microsoft опублікував Філіп Грінспун, професор Массачусетського технологічного інституту і відомий веб-підприємець. За його словами, Windows 8 – настільки незручна, що годиться тільки як підступний подарунок на майбутній Новий рік тому, кого ви просто люто ненавидите.

Хоча, звичайно, в Microsoft будуть стверджувати як і раніше, що без роботи над Windows 8 не вийшло б і наступної версії.

10.6. Сучасні системи UNIX

Головним суперником Windows у світі персональних комп'ютерів все ж таки була і залишається система UNIX (та її різні похідні). UNIX є найсильнішою системою для робочих станцій та інших комп'ютерів старших моделей, таких як мережні сервери. Вона стала особливо популярною на машинах з високопродуктивними RISC-процессорами (RISC, Reduced Instruction Set Computer – комп'ютер зі скороченим набором команд).

У процесі розвитку ОС UNIX з'явилося багато її реалізацій, кожна з яких має свої корисні можливості. Згодом виникла необхідність створити реалізацію, в якій було б уніфіковано багато важливих нововведень, додані можливості інших сучасних ОС. Розглянемо деякі приклади сучасних ОС UNIX.

System V Release 4 (SVR4). Версія SVR4, розроблена спільно компаніями AT&T та Sun Microsystems, поєднує у собі особливості версій SVR3, Microsoft Xenix System та SUNOS. Внаслідок цього з'явилася очищена від усього зайвого, хоча і складніша за реалізацією ОС.

Solaris 2.x. Система Solaris – це версія ОС UNIX, яка розроблена фірмою Sun на основі SVR4. На час цього огляду останньою версією, що вийшла, була Solaris 2.8. Solaris версії 2.x володіє всіма можливостями системи SVR4, а також деякими додатковими, такими, як повне "витиснення", наявність багатопотокового ядра, напівфункціональна підтримка SMP (Symmetric MultiProcessing – технологія **симетричної багатопроцесорності**) та об'єктно-орієнтований інтерфейс файлових систем. Solaris – це найпоширеніша реалізація ОС UNIX, яка користується комерційним успіхом.

10.7. OS/2. Битва двох гігантів

Історія появи, розквіту та практичного відходу зі сцени ОС під загальною назвою OS/2 (перша версія системи побачила світ у 1987 році) і дивна і повчальна. Будучи однією з найкращих ОС на ПК, вона не стала найпоширенішою, хоча і могла б стати досить легко. Основна причина тому – закони бізнесу, а не якість самої ОС. По-перше, IBM не просувала свою систему на ринок ПЗ з орієнтацією на кінцевого користувача, а вирішила продовжувати свою практику роботи з корпоративними клієнтами, ринок яких є істотно вужчий. По-друге, основні доходи компанія отримувала від продажу дорогих серверів та іншого устаткування. Для успіху на ринку ОС для ПК необхідно було забезпечити всебічну підтримку своєї ОС навчальною літературою, широкою рекламою і тому подібним. На жаль, цього не відбулося, і сьогодні вже мало хто знає про системи OS/2.

Аналітики, що займаються 32-х бітовими операційними системами для персональних комп'ютерів, завжди концентрують свою увагу на битві між Microsoft Windows та IBM OS/2, припускаючи, що Microsoft має перевагу. Але не всі згодні з такою точкою зору. OS/2 v.2.0 була першою доступною та такою, що працює з 32-х бітовою операційною системою для персональних

комп'ютерів. І вона першою почала чергове коло змагань – версія OS/2 Warp, яка призначена для клієнтських машин мереж клієнт-сервер та однорангових мереж, з'явилася на ринку раніше Windows 95, яка була представлена аналогічним чином. OS/2 Warp була також першою системою, яка включила набір засобів підтримки Internet, а також засобів об'єктної орієнтації.

Коли бета-тестери отримали Chicago, першу публічну версію Windows 95, ті, хто вже використовував OS/2, відзначили надзвичайну схожість двох систем. Наприклад, обидві запрошують користувача до роботи за містким робочим столом; обидві системи розглядають іконки та програми як об'єкти; обидві використовують праву кнопку миші для управління поведінкою об'єктів. Призначений для користувача інтерфейс обох систем має однаковий рівень витонченості, вимоги до апаратних ресурсів комп'ютера схожі, і вони обидві засновані на використанні однакового набору технологій, що лежать в основі системи. Ці технології включають багатозадачність та багатопоточність, здатність виконувати DOS-програми (в основі OS/2 лежала DOS) за допомогою віртуальних машин процесорів Intel 80x86 та повну 32- бітову організацію.

І це не випадковість. Через деякий час компанії IBM та Microsoft розійшлися у думці з приводу напряму розвитку OS/2, оскільки Microsoft хотіла присвятити більше часу ОС Windows, ніж OS/2. У 1990 році IBM та Microsoft вирішили працювати незалежно над двома гілками проекту OS/2. IBM взялася контролювати усі подальші версії OS/2 1 та OS/2 2, а Microsoft приступила до розробки OS/2 3. Незабаром, після підписання даної угоди, компанія Microsoft перейменувала OS/2 3 у Microsoft NT.

З тих пір, як IBM випустила версію 2.0 OS/2 (1992 р.), а Microsoft вирішила позиціонувати Windows NT як корпоративну ОС самостійно, став зрозуміло видимий важливий провал у лінії операційних систем Microsoft, який і заповнила IBM. Спроби Microsoft висунути Windows 3.1 на ту ж роль розвиненої ОС для настільних систем, що і OS/2, мали обмежений успіх.

В результаті Microsoft почала нести втрати в об'ємах продаж, і, що важливіше, втрачати твердий грунт для своїх операційних систем. Коли стало зрозуміло, що Windows NT навряд чи повною мірою стане лідером настільних ОС вищого класу, маркетингова машина Microsoft почала менше говорити про можливості Windows NT і почала говорити про можливості Windows 95. Зрозуміло, що IBM та OS/2 зробили значний вплив на стратегію Microsoft в області операційних систем.

IBM, у свою чергу, постійно створює здорову конкуренцію для лінії Windows. Windows 95 не порівнянна з OS/2 2.2. Швидше конкурувати почали Windows 95 та OS/2 Warp 3/0. Warp – це постріл з дальнім прицілом, який спрямований на витіснення Windows. І, хоча Warp має деякі початкові переваги і як система виглядає "кращою", Windows, як і раніше, є надійним вибором.

Імена операційних систем можуть змінитися, але рівновага у битві IBM/Microsoft залишиться тією ж. Існують дві причини – фактична та емоційна, які заважають встановленню перемир'я між цими двома компаніями.

Фактично, IBM була в цій області першою. OS/2 перетворилася на працюючий продукт зі своєю версією 2.0 у 1992 році. З того часу вона стала багатозадачною, багатопотоковою системою із зручним об'єктно-орієнтованим інтерфейсом. Зусилля з розвитку OS/2 були неквапливими та постійними, і система отримувала прихільність та підтримку на всьому шляху свого розвитку. Проте Windows, як і раніше, тримала найбільшу частку ринку. Емоційно, IBM відчуває себе "зрадженою" Microsoft, яка втекла з лав розробників OS/2. Це не зовсім справедливо по відношенню до Microsoft, оскільки компанія має право вкладати свої капітали у ту сферу діяльності, яка на її думку принесе найбільший прибуток. Хоча Microsoft могла б поводитися тактовніше і продовжувати партнерство по OS/2.

Не дивлячись на те, що зараз IBM далеко не та компанія, якою вона була у ті далекі часи, коли вона домінувала на ринку персональних комп'ютерів, їй теж не вистачає такту. Ця компанія була першою так довго, що вона не вміє виступати в інших ролях. Тому з моменту появи версії OS/2 2.0 IBM змінила свою стратегію. Вона почала грати за тими ж правилами, що і решта компаній.

У кінці 1994 року IBM випустила третю головну версію OS/2, яку назвала OS/2 Warp 3 (warp – основа). Зараз ми маємо справу вже з версією OS/2 Warp 4. OS/2 Warp має добре обміркований об'єктно-орієнтований інтерфейс із застосуванням техніки drug-and-drop при виконанні операцій копіювання, вилучення, друку, а також деяких інших. До складу OS/2 Warp входить набір утиліт BonusPack, який містить IBM Works – інтегрований програмний пакет початкового рівня, та Internet Access Kit – найповніший набір засобів для мережі Internet з усіх засобів, що постачаються у складі ОС, Web Browser та пошта Internet Mail. У публікаціях зустрічаються твердження, що він досконаліший, ніж набір для доступу до Internet, який реалізований у Windows 95. У лютому 1995 року IBM почала продавати пакет OS/2 Warp 3 Full Pack, який містив бібліотеки Win-OS/2. Ці бібліотеки дають можливість виконувати Windows-програми, не набуваючи ліцензійних копій Microsoft Windows.

У той же час в OS/2 Warp відчувався недолік мережних функціональних можливостей. Але становище змінилося, оскільки влітку 1995 року IBM почала продавати наступну версію OS/2 – Warp Connect, яка містить найважливіші драйвери та утиліти. Крім того, Warp Connect надає давно очікувані в OS/2 засоби однорангового мережного зв'язку.

Список літератури

1. Майер Б., Бодуэн К. Методы программирования: В 2-х томах. Том 1. – М.: "Мир", 1982. – 356 с.
2. Мартин Дж. Разработка приложений без программистов. – София, 1983. – 404 с.
3. Семенюк Э. П. Информатика: как ее понимать? – М.: НТИ, сер. 2, №7, 1984.
4. Уэзерелл Ч. Этюды для программистов. – М.: "Мир", 1982. – 288 с.
5. Лебедев А. Clarion для Windows. – М.: "ПРИОР", 1997. – 252 с.

6. Знакомьтесь: компьютер. //Под ред. Курочкина В.М. – М.: ”Мир, 1989. – 240 с.
7. Якобсон А., Буч Г., Рамбо Дж. Унифицированный процесс разработки программного обеспечения. – СПб: ПИТЕР, 2002. – 496 с.
8. Язык компьютера. //Под ред. Курочкина В.М. – М.: ”Мир, 1989. – 240 с.
9. Майерс Г. Надежность программного обеспечения. – М.: “Мир”,1980. – 360 с.
10. Компьютеры приобретают разум. // Под ред. Курочкина В.М. – М.: ”Мир, 1989. – 244 с.
11. Зиндер Е. З. Бизнес-реинжиниринг и технологии системного проектирования. Учебное пособие. – М.: МГУ, Центр Информационных Технологий, 1996. – 382 с.
12. Горчинская О. Ю. Designer/2000 – новое поколение CASE-продуктов фирмы ORACLE. – М: "СУБД", №5, 2004.
13. Архангельский А. Я. Программирование в Delphi 7. – М.: “БИНОМ”, 2003. – 1152 с.
14. Сергей Алексеевич Лебедев. К 100-летию со дня рождения основоположника отечественной электронной вычислительной техники /Отв. ред. В.С. Бурцев. Составители: Ю.Н. Никольская, А.Н. Томилин, Ю.В. Никитин, Н.С. Лебедева. – М.: ФИЗМАТЛИТ, 2002. – 440 с.
15. Митио Каку. Физика будущего; Пер. с англ. М: Альпина нонфикшн, 2012. – 584 с.
16. Ю.Н. Карпов. MODEL CHECKING. Верификация параллельных и распределенных программных систем. – СПб: БХВ-Петербург, 2010. – 560 с.
17. Максимов Н. В., Партыка Г. Л., Попов И. И. Архитектура ЭВМ и вычислительных систем: Учебник. М.: ФОРУМ: ИНФРА-М, 2005. – 512 с.

ЧАСТИНА ІІ. ІСТОРІЯ КОМП'ЮТЕРНИХ МЕРЕЖ

Вступ

Який би не був могутній та досконалий комп'ютер, яка б у нього не була апаратна та програмна потужність, у наш час ця потужність – ніщо без засобів комунікації. Людина не може жити одна – їй постійно потрібна допомога та підтримка інших людей. Комп'ютер мало чим відрізняється у цьому сенсі від людини.

Комп'ютер може зберігати та обробляти інформацію, але не здатний передавати її на відстань. Зв'язок – це той же транспорт, але для інформації. У нинішню інформаційну добу його значення неможливо переоцінити.

Комп'ютер та зв'язок – два ключові поняття, два рівноцінні важелі інформаційної революції. Довгий час вони розвивалися незалежно одне від одного, але у 80-х та 90-х роках почався процес їх інтенсивного зближення. З одного боку, електрозв'язок дав комп'ютерам можливість об'єднуватися у локальні та глобальні комп'ютерні мережі. З іншого боку, усі сучасні засоби зв'язку, чи то телефонна станція або студійний телевізійний комплекс, по суті, є складними програмно-керованими мікропроцесорними системами.

З початком XXI століття цей розвиток та співпраця стали виявлятися ще більше. Технічні та соціальні наслідки інтеграції комунікаційних та комп'ютерних технологій настільки значні, що фахівці порівнюють цей процес із злиттям двох половинок ядерного заряду в атомній бомбі. У результаті в останні кілька років спостерігається інформаційний вибух небаченої сили. Стільникові телефони, Internet з його безмежними можливостями в спілкуванні і пошуку інформації, пластикові гроші, електронна торгівля, дистанційна освіта – перші вияви цього незвичайного явища. Загальна кількість сайтів, що увійшли в структуру глобального Веба, значно збільшилася – ще на початку 2013 року кількість URL становило близько 630 млн, тоді як зараз (початок 2014 р.) вона перевищує 861 млн (+37%). Правда, в 2011 році приріст був ще більш вражаючим – майже 50%.

Цікаво, що фактично Internet (від електронної пошти до форумів) був передбачений ще у 1911 році в романі американського письменника-фантаста Хьюго Гернсбахера (Гернсбек) – "Ральф 124С 41+".

Герой твору – вчений та винахідник «Ральф 124С 41+» – в результаті невдалого досвіду з радіопередавачем знайомиться в ефірі з дівчиною з не менш дивним ім'ям Еліс 212В 423, і через неї опиняється у вирі небезпечних пригод, на космічному кораблі вирушає на Марс, де рятує свою кохану від одного з претендентів на її руку та серце. Треба зазначити, що історія кохання у романі була необхідна тільки для побудови загального сюжету, в якому присутні багаточисельні описи фантастичних наукових ідей та технічних винаходів майбутнього в області автоматизації, комунікації, космічних польотів, транспорту. У цьому вирі винаходів сьогоднішньому читачеві цікаво

виявити радар та відеофон, анабіоз (оживлення, від греч. ανα- – «знову» и βιος – «життя»), телебачення, використання сонячної енергії та багато іншого.

Цікава історія сталася з Робертом Уотсоном-Уаттом, який у 1935 році винайшов пристрій для стеження за літаками і ввів у вжиток слово «радар». Коли він прочитав роман «Ральф 124С 41+» з детальним описом принципів роботи майбутнього радара з проілюстрованою принциповою схемою, то був настільки вражений цим, що пізніше публічно віддавав Хьюго Гернсбеку пріоритет свого винаходу.

Для того щоб зрозуміти принципи побудови та напрями еволюції сучасних комп'ютерних мереж, необхідно знати, хоча б коротко, історію їх розвитку, основні поняття теорії та тенденції розвитку.

1. Передісторія сучасних комп'ютерних мереж

1.1. Покоління комп'ютерних мереж

Комп'ютерні мережі, як і самі комп'ютери, мають достатньо довгу та непросту історію. За ступенем розвитку їх можна поділити на три покоління.

Перше покоління (50-і та 60-і роки) – підключення віддалених терміналів до комп'ютерів (теледоступ). В перші десятиліття комп'ютерної доби (50-і роки), коли ЕОМ були великою рідкістю і стояли далеко не в усіх містах, великим успіхом вважалася сама можливість поспілкуватися з ними, хоча б через повільний та ненадійний телетайп. Розвиток техніки зв'язку та винахід модемів у 60-і роки дали можливість організовувати розгалужені мережі теледоступу.

Друге покоління (70-і роки) – підключення комп'ютерів один до одного. З початку 70-х років у світі почали створюватись чисельні ***територіальні (Wide Area Networks – WAN)*** та ***локальні (Local Area Networks – LAN)*** комп'ютерні мережі у сучасному розумінні, що надають високорівневі мережні послуги (віддалені обчислення, передача файлів, електронна пошта, тощо). Спочатку для передачі повідомлень між комп'ютерами використовувався принцип фізичної комутації каналів, але згодом він був повністю витиснений надійнішим та економнішим принципом комутації пакетів. Мережі другого покоління, як і ЕОМ другого покоління, відрізнялися різноманітністю та несумісністю. Два абоненти, які знаходились навіть на невеликій відстані один від одного, могли спілкуватися, якщо тільки вони належали до однієї мережі.

Третє покоління (80-і та 90-і роки) – підключення ізольованих мереж та утворення всесвітньої «мережі мереж» Internet.

1.2. Перші кроки телеобробки

Теледоступ (remote access) або, інакше, ***телеобробка*** – процес спілкування користувача по каналах зв'язку з обчислювальною машиною, яка знаходиться на значній відстані від нього. Експерименти з телеобробки були проведені ще до винаходу електронних обчислювальних машин.

Історично першим вважається досвід, який був запропонований Джорджем Стібіцем у 1940 році на засіданні Американського математичного товариства, яке проходило у Дармутському коледжі (він відомий як батьківщина Бейсика) у місті Ганновере. Телеграфний апарат, що був розташований там, був підключений до релейного обчислювача BELL-I, який знаходився за сотні кілометрів у Нью-Йорку. Два комплексні числа були відправлені телеграфом до машини, перемножені, а результат повернувся назад у Ганновер.

Одразу же після появи перших ЕОМ почали реалізовуватися різні технічні ідеї з телеобробки. Оскільки найдоступнішим діалоговим пристроєм введення-виведення на той час був букводрукувальний телеграфний апарат – телетайп, то більшість систем телеобробки у 50-і роки ґрунтувалися на

телеграфних каналах зв'язку. Принциповим недоліком такої схеми є вкрай низький коефіцієнт використання комп'ютера. Річ у тім, що швидкість роботи електромеханічного телетайпу дуже мала – до 10 символів за секунду, а операційних систем із розділенням часу ще не існувало. Таким чином, при роботі по телеграфній лінії процесор майже весь час простоював. Тільки після того, як до одного комп'ютера стало можливим підключати декілька одночасно працюючих терміналів, телеобробка стала економічно доцільною.

1.3. Телеобробка у 60-і та 70-і роки

ЕОМ третього-четвертого поколінь, що набули широкого поширення у кінці 60-х та 70-х роках, були вже достатньо потужними, але ще дуже дорогими та громіздкими, щоб їх могли придбати звичайні, не дуже багаті, підприємства та організації. Надії користувачів на той час були пов'язані з ідеєю колективного користування комп'ютерів, тим більше, що технічна можливість підтримки багатокористувацького режиму вже з'явилася. На зміну телетайпам прийшли електронні дисплейні термінали, були створені діалогові операційні системи з розподілом часу. Для підключення віддалених терміналів до ЕОМ почали широко використовуватися телефонні канали. Швидкості у декілька Кбіт/с (біт/с або bps (bit per second) – визначає кількість бітів переданої інформації за секунду), яку забезпечував стандартний телефонний канал та модем, було цілком достатньо для комфортної роботи за терміналом. Центральна ЕОМ, яка працювала в режимі розподілу часу, була оснащена спеціальним пристроєм – ***мультиплексором передачі даних***, який дозволяв підключити декілька (у системі IBM 360 – до 256) віддалених абонентів. Абоненти підключалися до системи каналами зв'язку, які були або **виділеними** (**некомутовані**), або звичайними телефонними каналами, що встановлювались на час сеансу зв'язку через комутуюче устаткування міських телефонних станцій.

Абонентами у системі телеобробки могли виступати не тільки дисплейні термінали, але й інші пристрої – касові апарати, банківські термінали, давачі фізичних величин, тощо. Зокрема, у ролі абонента могла виступити інша ЕОМ. У цьому випадку маємо вже не систему телеобробки, а просту комп'ютерну мережу.

Характерним для наведеної схеми є встановлення та підтримка на весь час сеансу зв'язку фізичного каналу між абонентами. Тому вона називається ***схемою з комутацією каналів***.

За схемою з комутацією каналів у 70-і роки на Заході були створені численні обчислювальні мережі, причому деякі з них розрослися до великих масштабів. Відзначимо, що це були саме **обчислювальні мережі**, оскільки основним їх призначенням було використання процесорних ресурсів, які колективно використовували ЕОМ. Цим вони відрізняються від **комп'ютерних мереж** 90-х років, де основним ресурсом є інформація.

Класичним прикладом є мережа **Cybernet**, яка була створена у 1969 році фірмою Control Data Corporation (CDC). Обчислювальні вузли мережі були оснащені потужними комп'ютерами CDC-6600, що дозволило обробляти запити від тисяч терміналів, які були розкидані по всьому світу.

1.4. Особливості мереж із комутацією каналів

В системах телеобробки з комутацією каналів є одна важлива перевага – відсутність затримок при передачі даних. Оскільки комутаційна апаратура на вузлах нічого не робить з повідомленнями, а тільки підключає одну з вихідних ліній до вхідної, електричний сигнал пролітає весь шлях між абонентами з великою швидкістю. Проте ця позитивна якість досягається ціною двох принципових недоліків.

По-перше, необхідність підтримувати наскрізний фізичний канал на весь час сеансу неминуче призводить до невисокої надійності системи, коли канал утворюється ланцюжком із багатьох сотень окремих ділянок ліній передачі та з'єднань на вузлах комутації.

По-друге, зайняття окремого фізичного каналу для організації зв'язку між терміналом та ЕОМ не виправдане економічно, особливо при дальніх міжміських з'єднаннях, оскільки канал використовується на малу частку своїх можливостей. Весь час, поки користувач думає перед натисненням чергової клавіші, канал простоює.

Вказані органічні недоліки принципу комутації каналів призвели до того, що у 80-і роки він був усюди витиснений принципом **комутації повідомлень**, про який детально йтиметься далі. Проте існують області застосування, де комутація каналів ще і досі використовується. Ось декілька прикладів.

Масові абонентські мережі. Підключення домашніх комп'ютерів до провайдерів послуг мережі Internet у межах одного населеного пункту у переважній більшості випадків проводиться через телефонну мережу загального користування. До того ж, сучасні мережі розділяють телефонні канали за допомогою спеціальних модемів на дві частотні смуги, які можуть використовуватись одночасно: низької частоти – для звичайного телефонного зв'язку та високої частоти – для підключення до Internet.

Інший показовий приклад – створена у Франції на початку 80-х років мережа Minitel. Майже у кожному французькому будинку поряд з телефонним апаратом розміщений невеликий термінал у вигляді портативного чорно-білого алфавітно-цифрового дисплея з клавіатурою. Зателефонувавши в один із центрів мережі, абонент перемикає телефонну лінію на терміналі, спілкуючись з комп'ютером центру, може отримати довідку про номер телефону, розклад роботи транспорту або програму передач, замовити квиток, зробити покупку. Те, що останнім часом рекламується як новітнє досягнення **Internet-комерції**, було доступно французам завдяки Minitel вже близько двадцяти років.

Спеціальні відомчі мережі. Деякі організації, що створили у минулому власні мережі з комутацією каналів для технологічних потреб, продовжують їх

експлуатувати і зараз. У такому режимі, зокрема, працюють багато банків зі своїми клієнтами та філіями.

Аматорські мережі. Простота організації міжкомп'ютерних з'єднань через телефонну мережу загального користування дала можливість, після того, як з'явилися персональні комп'ютери, організувати аматорські мережі. Найбільшою з них є мережа **FidoNet,** яка заснована у 1984 році програмістом з Лос-Анджелеса Томом Дженнінгсом (Jennings, Tom). Разом з Джоном Меділлом (Madil, John) з Балтімора вони написали програму, яка давала можливість автоматично обмінюватися електронними листами через усю країну по телефонній лінії. Ідея виявилася продуктивною, їхня мережа почала швидко зростати. У кінці 1984 року у мережі було 50 вузлів, на початку 1995 року – 200, на початку 1997 року – вже 37 000. Кількість кінцевих користувачів мережі не піддається обліку, за грубими оцінками воно перевищує мільйон. У самій назві мережі міститься натяк на неофіційність проекту: слово **Fido** використовується як призвісько безпородного собаки, дворняжки (звідси й емблема – Fido).

2. Мережі пакетної комутації

2.1. Великий ідеолог Internet

Якщо електронні обчислювальні машини – дітище другої світової війни, то сучасні комп' ютерні мережі – побічний результат "Холодної війни", яка почалася майже одразу після перемоги над гітлерівською Німеччиною. Колишні союзники США та СРСР, які були розділені залізною завісою ідеологічних суперечностей, спішно нарощували військово-технічний потенціал, готуючись до неминучої, як тоді здавалося, третьої світової війні. Насправді все розпочалося із запуску радянського супутника 4 жовтня 1957 року.

Спочатку у гонці озброєнь лідирували США, вони першими створили ядерну зброю і розгорнули навколо СРСР мережу військових баз, звідки американські бомбардувальники могли за лічені години доставити атомні бомби до будь-якого радянського міста. Сама ж територія США, відокремлена океанами від решти світу, здавалася недосяжною для вторгнення. Тому, коли 4 жовтня 1957 року американці дізналися про запуск першого радянського супутника, вони були приголомшені. Ракета, що доставила в космос мирний супутник, могла з таким же успіхом скинути на Нью-Йорк або Вашингтон водневу бомбу. Благополуччя нації одразу опинилося під загрозою.

У США все прийшло у рух. Розуміючи, що у гонці озброєнь виграє та сторона, у якої краще поставлені освіта та наука, уряд корінним чином перебудував свою політику у цій галузі. Вже у лютому 1958 року було створено два могутні відомства, які зіграли ключову роль в історії «холодної війни». Перше займалося безпосередньо космосом і називалося Національним управлінням з аеронавтики та досліджень космічного простору (National Aeronautics and Space Administration – **NASA**), перед другим було поставлено загальніше завдання забезпечити стратегічну перевагу США у сфері високих технологій. Воно працювало при міністерстві оборони і називалося Управлінням перспективних досліджень (Advanced Research Projects Agency – **ARPA**).

Управління підключило до військових програм найбільші університети та дослідницькі центри, витрачаючи на них величезні гроші. Шестидесяті роки – це час, коли у США, які підтримувались військовим бюджетом, розквітали кібернетика, інформатика, прикладна математика та інші точні науки.

У самий розпал «холодної війни», у дні кубинської кризи жовтня 1962 року, в ARPA була організована дирекція комп'ютерних програм, а її першим керівником був призначений Дж. Ліклайдер (Licklider, Joseph; 1915-1990) з Массачусетського технологічного інституту (MTI). Хоча його діяльність на високому посту була недовгою, ідея Мережі вже зацікавила керівництво ARPA.

У 1957 р. Д. Лікляйдер перейшов на роботу у компанію Bolt Beranek and Newman (BBN). Він залучив у штат BBN цілу групу професійних комп'ютерних інженерів, плідна робота яких призвела до того, що BBN з фірми

по виробництву акустичного устаткування перетворилася на фірму-консультант з комп'ютерних технологій. А вже у 1968 р. BBN виграла тендер на розробку першого комутатора для майбутньої мережі ARPANET. На думку Д. Лікляйдера, до кінця 50-х років потужність комп'ютерів вже була достатньою для того, щоб втілити на практиці передбачені В. Бушем ще в середині 40-х років, ідеї баз даних (у той час вони називалися автоматичними бібліотеками) для зберігання великих обсягів інформації. У 1959 р. Д. Лікляйдер опублікував книгу "Бібліотеки майбутнього" (Libraries of the Future), в якій детально описав технологію створення на перших мінікомп'ютерах баз даних, з одночасним віддаленим доступом кількох користувачів.

На початку 60-х років Д. Лікляйдера починають цікавити питання об'єднання комп'ютерів в єдину мережу з розподіленими ресурсами доступу. У жовтні 1962 р. він прийняв пропозицію директора агенції ARPA Д. Руна очолити в ARPA два департаменти, об'єднаних згодом у Бюро з методів обробки інформації – IPTO (Information Processing Techniques Office). Як додатковий стимул Д. Рун запропонував Д. Лікляйдеру використовувати найпотужніший на ті часи комп'ютер Q-32, який спочатку призначався для військово-повітряних сил США, але пізніше був відданий в агенцію ARPA. У серпні 1962 р. Д. Лікляйдер спільно з В. Кларком опублікував роботу "On-Line Man Computer Communication", яка стала першою серйозною теоретичною основою створення майбутньої глобальної мережі Internet. Була запропонована модель об'єднання безлічі різних комп'ютерів в єдину глобальну мережу, в якій кожен окремий користувач міг мати доступ до програмних продуктів або баз даних, які розміщені на інших комп'ютерах. Д. Лікляйдер став першим, хто детально розробив сучасну концепцію побудови глобальної комп'ютерної мережі, яка охопила всю Земну кулю.

У 1963 році Д. Лікляйдер опублікував серію документів, так звані нотатки для Міжгалактичної комп'ютерної групи, в якій з практичної точки зору спромігся описати створення майбутньої інтегрованої мережі. Він детально зупинився на двох основних проблемах, які необхідно було розв'язати при створенні майбутньої мережі. По-перше, достатня віддаленість окремих вузлів і, по-друге, несумісність операційних систем комп'ютерів, що сполучаються. Д. Лікляйдер запропонував використовувати програмне забезпечення, яке існувало б лише у мережі, і використовувалося б будь-якою машиною, яка б підключалася при необхідності. Ці теоретичні ідеї вдасться реалізувати лише декілька десятиліть потому, створивши мову Java. Очоливши першу дослідницьку програму в агенції ARPA, він зумів переконати у важливості своїх теоретичних викладень А. Сазерленда та Б. Тейлора, а також дослідника з MIT Л. Робертса. Ці три фахівці стануть наступниками Д. Лікляйдера при створенні майбутньої мережі ARPANET.

Надалі ми інколи називатимемо Internet Мережею з великої літери, на відміну від "мережі" з маленької літери, якою позначається взагалі будь-яка мережа, локальна або глобальна. Тут ситуація схожа зі словом "галактика":

наша галактика називається Галактикою з прописної літери, а "галактика" з малої літери відповідає будь-якій іншій зоряній системі подібних розмірів.

Наступники Ліклайдера – Іван Сазерленд (Ivan Sutherland), якого вважають родоначальником комп'ютерної графіки, та Боб Тейлор (Bob Taylor) почали опрацьовувати питання реалізації цього фантастичного проекту. Експерименти, що були проведені у 1964 – 1966 роках за замовленням ARPA в MTI під керівництвом Лоуренса Робертса (Roberts Lawrence) продемонстрували принципову можливість зв'язку кількох ЕОМ, але у той же час показали, що для великомасштабної та надійної обчислювальної мережі принцип комутації каналів неприйнятний. Потрібна була принципово нова схема передачі даних, і ця схема була знайдена. Вона називається ***комутацією повідомлень (message switching)*.**

2.2. Принцип комутації повідомлень та пакетів

Принцип комутації повідомлень був розроблений майже одночасно та абсолютно незалежно один від одного у трьох місцях. Головні теоретичні дослідження у цьому напрямі належать Леонарду Клейнроку (Kleinrock, Leonard; нар. 1934), який, будучи аспірантом MTI, запропонував базові принципи пакетної технології, опублікував у 1961 році піонерську статтю, а у 1964 році – першу монографію з комутації пакетів. У 1963 році, отримавши вчений ступінь, Клейнрок почав працювати у Каліфорнійському університеті Лос-Анджелеса, де організував дослідницький мережний центр.

Паралельно схожі принципи розвивалися у знаменитій некомерційній фірмі RAND Corporation (розшифровується як Research ANd Development – Дослідження та розробка). Ця невелика за чисельністю дослідницька організація, розташована у мальовничому місці на березі Тихого океану, прославилася видатними досягненнями в області аналізу складних систем, прикладної математики, дослідження операцій. Зокрема, за замовленням міністерства оборони там створювався проект системи зв'язку для управління збройними силами та країною у разі ядерної війни. У 1964 році під керівництвом Пола Берена (Baran, Paul) вийшов звіт, у якому пропонувалася самоорганізуюча комп'ютерна мережа, яка зберігає працездатність навіть при знищенні деяких вузлів комутації та ліній зв'язку.

Третя група дослідників працювала за океаном, у Великобританії, де в Національній фізичній лабораторії (National Physical Laboratory – NPL) під керівництвом Дональда Девіса (Davies, Donald) була створена експериментальна комп'ютерна мережа з комутацією повідомлень.

У системі з комутацією повідомлень вузли мережі, на відміну від комутації каналів, не займаються простим перемиканням ліній. Вузлу можна дати таке визначення: будь-яка сутність, що має унікальну адресу в Internet, називається ***вузлом***. З цієї точки зору Internet можна розглядати, як велику множину вузлів, кожен з яких потенційно може зв'язатися з будь-яким іншим вузлом.

Кожним вузлом є справжній комп'ютер з процесором та пам'яттю. У перших мережах ці комунікаційні комп'ютери позначалися як **IMP** — Interface Message Processor, а пізніше почали називатися ***маршрутизаторами (router,*** або ***роутерами)***. Коли від абонента надходить повідомлення, маршрутизатор запам'ятовує його у своїй пам'яті, визначає, якому з сусідніх вузлів потрібно відправити повідомлення далі, і, як тільки потрібний канал звільняється, передає його наступному маршрутизатору. Якщо від сусіднього вузла надходить підтвердження про успішне надходження повідомлення, то воно вилучається з пам'яті, а якщо відбулася помилка, то передача повідомлення повторюється доти, доки не прийде позитивна відповідь. Так, передаючись по естафеті від маршрутизатора до маршрутизатора, повідомлення за декілька стрибків (***hops***) доходить до адресата.

Як бачимо, принцип комутації повідомлень має такі характерні особливості:

– сполучний канал займається конкретним повідомленням тільки на час передачі та одразу ж звільняється. Тим самим забезпечується високий коефіцієнт використання дефіцитних сполучних каналів;

– передача повідомлення між сусідніми вузлами відбувається за процедурою з перезапитом. Більш того, якщо сусідній вузол або сполучний канал взагалі виявляться непрацездатним, маршрутизатор відправить повідомлення по обхідному шляху. Тим самим, навіть при ненадійних каналах зв'язку та руйнуванні окремих вузлів, забезпечується високий рівень надійності всієї мережі.

Таким чином, схема комутації повідомлень виявляється позбавленою від головних недоліків схеми з комутацією каналів, проте це досягається не безкоштовно, а ціною виникнення інших проблем.

По-перше, маршрутизатори – це не прості з'єднувачі, подібні телефонним реле на АТС, а повнофункціональні комп'ютери. Тому на практиці комутація повідомлень не могла реалізуватися раніше, ніж з'явилися відносно дешеві МІНІ-ЕОМ, тобто в середині 60-х років.

По-друге, повторний прийом та передача призводять до затримок, які можуть сягати значних величин, якщо шлях сполучення проходить через довгий ланцюжок вузлів. Цей недолік є принциповим, від нього неможливо позбавитися зовсім, можна тільки спромогтися зменшити затримку, підвищуючи швидкодію каналів зв'язку та самих маршрутизаторів.

У чистому вигляді комутація повідомлень незручна ще з однієї причини. Якщо необхідно передавати довгі повідомлення, до декількох мегабайт, то на кожному вузлі доводиться зберігати та повторно передавати такі масиви даних. Це довго та дорого, робота мережі блокується довгими повідомленнями. Для боротьби з цією проблемою була запропонована ідея пакетної комутації. ***Пакет (packet)*** – це повідомлення обмеженої довжини, зазвичай до 1500 байт. Довгі повідомлення джерела розбиваються на пакети, які незалежно один від одного проходять через мережу, і у одержувача знову збираються у цілісні

повідомлення. Мережі, які організовані за таким принципом називаються ***мережами пакетної комутації (packet switching).***

Таким чином, теоретично метод пакетної комутації обіцяв великі переваги у порівнянні з комутацією каналів. Залишилося перевірити його на практиці, проте для цього були потрібні значні капіталовкладення у канали зв'язку та маршрутизатори.

2.3. День народження першої комп'ютерної мережі (70-ті роки)

Всю другу половину 60-х років концепція пакетної комутації обговорювалася в ARPA, нарешті було ухвалено рішення про практичну реалізацію мережі. Керівником проекту був призначений Лоуренс Робертс, який у жовтні 1967 року організував історичну зустріч всіх трьох шкіл пакетної комутації (MIT, RAND, NPL) та підготував конкретні пропозиції. У серпні 1968 року був оголошений конкурс на створення мережі з чотирьох комунікаційних комп'ютерів з можливістю розширення до дев'ятнадцяти. Передача даних між вузлами передбачалася на швидкості 50 Кбіт/с.

Провідні фірми, такі як AT&T або IBM, не виявили цікавості до проекту, вважючи його реалізацію неможливою. Конкурс виграла маленька фірма Bolt Beranek and Newman (BBN), з Кембріджу, штат Массачусетс, яка була заснована фахівцями Массачусетського технологічного інституту. Група розробників під керівництвом Френка Харта (Heart, Frank) за мільйон доларів взялася протягом року втілити теорію у життя.

Комунікаційні процесори були замовлені у фірми Honeywell, яка вже у березні 1969 року поставила прототип маршрутизатора на базі MIHI-EOM Honeywell 516 з оперативною пам'яттю 12 Кбайт. У квітні було розроблено програмне забезпечення маршрутизатора і описаний стандарт на інтерфейс між абонентськими ЕОМ, так званими ***хостами (host – господар***), та маршрутизаторами. Автор цього стандарту Стів Крокер (Crocker, Steve), щоб уникнути звинувачень у самозадоволенні, назвав його скромно Request for Comment (**RFC 1**) – запит для коментарів та пропозицій. Так народилася майбутня система стандартів Internet, яка налічує зараз більше тисячі документів.

Перший вузол мережі був встановлений у вересні в Каліфорнійському університеті в Лос-Анджелесі (UCLA), де він був підключений до великої ЕОМ SDS Sigma-7. Ця честь була надана університету, оскільки в UCLA працював колектив Клейнрока, який взяв на себе функції мережних вимірювань.

Другий вузол був змонтований у жовтні, він розташовувався в Стенфордському дослідницькому інституті (SRI) з його ЕОМ SDS-940, там у цей час над проблемами людино-машинного спілкування працював знаменитий Дуглас Енгельбарт. На базі інституту був створений Мережний інформаційний центр (Network Information Center – NIC), який був зобов'язаний збирати, зберігати та поширювати всю документацію, що пов'язана з мережею, перш за все мережні стандарти RFC.

Колі 29 жовтня 1969 року рівно о дев'ятій вечора включили другий вузол, фахівці UCLA зробили спробу підключитися до комп'ютера у Стенфорді. Ось як Клейнрок згодом згадував цю подію в інтерв 'ю агенції «Рейтер»:

«Наш хлопець сидів за консоллю комп'ютера з телефонною трубкою, розмовляючи з хлопцями з SRI. Коли все було встановлено, він зібрався надрукувати слово «LOG», з тим, щоб комп'ютер в Стенфорді автоматично додав «IN», отримуючи «LOGIN». Наш хлопець набрав букву «L» і запитав по телефону свого колегу в Стенфорді:

– Ви бачите L?

– Так, ми бачимо L.

Він набрав «O» і знову запитав:

– Ви бачите O?

– Так, ми бачимо O.

Тоді він набрав «G», і тут вся система зупинилась».

Тобто не усе пішло гладко (після передачі перших трьох букв сервер вийшов з ладу), і завершити дію вдалося тільки о пів на одинадцяту. Проте, передача даних була здійснена, і саме цей день залишився в історії як день появи глобальної мережі Інтернет.

Так почалася мережна революція. Через 30 років день народження першої комп'ютерної мережі – **29 жовтня 1969 року** – був проголошеним як професійне свято – International Internet Day.

У грудні 1969 року перша черга мережі, названої **ARPAnet**, була запущена в експлуатацію. Мережа ARPAnet була не тільки експериментальним полігоном для відпрацювання технології пакетної комутації, але мала і практичну мету – об'єднати могутні комп'ютерні ресурси університетів, що працюють за контрактами Міністерства оборони США. Для цього розроблялися спеціальні програми, зокрема, Telnet, яка дозволяє працювати у режимі віддаленого терміналу з комп'ютером, який знаходиться на відстані тисяч кілометрів.

Мережа ARPAnet була створена для роботи без централізованого управління, тобто в ARPAnet не існувало структури «головний комп'ютер – підлеглий комп'ютер». Це означало, що у разі виходу з ладу будь-якого вузла мережі, вузли, які залишилися у робочому стані, зберігали можливість передачі пакетів даних від відправника до одержувача альтернативним маршрутом.

Однак, як відомо, одних дротів для передачі інформації недостатньо – передаючій та приймаючій сторонам необхідно також дотримуватися низки угод, які дозволяють строго регламентувати передачу даних, а також гарантувати, що ця передача пройде без спотворень. Такий набір правил називається ***протоколом передачі даних***. Простіше кажучи, протокол – це набір правил, який дозволяє системам, що взаємодіють у рамках мережі, обмінюватися даними у найбільш зручній для них формі.

Як основний мережний протокол ARPAnet був прийнятий ***міжмережний протокол*** *(Internet Protocol, IP).* Саме протокол IP стане згодом головним протоколом Всесвітньої Мережі — Internet. Міжмережний протокол є

універсальним кроссплатформенним стандартом, який дозволяє об'єднувати у мережу абсолютно різнорідні обчислювальні машини, які працюють під управлінням різних операційних систем. Важливо лише, щоб ці системи підтримували протокол IP.

Принципи, які були покладені в основу ARPAnet, виявилися виключно вдалими, мережа почала швидко розвиватися. У 1972 році вона налічувала вже 37 вузлів. У жовтні 1972 року Роберт Кан (Kahn, Robert; нар. 1938), професор математики з MTI, який перейшов на роботу у BBN, влаштував успішну публічну демонстрацію мережі на Першій міжнародній конференції з телекомунікацій у Вашингтоні. Ця демонстрація викликала величезний резонанс наукової громадськості. До кінця 70-х років ARPAnet охоплювала всю територію США, включаючи Гавайські острови, були встановлені перші міжконтинентальні з'єднання з Великобританією та Норвегією.

В мережі ARPAnet вперше з'явилася програма Creeper, яка вважається першим у світі комп'ютерним вірусом, якому у 2011 році виповнилося 40 років. Creeper був написаний у 1971 році співробітником компанії, яка займалася обслуговуванням ARPAnet.

Creeper не був комп'ютерним вірусом у сучасному розумінні цього терміну. Його творець – інженер Боб Томас просто намагався написати програму, яка могла б сама «пересуватися» між комп'ютерами, не припускаючи, що таким чином можна нанести будь-яку шкоду.

На відміну від комп'ютерних вірусів та хробаків, створених пізніше, Creeper вилучав свою копію з системи при переміщенні на новий комп'ютер.

Потрапляючи на нову машину, програма виводила повідомлення «I'm the creeper, catch me if you can» і негайно намагалася перебратися далі. Creeper працював на комп'ютерах DEC PDP-10 з операційною системою TENEX.

Для вилучення настирного, але у цілому нешкідливого вірусу невідомим була створена програма Reaper. По суті це був вірус, що виконував деякі функції, властиві антивірусу: він поширювався по обчислювальній мережі і у разі виявлення тіла вірусу Creeper знищував його. Проте слово «вірус» тоді відносно комп'ютерних програм ще не використовувалося.

2.4. Розвиток мереж пакетної комутації

До середини 1970-х років комп'ютерні мережі пакетної комутації почали рости, як гриби після дощу. По-перше, проявили ініціативу урядові та академічні кола. Успіх міністерства оборони у побудові мережі ARPAnet стимулювало міністерство енергетики США на створення мережі MFENet для дослідників термоядерного синтезу, потім фахівці в області фізики високих енергій отримали мережу HEPNet. Для астрофізиків з NASA побудували мережу SPAN, а для фахівців з інформатики з академічних та промислових кіл – CSnet. Безкоштовне розповсюдження компанією AT&T операційної системи UNIX, в яку спочатку був вбудований комунікаційний протокол UUCP (Unix-

Unix CoPy), породило безліч некомерційних академічних та аматорських мереж, деякі з них розвинулися до національних та міжнародних масштабів. Наприклад, мережа обміну новинами Usenet, про яку йтиметься далі.

По-друге, почали створюватися приватні комерційні мережі. Підприємці відчули, що метод пакетної комутації дозволяє створювати мережні передачі даних, такі ж відкриті для всіх охочих, як телефонна мережа загального користування. Звичайно, розраховувати на те, що рядовий користувач підключатиметься до віддалених комп'ютерів з метою проведення наукових обчислень було нереально, комерційний успіх публічних мереж бачився у наданні інших масових послуг, перш за все для передачі повідомлень електронної пошти. Електронна пошта, яка народилась у надрах ARPAnet, і про яку буде мова далі, відкрила новий та дуже перспективний ринок, який реабілітовував капіталовкладення у великомасштабні публічні мережі пакетної комутації. Компанія BBN, яка побудувала некомерційну ARPAnet, утворила крупну комерційну мережу Telenet, конкуренти створили мережі Tymnet, Compuserv, Datapac, Transpac тощо, що відрізняються технологією. За межами США також почали будуватися мережі передачі даних. У Франції у 1972 році почалася розробка мережі з комутацією пакетів Cyclades, у Великобританії створена мережа BTnet і т.і.

На жаль, всі ці мережі були несумісні одна з одною, оскільки грунтувалися на несумісних технологічних рішеннях та протоколах. ***Протокол (protocol)*** – це угода (стандарт), яка визначає формати даних та процедури взаємодії між об'єктами у мережі. Даний термін фахівці запозичили з дипломатичної практики, де правила поведінки учасників зустрічей та переговорів жорстко регламентовані, інакше можуть виникнути непорозуміння. Відмінність протоколів призводила до того, що комп'ютери, які були підключені до різних мереж та фізично сполучені один з одним, не могли обмінюватися повідомленнями. Ця обставина об'єктивно гальмувала широке розповсюдження мереж і заважала розвитку мережного бізнесу.

Вихід з положення, що створилося, був можливий тільки двома способами: або уніфікувати внутрішню технологію побудови мереж, з тим, щоб вони були спочатку сумісними, або шукати шляхи об'єднання різних мереж на зовнішньому рівні.

У 1976 році група провідних виробників мережного устаткування та телефонних компаній, які були зацікавлені в уніфікації мереж пакетної комутації, об'єднавшись під егідою МККТТ, розробила систему протоколів передачі даних під назвою ***Рекомендація X.25***, в основу якої була покладена технологія DECnet. Багато фірм почали виробляти устаткування, яке здійснювало комутацію пакетів за цим протоколом, і в результаті у 80-і роки мережі X.25 набули дуже широкого поширення в усіх країнах, оскільки вони забезпечували гарантовану та захищену передачу пакетів на ненадійних та повільних каналах зв'язку того часу. Протоколи X.25 стали загальноприйнятим стандартом для приватних та комерційних мереж, на них до

цих пір працюють численні міжнародні банківські системи, системи резервування квитків на авіалініях і т.і.

2.5. Виникнення Internet (80-ти роки)

Іншим шляхом пішли творці некомерційних мереж. З метою створення можливостей для об'єднання різнорідних мереж Роберт Кан, що перейшов у 1972 році на службу в ARPA, запропонував концепцію відкритої мережної архітектури, яка заснована на універсальних протоколах передачі даних. Для реалізації проекту, названого «Internating», Канн запросив Вінтона Серфа (Cerf, Vinton; нар. 1943) – аспіранта Стенфордського університету, який вивчав реакцію мереж на різні стресові ситуації.

Співпраця двох фахівців дала прекрасні результати. У 1974 році була опублікована перша версія сімейства протоколів **TCP/IP** (Transfer Control Protocol / Internet Protocol – Протокол управління передачею / Міжмережний протокол), а потім почався довгий процес його реалізації, доопрацювання та відлагодження, який тривав до початку 80-х років. У 1980 році TCP/IP був прийнятий як військовий стандарт, а 1 січня 1983 року на нього перейшла вся мережа ARPAnet. Незабаром до неї підключилися CSnet, Usenet, далі процес розповсюдження TCP/IP прийняв лавиноподібний характер. На базі ARPAnet почала розвиватися «**мережа мереж**» – **Internet**.

Що ж таке Internet? Точне визначення цього терміну було дане тільки у 1995 році, коли Федеральна мережна рада одностайно схвалила відповідну резолюцію. Internet – це не конкретна обчислювальна мережа, а співтовариство мереж різної фізичної природи, що належать різним власникам, які добровільно об'єдналися на базі логічних стандартів передачі даних. Визначальною умовою входження деякої мережі до Internet є ухвалення на зовнішньому рівні адрес, форматів та процедур, які були регламентовані базовими протоколами TCP/IP, при цьому внутрішня структура мережі може бути якою завгодно, у неї ніхто не втручається. Міжмережний обмін здійснюється через прикордонні ***шлюзи-маршрутизатори (gateways).***

Хоча мережа Internet ніким оперативно не керується, порядок у ній, враховуючи кількість та різноманітність учасників, повинен бути дуже суворий. Вже у 1979 році були створені перші координуючі органи – Дослідницька група (Internet Research Group) та Рада з конфігураційного управління Internet (Internet Configuration Control Board, ICCB). У подальшому вони багато разів реструктурувалися та міняли назви, при цьому їхня роль та авторитет у співтоваристві Internet постійно посилювалися.

80-і роки – час бурхливого зростання Internet. Якщо на початок цього десятиліття кількість підключених **користувальницьких комп'ютерів (хостів)** була близько 200, то на кінець вона складала більше 100000. Кількість комп'ютерів зростала експоненціально, подвоюючись щороку. Спочатку основу інфраструктури **Мережі** (ось так, із заголовної букви, часто називають «мережу мереж») складала ARPAnet, оскільки підключатися до неї ніхто не

забороняв, а програмне забезпечення протоколів розповсюджувалося безкоштовно. Проте, коли кількість учасників збільшилася до сотень, затурбувалося Міністерство оборони США. Для того щоб відгородитися від чужих користувачів, воно у 1983 році утворило самостійну військову мережу MilNet. Демократична ARPAnet виявилася безпритульною, а майбутнє Internet – під загрозою.

Врятував мережу Internet та надихнув у неї нове життя національний науковий фонд США (National Science Foundation – NSF). Він узяв під опіку ARPAnet, а у 1986 році приступив до створення мережі NSFnet, основу якої складала високошвидкісна магістраль передачі даних NSFnet Backbone. У наступні вісім років фонд вклав в інфраструктуру NSFnet 200 мільйонів доларів. Магістраль виросла з шести вузлів, сполучених каналами по 56 Кб/с, до 21 вузла з множинними зв'язками по 45 Мб/с. Кількість мереж в Internet у 1995 році перевищило 50 тисяч, з яких приблизно 29 тисяч розташовувалося на території Сполучених Штатів, а решта – в усіх частинах світу і навіть у космічному просторі.

Згідно закладеної спочатку у даний проект ідеї, згодом вчені планували об'єднати більшість американських дослідницьких центрів у глобальну інформаційну систему, створивши своєрідну «мережу мереж» (Internetwork, скорочено — Internet). Ця система повинна була містити найсвіжішу інформацію, що постійно оновлюється, про наукові дослідження дослідницьких установ США. За задумом Національного наукового фонду поява такої мережі дозволила б більшості дослідницьких інститутів Америки мати швидкий доступ до найсучасніших розробок вчених.

У 1990 році мережа ARPAnet, проживши двадцять років, офіційно припинила існування, ставши щасливою жертвою власної популярності. Народжена нею **глобальна супермережа Internet** почала переможну ходу планетою.

2.6. Комплексне інформаційне обслуговування в Internet (90-і роки)

Справжнього розквіту комп'ютерні мережі досягли у 90-і роки. Цьому сприяло кілька чинників.

По-перше, досягнення в області технології локальних обчислювальних мереж, про які йтиметься у наступному параграфі, дозволили створювати виключно дешеві та надійні обчислювальні мережі для підприємств будь-якого масштабу. Практично кожен офісний комп'ютер тепер працює у локальній мережі.

По-друге, розвиток високошвидкісних магістральних (оптоволоконних та супутникових) каналів передачі даних різко здешевив міжміські та міжнародні з'єднання і зробив їх доступними рядовому користувачеві.

По-третє, і це найголовніше, принципово змінилося ставлення до комп'ютера у масовій свідомості. Якщо раніше він використовувався, перш за все, для виконання обчислень (і називався в літературі найчастіше електронною

обчислювальною машиною – ЕОМ), то тепер комп'ютер в основному є інформаційним пристроєм, за допомогою якого людина обмінюється поштою, дізнається новини, читає книжки, слухає музику і навіть дивиться кінофільми.

Саме інформаційна потреба суспільства з'явилася тим каталізатором, який призвів до вибухового розвитку Internet в останнє десятиліття XX століття. Спочатку ці потреби були відносно скромними і обмежувалися, в основному, електронною поштою, проте після створення у 1991 році технології Всесвітньої павутини – World-Wide Web (детально про неї йтиметься далі) процес залучення широких мас до Internet почав набувати характеру епідемії. Експоненціальне зростання кількості комп'ютерів у мережі продовжувалося впродовж усіх 90-х років, до 2001 року їх там стало вже більше 100 мільйонів. З технічного феномену мережа Internet перетворилася на економічний і соціальний феномен.

У цих умовах ніякий уряд не міг би нести тягар витрат на експлуатацію гігантської публічної Мережі, її необхідно було приватизувати та зробити самоокупною. У США прийшли до ідеї самофінансування Internet у кінці 1980-х років, поступово мережа комерціалізувалась, а до 1995 року бюджетне фінансування NSFnet Backbone було повністю припинено. На зміну державним організаціям прийшли приватні компанії, які відкрили нову і дуже прибуткову ринкову нішу – організацію комерційних комп'ютерних мереж та надання послуг Internet **кінцевим користувачам** (**Internet providing**).

Досить швидко сформувалася ієрархія **провайдерів**. На верхньому рівні розташовуються найбільші мережні компанії, які, орендуючи магістральні канали первинної мережі у операторів телекомунікації, організовують вторинні комп'ютерні мережі в межах країни або навіть кількох країн. Найбільшим у світі провайдером Internet є компанія America On-Line (AOL), яка була утворена у 1985 році. У 1999 році її дохід склав 2,6 мільярда доларів, а кількість абонентів перевищила 20 мільйонів.

Дрібніші провайдери діють в межах окремих міст або географічних регіонів. Вони «оптом» купують послуги Internet у провайдерів першого рівня і продають їх «у роздріб» кінцевим користувачам, використовуючи для цього, як правило, місцеву телефонну мережу загального користування.

Прагнучи збільшити свої доходи, провайдери постійно розширюють перелік послуг, що надаються, переходячи від простого технічного підключення до Мережі до комплексного інформаційного обслуговування (електронна пошта, служба миттєвих повідомлень, збереження даних абонентів, оренда додатків, електронна комерція, Internet-віщання тощо). У цьому відношенні характерна доля AOL. Почавши з організації модемних пулів, компанія почала розширюватися, набуваючи фірм, які прославилися розробкою нових мережних технологій. У 1998-1999 роках вона купила Netscape Communications, яка створила знаменитий браузер та однойменний інформаційний портал, та ізраїльську фірму Mirabilis, яка розробила технологію **ICQ** для миттєвого обміну короткими повідомленнями.

2.7. Internet нового покоління

Web-революція середини 90-х років докорінно змінила зовнішність Internet з погляду користувача. Подорожі по Web (***Web-surfing***) стали звичними для таких людей, які ніколи раніше не мали справ з комп'ютером. З'явився новий вигляд послуг: **Intrnet-телефонія, Intrnet-радіомовлення** та **Intrnet-телебачення**. У цих умовах обсяг мережного трафіку різко зріс, пропускна спроможність магістральних каналів була швидко вичерпана, і в мережі стало тісно. Аналогічна ситуація відбулася півстоліття тому на дорогах, коли автомобіль став масовим видом особистого транспорту.

Для вирішення проблеми було необхідне будівництво ***інформаційних супермагістралей (information superhighway)***, пропускна спроможність яких набагато разів перевищувала б показники існуючих каналів Internet, і які зв'язували б найрізноманітніші інформаційні ресурси у масштабах планети. Цей помітний термін придумав американський сенатор Альберт Гор молодший (Gore, Albert Jr.; нар. 1948), батько якого, також сенатор, свого часу відстоював ідею будівництва автомобільних супермагістралей. Будучи законодавцем, а потім і віце-президентом в уряді Клінтона (1992 – 2000 р.р.), Ел Гор зробив надзвичайно багато для розвитку мережі Internet, впровадження комп'ютерних технологій в освіту. За його ініціативою у США були прийняті закони, які регулювали мережну діяльність. Розгорнулися також великомасштабні роботи з модернізації мережі Internet, перетворення її на справжню інформаційну супермагістраль для суспільства XXI століття.

Восени 1996 року стартували два багатомільйонні проекти: урядовий Next Generation Internet – NGI та проект Internet2, що спонсорується вищими навчальними закладами та компаніями інформаційних технологій.

Мережа NGI створюється для федеральних структур – ARPA, міністерства оборони, міністерства енергетики, які відповідають за атомну програму, національного наукового фонду NSF, космічного відомства NASA, національного інституту стандартів, найбільших наукових лабораторій та суперкомп'ютерних центрів. Вона передбачає тисячоразове збільшення швидкості передачі даних у порівнянні з Internet і призначена для забезпечення наукових досліджень та медичної діагностики.

Мережа Internet2 повинна об'єднати гігабітними магістралями 135 університетських кампусів (територія університетського містечка, а також саме таке містечко). Передбачається побудувати щось на зразок віртуального університету для студентів та викладачів з тим, щоб вони могли читати книги з бібліотек, які знаходяться на відстані тисяч кілометрів, брати уроки в інших ВНЗ, спільно проводити наукові дослідження.

У лютому 1999 року перша магістраль Internet2 під назвою Abilene була введена в експлуатацію. Вона зв'язала оптичним кабелем довжиною 21 тис. км. два узбережжя Сполучених Штатів – від Нью-Йорка до Сіетлу. Швидкість передачі даних складає – 2,4 Гбіт/с, поступово вона буде збільшена до 9,6 Гбіт/с, відповідальність за повсякденний супровід Abilene покладено на

Університет штату Індіана. Abilene стане однією з двох головних магістралей Internet2. Інша базуватиметься на vBNS – very high speed Backbone Network Service, яка є модернізованою магістраллю NSFnet, що належить Національному науковому фонду США та компанії MCI WorldCom.

Крім збільшення швидкості передачі даних, проект Internet2 ще має на меті впровадити нову (шосту) версію протоколу IP, під назвою **IPv6**. Вона принципово відрізняється від усіх попередніх, зокрема, тим, що кардинально вирішує проблему нестачі IP адрес. Коли у 70-і роки Роберт Кан та Вінтон Серф розробляли першу версію протоколу IP, вони передбачили 32-бітову адресу абонента, припускаючи, що в осяжному майбутньому буде не більше 256 мереж, а кількість хостів аж ніяк не перевищить сотні мільйонів. Дійсність, як ми бачимо, спростувала цей прогноз, на даний час можливості 32-бітової адресації практично вичерпані. Протокол IPv6 відводить на адресу 128 розрядів, це досить для того, щоб адресувати комп’ютери, навіть якщо їх буде по півтори тисячі на кожен квадратний метр земної поверхні.

3. Локальні обчислювальні мережі

На відміну від розглянутих вище територіальних мереж (їх називають ще ***глобальними***), що простяглись на десятки, – сотні – тисячі кілометрів, ***локальні мережі – ЛВС (Local Area Network – LAN)*** охоплюють територію однієї будівлі або групи близько розташованих будівель. Проте не відстань є визначальною характеристикою локальних мереж, а спосіб доставки повідомлень. Локальні мережі також базуються на передачі пакетів, проте в них відсутні маршрутизатори, а замість принципу комутації використовується принцип селекції пакетів.

Історія розпорядилася так, що локальні мережі з'явилися дещо пізніше глобальних. Перші експерименти були проведені на початку 70-х років, а широке розповсюдження локальних мереж почалося вже у 80-х.

3.1. Широкомовні мережі

Ідея створення мережі із селекцією пакетів народилася на Гавайських островах і була навіяна самою географією регіону. Коли у Гавайському університеті, який розташований на центральному острові Оаху, була встановлена велика ЕОМ, яка була здатна працювати в режимі розподілу часу, виникла проблема підключення до неї безлічі терміналів, які були розкидані по всьому архіпелагу. Зробити це за допомогою звичайних дротяних ліній було неможливо, оскільки глибокі протоки між островами ускладнювали прокладку телефонних кабелів, тому основним засобом спілкування міг бути тільки радіозв'язок.

Надзвичайно дотепне та вдале розв'язання проблеми запропонував професор Норман Абрамсон (Abramson, Norman), який до приїзду на Гаваї працював у Стенфорді і Массачусетському технологічному інституті. Його ідея полягала у тому, щоб побудувати ***широкомовну (broadcasting)*** радіомережу, в якій кожен абонент мав бути оснащений приймачем, причому всі вони повинні працювати на одній і тій самій частоті.

Принцип роботи широкомовної мережі простий. Припустимо, абонент 1 бажає передати пакет абонентові 3. На передавальній стороні цей пакет забезпечується адресами відправника і одержувача. Пристрій доступу до середовища постійно прослуховує ефір, і, як тільки там наступає пауза, відправляє повідомлення. Оскільки ефір загальний, то пакет буде зареєстрований усіма приймачами. Проте всі абоненти, окрім третього, його не отримають, оскільки спеціальний пристрій – **селектор** порівнює адресу одержувача у пакеті із власною і пропускає чужі пакети. Таким чином, з потоку пакетів, що циркулюють в ефірі, кожен абонент обирає тільки ті, які адресовані саме йому.

На практиці реалізувати принцип селекції було не так просто. Оскільки до передавального середовища одночасно підключаються всі абоненти, його пропускна спроможність повинна бути достатньо високою. Крім того,

необхідно було вирішити проблему зіткнення (колізії) пакетів, які можуть одночасно посилатися кількома передавачами. Проте, всі технічні труднощі були успішно подолані, і у 1970 році перша широкомовна мережа, яка була названа **Aloha** (це слово на місцевій мові відповідає англійському hello, тобто «привіт!») успішно запрацювала. Спочатку швидкість передачі даних в мережі складала всього 9600 біт/с, надалі вона постійно зростала. Мережа Aloha функціонує донині, а Норман Абрамсон є віце-президентом компанії AlohaNet.

3.2. Технологія Ethernet

Подальший прогрес комп'ютерних мереж, які використовують принцип селекції пакетів, пов'язаний з ім'ям Роберта (Боба) Меткалфа (Metcalfe, Robert; нар. 1946). Після закінчення Массачусетського технологічного інституту він поступив на роботу у знаменитий Xerox PARC. На початку 70-х років там розроблявся проект Alto і був створений перший, ще дуже дорогий, лазерний принтер, який був спроможний друкувати з роздільною здатністю 500 крапок на дюйм. Було поставлено завдання забезпечити можливість друку документів на одному лазерному принтері з будь-якого комп'ютера центру. Меткалф блискуче вирішив проблему, запропонувавши створити локальну комп'ютерну мережу, подібну Aloha, але зменшену до розмірів однієї будівлі. При цьому як загальне передавальне середовище пропонувалося використовувати звичайний коаксіальний кабель. У 1973 році мережа була побудована, вона називалася Alto Aloha Network і працювала зі швидкістю 2,94 Мбіт/с.

Через декілька років експериментів, у 1976 році, нова технологія локальних мереж була опублікована і отримала назву ***Ethernet*** (від слова ***ether – ефір***), проте офіційним днем її народження вважається 22 травня 1973 року, коли Меткалф поклав на стіл керівництву 13-сторінковий проект з описом ідеї Alto Aloha Network.

Як ми вже не раз відзначали, фірма Xerox, в якій народилися багато геніальних ідей (пригадаємо хоча б мишу, графічний інтерфейс комп'ютера Alto, мову програмування Smalltalk, растровий дисплей, лазерний принтер, мову опису сторінок PostScript), не зуміла довести ці ідеї до комерційної реалізації. Тому у 1979 році Меткалф пішов з Xerox і організував власну фірму 3Com (скорочення від слів computer, communication, compatibility) з метою просування на ринок технології Ethernet. Йому вдалося умовити трьох лідерів комп'ютерного ринку – DEC, Intel та Xerox прийняти Ethernet як загальний промисловий стандарт, а у 1980 році цей стандарт став міжнародним.

Впродовж 20 років технологія Ethernet пережила справжній вибух популярності, перевершивши найсміливіші прогнози. Близько 100 мільйонів комп'ютерів мають вихід у світ через мережні карти, які працюють за її протоколами. Масове виробництво пристроїв призвело до різкого зниження їх вартості: якщо в середині 80-х років мережна карта коштувала близько 1500 доларів, то зараз – декілька десятків. Одночасно постійно збільшується швидкість передачі даних. Почавши з 2,94 Мбіт/с, технологія швидко засвоїла 10, потім 100 Мбіт/с (**Fast Ethernet**). Нещодавно прийнято стандарт **Gigabit Ethernet**, не за горами і вищі швидкості.

Фізично середовище передачі в мережі Ethernet може бути організоване різними способами. Ранній варіант, який був запропонований самим Меткалфом, використовує магістральний коаксіальний кабель, до якого у потрібних місцях підключаються абоненти. Цей варіант найпростіший і найдешевший, оскільки не вимагає додаткового активного устаткування, проте з експлуатаційної точки зору не найзручніший: поганий контакт у точці

підключення одного комп’ютера виводить з ладу всю мережу. Тому останнім часом застосовується топологія у вигляді зірки. В центрі зірки встановлюється активне устаткування, яке організовує загальне середовище, до нього витими мідними парами підключаються абоненти. У простому випадку цим устаткуванням є ***концентратор (hub)***, що просто транслює сигнали з кожного вхідного порту на усю решту, проте з розвитком технології активне устаткування ставало все «розумнішим». У сучасних мережах воно представлено вельми досконалими ***комутаторами (switch)***, які з метою зменшення імовірності колізій аналізують конфігурацію мережі та передають пакети тільки на той порт, до якого підключений клієнт. Таким чином, з розвитком технології Ethernet межа між мережами з комутацією та селекцією пакетів поступово стирається.

З іншого боку, для мереж, в яких комп’ютери не мають постійного місцезнаходження, наприклад, у виставкових залах, розроблена бездротова технологія RADIOLAN, коли кожна мережна карта оснащена мініатюрним приймачем. Тут ми бачимо чистий приклад спірального розвитку ідеї – радіомережа Aloha на сучасній технологічній базі.

Заради об’єктивності слід зазначити, що на сьогоднішній день Ethernet – не єдина технологія локальних мереж, проте її суперники (Token Ring, ArcNet, Fiber Distribution Data Interface – FDDI) за рівнем поширеності, співвідношенням ціна/якість та демократичністю значно їй поступаються.

4. Мережні послуги

Давайте відійдемо від технічних деталей передачі повідомлень та обговоримо практичну користь комп'ютерної мережі для кінцевого споживача. Взагалі кажучи, набір послуг для локальної та глобальної мережі розрізняється, але, завдяки поширеному використанню протоколів Internet, грань між ними поступово стирається. З одного боку, в локальних мережах починають широко застосовуватися принципи Internet, з'явився навіть спеціальний термін ***інтранет (intranet)***, що означає технологію створення корпоративних інформаційних мереж на протоколах «великого» Internet, але без виходу у зовнішній світ. З іншого боку, кілька локальних мереж однієї фірми, які розташовані в різних містах або країнах, можуть через Internet (це називається ***прокладкою тунелю***) з'єднуватися таким чином, що глобальна мережа буде абсолютно прозорою для будь-яких видів спілкування, а користувачам здаватиметься, що вони працюють в одній локальній мережі. Така технологія об'єднання отримала назву ***екстранет (extranet).***

4.1. Віддалений доступ до ЕОМ

Спочатку комп'ютерні мережі називалися обчислювальними і надавали своїм користувачам єдину високорівневу послугу – доступ до ЕОМ, що знаходиться в іншому місці, в режимі віддаленого терміналу. Власно кажучи, для цього вони і створювалися на початку 1970-х років, коли про персональні комп'ютери ще і мови не було. Пригадаємо, що першим повідомленням, яке було передано в ARPAnet на момент її народження 29 жовтня 1969 року, була команда «LOGIN» на підключення користувача з Лос-Анджелесу до ЕОМ SDS 940, яка була розташована в Стенфорді. У 1972 році був розроблений протокол віртуального терміналу Telnet, який після переходу ARPAnet до транспортних протоколів TCP/IP був адаптований до нього і став стандартним протоколом прикладного шару Internet. За допомогою клієнтської частини Telnet, яка встановлюється на домашньому комп'ютері, можна (знаючи ім'я користувача та пароль входу) підключитися до будь-якого сервера, який надає послугу віддаленого доступу, і працювати на ньому в абсолютно прозорому режимі, неначе між клієнтом та сервером немає ніякої мережі. Таким чином, не виходячи з будинку, можна виконувати будь-які обчислення і навіть працювати адміністратором інформаційної системи, сумлінно виконуючи всі пов'язані з цим обов'язки.

Оскільки згадали тут про сервер, давайте нагадаємо йому визначення. Сервер – це будь-який окремо узятий постійно працюючий комп'ютер у мережі, який дозволяє іншим комп'ютерам, спрощено кажучи, використовувати собе як "посередника" при передачі даних.

Об'єднання комп'ютерів у локальній мережі вимагає наявності потужного «головного» комп'ютера, до якого під'єднані решта робочих місць. Такі комп'ютери отримали назву серверів. Прикладні комерційні та бізнес-системи,

які передбачають роботу з базами даних, потужними видавничими системами, мережевими системами та системами обслуговування комунікацій, вимагають переходу до моделі обчислень «***клієнт-сервер***». У такому варіанті частину роботи может виконувати сервер, а частину – комп'ютер користувача цієї мережі.

Залежно від інформації, яка міститься на сервері, вони поділяються на такі типи: файл-сервер, сервер бази даних, принтсервер, обчислювальний сервер, сервер додатків. Таким чином, тип сервера визначається типом ресурсу, яким він володіє – файлова система, база даних, принтери, процесори, прикладні пакети програм.

Слово «сервер» часто використовується ще в наступних словосполученнях.

Виділений сервер. Виділений сервер – це звичайний сервер, тобто комп'ютер, постійно підключений до Інтернету і конфігурований для роботи в Інтернет. Слово «виділений» підкреслює, що цей сервер ні від кого не залежить і перебуває у повному розпорядженні його власника. Власник цього комп'ютера може розміщувати на ньому будь-яку інформацію і використовувати сервер, як йому заманеться. Власниками виділених серверів є, як правило, досить великі фірми і компанії, що спеціалізуються на роботі в Інтернеті.

Віртуальний сервер. Мати комп'ютер, цілодобово підключений до Інтернету по швидкісній лінії зв'язку, досить невигідно і не завжди по кишені охочим помістити свою інформацію в Інтернеті. Виходом з ситуації, що склалася, може стати використання віртуального сервера.

Припустимо, що власник якого-небудь виділеного сервера має дуже великий об'єм жорсткого диска. Тоді він може розбити цей диск на невеликі частини і передати їх в оренду іншим користувачам Інтернету. Отримавши частину дискового простору на видаленому комп'ютері, ви стаєте володарем віртуального сервера. Фізичного доступу до нього ви, природно, мати не будете – сервер може знаходитися на іншому кінці світу. Проте під час коротких сеансів роботи в Інтернеті ви зможете розміщувати, видаляти і редагувати будь-яку інформацію на своєму віртуальному сервері. І ця інформація буде доступна іншим користувачам Мережі постійно.

Віртуальні сервери бувають платними і безкоштовними. Платні сервери відрізняються більшою функціональністю і гнучким доступом до інформації. Безкоштовні віртуальні сервери, як правило, мають дуже великі обмеження у функціональності, розміщена на них інформація не повинна виходити за рамки дозволеного, об'єм дискового простору часто не перевищує декількох гігабайт. Крім того, на сторінках вашого сайту, найімовірніше, розташовуватиметься реклама – це ваша плата за послуги, що робляться.

Використання локальних мереж і наявність у них робочих місць, що працюють під керуванням сервера, привело до виділення такого класу комп'ютерів, як ***робочі станції***. За технічними якостями робочі станції можуть значно поступатися серверам, оскільки саме ті беруть на себе основне

навантаження під час сеансу роботи. Прабатьками робочих станцій були «міні-комп'ютери».

Сервер – це саме комп'ютер ("залізо"), а не логічна частина Мережі, він може мати декілька різних IP-адрес (не говорячи вже про доменні імена), так що цілком може виглядати з Internet як декілька незалежних систем. Термін "сервер" має ще одне, абсолютно інше, визначення. Це програма (процес), яка обробляє запити клієнтів. Наприклад, додаток, що обслуговує користувачів WWW, називається Web-сервером. Як правило, з подальшого контексту буде зрозуміло, що конкретно мається на увазі.

З появою персональних комп'ютерів значення віддаленого доступу для виконання обчислювальних завдань зменшилося, але одна область застосування цієї послуги завжди залишиться актуальною. Йдеться про надскладні обчислення на суперкомп'ютерах, які визначають передовий край технічного прогресу. Ці машини надзвичайно дорогі та унікальні за своєю сутністю. Фізично вони розміщуються у найбільших національних суперкомп'ютерних центрах, а доступ до обчислювальних потужностей відбувається через мережу.

4.2. Передача файлів

Наступною високорівневою мережною послугою стала передача файлів, яка у локальних мережах довгий час була основним видом сервісу. Відомо, що найперша локальна мережа, яка була створена в Xerox PARC у 1973 році, призначалася саме для передачі файлів на загальний лазерний принтер.

На файловому сервісі виросла і набула світової популярності мережна ОС 80-х років **NetWare**. Прикладний рівень мережі NetWare організований за технологією клієнт-сервер: файли, що використовуються колективно, зберігаються на виділеному сервері, а клієнтські частини, які встановлені на призначених для користувача комп'ютерах, дають можливість працювати з цими файлами так, як нібито ті розташовуються на звичайному локальному диску. Зручність та популярність технології NetWare змусили фірму Microsoft ввести аналогічну послугу в операційну систему Windows, проте доступ у ній до віддалених ресурсів (файлів та принтерів) ведеться за ***одноранговою схемою***: кожен комп'ютер може виступати в ролі як сервера, так і клієнта.

В Internet передача файлів ведеться за протоколом **FTP** (File Transfer Protocol – Протокол передачі файлів), який був розроблений для мережі ARPAnet у 1972 році та згодом вбудований у родину TCP/IP. На даний час у світовій Мережі є безліч FTP-серверів, які зберігають гігабайти корисної та цікавої інформації, – від програм, які безкоштовно поширюються до електронних книжок, музики та відео. Встановивши на своєму комп'ютері клієнтську частину FTP (його у просторіччі називають «**качалкою файлів**»), можна легко зібрати колекцію улюблених книг та музичних творів, заплативши при цьому тільки за час роботи в Internet.

4.3. Електронна пошта

Засвоївши до 1972 року міжміську пересилку файлів, творці першої комп'ютерної мережі ARPAnet зробили наступний логічний крок, створивши ***електронну пошту (E-mail)***. Це трапилося у березні того ж 1972 року. Батьком електронної пошти вважається Рей Томлінсон (Ray Tomlinson) з компанії BBN, тієї самої, яка виграла підряд на будівництво перших вузлів мережі. Він написав перші програми для відправлення та читання пошти – SNGMSG та READMAIL, і запропонував формат електронної адреси. У заголовку електронного листа потрібно було якось розділити поля «кому» та «куди». Рей розумів, що потрібно знайти такий символ, який напевно не міг зустрітися ані в імені, ані в адресі. Подивившись на клавіатуру свого терміналу, він побачив там знак @ (в англійській мові називається «Комерційне at»), який з тих пір використовується як розподільник.

Вважається, що лише російські та українські користувачі називають знак **@** собакою або песиком. Говорять, що @ корейською означає равлика, фінською – сплячу кішечку, угорською – хробака, китайською – мишеня, шведською – булочку з корицею.

Інші знаки електронної пошти складаються з латинських символів. На даний час міжнародна організація IETF, яка займається розвитком протоколів та архітектури мережі Internet, надала стандарти роботи електронної пошти, адреси в якій можуть складатися з нелатинських символів. Зараз в Internet проходить тестування багатомовної електронної пошти. Правда, адреси, що запропоновані в рамках експерименту, як і раніше частково пишуться латиницею. Національні символи в них використовуються лише після знаку "@". У майбутньому планується затвердження поштового протоколу, який передбачає можливість створення та використання e-mail адрес, де всі символи будуть не тільки латинськими. Таким чином, у майбутньому стануть можливими поштові адреси виду **остап@бендер.уа**.

До липня 1972 року Ларрі Робертс з ARPA написав програму RD для управління поштовими повідомленнями, яка підтримувала всі стандартні функції: читання, збереження, створення відповіді, пересилку листів. З тих пір більш ніж на двадцять років електронна пошта стала найпопулярнішим мережним додатком. Завдяки їй, спочатку тисячі, а потім і мільйони людей залучилися до Мережі, на візитній картці будь-якої ділової людини є адреса E-mail. Чи можна було припустити, що сучасна людина щодня писатиме та отримуватиме десятки листів, а загальна щоденна кількість повідомлень перевищить мільярд? Сьогодні в електронний конверт можна вкласти не просто текст, але й фотографію, звук і навіть відео, а для того, щоб завести безкоштовну поштову скриньку необхідно мати тільки вихід в Internet, цю послугу навперебій пропонують тисячі поштових серверів.

Функціонування електронної пошти в Internet підтримується кількома протоколами прикладного рівня, які надбудовані над TCP/IP. На даний час

найпопулярнішим протоколом для відправки пошти є **SMTP** (Simple Mail Transfer Protocol – Простий протокол передачі пошти), а для прийому – **POP3** (Post Office Protocol, v. 3 – Протокол поштового відділення, версія 3).

Якщо звичайна пошта – це пересилка паперових листів та документів за допомогою державних поштових служб, то електронна пошта – це передача електронних листів з використанням міжнародної мережі Internet.

Для ведення електронної пошти використовуються спеціальні поштові програми, які можна знайти в Internet або в сучасних операційних системах. Зокрема, в операційну систему Windows входить поштова програма **Outlook Express.**

На основі стандартної електронної пошти в Internet реалізовано ще декілька високорівневих сервісів, наприклад, ***списки розсилки (mailing lists)*** та поштові конференції. Списки розсилки призначені для розповсюдження інформації серед регулярних підписчиків. Послявши лист з проханням про підписку на один із спеціалізованих серверів, де встановлена відповідна програма-робот, можна регулярно отримувати електронною поштою новини в обраній області інтересів: прогнози погоди, курси валют та усіляку іншу корисну інформацію доти, доки це не набридне, і не буде надіслано листа з відмовою від підписки.

Поштові конференції на відміну від списків розсилки є двосторонніми. Тобто, при цьому можна не тільки одержувати листи із сервера розсилки, але можна посилати туди свої, при цьому вони будуть негайно розіслані всім учасникам списку. Така форма спілкування виключно зручна для груп людей, що пов’язані загальними інтересами, наприклад, для вчених різних країн, які працюють в одній вузькій області науки.

Серед поштових порталів Росії та України найвідоміші – Mail.Ru та Ukr.net. Відоміший Mail.Ru з’явився на світ у 1998 році, коли Рунет був ще дуже маленьким. Команда програмістів, яка працювала у пітерському офісі американської софтверної компанії, створила нове ПО для поштових веб-серверів, яке надалі передбачалося продавати західним компаніям. Щоб протестувати сервіс, його тимчасово виклали у відкритий доступ для російських користувачів на домені, сталося це 15 жовтня 1998 року. Тепер ця дата вважається офіційним днем народження порталу.

4.4. Групи новин, форуми

Ще зручнішу форму взаємодії людей, які спілкуються на задану тему, надають ***групи новин (newsgroups)***. На відміну від списків розсилки та поштових конференцій, при цій технології повідомлення не забивають поштові скриньки підписчиків, а зберігаються вони на спеціалізованих серверах, які постійно і автоматично обмінюються один з одним останніми новинами.

Групи новин – це дуже популярна інтернет-служба, що є деяким дискусійним клубом, розділеним по інтересах. Часто цю службу називають телеконференціями (або просто конференціями). Нині в Інтернеті налічуються

десятки тисяч телеконференцій. Відмінною особливістю конференцій є те, що вони допомагають обмінюватися думками за різними темами користувачам, які в реальному житті навряд чи коли-небудь би зустрілися.

Конференция организована у вигляді переписки, листи (статті) якої доступні усім її учасникам. Кожен користувач може приймати участь в обговореннях, відповідати на питання, а також задавати свої питання.

Підключившись до найближчого сервера новин та обравши групу, яка його цікавить, абонент може заздалегідь прочитати заголовки, а вже потім викачати листи, що цікавлять його. Ідея груп новин народилася у 1979 році на східному узбережжі США, коли між Університетом штату Північна Кароліна (University of North Carolina – UNC) та Університетом Дьюка (Duke University) був організований регулярний обмін останніми новинами. Перші програмні реалізації пов'язані з іменами Белловіна, Деніела і Траскотта (Bellovin, Steve; Daniel, Steve; Truscott, Tom), вони ґрунтувались на відомому вже комунікаційному Unix-протоколі UUCP. Надалі до двох засновників підключилася безліч інших вузлів, в результаті утворилася мережа Usenet (Unix User Network), яка у наступні роки стала швидко зростати, і перетворилася на одну з найбільших інформаційних мереж.

Коли спеціалізовані комп'ютерні мережі одна за одною почали підключатися до Internet, Usenet не уникнула цієї долі. Для обміну новинами у 1986 році до TCP/IP був доданий прикладний протокол NNTP (Net News Transfer Protocol), який замінив класичний UUCP. Доступ до серверів новин перетворився на одну із звичайних послуг Internet.

На даний час у світі існує більше 70 тисяч груп новин, як глобальних, так і місцевих, які розраховані на жителів одного регіону. Для перегляду новин розроблено безліч клієнтських додатків, деякі з них суміщені з електронною поштою (наприклад, Outlook та Lotus Notes).

Групи новин зіграли дуже велику роль в Мережі, і не тільки технічну. Мільйони користувачів спілкуються через мережу новин, ставлять питання однодумцям і отримують відповіді з різних спеціальних питань, які охоплюються тематичними групами.

Аналогічно групам новин працюють вельми популярні у FidoNet **електронні дошки оголошень** (Bulletin Board Systems – BBS). Групи новин досягли піку популярності у 1991-1993 роках, згодом в результаті тріумфального розвитку Всесвітньої павутини їх значення зменшилося. Аналоги багатьох груп новин тепер можна знайти на Web-сторінках, там вони називаються ***форумами – forums***. Як відомо, форумом у стародавньому Римі називалася головна площа, місце обміну думками.

В Мережі форум виконує аналогічні функції. Форуми – цілком унікальне явище, яке дозволяє вирішити безліч проблем. Уявімо собі, що на Вашому комп'ютері виникли якісь неполадки. Для їх усунення можна, звичайно, розпитати у друзів і знайомих, звернутися у службу підтримки того магазину, де ви придбали комп'ютер, викликати майстрів тощо. Проте всі ці способи вимагають витрат коштів, сил та самого головного – часу. До того ж, з цими ж

самими неполадками обов’язково стикався хтось з багатомільйонних користувачів Internet. Залишається тільки знайти відповідний форум з обговоренням вашої проблеми, а при її відсутності створити нову тему. Прикладом такого комп’ютерного форуму може служити Oszone.net (www.forum.oszone.net).

Зайшовши на певний форум, користувач Internet може не тільки прочитати повідомлення, але і прикріпити до нього свій коментар у вигляді відповіді. Інший відвідувач може додати відповідь на відповідь і т. д., в результаті іноді утворюються цікаві ланцюжки обговорень.

Більшість форумів – тематичні, тобто присвячені якійсь певній області. Це ще одна їх перевага, яка дозволяє швидко знаходити однодумців або людей, які цікавляться тією ж самою проблемою. Для того щоб взяти участь в обговоренні зазвичай необхідно зареєструватися. Для цього потрібно ввести своє ім’я, створити псевдонім (так званий "нік") та вказати адресу електронної пошти. Не обов’язково ці дані повинні бути реальними. За порядком стежать спеціальні учасники, які можуть вилучати та редагувати чужі повідомлення. Їх називають модераторами.

Для виразу емоцій учасники обговорень широко користуються різними значками на кшталт **:)** або **-)**. Історики Internet точно встановили, що вперше такий значок – ***смайлик*** (від ***smile*** – усмішка, проте англійською він називається ***emoticon***) з’явився у Мережі 12 квітня 1979 року у повідомленні Мак-Кензі (MacKenzie, Kevin), яке він надіслав до групи новин.

Форуми дозволяють залишати повідомлення не в режимі реального часу. Це дозволяє обдумати свою відповідь, а також спілкуватися лише тоді, коли це зручно вам.

Нині в мережі є величезна кількість форумів, які присвячені найрізноманітнішим питанням. Для пошуку потрібного форуму, як завжди, слід використовувати пошукові системи. Запит типу "Мобільні телефони форум" у першому десятку результатів поверне найпопулярніші форуми, що присвячені мобільним телефонам.

4.5. Чат та миттєві повідомлення

Електронна пошта, групи новин та форуми, не дивлячись на всі зручності, мають один загальний і принциповий недолік – вони не забезпечують живого спілкування. Електронний лист дійде до адресата у кращому випадку за декілька хвилин, це зовсім не те, що розмова у реальному часі. Неологізм ***чат (chat)*** в перекладі означає «дружня розмова, бесіда, базікання». Ця послуга Internet призначена в основному для молодих людей, які від нудьги або цікавості бажають поспілкуватися з однолітками по всьому білому світу. Для її реалізації у 1988 році фінн Ойкарінен (Oikarinen, Jarkko) запропонував спеціальний прикладний протокол **IRC** (Internet Relay Chat – Передача чату в Internet).

Чат – це засіб обміну текстовими повідомленнями кількох (аж до кількох сотень або навіть тисяч) користувачів в режимі реального часу. Зазвичай під чатом розуміється веб-сторінка, на якій можна прочитати повідомлення одразу всіх учасників. Ця сторінка оновлюється після закінчення заданого проміжку часу, що дозволяє бачити нові повідомлення. Спілкування в чатах здійснюється у так званих "кімнатах", де віртуально збираються кілька користувачів, які мають бажання поспілкуватися на задану тему.

Запустивши на своєму комп'ютері клієнт IRC, користувач підключається до одного з чат-серверів, де йому пропонується на вибір безліч каналів, кожен з яких присвячений певній улюбленій темі, як у групах новин. Всього у світі є близько 30 незалежних IRC-мереж, які налічують у цілому більше 40 тисяч каналів. Обравши мережу та канал, користувач опиняється як би у кімнаті (chatroom), де всі можуть одночасно говорити та чути один одного, тільки ця розмова не звукова, а письмова. Кожне набране на клавіатурі слово негайно передається у канал, тому виникає повна ілюзія безпосереднього спілкування з мешканцями даної кімнати. При цьому будь-які два учасники загальної бесіди можуть перемкнутися на приватний канал та продовжити розмову tete-a-tete, тобто наодинці.

Оскільки набирати довгі тексти на клавіатурі обтяжливо, учасники чату широко користуються смайликами, жаргонними словами та специфічними скороченнями на англійській та російській мовах, так що зрозуміти сенс бесіди сторонній людині буває непросто. Завдяки своїй демократичності, чат дуже популярний серед молоді. У чаті знайомляться, призначають побачення.

Безперечна перевага чату – робота у реальному часі, проте для того, щоб поговорити через IRC з конкретним абонентом, треба зачекати, поки він з'явиться у потрібному каналі. У зв'язку з цим в найостанніший час винятковою популярністю в Internet почала користуватися технологія ***миттєвих повідомлень (instant messaging – IM)***, що є гібридом електронної пошти та чату. Винахідником служби миттєвих повідомлень є невелика ізраїльська фірма Mirabilis, яка у 1997 році запропонувала прикладний протокол і технологію **ICQ** (фонетичне скорочення від «I Seek You» – «Я шукаю тебе»; у російському (українському) просторіччі «**Аська**»).

Принцип дії цієї технології такий. При установці клієнтської частини ICQ абонент отримує 8-значний цифровий ідентифікаційний номер (UIN), що повідомляється друзям та знайомим, які вносять його до ***контакт-листів*** своїх клієнтів ICQ. Поки комп'ютер працює поза Internet, «Аська» нічим себе не проявляє, але як тільки абонент підключається до Мережі, вона прокидається і посилає на центральний сервер ICQ повідомлення про те, що абонент такий-то перейшов із стану Offline у стан Online. Сервер розішле це повідомлення всім зацікавленим користувачам, і через декілька секунд їх клієнти, якщо на даний час вони підключені до Internet, прореагують на цю звістку зміною статусу абонента у контакт-листі. Після цього можливий прямий діалог клієнтів у режимі чату. Тобто, склавши відповідним чином контакт-лист, ви завжди будете у курсі, хто з ваших кореспондентів, що розкидані по всьому світу,

доступний для негайного спілкування. А якщо потрібний абонент знаходиться у стані Offline, то йому можна залишити на сервері записку. Як тільки він включить свій комп'ютер, записка буде негайно надіслана.

4.6. Портали

Декілька сервісів, таких як пошта, новини, форуми, чати, каталоги сайтів, які об'єднані в рамках одного ключового сервісу (наприклад, пошуку) називаються порталом. Прикладом порталу може служити пошук Яндексу (www.yandex.ru). Інший приклад – Mail.Ru (www.mail.ru) – портал, де ключовим сервісом є пошта. Портали орієнтовані на дуже велику кількість (в ідеалі – на всіх) користувачів. Це дуже крупні сайти, які здатні витримувати сотні тисяч (іноді – мільйони) відвідувань на добу. Іноді, великі тематичні сайти з розгалуженою структурою, які містять форуми, чати і навіть поштовий сервіс, називають порталами. Проте, якщо велика частина посилань цього сайту спрямована на його внутрішні ресурси, то називати такий сайт "порталом" не зовсім коректно.

4.7. Передача мультимедіа

Комунікаційні послуги, про які ми говорили вище, так чи інакше мали справу з передачею тексту або файлу в режимі off-line. Для електронного листа або «миттєвого» повідомлення час, який витрачається на передачу повідомлення, є некритичним, затримка у декілька секунд нічого не вирішує. Абсолютно інша ситуація складається, якщо мова йде про мультимедійні мережні послуги, коли передача звуку та зображень відбувається у режимі реального часу. У цьому випадку повідомлення повинні передаватися у тому ж темпі, в якому вони створюються у відправника.

При роботі з мультимедіа-додатком критично важливим є поняття ***бітрейту (bitrate=bit + rate - швидкість)***, який показує, який об'єм двійкових даних передається за секунду при роботі цього додатку.

Для того щоб мережа підтримувала мультимедійну послугу, її корисна пропускна спроможність (яка набагато нижче за технічну швидкість каналів зв'язку через численні накладні витрати протоколів всіх рівнів) повинна бути не менше бітрейту додатку. Оскільки пропускна спроможність мережі є поки що дорогим та дефіцитним ресурсом, загальною проблемою мультимедіа-сервісу є зменшення бітрейта за рахунок стиснення (**компресії**) даних.

Компресія (та зворотна декомпресія) потоку проводиться функціональним блоком, так званим ***кодеком (codec від кодер/декодер – coder/decoder)***. Кодек може бути реалізований програмно або виконаний у вигляді спеціалізованого пристрою. Стислий до прийнятного бітрейту потік далі розбивається на пакети, які звичайним чином передаються по мережі передачі даних до одержувача, де відбувається його декомпресія та вихідне цифро-аналогове перетворення.

Розробка швидких та ефективних алгоритмів стиснення (компресії) даних є одним з важливих напрямів інформатики, який вимагає глибоких теоретичних досліджень.

Стиснення буває ***без втрат*** та ***з втратами (lossy compression).*** Стиснення без втрат грунтується на формальних методах економного кодування, головна ідея яких була запропонована ще Клодом Шенноном. Її сутність досить проста: у початковому потоці потрібно знайти послідовності символів, які часто повторюються, і перетворити їх в короткі кодові комбінації. Класичні алгоритми такого кодування (наприклад, коди Шеннона-Фено та Хаффмана) та їх математичне обґрунтування вивчаються в курсах теоретичної інформатики.

Стиснення із втратами вимагає глибокого проникнення у ті області знань, які досліджують сприйняття людиною відповідних повідомлень, тобто в психофізіологію слуху та зору. Завдання полягає у тому, щоб ціною незначних втрат суб'єктивно сприйманої якості звуку або зображення добитися істотного зниження обсягу даних, що передаються. Наприклад, фундаментальні теоретичні дослідження в області психоакустики дозволили встановити, що гучні звуки на деякій частоті роблять вухо на деякий час несприйнятливим до наближених частот, тому останні можна без істотної шкоди для сприйняття виключити із звукового сигналу. На цьому принципі розроблено багато алгоритмів аудіостиснення, зокрема, німецькі фахівці з Фраунгоферовського інституту (Fraunhofer Institute for Integrated Circuits) запропонували ефективний алгоритм **Audio MPEG Layer 3**, відомий під скороченням **MP3**. Він забезпечує 10-12 разове стиснення аудіоданих при гарній якості звучання музичних програм. Формат MP3 призвів переворот у світі музики, у цьому форматі продаються записи на компакт-дисках, транслюються радіопрограми, кодується звуковий супровід до відеосюжетів.

Ще складніші та витонченіші алгоритми стиснення винаходяться для передачі зображень. Роботи у цьому напрямі почалися достатньо давно. У 1988 році дві найвпливовіші міжнародні організації зі стандартизації – International Standard Organization (ISO) та International Telecommunication Union (ITU) – утворили об'єднану робочу групу експертів в області фотографії **JPEG** – Joint Photographic Experts Group, – яка розробила однойменний алгоритм стиснення з втратами нерухомих реалістичних зображень, що забезпечує компресію у 5-15 разів. На даний час цей формат є одним з найпопулярніших в Internet, у ньому закодовані численні фотоархіви.

Спеціально для розробки методів стиснення рухомих зображень була створена робоча група **MPEG** – Motion Picture Expert Group, результатом роботи якої стали декілька форматів стислих відеопотоків. Ця група використовувала принципи JPEG у частині представлення одного нерухомого кадру та доповнила їх безліччю нових ідей, що стосуються стиснення послідовності кадрів. Скажімо, навіщо кожного разу передавати зображення на задньому плані, якщо воно від кадру до кадру залишається незмінним, а дія відбувається тільки на передньому плані?

Формат **MPEG-1** був розроблений у 1992 р., він призначався для передачі зображень невисокої якості, такої, як у домашнього відеомагнітофона. Формат **MPEG-2** призначений для обробки відеозображення, якість якого можна порівняти з професійною телевізійною якістю. На сьогодні цю технологію використовує цифрове телебачення: сигнал, який стиснутий відповідно до стандарту MPEG-2, транслюється через супутники.

Найдосконалішим та універсальним є формат **MPEG-4**, перша версія якого була прийнята у 1999 році. Формат увібрав в себе все краще, що було створене за останні 10 років в області мультимедіа. Він не тільки дозволяє у сотні разів стискувати реальні відеопотоки, але, за задумом розробників, повинен надавати різноманітні інструменти та алгоритми для роботи з віртуальними об'єктами, за допомогою згенерованих засобів комп'ютерної графіки. Подібні об'єкти дозволяють значно скоротити обсяг даних, що передаються, оскільки для їх анімації буває достатньо передати всього декілька параметрів – все інше буде зроблено у декодері.

У частині генерування музики стандарт MPEG-4 також пропонує багато нового у порівнянні із сьогоднішнім MIDI: за допомогою спеціальних мов можна описати віртуальний оркестр, виконавця твору тощо. У цілому стандарт MPEG-4 є революційним, його деталі продовжують уточнюватися, а реалізація у програмних та апаратних кодеках різних виробників здійснена лише частково. Випереджаючи сучасний рівень техніки, він визначає шляхи розвитку мультимедіа на найближчі роки.

До сучасних мережних мультимедіа-послуг належать:

– комп'ютерна телефонія;

– потокове радіомовлення;

– відеоконференції та потокове відео.

Комп'ютерна телефонія (IP-телефонія). Як відомо, для передачі мови в принципі потрібна пропускна спроможність 64 кбіт/с, проте, використовуючи технології стиснення, можна зменшити бітрейт до 8-16 кбіт/с. Ці цифри роблять економічно доцільним передачу телефонних повідомлень не звичайними міжміськими та міжнародними телефонними каналами, а у потоці пакетів за IP-протоколом. Таким чином, якщо обладнати свій комп'ютер мікрофоном та навушниками, можна поспілкуватися з аналогічним абонентом в іншій країні за ціною, яку потрібно заплатити за користування каналом Internet без урахування дальності абонента. Проте при цьому вельми критичним є час затримки пакетів, який у звичайних умовах може виявитися неприпустимо великим (більше 0,1 с).

Як вид бізнесу у такому простому варіанті Internet-телефонія безперспективна: навряд чи знайдеться багато охочих телефонувати, сидячи біля комп'ютера з мікрофоном та навушниками; простіше скористатися міжміським телефоном. Комерційна послуга надається за іншою схемою. Абонент із звичайного телефону дзвонить в центр IP-телефонії свого міста, де знаходиться устаткування, що перетворює звуковий сигнал у потік пакетів, а

також шлюз у магістральну мережу, що гарантує прийнятний час затримки пакетів. Аналогічний центр влаштовується на іншому кінці магістралі, в місці призначення, де потік пакетів перетвориться в звуковий сигнал, що відправляється телефонною мережою загального користування абонентові. З урахуванням всіх витрат тарифи на міжміські та міжнародні дзвінки у провайдерів IP-телефонії виявляються на 40-50% нижче, ніж у традиційних операторів зв'язку.

Internet-радіомовлення. Близьким родичем IP-телефонії є Internet-радіомовлення. Принцип дії його такий самий: звуковий сигнал у джерела перетворюється у безперервний потік бітів, який стискується до прийнятного бітрейту та передається мережними транспортними засобами одержувачеві, де він за допомогою кодека та звукової карти знову перетворюється на звук. Для радіомовних компаній відкривається унікальна можливість, не збільшуючи потужності своїх передавачів, розповсюдити сигнал по всьому світу, а для користувачів, що мають вихід в Internet, з'являється можливість обрати будь-яку з сотень або тисяч Internet-радіостанцій.

Відеоконференції та потокове відео. Організація відеоконференцій між двома абонентами в Internet в принципі не відрізняється від комп'ютерної телефонії, тільки додатково до мікрофону потрібно встановити відеокамеру, а для кодування та декодування відеопотоку необхідно використовувати відповідні програмні або апаратні відеокодеки.

Під відеоконференцією зазвичай розуміють передачу картинки дуже невеликих розмірів із швидкістю зйомки 5-10 кадрів за секунду. При цьому, використовуючи сучасні методи стиснення даних, можна вкластися у 64-128 кбит/с.

Розвитком технології відеоконференцій є потокове відеомовлення. Для забезпечення цього сервісу розроблено низку високорівневих протоколів, зокрема, протокол передачі у реальному часі RTP – Real-time Transport Protocol, який замінює звичайний TCP і на відміну від нього підтримує багатоадресове розповсюдження інформації. Протоколи потокового відео дозволяють адаптивно налаштовувати бітрейт і дають можливість передавати зображення із самою різною якістю – від «напівживих» картинок, до повноцінних телепередач, які транслюються у Мережу відосерверами провідних телевізійних компаній.

4.8. Використання інформаційних ресурсів

Усі розглянуті високорівневі мережні послуги належали до категорії комунікаційних, таких, що забезпечують передачу текстових або мультимедійних повідомлень від абонента до абонента. Саме комунікаційні послуги зробили комп'ютерні мережі на ранніх етапах їх розвитку привабливими для мільйонів користувачів. Проте надалі все зростаючу роль почала грати вища форма інформаційного обслуговування – ***накопичення та спільне використання інформаційних ресурсів***. Елементи цього сервісу

містилися вже у групах новин та FTP-архівах, проте його дійсний розвиток авдбувся на початку 90-х років коли в Internet майже одночасно з'явилися дві нові технології – **Gopher** та **World Wide Web**, що засновані на абсолютно різних принципах. Ми стисло розглянемо історію та ідеологію Gopher, а розвитку Web, враховуючи її всесвітній характер, цілком присвятимо наступний параграф.

Слово «gopher» означає поширену в Північній Америці тварину гофера (гризуна), спорідненого з ховрахами. Цей невеликий симпатичний гризун є геральдичною твариною американського штату Міннесота. Крім того, воно за звучанням співпадає з жаргонним словом gofer (скорочення від go for), що перекладається як «хлопчик на побігеньках», «дорученець». Система Gopher була розроблена в Університеті штату Міннесота у 1989-1991 роках Полом Ліндером (Linder, Paul) та Марком Мак-Кахилом (McCahill, Mark). Створена спочатку для потреб університетського кампусу, ця перша система навігації в Internet почала швидко розповсюджуватися по інших університетах, до середини 90-х років у світі було більше 2000 Gopher-серверів, у сукупності створюючи інформаційний Gopher-простір (Gopher-Space).

Головною ідеєю системи було подання всієї інформації у вигляді ієрархічного дерева. Розробники Gopher вважали, що така форма дуже зрозуміла користувачам, оскільки вони щодня мають справу з ієрархічними каталогами бібліотек та ієрархічною структурою файлової системи. По суті Gopher є розподіленою по Gopher-Space файловою системою, кореневий каталог якої (Mother Gopher) до цих пір знаходиться в Університеті Міннесоти.

Коли Gopher-простір став достатньо великим, пошук потрібного документа за ієрархією меню перетворився на непросте завдання. Доводилося довго блукати по дереву каталогів, яке було складене людьми, уявлення яких про конкретну наочну область не співпадає з поглядами користувача. З цієї причини у кінці 1992 року для GopherSpace в Університеті штату Невада була розроблена інформаційно-пошукова система під дотепною назвою Veronica (Very Easy Rodent-Oriented Net-wide Index of Computerized Archives – дуже легкий всемережний індекс комп'ютеризованих архівів, орієнтований на гризунів). Ця система дозволяла знаходити потрібні документи за ключовими словами та немало сприяла зростанню популярності сервісу Gopher. Ще кращу можливість для пошуку інформації дала технологія World Wide Web, яка виникла майже одночасно з Gopher, але заснована на зовсім іншому принципі – принципі ***гіпертексту***. Якийсь час дві технології розвивалися паралельно одна одній, Gopher навіть випереджала Web, проте після створення у 1993 році графічного Internet-браузера розвиток сервісу WWW набув вибухового характеру та затьмарив усе інше.

4.9. Послуги у вигляді cookies

Для того щоб спростити вашу роботу у Мережі, виробники таких веб-браузерів, як Netscape Navigator та Internet Explorer, ввели спеціального типу

повідомлення, які дозволяють веб-сайту відстежувати деякі ваші дії при його відвідинах. Браузери зберігають цю інформацію на вашому комп'ютері (напевно, тому її і назвали **cookies – "домашнє печиво"**), аби полегшити відвідини даного веб-сайту надалі.

Ця інформація дійсно спрощує будь-які переміщення по Мережі, але **може стати і джерелом деяких проблем**. Коли ви звертаєтеся на сайт Internet-магазину, наприклад, з продажу компакт-дисків, він використовує cookies для того, щоб відокремити ваші замовлення від замовлень інших клієнтів. Припустимо, що ви користуєтеся кредитною карткою для покупок в Internet-магазині, і цей Internet-магазин запам'ятав номер вашої кредитної картки. Припустимо також, що цим ви займаєтеся на роботі, і після вас за той самий комп'ютер сідає хтось інший та потрапляє у той же Internet-магазин. Тоді, цілком можливо, йому вдасться зробити покупки за вашою кредитною карткою.

Можливо, cookies і роблять життя у Мережі зручнішим, тут вже кожен вирішує сам. Просто слід знати, що завжди є можливість зробити певні дії для забезпечення секретності ваших переміщень Мережею, якщо ви відчуваєте у цьому необхідність.

Файли, які містять cookies, можна виявити у спеціальній папці – теці Windows\Cookies\, зазвичай вони мають розширення *.txt. Можна вилучити їх, але браузер через деякий час створить нові файли. Сучасні браузери здатні повідомити про cookies та запитати, чи необхідно їх вживати, коли будь-який сервер пропонує це зробити. Файли з cookies не можуть прочитати дані з жорсткого диска вашого комп'ютера. Вони лише збирають інформацію, яку їм надає браузер.

Варто також відзначити, що Internet Explorer окрім "випікання" cookies створює ще й файли з історією ваших подорожей по Мережі. Ці файли можна виявити у теці Windows\History\History.IE7, та у теці Windows\Temporary Internet Files.

5. Web-революція

Розповсюдження Web-технологій в Internet у другій половині 90-х років минулого століття було настільки стрімким, що цей феномен не без підстав називають Web-революцією. Справа дійшла до того, що багато користувачів, використуючи надзвичайно прості та зручні Web-сервіси, і не підозрюють про те, що в Мережі було і є щось ще.

Для сторонніх спостерігачів Web-революція відбулася раптово, проте насправді вона готувалася протягом десятиліть. Основні її рушійні сили – гіпертекст, клієнт-серверні технології та мережні комунікації – до деякого моменту часу розвивалися незалежно один від одного. Об'єднавшись разом, вони породили концепцію глобальної інформаційної системи, яка не тільки визначила обличчя сьогоднішнього Internet, але, крім того, дала життя ще низці інших сучасних інформаційних технологій. Оскільки історію мережних комунікацій ми вже розглянули, звернемося до витоків гіпертексту.

5.1. Ідея гіпертекстових інформаційних систем

Вперше слово «web» у словнику інформатики з'явилося, мабуть, у працях видатного американського вченого та адміністратора Ванневара Буша (Boosh, Vannevar; 1890-1974). Це прізвище вже не раз зустрічалося нам в історії обчислювальної техніки. У 30-і роки, будучи деканом у Массачусетському технологічному інституті, він сконструював найбільшу електромеханічну аналогову машину – диференціальний аналізатор. Під його керівництвом починав свої дослідження легендарний Клод Шеннон. У 1945 році Буш опублікував у далекому від техніки журналі The Atlantic Monthly статтю «As we may think – Як ми можемо думати». Міркуючи про форми подання інформації, він відзначав, що лінійна, послідовна схема викладу матеріалу, яка прийнята у книжках, суперечить асоціативному механізму мислення людини.

Для реалізації своєї ідеї Буш запропонував побудувати фото-електромеханічну (електронних комп'ютерів на той час ще не було) машину «Memex», для зберігання та читання великих масивів перезнятих на мікрофільми документів. Успіхи мікрофотографії того часу були такі, що об'єму сховища вистачило б на сто років за умови, що користувач вводив би щодня по 5000 сторінок. Машина повинна була мати вигляд письмового столу з клавіатурою та екраном, на який передбачалося проектувати зображення документів, але найоригінальнішою її частиною був механізм, що дозволяв швидко переходити від одного документа до іншого за перехресними (гіпертекстовими) посиланнями.

Хоча проект Memex так і не був здійснений, він надихнув послідовників – Теда Нельсона, Дугласа Енгельбарта, Рея Оззі, Била Аткинсона та ін. – на створення ***гіпертекстових інформаційних систем***, при цьому всі вони визнавали пріоритет Ванневара Буша. У 1995 р. в Массачусетському

технологічному інституті була проведена конференція, присвячена Бушу та приурочена до п'ятидесятиріччя винаходу гіпертексту.

5.2. Універсальне сховище знань

Самі терміни *гіпертекст (hypertext)* та *гіпермедіа (hypermedia)* були вперше запропоновані Тедом Нельсоном (Nelson, Theodor Holm; нар. 1937) у 1965 р. Ця обдарована та вельми неординарна людина захоплювалася літературою, театром, музикою та філософією, виступала на професійній сцені, працювала кореспондентом The New York Times і навіть видавала власний журнал. Закінчивши комп'ютерні курси, він захопився ідеєю створення всесвітнього сховища знань, яке могло б зберегти у своїй бездонній пам'яті будь-яку написану або вимовлену думку, яка пов'язана несчиленними перехресними посиланнями з раніше занесеними поняттями або висловами. Тоді потенційний читач міг би прослідкувати будь-яку ідею ретроспективно, щоб з'ясувати її коріння. Для свого явно нездійсненного за технологічними мірками XX століття проекту Нельсон обрав екзотичне ім'я Xanadu (Ксанаду). Xanadu – реальне містечко в Монголії, де збереглися розвалини палацу Гнізда Хана – палацу літературної пам'яті та творчої свободи, де ніщо не забувається.

Не дивлячись на неможливість повномасштабного втілення проекту, окремі його частини були доведені до практичної реалізації. З його легкої руки слово «**гіпертекст**» увійшло у 70-х та 80-х роках у моду, а головна ідея Xanadu – універсальне сховище знань та свобода доступу до них – почала реалізовуватися у цілій низці працюючих документальних інформаційних систем. На відміну від СКБД такі системи зберігають не жорстко структуровані дані у вигляді таблиць, а довільні текстові або мультімедійні документи, доступ до яких здійснюється за системою посилань. Іноді до програм цього класу застосовується термін ***система управління знаннями).***

5.3. Реалізації документальних гіпертекстових систем

Система NLS. Першу працюючу гіпертекстову систему створив у 1965-68 р. Дуглас Енгельбарт. Ми вже говорили про нього як про винахідника миші, вікон та інтерактивного текстового редактора. Проте це були лише частини великомасштабного проекту під назвою NLS (oNLine System), що реалізовується під його керівництвом спочатку у Стенфордському центрі з посилення можливостей людини (Augumentation Reseach Center – ARC), а потім у Xerox PARC. Система NLS призначалася для організації колективної роботи групи людей, що пов'язані спільними інтересами. Основним об'єктом, що зберігається в системі, був «конспект» – деякий текстовий документ, створений учасником групи. Конспекти зв'язувалися у ланцюжки обговорень і посилалися один до одного гіпертекстовими посиланнями. Таким чином, проект NLS, який був реалізований ще не на мережі, а на одній великій ЕОМ, передбачив майбутні групи новин, електронні дошки оголошень та форуми.

Система NLS, як і інші революційні проекти PARC, сама по собі не мала комерційного успіху, проте її спадкоємці – HyperCard, Windows Help, Lotus Notes та ін., не говорячи вже про Web, стали поширеними бізнес-додатками.

HyperCard та **Windows Help.** Однією з перших документальних гіпертекстових систем, що були створені для персональних комп'ютерів була система HyperCard, яка була розроблена у 1987 році співробітником фірми Apple Вільямом Аткінсоном (Atkinson, William) для Макінтошів. Система має розвинені візуальні засоби, вона дозволяє створювати та редагувати «стопки карток» з різними текстами та зображеннями. Картки пов'язуються одна з одною гіпертекстовими посиланнями і можуть бути видимими у довільному порядку.

Близьку за духом гіпертекстову документальну базу застосувала фірма Microsoft як стандартну довідкову систему ОС Windows. Увійшовши до будь-якої теми з основного змісту, можна продовжити подорож по гіперпосиланнях, уточнюючи та конкретизуючи отриману інформацію.

Lotus Notes. Дві попередні системи належали до розряду персональних, розрахованих на одного користувача. На відміну від них, Lotus Notes призначена для корпоративного використання. Тому за своєю сутністю вона ближче до класичної NLS. Історія цієї дуже цікавої та популярної в усьому світі системи почалася у 70-х роках, коли в університеті Іллінойса (м. Чікаго) була розроблена система PLATO Group Notes для мейнфреймів.

Система дозволяла користувачам створювати нотатки (notes), аналогічні конспектам у NLS, та поміщати їх у загальні бази даних, а також писати відповіді та коментарі до нотаток. Система користувалася такою популярністю, що після закінчення ери великих ЕОМ почали розроблятися **PLATO**-подібні системи для міні- та персональних комп'ютерів.

Один з керівників розробки PLATO Рей Оззі (Ozzie, Ray) вирішив реалізувати таку систему для IBM PC, проте довго не міг знайти інвестора для фінансування розробки. Розуміння та підтримка прийшли з боку керівника Lotus Development Мітчела Кепора. Ця фірма здобула світову популярність завдяки класичній системі електронних таблиць Lotus 1-2-3, яка розглядалась раніше. Зіткнувшись із запеклою конкуренцією з боку аналогічного пакету Microsoft Excel, фірма почала втрачати ринок і почала замислюватися про новий стратегічний продукт. Познайомившись з проектом Рея Оззі, Кепор надихнувся його ідеями і вирішив зробити на нього ставку. У 1984 році на гроші Lotus була створена фірма Iris Associates Inc., яка приступила до реалізації перспективної системи, проте пройшло цілих п'ять років, перш ніж вийшла перша комерційна версія Lotus Notes.

В основі пакету лежить концепція ***документо-орієнтірованої бази даних***, яка радикально відрізняється від реляційної. Завантаживши систему, користувач бачить робочий стіл, на якому лежать папки з базами даних – повними аналогами підшивки документів. Документи можуть містити будь-яку інформацію – тексти, числа, малюнки, мультимедіа. Їх можна за заданими користувачами правилами створювати, перетворювати, пересилати один

одному по електронній пошті, складати у бази, забезпечувати гіпертекстовими посиланнями тощо. Для маніпуляції з документами у системі є декілька вбудованих мов програмування. За такими комплексними продуктами закріпилася назва ***системи групової роботи (groupware***), Lotus Notes став першим та недосяжним для конкурентів представником продуктів даного класу.

Подальші версії системи були спрямовані на розвиток її функціональності, масштабованості та платформної незалежності. Сервери та клієнти Lotus Notes одержали можливість працювати на всіх типах комп'ютерів та операційних системах – від мейнфреймів S/390 до Макінтошів та IBM PC.

Побачивши успіх Lotus, корпорація Microsoft вийшла на ринок groupware з системою **MS Exchange**. Третім конкурентом на цьому ринку була фірма Novell. Оскільки її мережна ОС NetWare почала здавати позиції, фірма доповнює її функціями групової роботи і сподівається таким чином повернути втрачену частку ринку. Нова версія ОС називається GroupWise і розраховує на успіх у конкуренції з IBM та Microsoft. Важко сказати, як розвиватимуться справи далі, але поки Lotus утримує лідерство.

5.4. Народження Web

До кінця 80-х років гіпертекстові документальні технології, які були реалізовані на одному комп'ютері (HyperCard) або у клієнт-серверному варіанті (Lotus Notes), були вже випробувані на практиці і швидко завойовували популярність. З іншого боку, росла аудиторія Internet, кількість підключених комп'ютерів вимірювалося вже сотнями тисяч. Проте користувачі Мережі, за винятком тих небагатьох, хто користувався послугами Telnet, працювали у режимі off-line: основним видом сервісу у цей час була електронна пошта, невелику частку трафіку займав обмін новинами та файлами.

Історична місія об'єднати комунікаційні технології Internet у режимі реального часу з чудовими можливостями документальних гіпертекстових систем і тим самим перетворити його на глобальну базу знань, яка була б доступна усьому людству, припала на долю молодого англійця **Тіма Бернерс-Лі** (Berners-Lee, Timothy; нар. 1955).

Закінчивши у 1976 році з відзнакою Королівський коледж в Оксфорді, він якийсь час працював у комп'ютерних фірмах, а у 1980 році уклав контракт на 6 місяців на роботу в якості консультанта з програмного забезпечення в лабораторії елементарних частинок Європейського центру ядерних досліджень (CERN) у Женеві (Швейцарія). У цей час він написав для власних потреб невелику програму «Enquire-Within-Upon-Everything», скорочено «Enquire» – «Довідник», яка була простою гіпертекстовою базою документів.

Коли контракт закінчився, Тім повернувся до Англії і декілька років займався різними проектами, зокрема пов'язаними з Internet. Проте у 1984 році повернувся в CERN і отримав завдання зайнятися надзвичайно актуальною для CERN проблемою зберігання та пошуку матеріалів з наукових досліджень у

режимі реального часу, причому доступ до документів повинні були мати вчені з різних країн, які співробітничають з цією міжнародною організацією. Ось тут-то і стала у нагоді програма «**Enquire**». На новому етапі розвитку вона вже повинна була не тільки підтримувати довільні гіпертекстові посилання, полегшуючи пошук у документальній базі, але і стати багатокористувацькою, незалежною системою, що працює через Internet. У **березні (13) 1989** року Бернерс-Лі представив керівництву проект під назвою «World-Wide Web: An Information Infrastructure for High-Energy Physics» – «Всесвітня павутина: Інформаційна інфраструктура для фізики високих енергій».

Основними складовими проекту були.

1. ***Мова розмітки гіпертекстових сторінок – HyperText Markup Language (HTML),*** якою описується зміст документів, що зберігаються. Мова дозволяє описувати структуру документу, виділяти заголовки, абзаци, розміщувати таблиці, ілюстрації і таке інше. Важливим елементом мови є гіперпосилання, які можна розставляти у довільних місцях документу і які посилаються на інші документи, що зберігаються не тільки у даному комп'ютері, але і в будь-якому іншому місці Internet. Так утворюється всесвітня павутина глобальних посилань, що і визначило назву проекту.

2. ***Протокол передачі гіпертексту – HyperText Transfer Protocol (HTTP***) є високорівневим прикладним Internet-протоколом, який надбудований над стеком TCP/IP.

3. ***Універсальний ідентифікатор документу – Universal Document Identifier (UDI),*** являє собою унікальне ім'я (адресу) і за якою даний документ можна знайти у глобальній Мережі. Якийсь час потому, коли поняття «документ» було замінене на більш загальне «ресурс», цей термін був перейменований і отримав назву ***універсальний локатор ресурсів – Universal Resource Locator (URL).***

Формат URL, який був запропонований Бернерс-Лі, сьогодні відомий будь-якому користувачеві Internet. Наприклад, сторінка, яка присвячена самому Бернерс-Лі у Массачусетському технологічному інституті, має адресу http://web.mit.edu/invent/www/inventorsA-H/bernerslee.html. Цей формат включає вказівку на протокол, за яким доступний даний ресурс (http://), доменне ім'я хосту, у даному випадку це Web-сервер MTI (web.mit.edu), шлях до потрібної HTML-сторінки у файловій системі сервера (/invent/www/inventorsA-H/) і, нарешті, ім'я файлу, що містить сторінку (berners-lee.html). На перший погляд цей формат здається складним, але до нього дуже легко звикнути, мільйони відвідувачів Web користуються ним без зусиль. Але згодом на симпозіумі, який був присвячений технологиям майбутнього у Вашингтоні, вчений зізнався: «Насправді, якщо подумати, то виявляється, що подвійний слеш не потрібний. Якби я знову створював це, я б обійшовся без нього». Сучасні браузери самостійно підставляють пропущений символ на адресу запитуваного ресурсу, якщо ви його пропустили. Проте раніше пропуск незручного для набору слешу призводив до повідомлення про

синтаксичну помилку. А тепер самі спробуйте ввести адресу у себе у браузері без подвійного слешу.

Якщо Ви запитаєте у свого знайомого, що протягом довгого часу користувався Internet, яка його адреса, то він напевно продиктує вам у відповідь щось типу http://www.myhomepage.domain.ua замість звичної фрази «місто, вулиця, будинок, квартира». Тому, хто має намір навчитися створювати власні проекти для Internet, треба спочатку сподобитися цьому абстрактному користувачеві глобальної Мережі та спробувати зрозуміти, яким чином здійснюється адресація в Internet.

Як вже згадувалось раніше, передача інформації в Internet здійснюється із застосуванням протоколу IP. Згідно із загальноприйнятим стандартом, кожен комп'ютер, який працює в Internet, використовує цей протокол і має унікальну IP-адресу, яка складається з чотирьох 8-розрядних умовних регістрів, що належать діапазону від 0 до 255, які розділені крапкою, наприклад, 193.75.101.12. Головний регістр, що розташований у записі IP-адреси і вказує на клас мережі, до якої підключений комп'ютер. Регістр, розташований праворуч, (це адреси хосту, тобто номер самого комп'ютера у локальній мережі. Усе, що розташоване між «праворуч» та «ліворуч», описують підмережі нижчого порядку. Спробуємо розібрати цей механізм запису адреси на простому прикладі. Припустимо, що нам необхідно направити пакет інформації за наведеною вище адресою (193.75.101.12. Спочатку пакет передається в 193-у мережу, далі у 75-у підмережу, куди і пересилаються відправлені нами дані. З 75-ої підмережі пакет інформації буде відісланий у 101-у підмережу, а її сервер, у свою чергу, відшукає у своєму сегменті мережі комп'ютер з номером 12 і передасть йому відправлену нами інформацію. Описаний вище процес визначення маршруту інформаційного пакету називається **маршрутизацією**, а здійснюють її спеціальні сервери, які носять назву, як вже було згадано раніше (маршрутизаторів, або роутерів.

Все це добре, але мало хто стане запам’ятовувати усі ці величезні набори цифр. Тому фахівці, які створювали Internet, вирішили спростити користувачам Всесвітньої Мережі задачу, придумавши корисну та зручну річ, яка називається *DNS* (Domain Name System), та перекладається як «доменна система імен». *DNS* дозволяє значно полегшити користувачеві процес роботи в Internet та запам’ятовувати імена хостів, з якими його комп’ютеру доводиться спілкуватися. Ось тільки комп’ютеру-то якраз набагато легше працювати з набором цифр, що він, власне, і робить, отримуючи за введеним оператором доменним іменем вже знайому нам IP-адресу машини. Яким чином здійснюється цей процес?

Основний механізм функціонування DNS зовсім нескладний. Навіщо винаходити велосипед, коли можна застосувати вже двісті років відомий людству алгоритм — принцип, який використовувся людьми при обміні кореспонденцією через звичайну пошту. Інженери, створюючи DNS, саме так і

поступили, взявши за основу технологію пересилки традиційних поштових відправлень.

Шлях нашого листа до кінцевого одержувача можна описати наступним рядком: «Квартира 1. Будинок № 2. Вулиця Громова. Черкаси. Україна». Скажіть, а чим гірше рядок віртуальної адреси: myhost.mysubdomain.mydomain.ua? Та нічим – це практично одне й те ж саме. Відповідно, рядок виклику розміщеного за цією адресою веб-сайту виглядатиме як http://www.myhost.mysubdomain.mydomain.ua, де http позначає протокол передачі гіпертексту (Hyper Text Transfer Protocol), а www (абревіатура англійського «World Wide Web») вказує на те, що запрошувані користувачем сервера документи є саме веб-сторінками. Таке позначення назвали **URL** (Uniform Resource Locator), що можна просто перекласти, як «**універсальний визначальник місцезнаходження ресурсу**». У загальному вигляді URL прийнято записувати наступним чином: protocol://host:port/path/filename, де port – номер порту сервера, через який передається документ, path – шлях до потрібного файлу у файловій структурі сервера, а filename – ім'я файлу, що запитується. Наприклад, якщо для пошуку будь-якої веб-сторінки ми використовуємо URL вигляду http://www.server.ua:8080/folder/file.html, то це означає, що нам необхідний файл file.html, який розташований у folder на сервері www.server.ua, який ми повинні отримати через серверний порт 8080 за протоколом HTTP.

Таким чином, ми бачимо, що адреса того чи іншого ресурсу Всесвітньої Мережі, записана у стандарті DNS, розбивається на декілька складових, що відокремлені одна від одної крапкою. Ці елементи носять назву «***домени*».** Домен – це деякий логічний рівень Internet, тобто група мережних ресурсів, що має власне ім'я і керується своєю мережною станцією.

Але ми відповіли лише на перше поставлене питання – як адресувати окремі машини в Internet. Тепер давайте подивимося, як нам бути з програмним забезпеченням, яке використовує Мережа для обміну даними.

До цього ми розцінювали машини, що підключені до Internet, як деякі неподільні сутності. Так воно взагалі і є (але з деякими обмовками) з точки зору протоколу IP. Але TCP використовує у своїй роботі дещо інші поняття. Для нього окремою сутністю є ***процес*** – програма, яка запущена десь на комп'ютері в Internet разом з необхідними для неї ресурсами. Саме між процесами, а не між машинами, і здійснюється обмін даними у термінах протоколу TCP. Тепер розглянемо, як саме TCP визначає той процес, якому потрібно доставити дані.

Нехай на деякій системі виконується програма (назвемо її Клієнт), яка хоче через Internet з'єднатися з якоюсь іншою програмою (Сервером) на іншій машині у Мережі. Для цього повинна виконуватися низка умов, а саме:

– програми мають «домовитися» про те, як вони одна одну ідентифікуватимуть;

– програма Сервер повинна знаходитися у *режимі очікування,* коли до неї зараз хтось підключиться.

Давайте зупинимося на першому пункті детальніше. Термін «домовитися» тут не зовсім доречний. Насправді програма Сервер, як тільки вона запускається, говорить драйверу TCP, що вона збирається використовувати для обміну даними з Клієнтами деякий ідентифікатор, або ***порт***, ціле число у діапазоні від 0 до 65 535 (саме такі числа можуть зберігатися в елементі пам'яті розміром у 2 байти). TCP реєструє це у своїх внутрішніх таблицях лише у тому випадку, якщо деяка інша програма вже не "зайняла" потрібний нам порт (в останньому випадку відбувається помилка). Потім Сервер переходить у режим очікування надходження запитів, що приходять на цей порт. Це означає, що будь-який Клієнт, який збирається вступити у «діалог» з Сервером, повинен знати номер свого порту. Інакше TCP- з'єднання неможливе: куди передавати дані, якщо не знаєш, до кого підключитися?

Тепер подивимося, які дії робить Клієнт. Він, як ми домовилися, знає таке:

– IP-адресу машини, на якій запущений Сервер;

– номер порту, який використовує Сервер.

Як відомо, цієї інформації цілком достатньо. Тому Клієнт посилає драйверу TCP команду на з'єднання з машиною, яка розташована за заданою IP-адресою із вказівкою потрібного номера порту. Оскільки Сервер "на тому кінці" готовий до цього, він відгукується, і з'єднання встановлюється.

Як тільки обмін "вітальними" повідомленнями закінчений (його ще називають "потрійним рукостисканням", тому що у цілому посилається 3 таких повідомлення), між Клієнтом та Сервером встановлюється *логічний канал зв'язку* (це нагадує випадок файлу, відкритого на читання та запис одночасно). Іншими словами, Клієнт може передати дані Серверу, записавши їх за допомогою системної функції у канал, а Сервер – прийняти їх, прочитавши з каналу.

4. ***Загальний інтерфейс шлюзу – Common Gateway Interface (CGI***), що є системою угод, які дозволяють запускати прикладні програми на Web-сервері та передавати їм дані з HTML-сторінок.

Сервіс WWW організований за клієнт-серверною технологією. Бази HTML-документів зберігаються на ***Web-серверах (Web-server)***, а у користувачів встановлюється клієнтська програма, що має назву ***Web-браузера (browser)***, яка відповідно до протоколу HTTP посилає запити на відповідний сервер, отримує сторінки і відображає їх на екрані.

Перший сервер і браузер (він же редактор HTML-сторінок) були реалізовані самим Бернерс-Лі протягом декількох місяців на комп'ютері NEXT в операційному середовищі NEXTEP. До появи Windows це була найсучасніша з погляду зручності розробки додатків об'єктно-орієнтована платформа, проте, сумістити текст та графіку в одному вікні не вдалося, браузер був чисто текстовим.

У грудні 1990 року перша інформаційна система на основі WWW запрацювала у локальній мережі CERN, а до літа 1991 року вона була відкрита в Internet. Хоча спочатку система замислювалась як вузько галузева,

розрахована на обслуговування фахівців в області ядерної фізики, її потенційні можливості виявились значно ширші. Після декількох демонстрацій на різних конференціях ідея Всесвітньої павутини привернула вчених у самих різних областях. Протягом 1991 року текстові Web-браузери були реалізовані на різних апаратних платформах – VAX, Sun, RS/6000 та ін. До кінця 1992 року у світі налічувалося вже близько 50 діючих HTTP-серверів. Всесвітня павутина з'явилася на світ.

Подальший розвиток Web-технологій відбувався в основному по іншу сторону океану, тому не дивно, що їх засновник Бернерс-Лі перебрався у 1994 році у США. Тут виявилася ще одна риса характеру цієї видатної людини. Відхиливши безліч привабливих пропозицій, які могли б зробити його мільярдером, Бернерс-Лі виступив рішучим супротивником комерціалізації Web. Кар'єрі бізнесмена він віддав перевагу роботі на кафедрі у комп'ютерній Мецці – Массачусетському технологічному інституті.

5.5. Перші версії браузерів

Не дивлячись на великі потенційні можливості, у перші два роки Всесвітня павутина розвивалася досить повільно, на початку 1993 року обсяг HTTP-трафіку у мережі NSFnet складав всього 0,1%. Значно більшою популярністю користувався Gopher, та навіть розвивався він швидше: у 1993 році об'єм цього сервісу виріс на 997%, тоді як WWW – тільки на 341%. В значній мірі це було пов'язано з тим, що всі створені у цей час Web-браузери були чисто текстовими, розрахованими на відтворення наукових та технічних документів, до того ж складними у засвоєнні і не дуже зручними для користування. Для професійних вчених це було невеликою перешкодою, проте рядовому користувачеві одноманітність текстових сторінок та безліч незрозумілих функцій швидко набридає.

Перетворити Мережу на універсальну інформаційну систему, цікаву та доступну для всіх верств населення допомогли графічні Internet-браузери, початок яким поклали роботи, що були виконані у Національному центрі суперкомп'ютерних додатків (National Center for Supercomputing Applications – NCSA) університету Іллінойса групою студентів та співробітників під керівництвом Марка Андріссена (Andreessen, Marc; нар. 1971).

У листопаді 1992 року 21-річний Андріссен вчився на останньому курсі університету і підробляв у NCSA. Цей центр був одним з основних вузлів NSFnet і мав доступ до всіх мережних ресурсів Internet – E-mail, FTP, Telnet, NNTP, Gopher, WWW. Пізнавши на своєму досвіді, наскільки незручно користуватися розрізненими клієнтами, Андріссен разом зі своїм другом Еріком Біна (Bina, Eric) замислив створити універсальну клієнтську програму, яка б об'єднала всі види сервісу і до того ж мала б простий та дружній графічний інтерфейс. При цьому, переходи за гіперпосиланням Web повинні були відбуватися одним клацанням миші. За шість тижнів була створена перша пробна версія браузера для Unix, названого **Mosaic**, вона налічувала

9000 рядків початкового коду. До осені 1993 року колегами Андріссена були здійснені реалізації Mosaic для двох інших популярних платформ – Macintosh та Windows. Оскільки NCSA була некомерційною організацією, браузери завантажувалися через Internet безкоштовно.

Успіх Mosaic був миттєвим та приголомшливим. Цей браузер узяв Internet штурмом, протягом року були поширені мільйони копій. Саме завдяки ньому Всесвітня павутина отримала щонайпотужніший імпульс розвитку, а Web-сервіс став у ній домінуючим і практично витіснив Gopher. Вже до кінця 1993 року трафік Web у мережі NSFnet виріс у 10 разів, а число HTTP-серверів перевищило 600.

У грудні 1993 року Андріссен закінчив університет та перебрався у Кремнієву долину, де влаштувався на роботу у невелику софтверну компанію, яка займалася за замовленням ARPA криптографією. Про продовження робіт над браузером мова не йшла: для організації власного підприємства потрібні були гроші та досвід підприємницької діяльності, яких у вчорашнього випускника університету, звичайно ж не було.

Вирішальну роль у подальшій долі Андріссена зіграла зустріч з Джимом Кларком (Clark, James), засновником та головою процвітаючої компанії Silicon Graphics Incorporated (SGI). Кларк ніколи раніше не чув ні про Андріссена, ні про Mosaic, але інтуїція досвідченого бізнесмена підказувала йому, що настав час вкладати гроші у мережні технології. Дізнавшись від знайомих про юного розробника графічного браузера, Кларк послав йому листа електронною поштою та призначив побачення. Переговори пройшли успішно, і до квітня 1994 року на гроші Кларка була створена компанія «Mosaic Communications Corp», яку через протест з боку NCSA, яка претендувала на фірмову марку «Mosaic», довелося незабаром перейменувати у «Netscape Communications Corp.», скорочено «**Netscape**».

Талановитий організатор, Джим Кларк зробив усе, щоб робота пішла швидко. Він умовив перейти у нову фірму більшість колег Андріссена з NCSA та запросив на посаду головного керівника найдосвідченішого Джеймса Барксдейла (Barksdale; James) з AT&T.

Перед розробниками було поставлено амбітне завдання: на основі Mosaic створити новий комерційний багатоплатформенний браузер, який витіснить з ринку свого батька. Робота над проектом йшла майже цілодобово і вже у жовтні була готова перша версія браузера Mosaic Communicator, який був перейменований згодом у Netscape Navigator.

Успіх підприємства Андріссена та Кларка перевершив усі очікування. Вже до літа 1995 року браузери Netscape були встановлені на мільйонах комп'ютерів, компанія почала розвиватися з рекордною швидкістю, об'єм продажу у 1995 році склав 80 мільйонів, у 1996 р. – 318 мільйонів, а в 1997 р. – 534 мільйони доларів. Кількість працівників у 1997 році перевищило 1800 осіб, компанія обзавелася шикарним офісом у містечку Маунтін-Вью у центрі Кремнієвої долини, а Марк Андріссен у віці 25 років став мультимільйонером.

Уявімо собі, що Internet – це величезне місто (мегаполіс), де багато-багато різних будинків та магазинів, великих і маленьких, прикрашених красивими вивісками та ілюмінаціями. Будинки, магазини – це сайти, але про них пізніше. Для того щоб правильно побачити всі деталі та красу цього мегаполісу обов'язково необхідно мати спеціальні окуляри. Окуляри – це браузер. Якщо ви знімете окуляри, то замість красивих будинків побачите лише дивні букви та значки. Саме це є вмістом Internet-сторінки (HTML-сторінки).

5.6. Веб-сайт

Завдяки графічним браузерам, що незмірно спрощували роботу користувача, Всесвітня павутина, а разом з нею і весь Internet стали розростатися з експоненціальною швидкістю. Своїми сайтами обзавелися не лише ВНЗ, комерційні організації, але й урядові установи всіх країн. Термін "***сайт***", утворений від англ. **site**, – "місце, ділянка", означає у WWW цілісний інформаційний ресурс, який складається з пов'язаних HTML-сторінок, – у комп'ютерній мережі об'єднаний під однією адресою (доменним ім'ям або IP-адресою) сукупність електронних документів (файлів) приватної особи або організації.

Апаратні сервери для зберігання веб-сайтів називаються *веб-серверами*. Сама послуга зберігання називається *веб-хостингом*. Раніше кожен сайт зберігався на своєму власному сервері, але із зростанням Internet та технологічним покращенням серверів на одному комп'ютері стало можливе розміщення безлічі сайтів (*віртуальний хостинг*). Зараз сервери для зберігання тільки одного сайту називаються *виділеними* (англ. – dedicated).

Один і той самий сайт може бути доступний за різними адресами і зберігатися на різних серверах. Копія оригінального сайту у такому разі називається *дзеркалом*. Існує так само поняття «оффлайновая версія сайту» – це копія сайту, яка може бути переглянута на будь-якому комп'ютері без підключення до комп'ютерної мережі та використання серверного програмного забезпечення.

Перший у світі сайт info.cern.ch з'явився у 1990 році. Його творець, той самий Тім Бернерс-Лі, опублікував на ньому опис нової технології WorldWideWeb, що була заснована на протоколі передачі даних HTTP, системі адресації URL та мові гіпертекстової розмітки HTML. Також на сайті були описані принципи установки та роботи серверів і браузерів. Сайт став і першим у світі Internet-каталогом, оскільки пізніше Тім Бернерс-Лі розмістив на ньому список посилань на інші сайти.

Усі інструменти, які були необхідні для роботи першого сайту, Бернерс-Лі підготував ще раніше – у кінці 1990 року з'явилися перший гіпертекстовий браузер WorldWideWeb з функціоналом веб-редактора, перший сервер на базі NeXTcube та перші веб-сторінки.

"Батько" Вебу вважав, що гіпертекст може бути основою для мереж обміну даними, і йому вдалося втілити свою ідею у життя. Ще у 1980 році Тім Бернерс-Ли створив гіпертекстове програмне забезпечення Enquire, яке використовувало для збереження даних випадкові асоціації. Потім, працюючи у Європейському центрі ядерних досліджень у Женеві (CERN), він запропонував колегам публікувати гіпертекстові документи, які пов'язані між собою гіперпосиланнями. Бернерс-Ли продемонстрував можливість гіпертекстового доступу до внутрішніх пошукових систем та документів, а також новинних ресурсів Internet.

Тім Бернерс-Лі є "батьком" засадничих технологій вебу – HTTP, URI/URL та HTML, хоча їх теоретичні основи були закладені ще раніше. У 1940-х роках Ванневар Буш висунув ідеї розширення пам'яті людини за допомогою технічних пристроїв, а також індексації накопиченої людством інформації для її швидкого пошуку. Теодор Нельсон та Даг Энгельбарт запропонували технологію гіпертексту – тексту, що "розгалужується", надає читачеві різні варіанти читання. Не закінчена гіпертекстова система Нельсона, була призначена для зберігання та пошуку тексту, у який введені взаємозв'язки та "вікна". Нельсон мріяв зв'язати перехресними посиланнями усі створені людством тексти.

5.7. Війна браузерів

Як не дивно, Білл Гейтс, що володіє надприродним чуттям на нові інформаційні технології, що можуть принести прибуток, прогавив початок Web-революції. Гігантська Microsoft, що отримала у 1995 році 5,9 млрд. доларів доходу, була дуже зайнята своєю ОС Windows-95 і не звертала уваги на Internet та Web. Лише на початку 1996 року Гейтс остаточно усвідомив той факт, що Web – це технологія, яка здатна змінити бізнес, тоді як Windows-95 усього лише нова операційна система. Але час був втрачений, ринок мережного програмного забезпечення вже сформувався, близько 20 млн. чоловік використовували Netscape Navigator та інші продукти, які були призначені для роботи з Web.

І тоді Гейтс зробив рішучий крок. Він гучно оголосив Internet ареною стратегічних інтересів своєї фірми і кинувся у бій з конкурентами, поставивши завдання перерозподілу ринку на користь Microsoft.

Наступні декілька років численні користувачі Мережі спостерігали драматичну «війну браузерів». Початкова розстановка сил не обіцяла Microsoft нічого доброго. Випущений у серпні 1995 року **Internet Explorer (IE) 1.0** був поспішно переробленим Mosaic і не йшов ні в яке порівняння з достатньо зрілими продуктами Netscape. На початок 1996 року він займав всього 2% ринку, тоді як Netscape Navigator (NN) – 71% (частка, що залишилася, доводилася на Mosaic та інші, зараз вже забуті браузери).

Перший удар зробила Microsoft: не зумівши поки наздогнати Netscape за якістю, вона свій наступний браузер **IE 2.0** зробила безкоштовним, включивши його до складу ОС Windows-95. Піднявся страшний шум, Гейтса звинувачували у нечесній конкуренції та спробі монополізувати ринок браузерів. Уряди декількох штатів, які лобіювались конкурентами, подали скаргу до суду, в результаті зав'язалася гучна «справа Microsoft», яка, вщухаючи та знов розгортаючись, тягнулася багато років і закінчилася, загалом, нічим.

Тим часом на доопрацювання стратегічного продукту Microsoft кинула великі інтелектуальні сили та гігантські грошові кошти (у бюджеті 1996 року на дослідження та розробки було виділено 2 млрд доларів). Нарешті у серпні 1996 року майже одночасно з **Netscape Navigator 3.0** вийшла версія **Internet Explorer 3.0**.

Обидва браузери мали масу нових можливостей, зокрема, підтримку мови Java та нових образотворчих засобів HTML. При цьому з'ясувалося, що розробники IE і NN реалізували ці нові можливості по-різному, через що браузери виявилися частково несумісними. Зрозуміло, це робилося спеціально, проте у програші опинилися ні в чому не повинні користувачі. Web-сторінки, які були оптимізовані для одного браузера, некоректно відображалися в іншому. Справа дійшла до того, що розробникам доводилося створювати по два варіанти одного і того ж сайту, кожен для свого браузера.

Так чи інакше, до кінця 1996 року сектор IE збільшився до 15%, але до перемоги було ще далеко. У подальші роки Microsoft методично продовжувала наступ, від версії до версії вдосконалюючи браузер. Розрив між конкурентами поступово скорочувався, до середини 1999 року позиції сторін вирівнялися, а у жовтні, коли на долю Internet Explorer доводилося вже 64%, стало зрозумілим, що Microsoft перемагає. Далі події розвивалися у повній відповідності із законами капіталізму: у 1999 році компанія Netscape була продана гігантові America On-Line за 4,2 млрд доларів акціями і, за деякими повідомленнями, взагалі покидає ринок браузерів, тим більше, що цей ринок сам по собі перестав бути прибутковим, оскільки, слідом за Microsoft, її конкуренти змушені були зробити браузери безкоштовними.

Проте Netscape Navigator залишився живий, він входить у комплект комунікаційних програм Communicator і займає природне місце у ніші не-Windows додатків, вільних від засилля Microsoft. Більш того, фірма Netscape опублікувала в Internet початковий код цього продукту і віддала його у руки світової громадськості програмістів (проект **Mozilla**). Також існують і інші браузери, наприклад, знаменита **Opera**. Відрізняються вони один від одного тим, що деякі елементи (функції) Web-сайтів працюють в одних і не працюють в інших браузерах.

6. Пошук в Internet

Вибуховий розвиток Internet останніми роками XX століття окрім очевидних переваг мав і зворотну сторону. Інформаційних ресурсів у Мережі стало так багато, що пошук потрібного документу серед десятків та сотень мільйонів непотрібних перетворився на серйозну проблему. Гіпертекст та будь-які браузери самі по собі не вирішують цю задачу, тому що вийти на потрібний ресурс тільки за гіперпосиланнями малоймовірно. Паперові довідники типу «Жовті сторінки Internet» допомагають лише частково через те, що, по-перше, Мережа дуже велика для будь-якої книги розумних розмірів, а, по-друге, ситуація у ній змінюється настільки швидко, що довідник встигає застаріти, перш ніж вийде з друку. У зв'язку з цим життєво важливим для подальшої долі всесвітньої інформаційної системи стало завдання пошуку різноманітних об'єктів (файлів, Web-сторінок, користувачів, та ін.).

Статистика показала, що троє з чотирьох користувачів, відповідаючи на питання: «Для чого ви використовуєте Інтернет»?, називають пошук інформації. І це не дивно – в Мережі, без перебільшення, є практично усе! Інтернет є неосяжним океаном мислимої і немислимої інформації. Він настільки великий, що не будь спеціальних засобів пошуку, знайти у ньому будь-що просто не представлялося б можливим.

Як знайти потрібну Вам інформацію в Internet? Якщо Ви не знаєте її адресу, то без засобів пошуку це не простіше, ніж знайти квартиру людини у великому місті, не знаючи її прізвища та адреси. Для того щоб не "заблукати" у мережі Internet, були створені спеціальні засоби пошуку. Компанії, які надають цей вид сервісу, збирають та регулярно оновлюють дані про мережні ресурси певного виду. Систематизують та накопичують їх у серверних базах, доступних для читання всім користувачам через клієнтські програми (спеціалізовані або універсальні). Засоби пошукового сервісу діляться на два типи: ***каталоги (directories)*** та ***пошукові системи (search engines).***

6.1. Internet-каталоги

Каталоги влаштовані за принципом бібліографічних довідкових систем. У них кожна книга або стаття знаходиться на певному місці у предметному або авторському покажчику. У мережному каталозі посилання розсортовані за тематичними рубриками та супроводжуються анотаціями. Мережний каталог, на відміну від бібліотечного, дозволяє значно прискорити роботу: на його головній сторінці є віконце для пошуку. Після введення ключового слова ви одразу отримуєте список рубрик та посилань, в яких вони зустрілися.

Прообразом усіх каталогів є розглянута нами система Gopher, яка систематизує документи Gopher-простору. Першим широко відомим каталогом для World Wide Web стала система **Yahoo!** (із знаком оклику), яка була створена у 1994 році докторантами Стенфордського університету Джері Янгом (Yang, Jerry) та Девідом Філо (Filo, David). Назва Yahoo є скороченням цілої

фрази «Yet Another Hierarchical Officious Oracle – Ще один ієрархічний дружній путівник». Ця система виросла з колекції Internet-посилань, яку розробники склали для своїх дисертацій.

Коли список став дуже великим та неосяжним, вони розбили його на декілька рубрик. Коли рубрики виросли в об'ємі, їх розділили на підрубрики і т. д., так народилася майбутня всесвітньо відома система. Система була встановлена на особистих комп'ютерах авторів, чутки про неї розповсюдився серед знайомих та друзів, до неї поступово почало звертатися все більше і більше Internet-відвідувачів. Коли до осени 1994 року був зареєстрований перший мільйон відвідин, друзі зрозуміли, що в їх руках знаходиться потенційний бізнес. Весною 1995 року за допомогою інвесторів були зібрані 2 мільйони доларів і утворена компанія Yahoo! Inc. з штаб-квартирою в містечку Сан Веллі у Кремнієвій долині.

Вже до 1997 фірма мала дохід 67 млн доларів та увійшла до списку 200 найбільших мережних компаній. На даний час її послугами щорічно користуються 237 мільйонів осіб, адреса www.yahoo.com є одним з найпопулярніших в Internet. Компанія не тільки веде гігантський каталог HTML-сторінок, але забезпечує безліч інших Internet-послуг: повнотекстовий пошук, електронну комерцію, Internet-мовлення тощо.

Прикладами інших сучасних каталогів є list.mail.ru, Rambler's "Top 100".

6.2. Пошукові системи

Щоб пошук в Інтернеті був зручним, швидким і ефективним, створені спеціальні пошукові системи. Практично кожен сеанс вашої роботи в Інтернеті не обійдеться без того, щоб ви не скористалися улюбленою пошуковою системою. Тому слід знати, як вона влаштована і яким чином максимально використовувати її можливості.

На відміну від каталогів, що зберігають лише анотації, пошукові системи Internet систематизують дані в автоматичному режимі та зберігають весь текст web-сторінок. Для цього пошукові машини щодня "повзають" по Мережі: вони відвідують web-сторінки та заносять їх у свої бази.

6.2.1. Пошукові машини

Людина може тільки ініціювати процес: як і у випадку з каталогами, автор сторінки повинен послати пошуковій системі заявку на свій новий матеріал. Якщо заявку не подати, пошукова система сама дістанеться до нової сторінки, використовуючи посилання, що веде до неї, але це відбудеться повільно. Тому після створення у Мережі своєї сторінки рекомендується «прописатися» в основних пошукових системах.

Пошукові системи – це спеціальні сервери, які займаються збором, впорядкуванням і пошуком інформації, яка є на просторах Інтернету. Основним

їх призначенням є забезпечення користувачів необхідною інформацією в результаті швидкого і зручного пошуку в Мережі.

Пошукові системи зазвичай складаються з чотирьох основних компонент:

– агент (павук або кроулер), який переміщається Мережею та збирає інформацію;

– індексатор;

– база даних;

– пошуковий механізм.

1. Агенти. Засоби пошуку типу агентів, павуків, кроулерів та роботів використовуються для збирання інформації про документи, які знаходяться у Мережі Internet. Це спеціальні програми, які займаються пошуком сторінок у Мережі, витягують гіпертекстові посилання на цих сторінках та автоматично індексують інформацію, яку вони знаходять для побудови бази даних. Кожен пошуковий механізм має власний набір правил, які визначають, як збирати документи. ***Агенти*** – «найінтелектуальніші» з пошукових засобів. Вони можуть робити більше, ніж просто шукати: вони можуть виконувати навіть транзакції від Вашого імені. Вже зараз вони можуть шукати сайти специфічної тематики та повертати списки сайтів, які відсортовують за їх відвідуваністю. Агенти можуть обробляти зміст документів, знаходити та індексувати інші види ресурсів, не тільки сторінки. Вони можуть також бути запрограмовані для отримання інформації із вже існуючих баз даних. Незалежно від інформації, яку агенти індексують, вони передають її у базу даних пошукового механізму.

Загальний пошук інформації в Мережі здійснюють програми, відомі як ***павуки***. Павуки повідомляють про зміст знайденого документа, індексують його та отримують підсумкову інформацію. Також вони проглядають заголовки, деякі посилання і надсилають проіндексовану інформацію у базу даних пошукового механізму.

Кроулери переглядають заголовки та повертають тільки перше посилання.

Роботи можуть бути запрограмовані так, щоб переходити по різних посиланнях різної глибини вкладеності, виконувати індексацію та навіть перевіряти посилання у документі. Проте є методи, що призначені для того, щоб заборонити роботам пошук по сайтах, власники яких не бажають, щоб вони були проіндексовані.

2. Індексатор (indexer) – програма, що служить для впорядковування інформації, знайденої павуками для подальшого застосування швидкого двійкового пошуку. Індексатор розбиває текст документу на складові його слова, при цьому запам'ятовуючи їх місцезнаходження, аналізуючи, написано слово прописними або рядковими буквами, чи являється воно назвою документу, текстом посилання тощо.

3. База даних (database) – є сховище усіх оброблених даних, накопичених пошуковою системою, які збирається павуками.

4. Пошуковий механізм і механізм видачі результатів (search engine) – призначений для взаємодії користувача з базою даних. З ним ви постійно

матимете справу при пошуку інформації. Механізм видачі результатів визначає, які веб-сторінки вам представляти відповідно до вашого запиту.

Після завдання вами ключового слова або фрази пошукова система здійснює пошук і видає вам результат, який є списком знайдених веб-сторінок, що відповідають вашому запиту. Для визначення порядку, в якому буде показані документи в списку, пошукова машина застосовує алгоритм ранжування. У ідеалі найбільш релевантні вашому запиту сторінки будуть розташовані в списку першими. Проте ідеальний алгоритм ранжування не знайдений, тому кожна пошукова система використовує власний, хоча відбір результатів в усіх системах грунтується приблизно на наступних критеріях:

- наявність ключових слів в заголовку;
- щільність ключових слів у вмісті документу, тобто кількість шуканих слів в тексті веб-сторінки;
- розташування ключових слів в документі;
- стиль ключових слів (напівжирний, курсив);
- індекс цитованості – кількість посилань, що ведуть на цю веб-сторінку з інших сторінок.

Прикладами найбільш популярних пошукових систем є Google, Yahoo!, Яндекс, Rambler, Апорт, mail.ru.

Слід зазначити, що практично усі представлені вище пошукові машини можуть працювати і з кирилицею. Але для пошуку інформації російською мовою все-таки краще рекомендувати російські пошукові системи: Яндекс, Rambler, Апорт, mail.ru.

6.2.2. Пошукова система Google

У 1996 році на сайті Стенфордського університету (штат Каліфорнія, США) з’явилася нова пошукова система. За назвою **"BackRub"** стояла наукова робота аспірантів Сергія Бріна та Ларрі Пейджа. Необхідний для роботи пошукової системи сервер з вінчестерами загальним об’ємом 1 терабайт мав у своєму розпорядженні Брін у кімнаті університетського гуртожитку. В основі BackRub лежала принципово нова система Internet-пошуку, коли всі знайдені за запитом сторінки ранжирувалися по кількості посилань на інші сторінки. Таким чином, у верхніх рядках опинялися документи, які мали найбільшій попит.

Пошук виявився настільки зручним, що скоро до нього почали звертатися люди далеко за межами університетського містечка. До літа 1998 року до BackRub щодня зверталося близько 10 тис. відвідувачів. У Стенфорді затурбувалися – сервіс почав заповнювати майже половину всього університетського Internet-трафіку. А тут ще додалися звинувачення у комп’ютерному хуліганстві. Річ у тім, що пошукова система не звертала уваги на обмеження доступу до університетських документів "для службового користування", відкриваючи їх для всіх. І BackRub, що перестав бути чисто

науковим проектом, пригрозили закрити. "У якийсь момент мені довелося зробити вибір: починати свою справу або продовжувати навчання", – говорив потім про це Сергій Брін.

Зробити вибір йому несподівано допоміг один із засновників Sun Microsystems Енді Бехтольшайм, виписавши Сергію чек на $100 000 після короткої розмови про неіснуючу ще компанію Google Incorporated. Брін створив Google у 1998 році разом з університетським другом Ларрі Пейджем. Назва компанії Google – це спотворений напис англійського слова "googol" ("гугол", або Googleplex – це велике число – асоціація з величезними масивами даних), що означає число – одиницю та сто нулів, 10^{100}. Тільки наступного дня їм підказали, що вони неправильно написали це слово. Воно пишеться як "googol". Але "неправильна" назва була вже зареєстрована.

Сергій народився у Москві у 1974 році і потрапив до Америки п’ятирічною дитиною: його батьки емігрували з СРСР у США.

Після школи Сергій поступив на факультет математики Університету Меріленда, достроково отримав диплом бакалавра та спеціальну стипендію, яка дозволила йому продовжити навчання у дорогому та престижному Стенфордському університеті. У Стенфорді Брін був прийнятий до докторантури.

В університеті Сергій подружився з Ларрі Пейджем, уродженцем штату Мічіган, і у них з’явилися спільні проекти.

Свою знамениту нині пошукову систему Брін та Пейдж вперше випробували на товаришах по навчанню.

Засновники системи Google Ларри Пейдж та Сергей Брин

Сан-Франциско – головне місцезнаходження компаній, що вкладають гроші у перспективні проекти. Дві з них погодилися інвестувати в Google 25 млн. доларів. Google швидко набирав силу, його популярність росла день за днем, і дуже скоро компанія перейшла з розряду початківців у категорію фірм, що швидко ростуть. Влітку 2000 року Google підписала контракт на обслуговування пошукових запитів Yahoo! – Internet-компанії, якій належить найпопулярніший сайт планети. Співпраця з Yahoo! дала могутній поштовх популярності Google, проте головним чинником зростання компанії стало схвалення, яке передавалось із вуст у вуста користувачів, які рекомендували Google друзям та знайомим.

Сьогодні Google вже перестала бути частиною виключно американського життя і культури. 67% користувачів Internet живуть за межами США, для 77% Internet-аудиторії англійська – не рідна мова. Але всі вони використовують для пошуку Google. Саме тому Google говорить на безлічі мов: в її арсеналі більше сотні живих мов. Google дозволяє користувачам знайти інформацію на будь-якій мові і за допомогою лінгвістичних технологій зрозуміти, про що йдеться.

Окрім головного пошуку Google пропонує доступ до цілого набору інших можливостей. Кожен може скористатися пошуковою системою, але чомусь знайти те, що треба, вдається далеко не всім. А все через те, що будь-який інструмент потрібно використовувати із знанням справи.

Числові діапазони. Пошукова машина Google розуміє числові діапазони – через знак «..» можна задати нижню та верхню межу деякого числового значення, яке має бути присутнім у документі. Приклад: **Домашній комп'ютер 2000..2002**

Пошук по конкретному сайту. Часто буває ситуація, коли ви точно знаєте, що потрібна інформація є на конкретному сайті, але ви ніяк не можете її знайти. Скористайтеся модифікатором site:. Приклад: в Internet-магазині shop.siriust.ru є великий вибір викруток для ремонту стільникових телефонів. Наберіть у Google "**викрутка" site:shop.siriust.ru**.

Враховуємо назву документа. Набагато більшої ефективності пошуку вдається досягти у Google, якщо за допомогою модифікатора intitle вказати слова, які обов'язково повинні входити у заголовок документу. Приклад**: intitle:статьи site:www.xakep.ru**.

Пошук схожих сайтів. Кращий спосіб знайти дружні (та конкуруючі) сайти – запитати про це Google. У відповідь на модифікатор related:[URL сайту] він з радістю видасть сайти із схожою тематикою. Приклад: На запит **related:www.whatis.ru** система видасть посилання на сайти, що пов'язані з Windows.

Хто на нас посилається? Можна використовувати Google і для того щоб перевірити популярність конкретного проекту. Так, модифікатор link:[URL сайту] відображатиме всі сторінки, які посилаються на цей ресурс. Чим більше сторінок, тим ресурс популярніший. Приклад**: link:www.whatis.ru**.

Використовуйте синоніми. Якщо хочете, щоб до результатів увійшли сторінки не лише з конкретним вказаним словом, але і з його синонімами, поставте перед ним знак «~». У цьому випадку будуть знайдені документи, що містять не лише саме слово, але і його синоніми. Правда, словник синонімів Google представлений лише англійською мовою.

Знайти конкретний тип документу. Якщо у Google виконується пошук конкретного типу документу, будь це звичайна сторінка, презентація, PDF або що-небудь ще, то використовується модифікатор filetype:. Приклад: "**SQL-injection" filetype:pdf**. У відповідь система видасть посилання на PDF-документи за SQL-injection.

Словник термінів. Щоб швидко знайти визначення будь-якого терміну, використовується модифікатор define:. Такий самий результат можна отримати, якщо перед словом поставите людські фрази «What is» або «що є». Приклад: **define:LDAP**.

Онлайн-переклад. Google за допомогою власного механізму (translate.google.com) перекладає вміст сторінки потрібною мовою. Онлайн-переклад на англійську (російську, українську) може сильно допомогти,

якщо потрібний текст виявився німецькою мовою, а німецькою вам відомо лише "Хенде хох". Якщо при перегляді результатів пошуку праворуч від назви документу стоїть [перекласти вміст], то, клацнувши по ньому мишкою, буде запропоновано вміст сторінки російською мовою.

Калькулятор. За допомогою пошукової системи Google можна шукати не лише слова і цілі фрази на сторінках, але і результати математичних обчислень! Працює це все так само, як і звичайний пошук. Спробуйте ввести у пошуковий рядок щось на кшталт «15+78*4,5», натискайте «Знайти» і поглянете, що при цьому вийде. Детальний опис синтаксису калькулятора (наприклад, як вам узяти натуральний логарифм числа п'ять) – на http://google.ru/help/calculator.html. Крім того, Google може працювати і як перекладач одиниць виміру: введіть в пошуковий рядок «1 mile in kilometers» і взнайте, скільки кілометрів в одній милі.

Пошукову систему можна використовувати як зручний конвертер валют!

Приклад 1: (31337-3.14)/87. Приклад 2: 600 USD in GBP.

На початок 2001 року Google почав приносити прибуток, залишаючись приватною компанією, акції якої були розподілені всередині вузького кола засновників та інвесторів. Логіка розвитку підказувала необхідність виходу на біржу як для компанії відкритого типу. У серпні 2004 року довгоочікуване первинне розміщення акцій відбулося. Але не звичайним шляхом: замість розповсюдження через брокерські контори керівники Google організували відкритий аукціон. Сергій Брін говорить, що це чесніший шлях. Акції компанії були випущені двох типів: А та Б. Більша частина акцій першого типу залишилася в руках Бріна та Пейджа.

У 2000 році Google став найбільшою світовою пошуковою системою. Наступні роки пошукова машина вдосконалювалася, додавалися нові служби та можливості. Сьогодні його індекс містить відомості про більш ніж 8 млрд різних URL, а сама система обробляє 200 мільйонів запитів щоденно.

Хоча ані Брін, ані Пейдж так і не отримали докторських ступенів, покинувши стіни Стенфордського університету у період створення власної компанії, у своїй кадровій політиці вони віддають перевагу докторам наук – їх у Google налічується більше ста. А всього у компанії працюють близько двох тисяч чоловік. Пошуковий механізм Google сьогодні – це 10000 замкнених в єдиний ланцюг могутніх комп'ютерів, які ведуть пошук по 3 млрд сторінок Internet. Користувачі Google можуть отримати низку інших інформаційних послуг. Тисячі рекламодавців розміщують свої банери в Internet через Google. Але основний наголос керівництво Google робить як і раніше, на пошук – безпрецедентно ефективний.

Компанія з величезним обігом на одного співробітника свідомо дозволяє їм 20% свого робочого часу займатися тим, що їм цікаво і при цьому годувати їх обідами, розставляти шафи із соками, горіхами, дозволяти грати на більярді.

Засновники Google діють відповідно до законів бізнесу. Головою ради директорів та головним виконавчим директором компанії Google є не Брін і не

Пейдж, а набагато досвідченіший професіонал галузі Ерік Шмідт, який раніше працював у Novell. У заснованій ними корпорації Брін служить президентом з технології, а Пейдж – президентом з продукції. У 2006 році особисті статки Сергія Бріна та Ларрі Пейджа складали по 11 млрд доларів.

Світовий лідер в сфері пошуку в останні роки показує схильність до певних проривів і в інших сферах. Все почалося з пошукової системи, але цим не обмежиться. Перед вами план із п'яти пунктів, якого дотримуються Ларрі Пейдж і Сергій Брін в процесі наближення до світового панування [http://ilife-news.com/7615-5-shagov-google-na-puti-k-mirovomu-gospodstvu.html, 25.12.2013].

1. Контроль потоку інформації. Google з'явився в процесі розвитку проекту аспірантів Стенфорда Пейджа і Бріна, який повинен був поліпшити пошук інформації.

З моменту заснування компанії у 1998 році вона швидко вийшла на перше місце в сфері світового пошуку і досі тримається там. Протистояти їй поки нікому – хіба що IBM, почне розкручувати свій Watson. Сьогодні фірмова технологія компанії обслуговує більше мільярда пошукових запитів на 146 мовах світу щодня. Перехід від ПК до мобільних пристроїв ще більше допоміг пошуку Google, оскільки загальна частка Google у сфері мобільного пошуку досягає 90% , в той час як на ПК тільки 70%, згідно з даними NetMarketShare.

Залізна хватка Google буквально тримає всю інформацію світу – адже всі ми користуємося послугами компанії. Допоміжні послуги від Google, від Gmail до Карт, від YouTube до Play Store на Android, тільки покращують доступ до інформації і забезпечують пошуковому гігантові гнучкий контроль.

Якщо завтра Google зникне, ми все ще зможемо користуватися альтернативними сервісами, наприклад, такими як «Яндекса» і Bing від Microsoft. Але поки ці опції не стали набагато краще тих, що пропонує Google, ми будемо ходити до нього в гості. Змінити пошукову систему дуже просто і безвитратно. Google це знає і розуміє, тому намагається забезпечувати кращу якість пошуку, однак одним цим не забезпечити собі вічну славу, а значить потрібно переходити до наступного кроку.

2. Управління доступом до інформації. Пошукова система – це всього лише ворота до інформації, а ось апаратне забезпечення вашому пошуку – це воротар. У зв'язку з цим вже сьогодні Google обслуговує найпопулярнішу мобільну операційну систему світу, Android , яка зайняла 81 % світового ринку смартфонів станом на другий квартал 2013 року. Android також обійшла Windows, ставши найпопулярнішою операційною системою в принципі. Таке домінування складно перервати, і Apple чітко зрозуміла це в 80-х роках, коли комп'ютери з операційною системою від Microsoft стали домінувати на ринку.

Однак на цьому Google не зупиниться. Її комп'ютери Glass стануть наступним логічним ланцюгом в гонці мініатюризації, яка перенесе комп'ютери з кишені на обличчя, і ті будуть перед вами постійно. Жодна інша компанія не може бути настільки близька до користувача з цим типом пристроїв, хоча очікується, що на початку 2014 року пристрої хлинуть на ринок. Залишилося

всього кілька місяців, а Google вже багато чого виграла від поширення програми Glass Explorers серед невеликого числа бета-тестерів.

Комп'ютери за своєю суттю представляють трубопроводи інформації, і Google Glass надаватиме інформацію точно так само, як її сприймають люди, ставши, таким чином, якоюсь другою шкірою з технологій. Люди вкрай лояльно ставляться до iPhone, але вони стануть фанатиками Glass, оскільки очки втілять в собі все, що потрібно таким людям з дня на день.

3. Контроль транспортної мережі. Контролювати інформацію – це тільки початок. Google не може по-справжньому правити світом, якщо візьме на себе тільки онлайн-частину. Самокеровані автомобілі – це важливий крок вперед. Вони нададуть Google контроль над транспортною мережею, оскільки саме на цих автомобілях люди будуть з легкістю і безпечно переміщатися з одного пункту в інший. Зараз ми хіба що використовуємо Google Maps, щоб добратися від А до Б, однак при цьому під час шляху ми всі так само звертаємося до Інтернету.

Велика частина витраченого марно в пробці часу є результатом людської помилки. Дорожні пригоди практично завжди є результатом помилки водія, і щороку по всьому світу відбуваються мільйони аварій. Усунення слабких місць у транспортній мережі, швидше за все, заощадить трильйони доларів, які витрачаються на лікування, купівлю нових машин, часу в пробці і інші непотрібні речі. Як тільки світ наповниться самокерованими автомобілями від Google, пошуковий гігант накладе лапу на наше повсякденне життя, і ми ще будемо вдячні за це. Нам залишиться тільки займатися щоденною чорною роботою, але тут у гру вступає останнє придбання Google.

4. Контроль ринку фізичної праці. Boston Dynamics має репутацією творця самих передових роботів на планеті. Її чотириногі боти були розроблені як в'ючні мули для військових, а двоногий PETMAN страхітливо нагадує людини у своїх рухах.

Звичайно, нікого не лякає, що робот може вкрасти вашу роботу лише завдяки своєму вмінню ходити. Але варто відзначити, що створення робота, який успішно пересувається по пересіченій місцевості, як BigDog, це величезний прорив у світі робототехніки. У процесі підкорення важких проблем буднів роботів ходьба – це перший крок. Уявіть світ, в якому ваш смачний обід буде готувати робот, який просто знає, як потрібно чистити картоплю, подавати до столу і говорити «смачного». Майстерність складається з маси простих завдань, яким можна навчити робота, і як ми вже сказали, ходьба це тільки початок. Не кажучи вже про те, що роботи вже замінюють робочих на складальних конвеєрах.

Оскільки роботи беруть на себе все більше і більше ручної роботи, люди будуть витрачати більше часу на розумову працю (у якому, до речі, Google процвітає) . Якщо Google вдасться створити симбіоз розумових і фізичних пропозицій, які зроблять наше життя простіше, краще і приємніше, то компанія увійде в кожен сегмент світової економіки і міцно влаштується там.

5. Контроль над життєвими таємницями. Кілька місяців тому Google анонсувала створення Calico, якій мав зайнятися колишній CEO Genentech і яка кидає виклик скромній меті продовження життя. Преса негайно відреагувала на «гугловський» проект безсмертя, який може бути не так вже й далекий від цього. Згадаймо про геніальне мислення Ларрі Пейджа, який завжди дивиться тільки в майбутнє і намагається випускати продукти, які разів у десять краще всього того, що є зараз. Нехай і не відразу. Безсмертя знаходиться за межами досяжного в найближчі десять років, а от продовження життя до 150 років – цілком. Обрі ді Грей, відомий геронтолог (геронтолоогія – наука, що вивчає процес старіння людини), стверджує, що перша людина, яка доживе до 150 років, уже народилася. А перша людина, яка доживе до тисячі років, народиться наприкінці 2030-х.

Google, по всій видимості, цілком здатна дати нам комп'ютери, що носяться на голові, автомобілі без водіїв і роботизовану силу до того, як середньостатистична людина буде жити 150 років. Але перші рішення блякнуть порівняно з вирішенням проблеми старіння. За допомогою цього рішення Google не тільки контролюватиме доступ до інформації, але також час, що відведений вам на цей доступ до інформації.

Можливо, Google ніколи не досягне цієї мети, але давайте на хвилинку припустимо, що досягне. Уявіть собі суспільство працівників розумової праці, які живуть вічно під захистом технологій Google. Чи це мрія Ларрі і Сергія? Що, якщо не це?

6.2.3. Пошукова система Яндекс

Якщо 75% усіх пошукових запитів мережі Internet обробляє пошукова система Google, то близько 80% користувачів російської частини мережі віддають перевагу Яндексу. Причиною тому – якісний евристичний аналіз тексту індексованих сторінок, що враховує морфологію російської мови. По цьому параметру Яндекс значно випереджає Google.

Підприємець та програміст Аркадій Волож заснував у 1988 році компанію CompTek. Спочатку вона займалася продажем комп’ютерів та розробкою програмного забезпечення для автоматизації робочих місць. Але Аркадія завжди цікавили алгоритми пошуку інформації по великих масивах даних, і всього рік потому, тепер вже у компанії з Аркадієм Борковским, він відкрив фірму «Аркадії», яка почала розробляти інформаційно-пошукові системи «Міжнародна класифікація винаходів» та «Класифікатор товарів та послуг».

Розробка найпопулярнішої пошукової системи **Яндекс** у російськомовному Internet (Рунеті) як алгоритму пошуку у текстових документах почалася на початку дев’яностих років минулого століття. У 1993 році народилося слово "Яndex", що ніяк не пов’язане з пошуком в Internet. Придумав його Ілля Сегаловіч, один із головних розробників пошукового механізму, зараз – технічний директор компанії Яндекс. Засновником системи є також Аркадій

Волож, а основні розробники – Сергій Ільінський, Михайло Маслов, Дмитро Тейблюм – до цих пір працюють у компанії Яндекс.

Спочатку "Яndex" означало "Мовний index", або, за традицією програмістів, "yandex" – це "Yet Another indexer" (ще один індексатор).

Девіз системи – "Знайти все". Прогресуюча конкуренція серед пошукових машин примушує більш упереджено відноситися до подібних обіцянок, оскільки мало знайти все, треба вміти знайти найголовніше. Проте пошукова система Яндекс досить успішно з цим порається, чим і виправдовує право називатися сьогодні самим відвідуваним ресурсом **Рунету**. Рунет – скорочення від "Russia Internet" позначає частину Internet. У цій зоні розташовуються сайти, які відносяться до Росії. Втім, до Рунету також відносять сайти інших доменних зон, зміст яких представлений російською мовою. У першу чергу до них відносяться доменні зони республік колишнього СРСР.

Вперше про існування пошукової Internet системи http://www.yandex.ru світ дізнався 23 вересня 1997 року. Вона була представлена на виставці Softool, та одразу ж зацікавила фахівців здатністю здійснювати пошук з урахуванням морфології російської мови.

Періодом найбільш бурхливого розвитку Яндекса був 1999 рік, що характеризується блискавичним збільшенням Рунета як в об’ємах текстів, так і за кількостю користувачів майже на порядок. Надалі, з бурхливим зростанням російської частини Internet, вдосконалювався і Яндекс. Саме ця пошукова система (у російській частині всесвітньої павутини) почала застосовувати індекс цитування (аналог гугловскому PageRank. Крім того, вона постійно випереджала світового лідера з морфологічному розбору російського контенту сторінок. Як наслідок, при пошуку по російськомовних сайтах Яндекс видає коректніші результати. Остання обставина стала її запорукою популярності у Росії. Засновники Яндекса, які і на сьогодні є його головними акціонерами, дуже добре розуміють, що саме у цій особливості криється їхня головна конкурентна перевага і продовжують її посилювати. Для користувачів же це означає ще швидший та коректніший пошук по російських ресурсах мережі.

В кінці 2002 року система Яндекс вийшла на самоокупність, тобто досягла перевищення доходів над витратами. На даний момент більше 60% доходів Яндекс отримує від контекстної реклами та близько 20% – за переходи користувачів в електронні магазини. Популярність принесла не тільки славу, але і труднощі, що виявилися у чергах на покупку рекламних місць на порталі.

Сьогодні Яндекс – це найбільший портал, що надає окрім пошукового сервісу та каталогу велику кількість різноманітних веб-служб: новини, енциклопедії, словники, каталоги товарів, безкоштовну пошту, хостинг, платіжну систему та багато чого іншого.

Щодня портал відвідують близько двох мільйонів чоловік з урахуванням зарубіжних гостей. При цьому з усіх російськомовних користувачів, що заходять у Мережу протягом дня, в середньому близько 75% відвідують і сайт Яндекса.

Останнім часом Яндекс особливо підсилив свої позиції, що відбилося у збільшенні відвідуваності приблизно на 60% та досягненні цифри у майже 10 млн. чоловік на місяць (близько 80% приходять з Рунета). На даний момент Яндекс удвічі популярніший Рамблера та на стільки ж популярніший останніх пошукових систем російськомовного Internet разом узятих. Пошук залишається функцією порталу, яка має найбільший попит і яка обробляє за місяць близько 150 млн. запитів.

Яндекс володіє найповнішим та найактуальнішим індексом сторінок російськомовного Internet. Якісний пошук забезпечується ретельним лінгвістичним та технічним опрацьовуванням текстів. Алгоритм нечіткого пошуку, а також якісний морфологічний та синтаксичний аналіз дозволяють Яндексу розуміти запити на природній мові. Тому користувачі-початківці, щоб не ускладнювати себе правилами використання службових слів при побудові пошукових запитів, можуть їх формулювати, наприклад, так: "де орендувати офіс у центрі Києва".

Яндекс чудово розуміє російську мову і може утворювати та розуміти будь-які словоформи. Наприклад, якщо введений запит на слово "думати", то машина видає результати пошуку також і за словами "думав", "подумавши" та навіть "розум".

Павук Яндекса індексує усі документи у російськомовній частині Internet, у тому числі і в країнах СНД.

Розмір індексної бази разом з якістю пошуку є одним з основних показників, що характеризують пошукову систему. Яндекс, як могутня пошукова система, обходить Мережу за декілька днів. Проте по мірі зростання системи основне навантаження лягає вже не на індексуючого робота, а на пошук та видачу результатів. Найближчим часом планується активне впровадження на Яндексі традиційних для багатьох систем зі світовим іменем технологій **ешелонування** і **прюнінга**. Сутність першої методики полягає у розподілі індексу на дві частини: більш релевантну та менш релевантну. Спочатку пошук здійснюється у першій та, якщо результат не задовольняє кількісно, пошукова машина звертається до другої частини індексу.

Технологія прюнінга (від англ. pruning – відсікання, скорочення) дозволяє динамічно припиняти обробку запиту при накопиченні достатньої кількості релевантної інформації. Ця технологія значно скорочує час пошуку та навантаження на пошукача, тим більше, заздалегідь відомо, що більшість користувачів при перегляді результатів пошуку частіше за усе не заходять далі третьої сторінки результатів.

Починати пошук в Яндекс краще за все взагалі не відкриваючи ніяких сторінок. Браузери Mozilla Firefox, Opera та Internet Explorer 7.0 у своєму інтерфейсі містять рядок пошуку. Тому краще одразу вводити в них запит, а потім переходити до результатів. Це прискорює роботу. Браузер Internet Explorer за замовченням пропонує як пошукову систему Windows Live Search. Для зміни цього значення натискаємо на значок стрілки у полі пошуку, у меню, що з’явилося, обираємо пункт "Змінити параметри пошуку за замовченням".

З’являється діалогове вікно "Настройка умолчаний для поиска", в якому необхідно натиснути на посилання "Другие поставщики".

У 2003 році Yandex навчився шукати документи у форматах RTF, PDF та DOC. На сьогоднішній день він зберігає інформацію про більш ніж 150 мільйонах документів, що складає більше 4000 Гбайт.

Яндекс повідомляє про досягнення рубежу в 50000000 пошукових запитів, які оброблюються за день (з урахуванням mail.ru). Це у п’ять разів більше, ніж у червні 2004 року, коли цей показник досяг 10 мільйонів, та у 25 разів більше, ніж у квітні 2002 року, коли кількість щоденних запитів до пошукової машини Яндекса перевищила 2 мільйони. Рубіж у мільйон запитів Яндекс перейшов навесні 2001 року. Зараз кожен пошуковий запит "перелопачує" масив інформації у 20 терабайт на географічно розподіленому кластері з кількох тисяч серверів.

6.2.4. Пошукова система Rambler

Англійське слово **rambler** має безліч значень. Самим працівникам компанії більше до душі переклад «бродяга», під яким мається на увазі бродяга по Internet. Розробляти пошуковий механізм група однодумців з підмосковного наукового міста Пущино почала у 1991 році. Через п’ять років, у 1996 році, програміст Дмитро Крюков створив першу унікальну російську пошукову програму, яку одразу ж і запустили в експлуатацію. Перша в Росії пошукова система з самого свого початку розташувалася за адресою rambler.ru (полегшений варіант сайту – http://r0.ru). Поступово невелика група однодумців виросла у потужний Internet-холдинг. У лютому 1997 року запрацювала рейтингова система Rambler’s Top100 (top100.rambler.ru). Через 3 роки, 7 березня 2000 року, був зафіксований мільярдний відвідувач сторінок, який був зареєстрований у рейтингу. Тобто всі ресурси, які стоять у рейтингу Rambler (і більш до Rambler жодного відношення не мають), за три роки отримали мільярд відвідувачів.

Рамблер-пошта. У зв’язку з широким поширенням цифрових фотоапаратів користувачі все активніше використовують «Рамблер-пошту» для обміну фотографіями. Тому тепер для них стала доступною опція перегляду зменшених копій вкладених у лист фото-файлів, що дозволяє заощадити час та обрати найцікавіші кадри.

Також можна прослухувати вкладені у листи аудіо-файли формату mp3. Для цього досить клікнути на кнопку запуску програвача, яка розташована поряд з кожним mp3-файлом в інтерфейсі пошти.

З’явилася можливість перегляду відеороликів без виходу з листа. Отримавши у листі посилання на відеоролик, користувач пошти «Рамблера» може одразу подивитися його, не здійснюючи зайвих переходів. «Рамблер-пошта» підтримує перегляд роликів більшості видеохостингів Мережі.

6.2.5. Синтаксис мови запитів

Різні пошукові системи використовують різні алгоритми та формули для обчислення ваги та різні способи співставлення всіх цих факторів. Тому релевантність документів (тобто, відсоток відповідності документів запиту) оцінюється по-різному. Тобто один і той самий запит до різних пошукових систем дасть різні результати.

Хоча розширений запит та призначений для уточнення критеріїв пошуку, пошук можна забезпечити і за допомогою застосування мови запитів. Мова запитів – це спеціальні символи та оператори, які пишуться у той же рядок для пошуку, що і ключові слова, і обробляються пошуковою машиною.

Не будемо розглядати синтаксис окремо взятої пошукової системи (детальний опис мови запитів будь-якої системи можна без зусиль знайти на її сайті), а поговоримо про основні оператори і правила, які працюють однаково у більшості пошукових системах. Їх, як правило, буває цілком достатньо.

Спеціальні символи. Google, Yandex та Rambler мають схожість у застосуванні деяких спеціальних символів. Оточений лапками рядок буде знайдений саме у тому вигляді, що і в запиті – слова, які розташовані у тому ж порядку і знаходяться у тій же формі. Таким чином дуже зручно шукати, наприклад, цитати.

Символ «+» перед словом говорить про те, що слово має бути обов'язково присутнім в знайдених документах. Насправді за замовченням між усіма словами і так мається на увазі логічний оператор «ТА», тобто будуть знайдені документи, які містять одночасно всі слова із запиту. Тому символ «+» має сенс для так званих «стоп-слів». Це такі слова, які часто зустрічаються у текстах і навряд чи можуть бути критерієм для пошуку. Наприклад, прийменники, сполучники, займенники, артиклі.

Протилежне значення має символ «-». Слово, якому передує цей знак, не повинне попадатися у документі. У Rambler замість «-» використовується знак «**!**». Виключення слів – дуже простий, але корисний прийом, який дозволяє одразу відсіяти безліч документів, які точно не підходять.

Для того щоб закінчення слів не впливали на результати пошуку, в запитах можна використовувати знак зірочки * та знак запитання ? для завдання шаблонів. Знак запитання у шаблоні «***информаци?***» означає, що на місці цього знаку може бути будь-яка буква, а знак зірочки «***информац****» означає, що на цьому місці можуть бути кілька будь-яких букв.

Приклади запитів з шаблонами слів:

запит: ***поиск информаци?***

запит: ***поиск информац* -Internet***

запит: ***поиск информац* Internet?***

Оператори. Для створення складних запитів російськомовні пошукові системи припускають використання логічних операторів «ТА», «АБО» та «НІ». В англійській мові їм відповідають оператори «AND», «OR», «NOT».

Незалежно від мови позначенням логічного оператора «ТА» («AND») є знак амперсанду &, для оператора «АБО» («OR») вертикальна риска |, а для оператора «НІ» («NOT») знак мінус (-).

Наприклад, запит «(*фотографія* | *фото* | *фотознімок*) & (*тигр* | *носоріг*)» видасть сторінки з фото якої-небудь з двох тварин. Yandex-оператор «&» вказує на те, що слова повинні знаходитися в одному реченні. Для Rambler же достатньо, щоб вони були просто присутні у документі. Для того щоб Yandex шукав слова по всьому документу, потрібно використовувати оператор «&&».

Деякі з російськомовних пошукових систем та практично усі зарубіжні пошукові системи не розуміють тонкощів російської мови. Тому вони можуть давати різні відповіді на запити «*пошук інформації*» та «*пошук інформація*», що відрізняються лише однією останньою буквою у слові «*інформації*».

Пошук з відстанню. Також Yandex та Rambler дозволяють вказати відстань між шуканими словами у реченні. У Rambler для цього використовується конструкція '(число, запитів) ', де число – це відстань між словами, які представлені у запиті і вимірюється у словах. У Yandex використовується конструкція вигляду «/(n m) », де n і m – відстань назад та вперед у словах між ключовими виразами. Крім того, можна застосовувати спрощену конструкцію («/n» або вказувати відстань не в словах, а в реченнях («&&/(n m)». Так, на запит «*операційна* /2 *система*» будуть знайдені документи, в яких є присутніми слова «*операційна*» і «*система*», причому відстань між ними складає не більше двох слів і вони знаходяться в одному реченні.

Yandex відрізняється чутливістю до регістра букв. Якщо у запиті присутнє слово, яке написане з малої букви, то будуть знайдені документи, де це слово написане як з малою, так і з прописною. Якщо ж у запиті міститься слово, яке написане з прописної букви, то будуть знайдені лише слова, що починаються з прописної (якщо це слово не перше у реченні). Для виключення слів у межах речення служить оператор «~», у межах документа — «~~» (тобто «~~» еквівалентно «-»). Для пошуку точної форми слова (без урахування морфології) потрібно поставити перед ним «!». За допомогою операторів «$» та «#» можна, як і у розширеному пошуку, задати зону пошуку (заголовок документу або текст посилання) або елемент документу (опис картинки, ключове слово і т. і.). Крім того, в Yandex існує можливість впливати на ранжирування результатів. Через двокрапку після ключового слова або виразу можна вказати число, яке впливатиме на вагу цього слова або виразу. Також можна використовувати оператор «<-» для завдання уточнюючого слова або виразу – це збільшить релевантність документів, що містять уточнюючий вираз.

Поради при складанні запитів:

1. Перевіряйте правильність написання запиту. Орфографічна помилка, як правило, зводить весь пошук нанівець. Правда, багато пошукових систем використовують алгоритми виправлення орфографічних помилок.
2. Використовуйте синоніми. Це дуже корисно, якщо список знайдених документів занадто малий або його вміст вас не влаштовує.

3. Як можна рідше використовуйте в тексті запиту популярні слова , такі як «E- mail» , «програма», «посилання» тощо. Подібні слова зустрічаються на мільйонах веб-сторінок, тому великої користі вони не принесуть, якщо, звичайно, ви не шукаєте саме ці об'єкти.
4. Обов'язково застосовуйте мову запитів. Повірте, якщо мовою запитів правильно користуватися, то вона стане дуже ефективним знаряддям пошуку і допоможе вам заощадити багато часу.

6.2.6. Додаткові можливості пошукових систем

Окрім головної своєї функції – повнотекстового пошуку по документах Internet – пошукові системи часто надають низку додаткових послуг. Наприклад, в усіх трьох розглянутих пошукових системах є можливість пошуку у каталозі.

Для пошуку графічних зображень на Yandex відведений окремий розділ. Звичайне зображення знаходиться у якомусь документі та пов’язане з деяким текстом. По цьому тексту і можна спробувати його знайти. Також інформацію про зображення можна почерпнути з тексту, який розташований у документі поряд з картинкою, та з назви графічного файлу. При цьому ключові слова піддаються і транслітерації, і перекладу англійською мовою. Таким чином, якщо ви шукаєте зображення по ключовому виразу, наприклад, «білий слон», то знайдуться, у тому числі, і файли, що містять у своїй назві поєднання «slon», «elephant», «white» і таке інше.

На Rambler є спеціальна форма для пошуку файлів. Файли можна шукати будь-які або певного типу: картинки, аудіо, відео. На відміну від Yandex пошук відбувається лише за іменами файлів або каталогів, без аналізу яких-небудь елементів, що пов’язані з файлом. Ім’я файлу можна задавати точним значенням або використовуючи шаблони (символи «*» та «?») та регулярні вирази (складніші форми шаблонів).

Заголовна сторінка кожної пошукової машини – це не просто форма для введення запиту, але ще і значний портал. На сайтах Rambler та Yandex можна знайти посилання на популярні ресурси, програму телепередач, прогноз погоди, гороскоп, курси валют, останні новини, поштовий сервіс, онлайн-словники, енциклопедії та безліч інших розділів.

6.2.7. Яка із пошукових систем шукає краще

Оцінити, наскільки результат пошуку відповідає запиту користувача, досить складно. Google та Yandex володіють найбільшими базами по російському Internety. Але Rambler, оскільки це перша пошукова машина, яка почала індексувати російський Internet, краще виконує пошук по старих документах, які за яких-небудь причин не стали популярними.

Особливість Google полягає у тому, що завдяки вживаній там системі присвоєння ваги PageRank добре шукаються авторитетні сайти. У цьому

відношенні Google був першим, але зараз подібні алгоритми посилання використовують майже всі пошукові системи.

Yandex відрізняється своєю розвиненою мовою запитів (якою користуються менше 1% користувачів) та великими пізнаннями у морфології російської мови, але розробники системи завжди бачили своєю задачею забезпечення точності пошуку при так званому природно-мовному запиті, тобто коли непідготовлена людина просто прийшла і просто запитала.

7. Internet-торгівля

Як відомо, в умовах розвиненої ринкової економіки підготувати товар або бути готовим надати деяку послугу – це ще півсправи, не менш важливо донести товар до покупця, а послугу до клієнта. На це спрямована діяльність гігантської сучасної індустрії маркетингу та торгівлі, яка шукає все нові та нові способи представити товар на ринку та полегшити сам процес покупки.

Розвиток Internet та особливо Всесвітньої павутини, що втягнула у свої мережі сотні мільйонів потенційних покупців, надав принципово нові можливості для розвитку торгівлі. Все, до чого додумалися винахідливі продавці, перерахувати неможливо, тут ми перерахуємо лише основні напрями використання Internet-технологій у ланцюжку виробник-покупець.

Перш за все, про рекламу, яка, як усім відомо, є двигуном торгівлі, і яка заполонила всі традиційні засоби масової інформації. Не дивно, що як тільки з'явилися Internet-послуги, з'явилася і паразитуюча на цих послугах реклама. Першими її жертвами стали групи новин Usenet та електронна пошта, де вона проявила себе у формі ***спаму (spam)***, що на комп'ютерному жаргоні означає непрошений потік рекламних повідомлень, ***інформаційне сміття***. Це слово має незвичайне походження. У стандартному англо-російському словнику воно перекладається як «консервований м'ясний фарш», переносне значення у комп'ютерному жаргоні з'явилося завдяки американському телесеріалу «Monty Python's Flying Circus», в одному з серіалів якого всі діалоги заглушалися до абсурду нав'язливою рекламою м'ясних консервів.

Перший крупний скандал з приводу поштового спаму вибухнув у США у вересні 1996 року. Клієнти найбільшого Internet-провайдера America On-Line (AOL) поскаржилися на те, що їх поштові скриньки засипаються рекламною поштою, яка надходить від іншої американської компанії, – Cyber Promotions. Ця фірма, яка складається з шести чоловік, розсилала щодня до 1,3 мільйонів електронних листів. Фірма обслуговувала близько 5000 рекламодавців, основними замовниками були представники дрібного бізнесу, вартість рекламної кампанії складала від 60 до 1500 дол. AOL подала до суду, але справу програла, оскільки мережний спам, на відміну від поширюваного факсом, американськими законами не заборонений. Таким чином, одержувачам електронної пошти доводиться боротися з цим злом власними силами, встановлюючи на клієнтських програмах спеціальні фільтри, проте спроби поставити спам поза законом не припиняються.

31 травня 2002 року Європейський парламент прийняв Ухвалу про захист даних електронних комунікацій. Відповідно до принципу «opt-in» маркетинговим компаніям заборонена розсилка масових електронних листів, факсів та миттєвих повідомлень без згоди адресата. На жаль, ухвала не може захистити користувачів від рекламних розсилок, які здійснюються з території держав, які не входять до складу ЄС, а також від дій **спамерів**, які нелегально

розсилають мільйони електронних листів та не підкоряються жодному законодавству.

Поява Web-інтерфейсу відкрила перед мережною рекламою нові можливості. На HTML-сторінках рекламовані товари можна не тільки згадати у тексті, але і показати.

Зміст реклами розширився, її об'єктами стали не тільки товари, але і самі Web-сторінки. Стандартною формою реклами у WWW є ***банери*** (***banner*** – «прапор») – невеликі картинки стандартних розмірів, дуже часто з анімацією, за пропорціями нагадують прапорці. Клацання по такій картинці викликає перехід на рекламовану Web-сторінку. Створення дотепних привабливих банерів та розміщення їх на чужих сайтах перетворилося на специфічний вид бізнесу. Організовані цілі мережі, що надають послуги з розміщення банерів як за плату, так і на умовах взаємного обміну.

Виставивши товари в Internet, спритні діячі мережного бізнесу поставили наступне логічне завдання – тут же їх і продати. ***Internet-торгівля (internet trade)*** зародилася у США в середині 1990-х років та лавиноподібно розповсюдилася по всьому світу, проте вона створювалася не на порожньому місці.

Споконвіку у США, та і в інших розвинених країнах процвітала торгівля поштою, коли покупець обирав товар з каталогу, а магазин, отримавши грошовий переказ, негайно відправляв покупку адресатові. Система була налагоджена до досконалості, термін виконання замовлення, не враховуючи часу знаходження товару при транспортуванні, складав 1-2 дні. В принципі таким чином можна торгувати будь-якими товарами, проте найбільш зручними для посилочної торгівлі вважаються книги, компакт-диски, відеокасети та програмне забезпечення.

Саме з цих товарів почав Internet-торгівлю її родоначальник Джеффрі Безос (Bezos, Jeffrey Preston; нар. 1964). Легенда бізнесу, удостоєний журналом Time звання «чоловік 1999 року», Безос народився в Альбукерку, закінчив з відзнакою Прінстонський університет по кафедрі електронних вимірювальних приладів та інформатики, через два роки поступив на роботу в один з банків на Уолл-Стріті, де розробив комп'ютерну систему для обслуговування активу банку розміром у двісті п'ятдесят мільярдів доларів. Ще через два роки став наймолодшим віце-президентом за всю історію компанії і мав чудові можливості для подальшого зростання. Але він мріяв про власну справу і наполегливо шукав точку докладання своїх сил. Одного разу на початку 1994 року до його рук потрапили статистичні дані, які круто змінили все його життя. Безос дізнався, що річний обіг Internet виріс на 2300%, і зрозумів, що це унікальний шанс для людини, яка хоче поєднати традиційні технології комерції з комп'ютерними.

Звільнившись з роботи, Безос перебрався на західне узбережжя до Сіетлу та поряд із знаменитим книжковим супермаркетом **Ingram** заснував віртуальний книжковий магазин **Amazon**. Сайт Amazon.com відкрився 30 червня 1995 року, в перший тиждень доходи фірми складали близько 5 тисяч

доларів, в другий – 10 тисяч, а до закінчення другого року роботи – 150 мільйонів доларів на рік. При цьому, операційні витрати вимірювалися всього 13 тисячами доларів на квартал. Надвисока рентабельність пояснювалася дуже просто: віртуальному магазину не потрібні торгові зали та склади. Замовлення, що поступали по Internet, тут же переадресовувалися реальним постачальникам продукції.

На початку XXI століття річний дохід Amazon наблизився до 3 мільярдів доларів, у каталозі компанії значиться понад 2,5 мільйонів назв книг та інших товарів, більше 10 мільйонів чоловік щорічно відвідують її сайт. За допомогою Internet був поставлений світовий рекорд швидкості збагачення. Генрі Форду знадобилося 23 роки, щоб заробити свій перший мільярд на автомобільному бізнесі. Геній роздрібної торгівлі Сем Уолтон (Walton, Sam) – засновник найбільшої у світі мережі магазинів **Wal-Markt** – зробив це за двадцять років. Навіть Біл Гейтс свій перший мільярд отримав за дванадцять років. Джеффрі Безос став мільярдером за три роки.

У цілому аналітики передбачають різке збільшення електронного бізнесу найближчими роками. Так, обсяг Internet-торгівлі у 2002 році прогнозувався величиною 400 млрд дол, проте досвід останніх років показує, що розвиток Мережі передбачати неможливо, дійсність випереджає найсміливіші прогнози.

Наприклад, роздрібна торгівля за допомогою Internet на території США підвищилась з $172 млрд., обіцяних аналітиками у 2005 році, до $329 млрд. у 2010 році.

Ринок Internet-торгівлі України знаходиться на етапі бурхливого розвитку. За останні 10 років кількість Internet-магазинів зросла приблизно у 60 разів, а обіг онлайн-торгівлі у 2010 році перевищив 1 млрд дол.

Останнім часом широко поширюється залучення до такого виду бізнесу, як робота на електронних біржах, де можна брати участь у торгівлі валютою, товарами та опціонами. Це так звана система FOREX (foreign exchange – торгівля валютою). Відповідні клієнтські програми широко рекламуються та поширюються через Internet. Сутність роботи полягає у заробітку за рахунок різниці у ціні купівлі та продажу товару. Так, якщо валюта або деякий інший товар (золото, срібло, платина, нафта, пшениця, тощо) купуються за одну ціну, а продаються за більшу ціну, то трейдер заробляє на різниці у ціні товару. Те ж саме відбувається і в разі продажу товарів та наступної їх купівлі за нижчою ціною. Інакше трейдер несе збитки. Ці послуги надають ділерські фірми, через мережу яких здійснюється така торгівля. Ділерські фірми у свою чергу, напряму пов’язані з реально працюючими біржами Азії, Австралії, Європи та Америки. Для реальної торгівлі ділерські фірми надають так зване кредитне плече, яке дозволяє оперувати лотами (мінімальна одиниця товару, вартістю приблизно у $100000). Так, маючи кредитне плече 1:100 та власний рахунок $1000, можна вести торгівлю з одним лотом. Крім того, ділери надають можливість вести торгівлю не з цілим лотом, а з часткою, наприклад з 0.1, 0.2 і т.ін.

8. Електронна пошта

Електронна пошта (Electronic Mail, або E-Mail) – одне з найпоширеніших на сьогоднішній день засобів спілкування між людьми. Вона відкриває можливість передавати на великі відстані – всюди, де є Інтернет – не тільки текст, а й web-сторінки, а також абсолютно будь-які файли у вигляді вкладень в лист, причому всього лише за кілька секунд. Жоден з інших відомих зараз людству способів обміну інформацією не дозволяє робити це настільки ж швидко і якісно.

Існує також ще кілька можливостей E-Mail, таких, наприклад, як використання електронної пошти в телеконференціях або для звернення за довідковою і технічною підтримкою на деяких серверах. У разі якщо користувач хоче приєднатися до телеконференції, він відсилає на спеціальний сервер (mail reflector) заявку на підключення до листування з якої-небудь тематики. Після цього сервер автоматично відображає численним абонентам всю пошту з відповідних тем, що надходить на адресу його поштової програми.

Щоб отримати допомогу або технічний опис принципів роботи з деякими з серверів Інтернету, іноді достатньо вислати на їх адресу лист, що містить лише одне слово: help. Протоколи, які використовуються при роботі з електронною поштою, називаються SMTP та POP3, перший «відповідає» за відправку вихідної кореспонденції, другий – за доставку, яка надходить. Вхідна пошта записувалась в «електронну поштову скриньку» – як правило, це просто каталог на вінчестері поштового сервера, доступ до якого мав тільки абонент цієї скриньки. Звернувшись до сервера і забравши пошту, абонент може прочитати її, вже розірвавши з'єднання з Інтернетом. Однак слід пам'ятати, що після того, як пошта була витягнута з поштової скриньки, на сервері вона не зберігалась.

Щоб відіслати звичайний лист іншій особі, потрібно вказати на конверті адресу. Адреси електронної пошти хоч незначною мірою, але відрізняються від адрес, прийнятих в системі DNS. У цілому адресу електронної поштової скриньки (іноді його також називають E-mail) виглядає наступним чином: **Імя_абонента@адрес_почтового_сервера** (без пропусків). Знак @ («ат», або «собака» чи «песик») служить для того, щоб комп'ютер міг відокремити адресу сервера, на який слід відправляти лист, від імені абонента, вірніше, його особистого каталогу, куди цей лист запише по отриманні сам сервер. Там воно і буде зберігатися до тих пір, поки його не забере користувач або поки не закінчиться термін його зберігання.

Абонентське ім'я користувач вибирає сам. Після цього він реєструє поштову адресу на сервері, налаштовує програмне забезпечення і може працювати з електронною поштою.

Не зайве буде нагадати також, що електронна кореспонденція – це теж документ, тому при складанні листів намагайтеся дотримуватися існуючих правил етики. В Інтернеті не прийнято публікувати особисту переписку, але це і не забороняється. Тому намагайтеся не робити доступним для кожного

користувача зміст електронних листів і не пишіть в листах нічого, що не хотіли б побачити виставленим на загальний огляд.

В порівнянні із звичайною поштою, електронна пошта має безліч переваг:

1. Електронний лист досягає адресата за лічені секунди, як далеко б він не знаходився.
2. Відправка електронних повідомлень не коштує практично нічого. Ви платите тільки за час, проведений в Інтернеті.
3. До електронних повідомлень можна прикріплювати будь-які файли: зображення, музичні композиції, програми тощо.
4. Використовуючи електронну пошту, дуже зручно працювати з кореспонденцією: можна пересилати отримані листи по іншим адресам, встановлювати фільтри від непрошених гостей, відповідати на послання автоматично, посилати повідомлення відразу декільком адресатам, сортувати, зберігати отриману пошту, тощо.

У E-mail існують і недоліки. Основним з них є можливість перехоплення електронного послання. Перехоплення електронної пошти під силу далеко не кожному користувачеві, а ті, кому ця справа під силу, навряд чи зацікавляться вашим листуванням. Ну а для тих користувачів, які серйозно бояться просочування інформації, існують спеціальні шифрувальні методи, які зводять практично нанівець спроби перехоплення поштових повідомлень.

9. Сервіси сучасного Internet

Ми розглянули декілька типів сайтів, які є "класичними", вони з'явилися майже одночасно з розвитком Internet. Так було приблизно до епохи Веб 2.0, коли виникли нові концепції, що змінили зовнішність сучасного Internet.

Термін "Веб 2.0" зазвичай пов'язується з виходом статті Тіма Орейлі "Що таке Веб-сервер 2.0" "TimOReilly-WhatIsWeb2.0", яка була опублікована 30 вересня 2005 року. Російською мовою переклад цієї статті доступний на сайті "Комп'ютера online": http://www.computerra.ru/think/234100/. Для розробників веб-сайтів відповідність епохи Веб 2.0 означає використання певних концепцій, стандартів та навіть елементів оформлення. Для користувачів епоха ознаменувалася новими сервісами, які отримали заслужену популярність. Саме їх ми зараз і розглянемо.

9.1. Блоги

У класичній російській літературі 19 століття часто можна зустріти опис того, як герої і героїні романів вели щоденники. Пристрасті, інтриги, переживання – все це довірялося паперу, причому щоденник найчастіше вівся для себе.

Двадцяте століття радянської епохи містило безліч атрибутів, у тому числі і стінгазети. Що буде, якщо надати кожному охочому вести свою персональну стінгазету? А всім іншим надати можливість залишати коментарі до нотаток? Додати можливість об'єднуватися у тематичні співтовариства? І найважливіше – надати стінгазетярам можливість створювати списки друзів, із зручним читанням нових нотаток та коментарів до них? Новий сервіс Живих журналів (блогів), який народившись у 1999 році, набув поширення та популярності вже у 21 столітті.

Блог (англ. blog, скорочення від weblog, web «павутина» та log(us) слово), мережний журнал, або загальнодоступний щоденник подій в Internet (мінісайт), що містить інформацію особистого характеру, яка регулярно поповнюється і яку власник блога (блоггер) надає користувачам мережі.

Якщо світу потрібна ваша допомога і ви готові її надати, але вам абсолютно не хочеться заводити свій власний веб-сайт, то це можна зробити простіше, використовуючи блог.

Є багато способів завести собі блог, але найпростіший – скористатися послугами безкоштовного веб-сайту. Безумовний лідер у цій області – Blogger (http://www.blogger.com). Цей сайт належить Google. Потрібно просто зареєструватися на сайті, ввести основну інформацію про свій блог та обрати для нього симпатичний вам шаблон. Після того як ви зареєструєтеся, привласніть своєму блогу ім'я та оберіть URL-адресу. URL буде закінчуватися на blogspot.com, отже ваш URL виглядатиме приблизно так: http://www.prestonsposts.blogspot.com. Після того, як ви привласнили своєму

блогу ім’я та URL-адресу, оберіть шаблон і починайте писати повідомлення. Як? Натискуйте кнопку Start Posting (Створити повідомлення), яка з’явиться, як тільки ви обрали шаблон. Напишіть будь-що та, якщо треба, відформатуйте за допомогою відповідних HTML-інструментів. Натискайте кнопку Publish Post (Опублікувати повідомлення). Все – ви написали свій перший пост. Тепер за тією адресою, яку ви задали при реєстрації, існуватиме ваш блог. Щоб написати ще що-небудь, перейдіть за адресою http://www.blogger.com. Ви попадете на панель інструментів, такий собі центр управління. Клацніть по назві свого блогу, а на сторінці, що з’явилася, натискайте Create a new post (Написати нове повідомлення). Ви знов побачите сторінку і зможете знов поділитися своїми думками з усім світом.

За даними статистики (http://www.livejournal.com/stats.bml) на 7 червня 2008 року зареєстровано більше 15 мільйонів користувачів живих журналів. Середній вік власників живих журналів – від 16 до 24 років. Проте не слід вважати це явище виключно юнацьким заняттям. Багато достатньо відомих людей також ведуть свої щоденники (http://ru.wikipedia.org/wiki/Известные_люди_Живого_журнала). А найпопулярнішим блогом, ось вже кілька років, є щоденник фотографа Адагамова Рустема (http://drugoi.livejournal.com/), який розміщує у своїх записах фотографії різноманітних куточків Землі, супроводжуючи їх невеликими коментарями.

Ведення блогу передбачає наявність на сервері програмного забезпечення (движок блога), який дозволяє додавати та змінювати записи. Він може бути розміщений на особистому веб-просторі автора або в одній із служб, які надають місце спеціально для блогів. Для створення власного блогу достатньо скористатися такими ресурсами, як www.blogger.com, www.livejournal.com, www.blogs.mail.ru та ін.

Корпоративний блог. Успіх персональних блогів, їх значний вплив, на користувачів Internet призвели до того, що компанії почали вести корпоративні блоги. Таким прикладом може служити блог Яндекса (http://company.yandex.ru/blog/).Компанії, які ведуть корпоративні блоги, роблять свою діяльність прозорішою, а через це привабливішою для клієнтів. Наявність зворотного зв’язку також покращує діяльність компанії. Разом з тим, через те, що відвідувачі можуть залишати небажані або навіть нецензурні коментарі, які не завжди можна оперативно вилучати, більшість компаній Рунету поки що з побоюванням ставляться до ідеї вести корпоративні блоги. Існують також й інші причини – неповна ясність, про що можна писати у таких блогах, можливість винесення "сміття з хати", можливість стати об’єктом критики. Отже, ведення корпоративних блогів поки-що не стає популярним заняттям.

Колективний блог. Якщо один користувач може вести один журнал, то чому б кільком користувачам не спробувати вести такий самий журнал? Така ідея лежить в основі колективних блогів, коли безліч користувачів залишають нотатки.

При спільному наповненні блога виникає проблема регулювання матеріалів. Вона вирішується введенням репутації, яка на сленгу називається кармою. Через деякий час після реєстрації користувач одержує можливість залишати коментарі, які по ходу читання оцінюються іншими користувачами. Кожен позитивний голос збільшує репутацію (піднімає карму), кожен негативний – зменшує. Після набору необхідної кількості карми користувач має можливість публікувати повідомлення, які також оцінюються іншими відвідувачами. Така загальна ідея саморегуляції колективних блогів. У цьому плані вони докорінно відрізняються від форумів, де за порядком стежать модератори – обмежене коло громадян, які несуть "чергування". У колективних блогах модератори, звичайно, теж є, але їх втручання не є головним регулюючим механізмом.

9.2. Internet-пейджери

Спілкування електронною поштою, як вже наголошувалося, не дозволяє вести діалоги в режимі реального часу. Іноді це буває необхідно, наприклад, при обговоренні роботи, коли учасники віддалені один від одного. В принципі, можна використовувати чати – вище ми розглядали їх опис. Проте постійно заходити на сайти з чатами, авторизуватися, очікувати інших співбесідників – незручно.

У 1996 році ізраїльською компанією "Mirabilis" була випущена програма ICQ (http://www.icq.com/) (англ. акронім фрази "I Seek You", що перекладається як "Я шукаю тебе"), що поклало початок новому класу програм – Internet-пейджерам. Зараз ICQ став таким же звичним засобом спілкування, як і електронна пошта.

Більшістю Internet-пейджерів є програми, що встановлюються на комп'ютер. Після реєстрації, користувач отримує ідентифікаційний номер, який у поєднанні з паролем дозволяє одержати доступ до сервісу. Програма запускається разом з операційною системою, отже доступна для спілкування одразу після включення комп'ютера.

Для прикладу розглянемо пару Internet-пейджерів: QIP та Mail.Ru Агент.

QIP. Програма ICQ доступна і російською мовою, зокрема у рамках спільного проекту Рамблер-icq (http://icq.rambler.ru/). Проте наявність реклами, яка відволікає під час спілкування, призвели до того, що багато хто з користувачів віддає перевагу альтернативним Internet-пейджерам, які підтримують протокол ICQ. До таких Internet-пейджерів відноситься QIP (http://qip.ru/), аналог класичної програми ICQ. QIP не містить реклами, а для отримання номера не потрібно заповнювати численні поля веб-реєстрації.

Після отримання номера можна одразу почати спілкування з іншими користувачами. Звичайно, вказівка своїх даних, наприклад, імені та прізвища, бажана, але навіть і без них можна працювати.

Mail.RU Агент. Mail.RU Агент – це Internet-пейджер, який локалізований на кількох мовах та окрім переваг пейджера, є зручною інтеграцією з поштою Mail.RU.

При запущеному агентові необов'язково заглядати у поштову скриньку на сайті або навіть відкривати поштову програму – при отриманні нового повідомлення Mail.RU Агент видасть повідомлення. Пошук користувачів за адресою електронної пошти Mail.RU, інтеграція з іншими сервісами Mail.RU, та всі інші можливості Internet-пейджера роблять цю програму однією з найзручніших у своєму класі.

9.3. Skype

Програми для голосового та визуального спілкування є окремим класом засобів комунікації, що дозволяють розмовляти через Internet в режимі реального часу. Все що потрібне для їх використання – це комп'ютер з широкосмуговим доступом до мережі, динаміки та мікрофон. Втім, на звичайному комутованому доступі (dial-up) найчастіше теж можна використовувати ці програми.

Багато Internet-пейджерів також підтримують голосове спілкування. Але безперечним лідером у цій області є Skype. Skype дозволяє абсолютно безкоштовно розмовляти двом або декільком співбесідникам, на комп'ютерах яких встановлена програма, навушники та мікрофон. До платних послуг відноситься здійснення вихідних дзвінків на стаціонарні та мобільні телефони у більшості країн світу. Оплата похвилинна, диференційована, залежить від регіону. Її вартість складає від кількох центів, до кількох доларів (http://www.skype.com/intl/en/prices/callrates/). Skype також дозволяє приймати вхідні дзвінки із стаціонарних та мобільних телефонів на свій обліковий запис.

9.4. Дистанційне навчання

Дистанційне навчання – це можливість підвищувати свою кваліфікацію у зручному для себе режимі, за довільно обраними предметами. Але це ж банальне читання книг? Не зовсім так, річ у тім, що вивчення предмету повинно перевірятися складанням кваліфікаційних іспитів, що підтверджують знання слухача. Крім того, бажано хоча б віддалене, але спілкування викладача та студента, а також спілкування студентів між собою. Internet може допомогти налагодити ефективну взаємодію сторін у процесі навчання, а також розробити відповідні методики.

Дистанційне навчання – це зараз модна тема. На момент написання цього курсу (липень 2011 року) при запиті "Дистанційне навчання" користувач видає до одного мільйона посилань. Але велика частина цих посилань містить дидактичні матеріали за дидактичними матеріалами дистанційного навчання.

9.5. Пошук роботи

Засвоївши нову спеціальність – дистанційним або звичайним способом, саме час приступити до пошуку роботи. Пошук роботи та співробітників через Internet – це найшвидший та найзручніший спосіб підбору персоналу. Роботодавці, що розміщують оголошення про вакансію у газетах та журналах вимушені платити за рекламу, а також чекати виходу видання. Розміщення ж вакансії в Internet часто абсолютно безкоштовно, а оголошення з'являється одразу ж після розміщення.

Зараз, в Internet можна зустріти велику кількість вакансій, які не пов'язані не тільки з Internet, але і взагалі з роботою в офісі. Тому даний вид пошуку може бути зручним для всіх претендентів. Сайти, які пропонують бази вакансій та резюме, як правило, містять докладний опис роботи з вказівкою заробітних плат. Прикладом такого сайту може служити Робота.RU (www.rabota.ru).

Існують інші сайти для пошуку роботи – www.job.ru, www.rabota2000.com.ua, www.zarplata.ru, alljob.com.ua. На цих сайтах також можна зустріти зразки правильного заповнення резюме.

9.6. Робота через Internet

Фріланс (від англ. freelance – вільний художник) – робота без контракту, що виконується віддаленим співробітником. Фріланс – це форма організації праці, коли працедавець та співробітник взаємодіють через Internet.

В основному, ринок фрілансу дозволяє виробляти продукти, які можна переслати через Internet. Він стосується сфери дизайну, програмування, розробки сайтів, написання текстів, перекладів, менеджменту, консультування тощо.

У порівнянні з класичною формою організації роботи, Фріланс містить як переваги, так і недоліки. До безперечних переваг фрілансу відносять відсутність транспортної проблеми – виконавець працює вдома, вільний графік роботи, самостійна відповідальність. До недоліків фрілансу можна віднести відсутність соціального пакету, нестабільність доходів, можливість шахрайства, а також брак спілкування.

У будь-якому випадку, фріланс є. Сайти, які присвячені фрілансу, дозволяють знайти як замовника, так і виконавця (http://www.free-lance.ru/). Існують і інші сайти фрілансу, наприклад, www.weblancer.net. Зараз ринок фріланса стає все більш популярним.

Для того щоб стати справжнім «віддаленим робочим» недостатньо просто увійти в Інтернет і ввести в Яндексі словосполучення «*Віддалена робота*». Нюансів і підводних каменів тут предостатньо [14].

При пошуку роботи вас рано чи пізно попросять надіслати своє резюме. Для роботодавців це ваша візитна картка. Роль резюме дуже велика: воно складає про вас перше враження, яке в даному випадку є вирішальним. Саме від якості і змісту вашого резюме залежить, на яку кнопку в своїй поштовій

програмі натисне роботодавець: Видалити або Відповісти. Тому резюме має бути найретельнішим чином підготовлено заздалегідь, а не під час активних пошуків роботи [14].

9.6.1. Оформлення резюме

Насамперед слід обміркувати дизайн вашого резюме. Однак це не означає, що ви повинні створити чарівний барвистий шедевр, виконаний в незвичайному стилі. Якраз навпаки – в резюме не повинно бути нічого зайвого, тільки корисна інформація. Якщо необхідно щось виділити (наприклад, заголовки), то допускається змінювати розмір шрифту і його стиль (напівжирний, курсив, підкреслений), але не більше.

9.6.2. Структура резюме

Одна з основних відмінностей хорошого резюме від поганого – це лаконічність. Роботодавці – люди, як правило, зайняті, а листів, подібних вашому, їм належить прочитати не один десяток, і витрачати свій час на вивчення вашої докладної біографії вони напевно не будуть. Хороше резюме повинне займати не більше однієї сторінки друкованого тексту.

Контактна інформація. Починати резюме слід з написання своїх прізвища, імені та по батькові вгорі сторінки великими літерами з вирівнюванням по центру. Якщо ви збираєтеся відправляти своє резюме в іноземну компанію, то по батькові краще опустити.

Далі слід вказати свої координати. У першу чергу, це адреса електронної пошти. Бажано вказати контактний номер телефону. Домашню адресу повністю можна не писати, досить вказати місто проживання.

Мета. Це дуже важлива частина резюме. Роботодавець може шукати претендентів на різні посади, тому йому потрібно швидко визначити, в якості кого ви маєте намір працювати. Якщо мета буде відсутня, то резюме з великою ймовірністю може бути просто не прочитано.

У цьому розділі слід написати, на яку посаду ви претендуєте. В ідеалі формулювання посади повинна збігатися з тим, як вона заявлена роботодавцем. Ніяких розмитих фраз типу: «*Хотів би отримати роботу у відповідності за своєю освітою*» або «*Враховуючи свій досвід, сподіваюся виявитися вам корисним*» – вони звучать наївно і можуть викликати у роботодавця тільки подив. Пишіть тільки по суті, наприклад: «Мета – здобуття посади програміста, мова – C++». Не варто перераховувати кілька варіантів можливої роботи. У цьому випадку створюється враження, що людина сама не знає, чого хоче. Роботодавцю потрібен фахівець у певній галузі.

Питання, вказувати чи ні розмір очікуваної заробітньої платні, спірне. Якщо ви вважаєте себе досить конкурентоспроможними на ринку праці, то ваше побажання щодо зарплати може служити своєрідним фільтром, що

відсіває нецікаві пропозиції. В іншому випадку має сенс обговорити заробітну платню пізніше.

Освіта. У даному розділі варто вказувати тільки ті ваші заслуги, які відносяться до розглянутої посади і можуть показати рівень вашого професіоналізму. Влаштовуючись на роботу програмістом, зовсім не обов'язково писати про те, що ви вчилися в художньому технікумі або з відзнакою закінчили школу. Записи повинні бути короткими, чіткими і ясними. Назви навчальних закладів слід писати повністю, оскільки абревіатури, зрозумілі вам, найімовірніше це буде загадкою для роботодавця.

Досвід роботи. Це найважливіша частина резюме. Наявний досвід, як правило, робить вирішальний вплив на роботодавця. Тому навіть якщо ви тільки недавно закінчили навчання, постарайтеся згадати хоч що-небудь: тимчасовий підробіток, дипломну практику, тощо. При перерахуванні місць роботи можна використовувати або принцип значущості (як у випадку з освітою), або зворотну хронологічну послідовність (починаючи з останнього місця роботи, і далі в зворотному порядку).

Необхідно вказати назву компаній, в яких ви працювали, профіль їх діяльності, посади, яку ви займали, а також короткий опис ваших посадових обов'язків. Якщо за час роботи в компанії ваша посада змінювалася (особливо при підвищенні), то можна вказати це, перерахувавши посади через кому. При перерахуванні своїх обов'язків бажано обійтися без займенників. Не повинно бути виразів типу: «*Я впроваджував нову програму*». Краще написати просто: «Впровадження нової програми». Акцентувати увагу слід на тих обов'язках і функцій, які більшою мірою близькі до роботи, яку ви хочете отримати. Якщо ваш послужний список довгий, то детально варто описувати тільки останні місця роботи. Вказувати мотиви звільнень небажано – це виглядає як виправдання.

9.6.3. Про що писати не потрібно

Не слід спотворювати дані: навмисне вказувати більший досвід роботи, сильно перебільшувати свої можливості, кривити душею щодо освіти. Таємне, як правило, стає явним. У результаті і ви, і роботодавець просто даремно втратите дорогоцінний час. Не рекомендується прикріплювати до резюме свою фотографію, якщо тільки це не обумовлено роботодавцем. Для віддаленої роботи зовнішність взагалі не важлива.

9.6.4. Супровідний лист

Відправлення резюме, яке прикріпленео до порожнього повідомлення, є поганим тоном. Обов'язково має бути присутнім супровідний лист, мета якого не стільки інформувати, скільки залучити роботодавця прочитати ваше резюме, виділити вас з всіх інших претендентів.

Супровідний лист повинен бути невеликим, навіть маленьким – двох–трьох абзаців цілком достатньо. Але воно має показати роботодавцю, що саме ви найкращим чином підходите на вакантну посаду. Прочитання резюме для нього має залишитися лише формальністю. Тому підходити до написання супровідного листа варто не менше, а може бути навіть більш серйозно, ніж до створення самого резюме.

Привітання. Намагайтеся уникати звернень типу: «*Шановні панове!*» або, що ще гірше, «*Тому, кого це стосується*». Завжди, коли це можливо, звертайтеся до адресата особисто. Якщо в оголошенні не вказано ім'я, то витратьте трохи часу на його пошук, наприклад, ввівши контактну адресу електронної пошти в пошуковій системі.

Введений абзац. Тут вкажіть посаду, на яку ви претендуєте, а також як ви дізналися про вакансію. Коротко обгрунтуйте, чому ви зацікавлені в отриманні саме цього місця. Бажано показати роботодавцю обізнаність про діяльність його компанії.

Текст листа. Тут викладіть всі свої основні козирі. Опишіть досвід і навички, пов'язані з розглянутою посадою. Вкажіть тільки саму кращу інформацію про себе: найбільш вражаючі досягнення, найвдаліші проекти. Весь лист повинен бути наскрізь пронизаний позитивом.

Висловлюйтесь коротко і ясно. Уникайте вживання загальних слів, а також гарних і витіюватих фраз типу: «Враховуючи мій досвід і освіту, сподіваюся бути корисним вашій компанії для її подальшого процвітання».

9.7. Оплата через Internet

А як замовник оплачує роботу віддаленого працівника? Як можна сплатити покупку в Internet магазині? Для цього можна використовувати як Банківський переказ, так і електронні гроші.

Банківський переказ є, мабуть, найпоширенішим способом передачі грошей. Щоб мати можливість отримувати гроші за допомогою банківського переказу, необхідно сходити в найближче відділення банку і відкрити в ньому рахунок. Поцікавтеся у касира, який рахунок вам краще підійде для отримання заробітньої платні, і він вам все популярно пояснить.

Western Union – це унікальна система, що здійснює грошові перекази по всій земній кулі. З її допомогою відправити і отримати гроші можна більш ніж в 190 країнах. Вражає швидкість переказу – завдяки використанню власної всесвітньої комп'ютерної мережі гроші можна отримати вже через кілька хвилин після їх відправлення. До речі, щоб скористатися Western Union, не потрібно відкривати рахунок у банку або відвідувати якийсь певний пункт – як відправити, так і отримати гроші можна в будь-якому пункті обслуговування Western Union.

Система WebMoney Transfer (http://www.webmoney.ru) дозволяє проводити грошові розрахунки в режимі реального часу за допомогою титульних знаків WebMoney (WM – units, http://www.WebMoney.ru/). В системі підтримується

обмін декількома титульними знаками (електронні гаманці), еквівалентними реальній валюті:

WMR – рублі, RUB; WMZ – долари, USD; **WME – євро, WMU – гривні.**

Для передачі грошей використовується спеціальна програма, яка встановлюється на клієнтські машини, – WM Keeper. Переказ електронних грошей у паперові підтримується багатьма банками. Існують, також, інші платіжні системи, наприклад, Яндекс Гроші (http://money.yandex.ru/).

9.8. Заміна десктопних додатків

Проблеми використання неліцензійного програмного забезпечення стають все більш і більш актуальними в колишніх країнах СНД. Виникає дилема – платити та користуватися або не платити та відмовлятися від звичних програм. А чи можна працювати із звичним інтерфейсом, не використовуючи піратський софтвер? Не купуючи ліцензію і взагалі нічого не встановлюючи на свій комп'ютер? Відповідь – можна, якщо використовувати Rich Internet Application. RIA – в дослівному перекладі збагачений Internet додаток, який покликаний замінити звичайні, десктопні додатки. Зараз відомо декілька розробок.

Наскільки вони здатні замінити звичні Windows – програми? На це питання кожен користувач може відповісти самостійно, просто випробувавши відповідний сервіс. Документи Google (www.docs.google.com) – набір веб-додатків, які дозволяють створювати текстові документи, електронні таблиці, а також презентації. Робота у текстовому редакторі Google дуже схожа на використання програми Microsoft Word 2003. Той самий інтерфейс, звичні кнопки, знайоме меню.

Можна також спробувати веб-редактор Buzzword (http://www.buzzword.com/), який призначений для створення та редагування документів у форматі Microsoft Word. Редактор таблиць Google нагадує книгу Microsoft Excel. Розробки інших компаній також дозволяють вирішувати безліч задач через Internet. Графічний редактор Picnik (http://www.picnik.com) цілком може замінити настільний додаток для обробки фотографій.

Створення подібних веб-додатків, які здатні замінити звичайні, десктопні програми – достатньо перспективний напрям, яким зараз займаються багато компаній.

9.9. Словники

Скільки потрібно часу для того, щоб знайти переклад англійського слова в паперовому словнику? Цей час можна провести з більшою користю, якщо використовувати онлайн-словники, які дозволяють вмить набути значень слова (наприклад, **http://www.lingvo.ru/**). Більшість порталів також включають онлайн-словники. Словники Mail.Ru (http://multilex.mail.ru/), Яндекс. Словари (http://slovari.yandex.ru/) також містять безліч словників та енциклопедій.

9.10. Газети та новини. Журнали. Карти

Новини в Internet – найшвидші, найдинамічніші, найактуальніші. Неможливо випустити 12 тиражів газети за добу, але можна розміщувати нові матеріали кожні дві години на сайті новинних стрічок. Наявність зворотного зв'язку, рейтинг, що відображається у лічильниках відвідин та перегляду, лавиноподібне тиражування новин та посилань на них, величезна аудиторія – все це робить новинні публікації в Internet поза конкуренцією у порівнянні з іншими ЗМІ.

Важко виділити одну з онлайн-газет. Для прикладу, Lenta.Ru (www.lenta.ru) – типовий представник російських онлайн-газет. У тій або іншій формі новини (або посилання на них) розміщують практично всі портали та великі сайти. Мабуть, кращий спосіб вийти на новинні сайти – це набрати у пошукачеві слово "Новини". Втім, Google також надає чудові інструменти пошуку і в новинах (http://news.google.com).

Щомісячні журнали, на відміну від газет, передбачають не стільки оперативну подачу інформації, скільки опрацьований та якісний тематичний матеріал. Проте Internet і тут виконує свої певні завдання. "Мир ПК" – один з популярних російських комп'ютерних журналів (www.pcworld.ru).

Пошук адрес у незнайомому великому місті завжди був достатньо непростим завданням. Сервіс Яндекс.Карти (http://maps.yandex.ru/) призначений для спрощення цього завдання – тепер досить ввести у рядок пошуку потрібну адресу, щоб побачити її розташування на карті. Сервіс дозволяє також прокласти маршрут проїзду, проглянути цю ж карту з супутника, відобразити ландшафт місцевості. Іншими словами, паперові географічні атласи та карти залишаються у минулому.

Ми розглянули деякі ключові можливості, які надає сучасний Internet. Залишилися осторонь від нашої уваги такі сервіси як отримання прогнозу погоди (www.gismeteo.ru) або розкладу авіарейсів (www.polets.ru) тощо.

10. Що ще необхідно знати про Інтернет?

10.1. Правила хорошого тону

Американці вважають, що найвільніша країна в світі – це Америка. Україна теж вільна країна. Однак ніхто, напевно, не стане сперечатися з тим, що найвільніша держава на землі – це неозорий віртуальний світ Інтернету.

Перебуваючи в Інтернеті, ви можете спілкуватися коли завгодно і з ким завгодно, географія вже не має значення. Ваш співрозмовник може перебувати в Австралії або в Америці, швидкість обміну даними або умови зв'язку від цього ніяк не змінюються.

Право людини на інформацію та індивідуалізм – невід'ємне право кожного «громадянина Інтернету». Будь-хто з нас може сповідувати різні точки зору на ті чи інші речі, дотримуватися абсолютно різних життєвих позицій. Гоніння за погляди суворо заборонені. Створюючи власний web-ресурс, необхідно пам'ятати, що ви стаєте будівельником, одним з творців Інтернету. Публікуючи інформацію для загального відома, ви можете викладати будь-яку точку зору, важливо лише дотримуватися елементарної коректності.

Власне, навіть вашу поведінку в Мережі нікого, крім вас самих (і модераторів на деяких сторінках), не хвилює. Однак, якщо ви зустрічаєте в спілкуванні з вами ввічливе ставлення до себе, то ваші співрозмовники, напевно, мають право розраховувати на таке ж ставлення і з вашого боку. В Інтернеті немає каральних або правоохоронних інстанцій, таких як, наприклад, поліція. Все залишається на совісті самих користувачів. І якщо ви увійшли в чистий і доглянутий будинок, постарайтеся там не смітити і не писати на шпалерах маркером непристойні слова. Це в першу чергу говорить тільки про ваше виховання та особистий інтелектуальний рівень.

10.2. Приватна власність

При роботі з Інтернет, а особливо при створенні власного web-ресурсу, часто виникає ситуація, коли користувач може скопіювати собі будь-яку інформацію: програму, нову книгу, пісню улюбленої рок-групи. Тут починаються взаємини, які регулюються міжнародними законами про авторські права. Не можна, наприклад, зробивши копію статті, публікувати її в пресі або на власній web-сторінці без згоди автора, мотивуючи це тим, що, мовляв, «там-то вона лежить вільно». Звідки ви знаєте, якими саме міркуваннями керувався автор, коли виклав свій твір для загального доступу? Дана стаття чи книга – його інтелектуальна власність, і охороняється вона законом нітрохи не гірше, ніж якщо б ви придбали її в кіоску або книжковому магазині.

Багато компаній-розробників програмного забезпечення пропонують на своїх серверах скопіювати демо- або бета-версію, а може бути, і комерційну копію програми, замінену вже більш досконалим її варіантом. Ви можете

використовувати цю програму, передавати її своїм друзям. Але ви не маєте права продавати її без згоди розробника.

Якщо вам важко вирішити, чи можна зробити з отриманою інформацією те, що ви задумали, скористайтеся розташованою практично на кожному сервері посиланням на поштову адресу творця сервера і напишіть йому електронного листа, де детально викладіть своє питання. Власник сервера вишле вам відповідь, де настільки ж докладно розповість, можна це робити чи не можна.

10.3. Сервер, сайт, домашня сторінка

Зараз ми трохи більш докладно зупинимося на таких поняттях, як сервер, сайт і домашня сторінка, спробуємо розібратися, чим вони відрізняються один від одного і які компоненти повинен включати в себе кожен із зазначених підвидів web-ресурсів.

Сервером мережі Інтернет називається комп'ютер, на якому встановлена спеціальна програма (вона теж називається сервером, web-сервером або http-сервером), яка відображає web-сторінки за запитом клієнтської машини, а також виконує безліч інших корисних функцій. Коли ваш домашній комп'ютер зв'язується з сервером і отримує від нього всі необхідні дані, наприклад код web-сторінки, він виступає в ролі «клієнта», а всю систему в цьому випадку прийнято називати зв'язкою «клієнт-сервер».

Сервери можуть бути різними, причому відмінності полягають насамперед в операційній системі, під управлінням якої вони працюють. На даний час на більшості інтернетівських вузлів використовують два типи серверних програм: або Internet Information Server, який розрахований на роботу під Windows NT, або Apache, призначений для платформ, сумісних зі стандартом UNIX.

Крім відповідної програми справжній сервер повинен мати власний домен, тобто адресу DNS, що відповідає стандартам Domain Name System.

Сайт на відміну від сервера виділеної серверної програмою не володіє. Як правило, він є інтегрованою частиною сервера, каталогом на серверному комп'ютері, незважаючи на те що більшість сайтів мають власне доменне ім'я. Ще один аспект, в якому сервер і сайт значно різняться, – це їх інформаційне наповнення. Сайт (від англ, site, ділянка) – це саме ділянка сервера, тобто розділ, повністю присвячений якій-небудь одній темі. Зрозуміло, практично всі сайти включають в себе безліч підрозділів, кожен з яких може дробитися на ще більш дрібні складові. Але в будь-якому випадку всі частини сайту об'єднує якась загальна ідея, смислова спрямованість.

Домашня сторінка (homepage) у більшості випадків не має навіть власного домену, і її адреса зазвичай виглядає як http://www.domain.zooe/your_name/. Винятки становлять хіба що сторінки, що розміщені на безкоштовних серверах, адміністрація яких охоче виділяє для своїх клієнтів домени третього рівня.

За змістом домашня сторінка – це насамперед розповідь про людину, його родину, способі життя, хобі та захоплення. Заводять домашню сторіночку, як правило, з двох причин. По-перше, в процесі пошуку в Інтернет нових знайомих неодмінно виникає ситуація, коли людині необхідно розповісти співрозмовнику про себе, щоб той отримав хоч якесь уявлення про те, з ким він спілкується. По-друге, опублікувавши на головній сторінці своє резюме, ви можете значно полегшити процес пошуку роботи.

11. Як зробити сайт самому

Створення сайту, можна, звичайно, замовити фахівцям, але це обійдеться вам не дешево. Та й потім, вони будуть постійно вас «тримати на гачку» – адже сайт потрібно регулярно оновлювати, інакше від нього користі ніякої не буде!
Є два перевірених варіанта [13]:

1. Зробити сайт самому на безкоштовному хостингу.
2. Зробити сайт самому, але з невеликими вкладеннями.

Перш ніж приступити безпосередньо до формування веб-сторінок, необхідно підготувати вміст майбутнього сайту (текст, зображення та ін.) Тому будемо вважати, що вся необхідна для наповнення веб-сторінок інформація вже є. Залишилося відформатувати її, зробити зручною для читання, вставити ілюстрації і прикрасити, тобто привести в належний вигляд. Все це і багато іншого вміє найважливіший у Всесвітній павутині інструмент – HTML.

Існує безліч різноманітних редакторів для створення HTML-сторінок. Всіх їх можна розділити на дві основні категорії.

1. Програми, що працюють за методом WYSIWYG (What You See Is What You Get) – що бачиш, те й отримаєш. Вони автоматично вставляють всі необхідні службові слова (теги) у вихідний файл. За допомогою даних програм можна створити непоганий сайт, не маючи жодного уявлення про мову HTML. Створення сайту в даному випадку нагадує збірку конструктора – все просто, наочно і інтуїтивно зрозуміло.
2. Звичайні текстові редактори (зокрема, стандартний Блокнот). В даному випадку не обійтись без знания синтаксису мови HTML. Робота буде рутинною, яка нагадуе програмування.

11.1. Зробити сайт самому на безкоштовному хостингу

Серед багатьох варіантів безкоштовних хостингів виділимо два кращих, які пропонують безкоштовно місце під сайт і конструктор для створення сайту.

uCoz. Один з відомих сервісів, який надає можливість безкоштовного розміщення сайту – це uCoz. На цьому сайті дозволяють розміщувати свою рекламу. Багато хто вважає цей конструктор сайтів кращим серед безкоштовних сервісів. Ви можете перейти на сторінку створення свого аккаунта на http://www.ucoz.ru і тут же приступити до створення свого першого сайту.

Narod.yandex.ru. Інший сервіс за безкоштовним створенню сайтів, на який варто звернути увагу, – narod.yandex.ru. На цьому сервісі менше варіантів дизайну сайту, ніж на uCoz. Але, великий плюс сервісу в тому, що це дітище Яндекса, найбільшого пошуковика Рунета. Яндекс швидко знаходить і індексує «народні» сайти. Навіщо їх шукати, коли вони у нього на серверах знаходяться. Щоб почати створювати сайт, потрібно, спочатку, зареєструватися на Яндексі,

завести пошту і реєструвати собі поштову скриньку. Після цього ви отримаєте доступ до всіх сервісів Яндекса, в тому числі і на «Народ».

Мінуси всіх безкоштовних хостингів. Якщо сайт створюється для того, щоб розповісти про себе, культурі та мистецтві, то сайти на безкоштовних хостингах цілком згодяться. Але робити на безкоштовному хостингу сайт, за допомогою якого ви збираєтеся продавати товари, будувати бізнес, несерйозно.

Заробляють ці сервіси на рекламі, яка буде висіти вгорі вашого сайту постійно.

Багато й інших мінусів в безкоштовних хостингах. Ви не можете забрати сайт на інший хостинг. Якщо ви і встановіть свою рекламу, то на кращих місцях буде реклама власників хостингу, а не ваша. Тому краще зробити невеликі вкладення, щоб бути повноправним господарем свого сайту і ні від кого не залежати.

11.2. Зробити сайт самому, але з невеликими вкладеннями

Чому сайт потрібно робити самому? Майбутні власники сайту зазвичай впадають в дві крайності.

1. Замовляють сайт за безцінь, наприклад, студенту. При цьому іноді виходять за дизайном непогані сайти. Але через кілька місяців власник помічає, що на сайт ніхто не заходить. А справа в тому, що для створення дизайну сайту досвід не обов'язковий. З першого разу може вийти гарний сайт. А от зробити правильну оптимізацію сайту і просувати його в пошукових системах без досвіду ну, ніяк не вийде. Потренувавшись на вашому сайті, за ваші гроші, студент за пару років, можливо, набереться досвіду в розкручуванні сайтів. Але тоді він буде брати вже інші гроші за створення та розкрутку сайтів. То чи не краще відразу самому зайнятися сайтом, а не платити гроші будь-кому.

2. Інша крайність – замовляють сайт у «крутій» фірмі вже за пристойні гроші. Додавання нових сторінок і сервісів на сайт в таких фірмах йде за окремим тарифом і варто також солідних грошей. Ну, а розкрутка сайту у них буде коштувати шалені (за нашими з вами поняттями) гроші.

Як самому встановити CMS (движок) на сервері. Вибираючи черговий раз хостинг, проаналізувавши якість послуг та рейтинг провайдерів хостингу, багато хто зупиняється на Beget. Ця хостінг компанія надає безкоштовний тестовий період на цілий місяць. Більшість компаній надають 3-10 днів.

На хостингу Beget предвстановлені на сервері «движки» сайтів (CMS – Content Management System – Система управління сайтом, движок для сайту). Вибравши потрібний CMS, ви можете натисненням однієї кнопки встановити, наприклад, WordPress (для блога), PHP BB (для форуму), Joomla для універсального сайту.

Немає проблем ні з правами на папки, ні з скриптами, ні з розширеннями. Залишилося тільки зайти в адміністраторський режим сайту, змінити стандартний шаблон на більш красивий і зайнятися наповненням сайту матеріалами.

Матеріали-зразки на сайті можна швидко замінити на свої. Шаблонів для Joomla в Інтернеті безліч – зробіть дизайн, який вам подобатися. Якщо базовій комплектації вам виявилося недостатньо, то є сотні різних розширень, модулів, плагінів. Ви можете завантажити на сайт красиву фотогалерею, каталог, форум, інтернет-магазин.

Якщо ви зупините свій вибір на хостингу Beget, то самий підходящий тариф для початківців вебмайстрів – Start вартістю приблизно 32 грн на місяць, або 300 грн на рік.

Як швидко навчитися робити сайти на CMS Joomla. Один з найпопулярніших CMS (движків) – Joomla. На Joomla можна зробити сайт практично будь-якого рівня. Але, щоб вивчити всі можливості движка, потрібно багато часу. Коли ви почнете працювати над сайтом, то переконаєтеся в цьому.

Найкраще вивчати створення сайтів по відеоуроку. Вивчення по книгах – заняття важке, хоча б як, наприклад, з книги набирати HTML-код. А з матеріалів, які додаються до відеоуроку, взяв – скопіював і вставив. Тому для тих, хто хоче якомога швидше створити свій бізнес-сайт, відеоуроки з Joomla просто необхідні.

11.3. Розкрутка сайту

Після того как ви створите свій сайт необхідно зробити його доступним широкому загалуи. Щоб зробити сайт по-справжньому доступним, необхідно рассказать о нем усім і кожному. В мережі існують спеціальні механізми, які дозволяють інформувати користувачів про ваш сайт. І від того, як вы ними володієте, залежить кількість ваших відвідувачів [14].

11.3.1. Пошукові системи

Реєстрація в пошукових системах – це найважливіший крок при розкручуванні свого сайту, оскільки переважна більшість користувачів Інтернету шукають інформацію саме за допомогою пошукових сайтів. Однак бездумна реєстрація не сильно підвищить відвідуваність вашого ресурсу, якщо пошукова машина буде ставити посилання на ваш твір де-небудь в кінці списку знайдених сайтів, то рідкісний користувач клацне на цьому посиланні. Основна увага найчастіше приділяється 10-20 ресурсам, які пошукова система пропонує на перших сторінках результату пошуку. Ваше завдання – потрапити в цю лідируючу групу.

Вибір ключових слів. Ключові слова – це ті слова, які ваші потенційні відвідувачі будуть вводити в рядок пошуку. Керуючись саме цими словами, пошукова система організовує результуючий список сайтів, який буде представлений користувачеві відповідно до алгоритму ранжирування.

Щоб потрапити у даний список, необхідно обов'язкову присутність цих слів у вашому документі. Найбільш ходові з них повинні бути у заголовку.

Визначити найбільш поширені запити користувачів, звернені до теми вашого сайту, нескладно. Популярні пошукові машини надають таку послугу. Буде не зайвим вивчити сайти конкурентів, що займають верхні рядки пошукових списків. Дізнайтеся, як вони домоглися таких успіхів і які ключові слова вони використовують, зверніть увагу на розташування цих слів у документах і їх щільність.

Особливості реєстрації. Щоб пошукова машина могла знайти ваш сайт, його необхідно зареєструвати. Причому реєстрацію доведеться проходити на всіх популярних пошукових серверах (якщо, звичайно, ви хочете, щоб їх користувачі могли вас знайти). Хоча є альтернатива: існують спеціальні програми, які автоматично реєструють ваш ресурс в сотнях різних пошукових системах. Але повністю довіряти таким програмам не варто, і в десятці найпопулярніших пошукових систем краще все-таки внести дані про свій сайт вручну.

При реєстрації пошукові системи запитують тільки одну сторінку вашого сайту – головну. Далі пошуковий робот буде просуватися за вказаними на ній посиланнях і таким чином омине весь сайт. Якщо у вас є веб-сторінки, на які немає посилань в інших документах, то їх доведеться реєструвати окремо.

Індекс цитування. Індекс цитування – це кількість посилань на ваш ресурс, розташованих на інших сайтах. Дуже часто індекс цитування застосовується пошуковими системами при ранжуванні ресурсів і є в цьому випадку, як правило, одним із пріоритетних параметрів. Логіка наступна: якщо на сайт веде безліч посилань, значить його вміст становить велику цінність.

Однак кількість посилань – це ще не все. Важливо ще й їх якість, тобто значімость того сайту, на якому розташоване посилання. На якість посилань впливає кілька факторів. Одним з них є тематична схожість цього сайту з вашим. Іншим важливим фактором є авторитетність сайту, на якому знаходиться посилання.

Для штучного збільшення індексу цитування можна обмінюватися посиланнями з іншими сайтами, реєструватися в різних каталогах і дошках оголошень, залишати разом з контактними даними адресу свого сайту у всіляких форумах тощо.

Довідатися, які сайти посилаються на ваш ресурс, дуже просто. Для цього достатньо ввести адресу свого ресурсу в рядок пошуку пошукової машини і натиснути кнопку Знайти.

11.3.2. Інші способи розкрутки

Розкрутка в пошукових системах дуже важлива, проте не варто забувати, що на сайти потрапляють не тільки з їх допомогою. В Інтернеті існують і інші способи заманювання відвідувачів на ваш сайт, і було б дуже неправильно нехтувати ними.

Каталоги. Каталог – це структурована колекція посилань, розбита на розділи і підрозділи. Щоб в одному з цих підрозділів перебувало посилання і на

ваш сайт, останній слід зареєструвати в даному каталозі. Реєстрація, як правило, безкоштовна. Від реєстрації в пошукових системах вона відрізняється тим, що перед включенням вашого ресурсу в один з розділів його обов'язково перевірять власники каталогу на предмет схожості з зазначеною тематикою і заявленим описом.

Внесення сайту в каталог не дає настільки сильною віддачі, як реєстрація в пошуковій системі, однак якщо посилання на ваш сайт буде не в одному, а в 10 або 100 каталогах, то приплив відвідувачів буде більш ніж істотним. Крім усього іншого, збільшується і індекс цитування вашого сайту.

Дошки оголошень. Дошка оголошень – це розділ сайту (а іноді й цілий сайт), присвячений оголошенням відвідувачів і дозволяє як читати, так і розміщувати повідомлення. Як правило, дошки оголошень структуровані на розділи і підрозділи. Їх використання в переважній більшості випадків є безкоштовним.

Існують спеціальні програми, що дозволяють автоматично розміщувати створене одного разу оголошення на сотнях різноманітних дощок. Якщо оголошення, яке рекламує ваш сайт, дійсно варте і доречне, то кількість ваших відвідувачів різко зросте. Крім того, пропорційно кількості оголошень зросте і індекс цитування, що також важливо. Однак хвиля збільшення індексу цитування і приросту відвідувачів скоро спаде, оскільки оголошення швидко застарівають. Щоб закріпити успіх, їх необхідно періодично оновлювати.

Обмін посиланнями. Тут логіка надзвичайно проста: чим більше посилань веде на ваш сайт, тим більше вірогідність того, що будь-який користувач клацне на одній з них і потрапить до вас у гості. Найпростіший спосіб розміщення посилань на інших сайтах – обмін. Для цього достатньо знайти інтернет-ресурс спорідненої тематики, зв'язатися з його автором і запропонувати обмінятися посиланнями. Як правило, власник ресурсу з радістю погоджується, оскільки обмін в даному випадку взаємовигідний.

Обмінюючись посиланнями, ви також підвищуєте свій індекс цитування, оскільки кількість сайтів з посиланнями на ваш ресурс збільшується.

Статті. Написання якісних статей є одним з кращих способів розкручування сайту. По-перше, при цьому збільшується інформаційне наповнення вашого ресурсу, отже, підвищується ймовірність знаходження сайту через пошукові системи. По-друге, якщо якому-нибудь користувачеві сподобається ваша стаття, і він вирішить розмістити її на сторінках свого сайту, то в кінці тексту він зобов'язаний буде поставити посилання на джерело, тобто на вас. Це, в свою чергу, веде до збільшення популярності та індексу цитування вашого ресурсу.

11.3.3. Платні послуги

Описані вище способи розкрутки сайтів безкоштовні. Непрямі витрати, звичайно, є: оплата часу роботи в Інтернеті, витраченого на розкрутку сайту, а також витрата особистого часу, який теж коштує недешево. Але все-таки

безпосередньо за послуги розкрутки ви нікому не платите. Проте, як завжди, є альтернативний варіант.

Розкрутка за гроші. Якщо у вас немає часу, але є гроші, то розкрутку можна доручити користувачам, що займаються цим професійно. Існує безліч сайтів, що пропонують послуги по збільшенню кількості відвідувачів інтернет-ресурсів. Знайти їх нескладно: достатньо ввести в будь-якій пошуковій системі слова «розкрутка сайту» – і вам буде надано довгий список подібних інтернетконтор.

Банерна реклама. Показами банерів можна не тільки обмінюватися, їх також можна купувати. Вартість показів різна і залежить від того, на яких сайтах демонструватиметься ваш банер, якого він буде розміру тощо. У загальному випадку, приблизна вартість 1000 показів дорівнює $0,5-5.

Існує також контекстна реклама – розміщення банерів на сторінках сайтів відповідної тематики. Подібні послуги часто надають каталоги та пошукові системи.

12. Для чого потрібен торрент?

Інтернет створювався для обміну інформацією. Для звичайних користувачів обмін інформацією полягав в обміні текстовими повідомленнями. Так почали створюватися сайти, потім почали додаватися картинки для барвистості і наочності. Технології не стояли на місці. З кожним роком швидкість обміну даними зростала, і врешті-решт люди задумалися про обмін файлами. З цим завданням добре справлялися FTP-сервери. Але й цього виявилося замало. FTP-сервери не могли зберігати нескінченно великі обсяги інформації. Так 4 квітня 2001 року програмістом Бремом Коеном був розроблений перший torrent-клієнт BiToггепt.

12.1. Що таке торрент-клієнт?

Торрент-клієнт (програма Торрент) – це програма, яка працює за протоколом P2P, що дозволяє користувачам обмінюватися файлами через Інтернет. Існує кілька різних torrent-клієнтів, але найбільш відомими і поширеними є uTorrent і BitTorrent.

Основним недоліком мережі BitTorrent є неможливість пошуку файлу в даній мережі за його назвою та неможливість спілкування користувачів між собою.

Для скачування необхідний спеціальний торрент-файл, в якому у зашифрованому вигляді зберігається унікальна інформація про даний файл. Даний торрент-файл створюється для кожного окремого файлу або папки і не може застосовуватися для скачування подібні фалів.

12.2. Як працює торрент?

Перед початком скачування файлу торрент-клієнт підключається до трекера, адреса якого прописана в торрент-файлі, і повідомляє йому свою адресу, а також контрольну суму файлу. У відповідь трекер повідомляє йому адреси інших клієнтів, які викачують або роздають цей же файл. Надалі торрент-клієнт періодично інформує трекер про хід процесу скачування і отримує оновлений список адрес. Цей процес називається оголошенням (англ. announce). Якщо ви зареєстровані на трекері і завантажили торрент-файл під своїм логіном (увійшли на сайт, використовуючи логін і пароль), то в статистиці вашого облікового запису після анонсів буде змінюватися статистична інформація про кількість викачаних і відданих мегабайт.

Клієнти, використовуючи отриманий список, починають обмінюватися між собою сегментами файлу без участі трекера. Порядок обміну вибирається так, щоб клієнти обмінювалися спочатку найбільш рідкісними сегментами файлу. Для більш стабільної роботи необхідно, щоб якомога більше клієнтів брало участь у роздачі і не відключалися після повного скачування. Але для

скачування файлу зовсім необов'язково наявність у повному обсязі файлу у більшості учасників. Завантаживши файл або сегмент файлу, клієнт одразу може приступити до його роздачі, таким чином досягається більш ефективне використання каналу віддачі.

При роздачі рідкісних файлів іноді бувають такі ситуації, коли клієнт, який має в повному обсязі єдиний файл, пішов з роздачі, не встигнувши передати всі сегменти різних учасників. Тоді, обмінявшись між собою всіма наявними сегментами, клієнти не можуть завершити скачування через відсутність у них деяких сегментів файлу. У цьому випадку залишається тільки чекати надалі підключення до роздачі клієнта, який має файл у повному обсязі. У графі бенкети торрент-екрану перераховані всі активні учасники роздачі з інформацією про IP-адреси.

12.3. Що таке торрент-трекер

Торрент-трекер – це спеціалізований сервер (сайт в Інтернеті), який працює по протоколу HTTP. У більшості випадків ці сайти виконані у вигляді форумів, мають зручний інтерфейс, розширений пошук і обширний опис наявних роздач. На торрент-трекерах ви можете завантажити торрент-файл цікавих вам роздач, а також самі можете створити роздачу наявних у вас файлів.

Постійно поповнюючи бази загальнодоступних файлів, торрент-трекери набрали більшу популярність. Довгий час більшість людей взагалі не знали про їх існування, а якщо чули, то не знали, як звідти що-небудь завантажити. Доступ до торрент-трекера був обмежений, на деяких реєстрація велася тільки за запрошенням. Вівся постійний контроль завантаженого і відданого трафіку, адже хтось повинен був все-таки «роздавати» файли.

По мірі зменшення вартості завантаження файлів з файлообмінників і збільшення швидкості Інтернету користувачів на торрент-трекерах приймалися різні заходи по збільшенню відвідуваності і залученню постійних користувачів, які бажають завантажити файл з торрентів. Підсумком всієї цієї роботи стала можливість скачати через торрент безкоштовно і без реєстрації будь-які доступні файли на максимально доступній у вашій мережі швидкості.

Список літератури

1. Леонтьев В.П. «Новейшая энциклопедия персонального компьютера 2001». – М.: «ОЛМА – ПРЕСС», 2001. – 848 с.
2. Левин В.И. История информационных технологий. Интернет университет. http://www.intuit.ru.
3. Закарян И., Рафалович В. Что такое Internet, WWW и HTML. – М.: Энергия, 1998. – 258 с.
4. Д. Ликляйдер – Великий идеолог Internet. http://internetua.com/.

5. Кон А. Секреты Интернета. Серия "Учебный курс". – Ростов н/Д: "Феликс", 2000. –320 с.
6. Коровченко Э.С. Энциклопедия Internet 2004. – М.: "Новый издательский дом", 2004. – 752 с.
7. Копыл В. И. Общение в Интернете. Простой и быстрый курс для самостоятельного изучения. Серия: Какие кнопки нажимать. Издательства: АСТ, Харвест, 2005. – 48 с.
8. Копыл В. И. Поиск в Интернете. Простой и быстрый курс для самостоятельного изучения. Серия: Какие кнопки нажимать. Издательства: АСТ, Харвест, 2005. – 64 с.
9. Т. В. Козлова. Интернет. Как эффективно пользоваться поисковиками. Серия: Спрашивали – отвечаем. Издательство: НТ Пресс, 2006. – 144 с.
10. Иванов В. Интернет для начинающих. Самоучитель. Издательства: Питер, Издательская группа BHV, 2005. –240 с.
11. Левин Дж, Левин-Янг М., Бароди К. Интернет для "чайников".Издательство: Вильямс, 2009. – 352 с.
12. Лапунов А. В., Ульянов О. В., Прокди Р. Г. Интернет. Самоучитель. Издательство: Наука и техника, 2010. – 224 с.
13. Работаем Интернете. Компьютер для начинающих, №8(32), 2012.
14. Александров Е. Л. Интернет – легко и просто! Популярный самоучитель. – СПб.: Питер, 2005. – 208 с.

ЧАСТИНА ІІІ. ОСНОВИ ІНФОРМАТИКИ

Вступ

Протягом кількох останніх десятиліть 20 століття та початку 21 століття інформатика – дослідження автоматичної обробки інформації та, зокрема, проблем, які пов'язані з принципами та застосуваннями ЕОМ, – більш-менш сформувалася у самостійну науку, яка об'єднує декілька більш менш незалежних дисциплін. Одна з таких дисциплін – програмування – стала предметом численних систематичних досліджень та досягла певного ступеню формалізації.

Розрив між останніми досягненнями інформатики та повсякденною практикою програмування, між складністю задач програмування та примітивністю доступного арсеналу засобів, між запаморочливим прогресом устаткування та інертністю програмного забезпечення, мабуть, зростає з кожним роком. Визнаючи кризовий стан практики програмування та демонструючи широкий спектр думок з приводу актуальної проблематики, вчені дійшли до висновку, що подолати його можна лише шляхом ретельно підготовленого загального та інтенсивного курсу навчання, який охоплює не тільки сукупність технічних прийомів, але й у певному сенсі формує саму особу програміста.

Особливої важливості цей підхід набуває у зв'язку з можливостями персональних обчислювальних засобів, що з'явилися, сенсаційним падінням цін на комп'ютери, могутнім вторгненням їх у будь-яку сферу людської діяльності.

Вже перше покоління ЕОМ дало людству новий інструмент пізнання навколишньої дійсності. Стало зрозуміло, що з часом ЕОМ стануть робочим інструментом людей багатьох спеціальностей. Проте для цього було необхідно, щоб обчислювальні машини були дешеві, надійні, доступні для безпосереднього використання не тільки програмістами-професіоналами, але й фахівцями з інших областей. У 50-60-і роки минулого століття рівень розвитку обчислювальної техніки ще не дозволяв забезпечити всі ці умови. ЕОМ були дуже громіздкими, коштували дорого, вимагали для експлуатації великих спеціально обладнаних приміщень, роботу машини забезпечували десятки операторів та інженерів.

В середині 70-х років у розвитку обчислювальної техніки почався якісно новий етап, який пов'язаний зі справді революційними науково-технічними досягненнями в області фізики твердого тіла. В результаті цих досягнень були створені та технологічно засвоєні великі та надвеликі (у надмалих вимірюваннях) інтегральні схеми, які забезпечують масове виробництво надійних мікромініатюрних вузлів, блоків та окремих пристроїв, зокрема мікропроцесорів як технічної бази нового покоління ЕОМ. Сьогодні застосування мікропроцесорної техніки у багато разів понизило вартість та

зменшило розміри обчислювальних пристроїв, підвищило надійність та спростило умови експлуатації ЕОМ.

Значно вдосконалилися технічні засоби спілкування людини з машиною, подальший розвиток отримали мови програмування, з'явилося безліч пакетів прикладних програм, які були розроблені програмістами-професіоналами для використання їх як іншими програмістами, так і непрофесійними користувачами ЕОМ. Все це створило умови для широкого впровадження обчислювальної техніки у виробництво та інші галузі народного господарства.

Зараз можна з упевненістю сказати, що знання основ інформатики та обчислювальної техніки стали необхідними кожній людині як у переважній більшості областей професійної діяльності, так і у побуті, тобто стали по-справжньому загальноосвітніми.

1. Загальні відомості про комп’ютер

Цей розділ – вступний. Він починається із знайомства з основними принципами «традиційної» архітектури комп’ютера, яка була запропонована фон Нейманом. Далі ми познайомимося з алгоритмами та основними алгоритмічними структурами. Розглянемо подання чисел у різних системах числення та особливості машинної арифметики. Завершальна частина розділу присвячена загальним принципам мов програмування, їхній класифікації, короткій історії створення та розвитку.

1.1. Як влаштований комп’ютер

Сказати, що комп’ютер – це дуже просто, буде дуже гучно, але що це не так вже і складно – цілком справедливо. Такому обнадійливому твердженню можуть не повірити тільки:

- ті, хто в очі не бачив комп’ютера;
- ті, хто купив собі комп’ютер, привіз додому, включив, ну й натужливо дивиться на незрозумілі слова та картинки, не знаючи, що з ними робити;
- нарешті, ті, кому показали, як запускати яку-небудь гру або потрібну для роботи програму, але все решта для них – темний ліс.

Звичайно, прочитавши цей розділ було б наївно розраховувати придбати бездоганні знання для роботи на комп’ютері. На цю тему написані окремі та досить-таки товстенькі книжки, в яких є все-все-все. Але, як з’ясувалося, все-все-все насправді мало кому потрібно. Мова йтиме не про ті хитрощі та тонкощі, що гріють душу професіонала або естета від комп’ютерної технології, але лише про те, що може стати у нагоді практично на перших порах, особливо початківцю на персональному комп’ютері.

Що таке комп’ютер, пояснювати особливо нікому не треба. Це така річ, щоб грати в ігри, набирати, красиво-красиво оформлювати та роздруковувати будь-які тексти, бланки, договори, вести бухоблік або архів, управляти фінансовими ресурсами банка або страхової компанії, робити наукові або інженерні розрахунки, ставити діагнози хворим та здоровим, малювати картинки та мультики, записувати та виконувати музику, обмінюватися листами по електронній пошті, вивчати будь-яку науку або іноземні мови тощо.

Не треба тільки плутати комп’ютер з ігровими приставками до телевізора типу Play Station, Dendy або Sega. Ці іграшки на справжні комп’ютери схожі тільки тим, що на них теж можна грати в ігри. На цьому схожість і закінчується.

З’являються також інші «майже комп’ютери», як, наприклад, персональні органайзери. Іноді це досить розумні та вмілі пристрої (особливо коли органайзер об’єднаний з мобільним телефоном так, що має можливість доступу до Internet), але все ж таки не настільки, щоб порівнювати їх з повноцінним комп’ютером. Тому про них ми тут говорити теж не будемо.

Комп’ютери бувають настільні (їх ще називають – на американо-інтернаціональному жаргоні – **десктоп**) та мініатюрні (**ноутбук**, тобто дослівно «записник»). А крім того, є великі (робочі станції), дуже великі, надпотужні та надшвидкісні, яких ми з вами, може, і в очі не бачили.

Є також порівняно новий тип комп’ютера, як **кишенькові комп’ютери** (palmtop – «на долоні»). Вони простіші великих комп’ютерів та позбавлені деяких важливих можливостей. Але вони дійсно кишенькові – на відміну від ноутбуків (такий «записник» не в кожен портфель вкладеш!).

Для того щоб водити автомобіль, зовсім не обов’язково знати, як він влаштований. Але щоб бути хорошим водієм, устрій та взаємодію окремих частин автомобіля уявляти все ж таки необхідно. Те саме можна сказати і про комп’ютер: можна на ньому працювати, використовуючи готові програми, та навіть самому програмувати, не знаючи внутрішнього устрою комп’ютера. Проте кваліфікованим користувачем або програмістом не станеш, не вивчивши пристрій та взаємодію окремих його частин.

В основу побудови переважної більшості комп’ютерів покладені такі загальні принципи, які були сформульовані у 1945 р. американським вченим Джоном фон Нейманом.

1. Принцип програмного управління. Програма складається з набору команд, які виконуються процесором автоматично у певній послідовності.

Вибірка кодів команд програми з пам’яті здійснюється за допомогою лічильника команд. Це спеціальний регістр процесора, який послідовно збільшує адресу поточної команди на довжину цієї команди після її виконання, та зберігає у ньому значення цієї адреси. А оскільки команди програми розташовані у пам’яті одна за одною, то тим самим організовується вибірка низки команд з послідовно розташованих елементів пам’яті. Якщо ж після виконання чергової команди необхідно перейти не до наступної команди, а до якоїсь іншої, то використовуються команди умовного або безумовного переходу, які змінюють природний порядок виконання команд та заносять у лічильник команд адресу елементу пам’яті, який містить код наступної команди. Вибірка команд з пам’яті припиняється після досягнення та виконання команди «стоп». Таким чином, процесор виконує програму автоматично, без втручання людини.

2. Принцип однорідності пам’яті. Програми та дані зберігаються в одній і тій самій пам’яті, тому комп’ютер не розрізняє, що зберігається у даному елементі пам’яті – число, текст або код команди. Над командами можна виконувати такі самі дії, як і над даними.

Більш того, команди однієї програми можуть бути отримані як результати виконання іншої програми. На цьому принципі засновані методи трансляції – перекладу тексту програми з мови програмування високого рівня на мову конкретної машини.

3. Принцип адресності. Структурно основна пам’ять складається з перенумерованих комірок. Процесору у будь-який момент часу доступна будь-яка комірка пам’яті.

Звідси з'являється можливість давати символічні імена областям пам'яті так, щоб до значень, які в них зберігаються (адрес), можна було згодом звертатися або міняти їх у процесі виконання програм з використанням наданих імен.

Комп'ютери, які побудовані на перерахованих принципах, відносяться до типу **фон-неймановських**. Але існують комп'ютери, які принципово відрізняються від фон-неймановських. Для них, наприклад, може не виконуватися принцип програмного управління, тобто вони можуть працювати без лічильника команд, який вказує на поточну команду програми, що виконується. Для звернення до будь-якої змінної, яка зберігається у пам'яті, цим комп'ютерам не обов'язково давати їй ім'я. Такі комп'ютери називаються не фон-неймановськими.

Вам частіше за усе доведеться працювати на так званому **персональному комп'ютері** (ПК). Щоб легко зрозуміти взаємодію різних частин комп'ютера, уявимо собі його структуру схематично (рис. 1.1). На схемі зображено тільки сім найнеобхідніших пристроїв та системна **шина**, хоча у комп'ютері їх може бути значно більше. Найпомітніша частина комп'ютера, настільки помітна, що її навіть під стіл найчастіше не ховають, – це екран, він же дисплей або **монітор**.

Уся найважливіша начинка міститься всередині корпусу – **системному блоці**, – поряд з яким стоїть ваш монітор. У системному блоці знаходяться: **процесор** – те «залізо», яке вміє обчислювати; **пам'ять** (memory; часто говорять «оперативна пам'ять» або просто пам'ять) – те місце, де зберігається вся інформація, яка необхідна для роботи програм прямо зараз, коли ви працюєте (при виключенні комп'ютера вся інформація у пам'яті зникає); жорсткий диск, або **вінчестер** (HDD – hard disk drive), де знаходиться вся інформація, яка на даний момент не використовується, але яку можна довгий час зберігати – навіть після виключення комп'ютера.

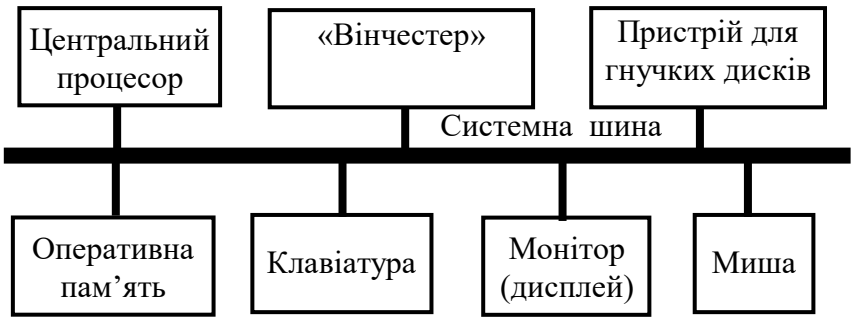


Рис. 1.1. Схематичний устрій персонального

Додатково до цього, у комп'ютера є клавіатура та миша, дисковод (floppy disk drive, FDD-привід) – пристрої для роботи з гнучкими дисками (дискетами) (floppy disk), дисковод для компакт-дисків (CD-ROM drive). Є також кілька так званих **портів** (port) – роз'ємів для під'єднання до комп'ютера додаткових пристроїв: **принтера** (для друку), **модему** (для зв'язку з іншими комп'ютерами,

зокрема через Internet, за допомогою телефонної мережі), **сканера** (для введення зображень у комп'ютер) і т.і.

Не дивлячись на різний зовнішній вигляд, у внутрішньому устрої сучасних комп'ютерів різних типів багато спільного. У загальному випадку до складу системного блоку входять усі основні вузли комп'ютера, багато з яких ми розглянемо окремо:

- системна плата;
- блок живлення;
- накопичувач на жорсткому магнітному диску;
- накопичувач на гнучкому магнітному диску;
- накопичувач на оптичному диску;
- роз'єми для додаткових пристроїв.

На системній (материнській) платі у свою чергу розміщуються:

- мікропроцесор;
- генератор тактових імпульсів;
- мікросхеми пам'яті;
- контролери зовнішніх пристроїв;
- звукова карта та відеокарта;
- таймер.

1.2. Архітектура комп'ютера

Архітектурою комп'ютера називають опис основних компонентів комп'ютера та способів їх взаємодії. Архітектура – це ті властивості та характеристики обчислювальної системи, які є «видимі» програмістові. Серед них – набір команд, розрядність машинного слова, механізми введення/виведення і т. і.

Центральний процесор та оперативна пам'ять утворюють найважливішу частину комп'ютера. Швидкість роботи комп'ютера визначається у першу чергу швидкодією центрального процесора. Зауважимо, що в спеціалізованих комп'ютерах програмна частина може бути реалізована «апаратно», тобто за допомогою набору мікросхем, що мають спеціальний пристрій. У цьому випадку комп'ютер може виконувати тільки одну програму.

Зовнішню пам'ять та пристрої введення/виведення називають «***периферією***» (периферійні пристрої). Під час роботи комп'ютера всі пристрої обмінюються між собою інформацією. Щоб інформація могла потрапляти з одного пристрою в інший, усі пристрої сполучені між собою загальною ***системною шиною***, яка являє собою набір керованих електричних провідників, по яким можуть передаватися певні сигнали.

Арифметичні операції виконуються над цілими та дійсними числами, для подання яких використовується двійковий формат. Для цілих від'ємних значень використовується додатковий код. Він дозволяє спростити реалізацію

арифметичних операцій. Докладніше про способи подання чисел йтиметься далі.

Команди також зберігаються у двійковому форматі. Подивившись на послідовність двійкових розрядів машинного слова, неможливо «очима» визначити, що це – значення або команда. Визначити це може лише пристрій управління. Щоб доповнити наше уявлення про комп'ютер та його реальні можливості, залишається уточнити природу горезвісних команд – "**базових операцій**" центрального процесора, тих, з яких можна складати програми. Можна розподілити ці операції, які звуться ще операторами "**машинної мови**", на чотири великі категорії, які слідують зі структури ЕОМ. Нижче показується, що команда описується постійною частиною, "**кодом**", та змінною, "**адресною**" частиною. Розрізняють:

1. **Обчислювальні команди,** або команди обробки, що виконуються арифметико-логічним пристроєм над величинами, що зазвичай містяться у загальних регістрах.

2. **Команди обміну** інформацією між пам'яттю та арифметичним пристроєм.

3. **Команди розгалуження.** Зазвичай команди виконуються одна за одною у порядку збільшення адрес. Команда розгалуження дозволяє дізнатися, що якщо виконується деяка умова, то наступна команда, всупереч звичайному порядку, повинна бути знайдена за адресою, яка задана в адресній частині команди, інакше виконуватимиться наступна команда.

4. **Команди введення та виведення,** забезпечують обмін інформацією між оперативною пам'яттю та периферійними пристроями.

Крім наведених груп команд в пізних комп'ютерах можуть бути деякі специфічні групи команд.

1.3. Монітори

Монітор є необхідним пристроєм виведення інформації. Монітор (або дисплей) дозволяє вивести на екран алфавітно-цифрову або графічну інформацію у зручному вигляді для читання та контролю користувачем. Відповідно до цього, існує два режими роботи: текстовий та графічний. У текстовому режимі екран представлений у вигляді рядків та стовпчиків. У графічному форматі параметри екрану задаються кількістю точок по горизонталі та кількістю точкових рядків по вертикалі. Кількість горизонтальних та вертикальних ліній екрану називається **роздільною здатністю**. Чим вона більше, тим більше інформації можна відображати на одиниці площі екрану.

Екрани моніторів бувають звичайного розміру (14 та 15 дюймів по діагоналі), збільшені (17–19) та великі, як телевізор (20, 21 та навіть 24 дюйми). Кольорові (від 4–8–16 до декількох мільонів кольорів) та монохромні (тобто двобарвні, рахуючи й колір фону: чорно-білі, чорно-жовті, чорно-зелені). Діляться вони ще і в залежності від своєї роздільної здатності, тобто

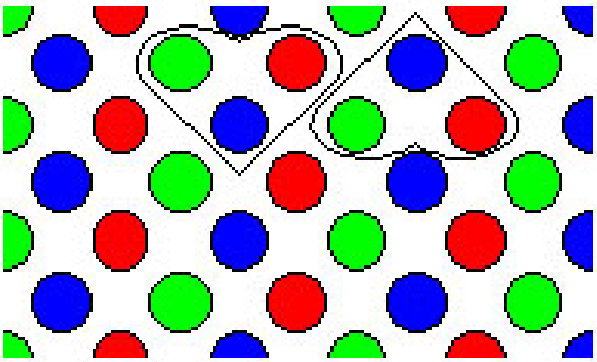
наскільки вони вам дозволяють насолоджуватися високою якістю зображення на них: чим більше крапок на екрані може намалювати ваш комп'ютер, тим тонше та точніше будуть малюнки та розбірливішим текст. Відрізняються екрани також за розміром зерна (чим зерно дрібніше, тим вище якість зображення), за ступенем захисту від електромагнітного випромінювання.

Особливо гарні плоскі монітори. Вони менше спотворюють форму зображення по краях та у кутах, значно менше блікують, не так сильно стомлюють очі.

Монітор – дуже важлива частина комп'ютера, приблизно, як колеса для автомобіля: на цьому не економлять. Перші монітори з'явилися лише в середині 70-их років XX століття.

Переважна більшість моніторів до 2007 року була сконструйована на базі **електронно-променевої трубки (ЕПТ)**, і принцип їх роботи аналогічний принципу роботи телевізора. Монітори бувають алфавітно-цифрові та графічні, монохромні та кольорового зображення. Сучасні комп'ютери комплектуються, як правило, кольоровими графічними моніторами.

Екран монітора покритий люмінофором – спеціальною речовиною, здатною випромінювати світло при попаданні на нього швидких електронів. Люмінофор наноситься у вигляді наборів точок трьох основних кольорів – **червоного, зеленого** та **синього**. Ці кольори називають основними, тому що їх поєднаннями (у різних пропорціях) можна представити будь-який колір спектру. Набори точок люмінофора розташовуються трикутними тріадами. Тріада утворює **піксел –** точку, з набору яких формується зображення (англ. pixel – picture element, елемент картинки). Відстань між центрами пікселів називається **точковим кроком монітора**. Ця відстань істотно впливає на чіткість зображення. Чим менше крок, тим вище чіткість. Зазвичай в кольорових моніторах крок складає 0,24 мм. При такому кроці око людини сприймає точки тріади як одну точку "складного" кольору.



Піксельні тріади

Найголовніше, чого ми хочемо від моніторів, це чіткість та стійкість зображення, нормальна яскравість та контрастність, комфорт для очей. Крім того, в дешевих моделях моніторів електронно-променева трубка швидше втрачає електронну емісію, що веде до погіршення яскравості та контрасту.

Комп’ютер не вміє малювати на екрані монітора нічого, окрім точок, що світяться (пікселів). Проте, розташувавши декілька точок у ряд впритул одна до одної, отримаємо лінію, а суцільно заповнивши ними деяку область екрану, отримаємо зображення фігури. Пікселі розташовані на екрані струнками лавами. Кожен піксел за вказівкою програми може бути потухшим або горіти заданим кольором (рис. 1.2).

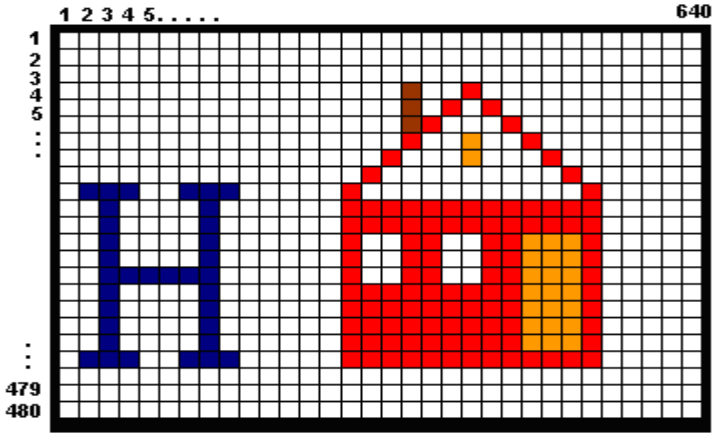


Рис. 1.2. Принцип створення зображення на екрані монітора

На рис. 1.2 ви бачите, що на екрані вміщується 640 стовпців та 480 рядків пікселів. Загальна кількість пікселів дорівнює 640x480=307200. Мінімальний помітний розмір пікселя залежить від якості монітора. Чим більше помітних пікселів уміщується на екрані, тим розмір цих пікселів менший, тим тонше та правдоподібніше рисунки на екрані. Загальна кількість пікселів управляється спеціальною електронною схемою, що знаходиться в комп’ютері, – **відеоадаптером** (**відеокартою**). Якщо ви змусите відеоадаптер розбити екран на дуже велику кількість пікселів, вони перестануть бути помітними і якість зображення не покращає, а то й погіршає. Кількість кольорів, які можуть використовувати різні відеоадаптери, теж різна – від 4 у відеоадаптерів типу CGA до багатьох тисяч у відеоадаптерів типу SVGA.

Альтернативу «трубковим» монітора знайшли давно – це **монітори на рідких кристалах (LCD)**, чудові тим, що вони, по-перше, цілком плоскі, а по-друге, шкідливих випромінювань від них зовсім немає. Раніше їх використовували тільки у портативних комп’ютерах, сьогодні їх все ширше застосовують і в настільних.

Рідкі кристали – це особливий стан деяких органічних речовин, в якому вони мають текучість та властивість утворювати просторові структури, подібні кристалічним. Рідкі кристали (РК) можуть змінювати свою структуру та світлооптичні властивості під дією електричної напруги. Змінюючи за допомогою електричного поля орієнтацію груп кристалів, та використовуючи введені в рідкокристалічний розчин речовини, які здатні випромінювати світло під впливом електричного поля, можна створити високоякісні зображення, передавати більше 16 мільйонів відтінків кольорів.

Більшість LCD-мониторів використовують тонку плівку з рідких кристалів, яка розміщена між двома скляними пластинами. Заряди передаються через так звану **пасивну матрицю** – сітку невидимих ниток, горизонтальних та вертикальних, створюючи у місці перетину ниток точку зображення (декілька розмитого через те, що заряди проникають у сусідні області рідини).

Активні матриці замість ниток використовують прозорий екран з транзисторів і забезпечують яскраве, таке, що практично не має спотворень зображення. Екран при цьому розділений на незалежні комірки, кожна з яких складається з чотирьох частин (для трьох основних кольорів та одна резервна). Кількість таких комірок по ширині та висоті екрану називають роздільною здатністю екрану.

За останні роки РК-екрани, загалом-то, не дуже змінилися, хіба що подешевшали і збільшилися в розмірах. Сама помітна новація – поява моніторів з чутливими екранами (тачскрін): вони дозволяють управляти комп'ютером без мишки, легким дотиком пальців. До речі, однією з найпомітніших «фішок» операційної системи Windows 7 якраз і стала підтримка таких дисплеїв.

Розміри LCD-моніторів у ноутбуках – від 10 до 21 дюймів, для десктопів їх роблять більше – 15, 17, 19, 21, 24, 30 дюймів; роздільну здатність замінюють буквеними позначками:

SVGA – позначення роздільної здатності екрану, 800x600, 1280x1024;
XGA – позначення роздільної здатності екрану, 1024x768;
WXGA – позначення роздільної здатності екрану, 1280x800;
SXGA+ – позначення роздільної здатності екрану, 1400x1050;
UXGA – позначення роздільної здатності екрану, 1600x1200;
WXGA+ – позначення роздільної здатності екрану, 1400x900;
WSXGA+ – позначення роздільної здатності екрану, 1680x1050;
UWXGA – позначення роздільної здатності екрану, 1920x1200.

Часто поряд із розміром екрану пишеться слово Wide (широкий) або просто буква W. Це вказівка на співвідношення сторін екрану. Якщо стандартом прийнято співвідношення розміру по горизонталі до розміру по вертикалі 4 до 3, то в Wide-екрані це співвідношення 15 до 10 (або 16 до 9). На розширеному екрані зручніше переглядати фільми. Також такі екрани люблять ті, хто працює з табличними редакторами, наприклад, з Excel: на витягнутий у довжину екран поміщається більше комірок з інформацією.

Таким чином, екран має від 1 до 5 млн. точок, кожна з яких управляється власними транзисторами. По компактності такі монітори не знають собі рівних. Вони займають в 2 – 3 рази менше місця, ніж монітори з ЕПТ, і у стільки же разів легше; споживають значно менше електроенергії і не випромінюють електромагнітних хвиль, які впливають на здоров’я людей.

Якщо екран зроблений за спеціальною технологією, то поряд з його розмірами буде вказівка на це. Багато виробників використовують однакові технології, але дають їм свої власні маркетингові назви (Crystal View, TrueBrite, XBriTE та ін.). У загальному випадку така назва означатиме покриття

антівідблиску екрану (блискучий екран) і, можливо, покращену яскравість, контрастність та якість відтворення кольорів.

Старі рідкокристалічні монітори для постійної багатогодинної роботи підходять мало: в них дуже низька яскравість, очі швидко втомлюються, а при зміні кольору точки «пливуть». Нові високоякісні монітори більшості цих недоліків позбавлені. І кольори в них відмінні, і кут огляду широкий.

Нормою сьогодні вважається коефіцієнт контрасності (співвідношення між середньою яскравістю білих і чорних прямокутників; яскравість чорного прямокутника приймається за 1) 1:700, деякі нові моделі володіють ще більшим ступенем контрастності (до 1:1000). Але це не означає, що чим більше контрастність – тим краще: надмірно різке, контрастне зображення теж не доставить очам задоволення.

Не менш важливим для хорошого РК-монітора вважається великий кут огляду. Ідеальний кут огляду повинен дорівнювати 180 градусам – в цьому випадку користувач зможе бачити зображення на екрані, навіть якщо його погляд падає паралельно йому. Ідеал, звичайно, поки недосяжний, але кут огляду градусів 170-176 сучасні РК-екрани вже забезпечують.

Дуже непогані також плоскі **газоплазмові дисплеї**, які не мають обмежень LCD-екранів. Поки їх основний недолік – велике споживання електроенергії.

Особливо треба виділити групу **сенсорних екранів** (моніторів), оскільки вони дозволяють не тільки виводити на екран дані, але й вводити їх за допомогою сенсорної сітки, тобто вони потрапляють у клас пристроїв введення/вивення. Ця відносно нова технологія поки що не набула широкого застосування. Такі екрани забезпечують найпростіший та найкоротший шлях спілкування з комп'ютером. Зазвичай цим обирається необхідний режим з меню, яке показане на екрані монітора і по якому можна зробити конкретний вибір. На даний час частіше за усе використовуються в інформаційно-довідкових системах.

1.4. Відеокарти

Монітор підключається до комп'ютера через особливу плату, яка знаходиться всередині системного блоку. Цю плату називають ***відеокартою*** (або ***графічним адаптером***). Один і той самий монітор може підключатися до комп'ютера через різні адаптери.

Комп'ютери, карти та монітори часто випускають різні фірми (інколи навіть із різних країн), тому не обійшлося без домовленостей про єдині стандарти. Сучасні монітори та відеокарти найчастіше відповідають стандарту SVGA, XGA і т. і. На старих і дуже старих комп'ютерах можна зустріти також графічні системи VGA, EGA, CGA, Hercules і т. і.

Для кожної відеокарти існує програма, яка забезпечує правильну роботу відеосистеми комп'ютера. Називається вона ***драйвером*** (driver – водій).

Головне, що треба знати про свій монітор та карту, це тип (VGA, SVGA, XGA і т. і.), а також максимальну роздільну здатність (наприклад, 600x480,

800x600, 1024x768, 1280x1024 точок (пікселів) та максимальну кількість кольорів, що відображаються (16, 256, 32 тис., 16 млн). Два останні параметри безпосередньо залежать від об'єму власної пам'яті відеокарти: чим більше відеопам'яті встановлено на карті, тим більше пікселів і з більшою кількістю кольорів вона може зберігати та виводити зображення на екран.

Чим більше розмір екрану, тим більше повинна бути відеопам'ять. Наприклад, для екрану 14" при нормальному показі та кольоровості потрібно мінімум 1 мегабайт (Мб), а для 17" потрібно вже 4 Мб. На старих комп'ютерах об'єм пам'яті відеокарти зазвичай не перевищував одного мегабайта. Усі ці цінні дані можна знайти у керівництвах до свого монітора та відеокарти або у паспортах на них.

Хочеться згадати також про сучасні могутні відеосистеми. Вони (особливо за наявності 17–20-дюймового екрану) значно перевищують за своїми показниками стандартні SVGA-монітори (Super Video Graphics Array – супервідеографічний масив) – як за здатністю зображення, так і за кількостю відображених кольорів (1280x1024 пікселів при 256 кольорах, 1024x768 пікселів при 16 млн. кольорів). На кодування кольору може відводитися вже не 8 або 16 двійкових бітів (256 або 65 тис. кольорів), а 24 біти (16 млн. кольорів) та навіть 32 біти (4 млрд). Пам'яті у таких систем буває і 16 МБ і 32 МБ і більше, що дозволяє працювати у режимі високої роздільної здатності з великою кількістю кольорів. Ще більший об'єм пам'яті (64 Мб і вище) потрібний в основному для ігор.

Зазвичай у прайсах на ПК ставиться позначка відеосистеми. Вона може бути як у повному вигляді (NVIDIA GeForce 6200 Go або ATI Mobility Radeon X700), так і у скороченому варіанті (NV44M або M26). Написи SMA (Shared Memory Architecture, архітектура пам'яті, що розділяється), Intel Extreme Graphics, GMA (Graphics Media Accelerator, графічний прискорювач) означають відеопідсистему, яка інтегрована у чіпсет, тобто не дуже продуктивну. Інколи можна зустріти слова TurboCash або HyperMemory за назвою відеокарти. Це аналогічні технології NVIDIA та AT, що означають, що якийсь об'єм відеопам'яті при необхідності може братися з оперативної пам'яті.

Відеокарта може мати спеціальні контакти для підключення телевізора або відеокамери, але треба враховувати, що для справжньої роботи з відео (монтажу та створенні різних відеоефектів) потрібні досить дорогі спеціалізовані відеокарти. Тому, поряд із традиційними відеоадаптерами широко використовуються різноманітні пристрої комп'ютерної обробки відеосигналів:

1. **Графічні акселератори** (прискорювачі) – спеціалізовані графічні співпроцесори, які покращують ефективність відеосистеми. Їхнє застосування звільняє центральний процесор від великого обсягу операцій з відеоданими, тому що акселератори самостійно обчислюють, які піксели відображати на екрані, та які їхні кольори.

2. **Фрейм-граббери,** які дозволяють відображати на екрані комп’ютера відеосигнал від відеомагнітофона, камери, лазерного програвача тощо, для того, щоб захопити потрібний кадр у пам’ять та згодом зберегти його у вигляді файлу.

3. **TV-тюнери –** відеоплати, що перетворюють комп’ютер у телевізор. TV-тюнер дозволяє обрати будь-яку потрібну телевізійну програму та відображати її на екрані у масштабованому вікні. У такий спосіб можна стежити за ходом передачі, не припиняючи роботу.

Всі сучасні графічні плати оснащені як мінімум 512 Мб пам'яті. Взагалі-то для повноцінної роботи з двовимірним зображенням використовується лише чверть цього обсягу, а от при роботі з тривимірним зображенням використовується вся пам'ять: в пам'яті відеоплати зберігаються текстури, якими обтягається в іграх тривимірний каркас. Чим більше такої додаткової пам'яті, тим краще будуть виглядати картинки в ваших улюблених іграх. Та й не тільки в них: сьогодні 256 Мб пам'яті вимагає навіть тривимірний Робочий стіл Windows 7. Саме тому норма з 2010 р. для карт середнього діапазону – 1024 Мб, а ігрові карти нового покоління мають власну пам’ять до двох гігабайт оперативної пам'яті.

1.5. Процесори

Процесор – це створений за особливою технологією кристал кремнію, який забезпечений спеціальними елементами, так званими транзисторами. Основне завдання процесора у комп’ютері – проводити обчислення. Процесор зчитує програму з оперативної пам'ятіі і виконує її. Результати роботи програми виводяться на виконавчий пристрій. Виконавчим пристроєм можуть бути монітор, принтер і т.п. Тому процесор – це основа основ комп’ютера. Без нього комп’ютер не може існувати. Від швидкості роботи процесора (швидкості, з якою він проводить обчислення) частково залежить той вельми умовний параметр, який називають «швидкістю комп’ютера».

Будь-який сучасний процесор складається із сотень мільйонів окремих електронних перемикачів-транзисторів, з'єднаних між собою за спеціальною схемою. Для їх виготовлення використовується вирощений за спеціальною технологією кристал кремнію, до якого внесено спеціальні добавки. Після виготовлення пластину покривають тонким шаром діелектрика – оксиду кремнію, який у свою чергу покривається світлочутливим лаком з особливими властивостями – фоторезистом. Потім на цей лак, як на звичайну фотоплівку, переноситься зі спеціального фотошаблона схема майбутнього процесора. Під час експонування (засвічення за допомогою ультрафіолетових променів) фоторезист змінює свої властивості: залежно від його типу засвічені (або навпаки, не засвічені) ділянки утворюють стійкий до зовнішніх впливів захисний шар. Лак на незахищених ділянках згодом протравлюється, а в утворени смужки на тілі пластини додаються спеціальні домішки, які проникають у кристалічну решітку кремнію.

Але одним шаром справа не обмежується: поверх напилення знову наноситься фоторезісторний лак – і процес повторюється знову і знову. Для зв'язку окремих шарів один з одним використовуються іони металів: заповнюючи спеціально залишені для них канавки, вони утворюють металеві доріжки-провідники.

Процесори персональних комп'ютерів відповідають єдиному стандарту, який заданий фірмою Intel – світовим лідером у виробництві процесорів для ПК. Перший мікропроцесор фірма Intel створила ще у 1971 році. Він називався Intel 4004 і був розроблений на замовлення однієї японської фірми, яка мала намір поставити його у свій новий калькулятор. Проте фірма разорилася, і процесор залишився у власності фірми Intel, яка продовжила роботу над ним.

Не дивлячись на те, що процесор Intel мав відкриту архітектуру, до сих пір Intel зуміла зберегти безумовну першість на ринку мікропроцесорів, її процесори займають левову частку (порядка 85%) в усіх персональних комп'ютерах. Але випускається ще як мінімум один дуже відомий тип Iintel-сумісних процесорів – це AMD. На ринку деякий час були присутні ще два типи Iintel-сумісних процесорів – Cyrix та VIA, – але їхня доля була нікчемна, популярністю ці процесори не користувалися, і врешті-решт зійшли зі сцени.

Проте процесори характеризуються далеко не лише фірмою-виробником. Перш за все, вони відрізняються такими важливими параметрами, як: *покоління, тактова частота, архітектура та форм-фактор*.

Покоління процесорів. Починалося все з процесорів 8088 (на них працювали перші IBM-сумісні персоналки IBM PC-XT). Далі, у найстаріших комп'ютерах (10-12-річної давності) можна знайти процесори типів XT, 086, 286, 386, 486. У дещо новіших (7-9 літніх) – Pentium, Pentium-Pro, Pentium-MMX, Pentium-II, Celeron, у новітніх – Celeron II, Pentium-III, Pentium-IV, Pentium-V, Core 2 – Core i7. Проте не виключено, що зараз, коли ви читаєте даний розділ, вже з'явилися інші марки процесорів.

Celeron – це спрощена і помітно дешевша модель Pentium, дуже популярна та, до того ж, не сильно поступається за продуктивністю самому Pentium.

Конкурент фірми Intel – фірма AMD – випускає процесори аналогічні інтеловским, але називаються вони трохи інакше: K5, K6, K7, Duron, Athlon (список за збільшенням потужності). За внутрішньою структурою ці процесори іноді принципово відрізняються від інтеловских. Інколи для них потрібні інші материнські плати. Але для нас з вами це все не так важливо, тому що як ті, так і інші працюють з тими самими програмами та операційними системами.

Тут потрібна обмовка. Ми ведемо мову про так звані IBM-сумісні персональні комп'ютери. На нашому ринку, як і в усьому світі, їх переважна більшість. Річ у тім, що дуже багато фірм-виробників персональних комп'ютерів у США, Європі та Азії прийняли стандарти фірми IBM. У розрахунку саме на ці стандарти робляться ігри та програми, створюються нові пристрої друку та інше.

Але є ще стандарт фірми Apple для персональних комп'ютерів «Макінтош» (Macintosh). Це досить могутні та прості в експлуатації

комп’ютери. Наприклад, у видавничому бізнесі професіонали дуже часто віддають перевагу саме «Макі». Проте у нас на Україні (та і в усьому світі) їх значно менше, ніж PC (у десятки разів), і вони дорожчі. Останніми роками з’явилися дешевші «Макі», стильні і красиві (прозорі, Mac).

Сьогодні все фірми дійшли висновку, що їм необхідно мати у своєму асортименті як мінімум чотири основні модифікації:

1. **Мобільна**. Самі економічні і слабкі процесори для ультрамобільних комп'ютерів (нетбуків) і офісних комп'ютерів початкового рівня (неттопів).
2. **Мобільно-офісна**. Не надто швидкі, зате економічні процесори для ноутбуків, офісних і домашніх систем початкового рівня.
3. **Домашня**. Універсальні процесори для домашніх комп'ютерів. Саме цей клас приносить виробникам найбільше прибутку. Сьогодні його займають застарілі, але досить продуктивні процесори сімейства Core i3 і i5. У сімействі AMD аналогічну нішу займають процесори Phenom П.
4. **Ігрова та професійна**. Потужні процесори, призначені для таких ресурсоємних завдань, як обробка відео, тривимірної графіки. Лідери в цій ніші – процесори родини Core i7 з частотою до 3.6 ГГц, а в родині AMD – процесори Phenom II X4 9xx і X6.

Тактова частота – це та кількість елементарних операцій (тактів), які процесор може виконати протягом секунди. Тактова частота вимірюється у мегагерцах, та показує кількість дій, які процесор може виконати за одну секунду. Від 8-16 Мегагерц на старих моделях, та до 1.3-3.6 Гігагерца – на нових. Чим більше тактова частота, тим швидше працює комп’ютер, хоча частота не є єдиним показником його продуктивності.

Ще недавно цей показник був найважливішим – єдино значимим, а збільшення швидкодії нових процесорів було пов'язано насамперед із збільшенням тактової частоти. У свій час вже відомий нам Гордон Мур з Intel передбачив (1968 р.), що кожні півтора року частота мікропроцесорів буде подвоюватися разом з числом транзисторів на кристалі. Однак років п'ять тому збільшувати частоту припинили – і сьогодні швидкодія процесора зростає за рахунок інших показників (наприклад, збільшення кількісті ядер). Тактова частота, втім, теж зростає, але не так помітно, як десятиліття тому: за два останні роки вона збільшилася всього лише на 25%. Сьогоднішній стеля – 3,6 Ггц на ядро. Ядро являє собою окремий процесорний елемент з власною кеш-пам’яттю, яке з’єднано з системною шиною через спеціальний пристрій – **арбітр шини**.

Форм-фактор. Ключове слово тут – *форм*, яке, безумовно, утворене від слова «форма». Це і є форма процесора. Річ у тім, що виробники одних і тих самих процесорів часто-густо випускають їх у різних модифікаціях з точки зору форми. Спочатку процесори – 286, 386, 486 – були маленьким керамічним квадратиком, із сталевими штирьками.

Але з появою другого «Пентіуму» виробники помістили його у великий прямокутний паралелепіпед, який назвали Slotl. Ідея була явно невдалою через

громіздкість конструкції. Проте, на щастя, виробники досить швидко усвідомили свою помилку, після чого повернулися до звичної форми процесорів, а Slot1 (та пізніша модифікація Slot2) залишилися лише у старих типах процесорів – PENTIUM-II та PENTIUM-III.

Нинішні ж форм-фактори покупцям особливих сюрпризів не підносять. Сучасними процесорами є традиційні квадратики на ніжках, а відрізняються вони один від одного лише розмірами. Посадкове гніздо для процесора (та, відповідно, його форм-фактор) називається сокетом (socket), а його розмір (як і процесор) позначається цифрами. Наприклад, форм-фактор для Pentium-III – Socket 370. Форм-фактор для Intel Pentium 4 – Socket 423, Socket 478, Socket 603.

У сучасних процесорах цифра в моделі сокета найчастіше позначає кількість процесорних контактів – «ніжок». Наприклад, нові процесори Intel зразка 2011 року призначені для роз'ємів LGA1366 (старші моделі родини Core i7) і LGA1155 (родини Core i5 і Core i7 2xxx), а старі моделі тих же серій – для роз'єму LGA 1156. А це означає, що новий процесор ви вже не зможете встановити на стару системну плату – і навпаки.

На даний момент, наприклад, Intel представляє на ринку три лінійки процесорів:

1. Atom – «бюджетне рішення» для нізкопродуктивних систем, на кшталт «неттопів» і нетбуків. Одноядерні і двоядерні процесори з частотою до 2 ГГц.
2. Core i3 – двохядерні процесори початковій категорії, для офісних і простеньких домашніх систем.
3. Core i5 – чотириядерні (у мобільних і нових серія – двоядерні) процесори середнього рівня і середньої цінової категорії.
4. Core i7 – двох-, чотирьох- і шестиядерні процесори вищого класу продуктивності.

Регістри центрального процесора – це швидка пам’ять невеликого розміру (іноді її називають надоперативною пам’яттю). Набір регістрів розрізняється у процесорах різного типу. Програміст має доступ до таких регістрів**:**

- регістри загального призначення (General Purpose Registers);
- регістри даних (Data Registers);
- регістри адреси та індексу (Address and Index Registers);
- регістри співпроцесора;
- MMX- та SSE-регістри SIMD–команд;
- сегментні регістри;
- деякі інші.

Є управляючі регістри та регістри стану, такі як лічильник команд (Program Counter або Instruction Pointer), регістр декодування команд (Instruction Register), регістр адресації пам’яті (Memory Addressing Register), регістр буфера пам’яті (Memory Buffer Register). Дані, які знаходяться у регістрах,

контролюються транслятором або програмістом, якщо він створює програму на мові Асемблера.

Кеш-пам'ять. Ще один важливий параметр, прямо пов'язаний зі швидкістю роботи процесора – обсяг «кешу», швидкої буферної пам'яті, вбудованої в кристал процессора – вона працює значно швидше звичайної оперативнї пам'яті.

Кеш-пам’ять є швидкою пам’яттю невеликого об’єму, що містить ту інформацію з оперативної пам’яті, яка використовувалася процесором під час його роботи найчастіше. Дослідження показали, що якщо програма звернулася до якогось елементу даних або команди в оперативній пам’яті, цей елемент (команда) з великою ймовірністю можуть незабаром знадобитися знов. Крім того, незабаром можуть знадобитися дані, адреси яких не набагато відрізняються від адреси даних, що використовуються на даний момент. Ці дані можна розмістити у більш швидшій пам’яті меншого об’єму – кеш-пам’яті, що збільшить середню швидкодію роботи пам’яті. Операційна система сама вирішує, яку інформацію слід зберегти у кеш-пам’яті; програміст не може управляти цим процесом.

Тобто алгоритм пошуку даних такий: спочатку здійснюється пошук потрібної інформації у кеш-пам’яті, а в разі її відсутності відбувається вибірка даних з оперативної пам’яті. Оскільки час доступу до кеш-пам’яті суттєво менший (вона фізично статична, тобто не вимагає постійного оновлення), ніж до оперативної пам’яті, то застосування кешування дозволяє суттєво підняти продуктивність системи.

Існують два види, або рівні, кеш-пам’яті. Кеш-пам’ять першого рівня (L1 cache) має об’єм у декілька десятків або сотень Кбайт і призначена для узгодження швидкості роботи процесора та зовнішньої кеш-пам’яті. Зовнішня кеш-пам’ять другого рівня власно і відповідає за процес кешування.

Коли говорять про кеш-пам’ять, то, як правило, мають на увазі кеш другого рівня. Для ефективного кешування використовуються алгоритми передбачення, тому ймовірність влучення у кеш є досить великою.

У перших процесорах Intel кеш-пам’ять другого рівня не була поєднана з процесором. Вона виконувалася у вигляді окремої мікросхеми та знаходилася на материнській платі.

Обираючи комп’ютер, часто шукають моделі з процесором Pentium. Керуються при цьому частенько думкою "компетентного" знайомого, який сказав, що Pentium краще, ніж Celeron. Питання "А чим краще?", як правило, ставить у безвихідь. Чим же все ж таки відрізняються Pentium та Celeron. Давайте, трішки заповнимо цю прогалину.

Перші Celeron з частотами 266 та 300 Мгц були звичайними Pentium II без кеш-пам’яті другого рівня, завдяки чому коштували набагато дешевше, ніж їхні «не урізані» аналоги. З офісними додатками вони впорувалися без зусиль, але для ігрових програм були зовсім непридатні. Тому Intel змінила напрям розвитку дешевих рішень.

Нові процесори Celeron з частотами від 300 Мгц отримали власну кеш-пам’ять другого рівня, причому вона була інтегрована в ядро і працювала на частоті процесора (а не на половинній частоті, як у Pentium II). Тому продуктивність виявилася приблизно однаковою, і ці Celeron цілком могли змагатися з Pentium II, хоча коштували вони дешевше. Тобто, не завжди можна було сказати, що Celeron однозначно гірше у порівнянні з Pentium

Надалі різні характеристики процесорів змінювалися, і на сьогоднішній день у загальному випадку залишилися три основні відмінності між Celeron та Pentium:

– менший об’єм кеш-пам’яті другого рівня в Celeron (впливає на продуктивність);

– як правило, Celeron працює на нижчій частоті (впливає на продуктивність);

– менша вартість процесора Celeron (на що впливає – зрозуміло).

Тобто, на загал маємо таку ситуацію: дещо знижена продуктивність у порівнянні з аналогічним Pentium цілком може компенсуватися низькою вартістю. Знижена ж продуктивність для більшості залишиться абсолютно непомітною. Якщо не брати до уваги сучасні тривимірні ігри, архівацію величезних баз даних та професійну роботу з відеоматеріалом, то врешті Celeron виявлятиме досить пристойну продуктивність.

У сучасних процесорів відразу три види кеш-пам’яті (LI, L2 і L3): чим вище «рівень» – тим пам'ять повільніше, однак і обсяг її більше. У дешевих Atom, наприклад, кеш L2 складає 512 Кб, а більш потужні процесори в доповненню до цього отримують ще і кеш L3 об'ємом від 3 Мб (Core i3) до 8 Мб (Core i7). Більший кеш дає деякий виграш в продуктивності (до 15-20%) при роботі з деякими програмами – наприклад, архіваторами, базами даних, математичним софтом.

1.6. Системна (материнська) плата

Багато хто вважає, що головний в комп'ютері – процесор («мозковий центр» системного блоку). Але і за процесором потрібне стеження, потрібен контроль, так само як за всіма іншими пристроями. Та й передачу сигналів від пристрою до пристрою не завадить забезпечити. Наразі, всій залізній братії потрібен будинок і турботлива господиня, яка б підтримувала в цьому будинку порядок. І така господиня в комп'ютері є – системна плата.

Знявши з комп’ютера кожух, можна побачити і блок живлення, і дисководи з вінчестерами, і купу плоских кабелів-шлейфів, які з’єднують окремі блоки. Але найголовніше тут – системна плата, так звана ***материнська плата*** (від англійського «Motherboard»), або просто на сленгу програмістів «материнка», «мама», або «мамка».

Материнська плата – це одна з найважливіших складових персонального комп’ютера. Можливо, найважливіша складова. Простіше кажучи, це майже

сам комп’ютер, тим більше що зараз дуже часто зустрічаються материнські плати, на яких все вбудовано: відеокарта, модем, звуковий та мережний адаптери.

На «материнку» встановлені мікросхеми процесора та пам’яті, сюди ж підходять шлейфи від інших частин комп’ютера. На материнській платі є роз’єми (**слоти**, slot – "паз, отвір для вставки будь-чого"), куди підключають додаткові пристрої – відеокарту, звукову карту, мережну карту, модем (якщо він внутрішній).

Розглянемо детальніше, з чого складається материнська плата, а точніше, які елементи на ній присутні.

Посадкове гніздо для процесора – той самий форм-фактор процесора. Ми вже говорили про те, що процесори бувають різних форм-факторів. Тому, підбираючи материнську плату під процесор, потрібно чітко шукати тип «материнки» з відповідним типом посадкового гнізда. Якщо процесор має форм-фактор P4 Socket 423, тоді і материнську плату потрібно обирати з відповідним посадковим гніздом – це завжди написано у назві плати.

Чіпсет. Чіпсет – це базовий набір мікросхем, від якого залежать досить важливі характеристики материнської плати: швидкість обміну даними, підтримка тих чи інших видів процесорів, робота з пам’яттю, підтримка різних режимів роботи з жорстким диском. Чіпсети для материнських плат випускає декілька фірм: Intel, VIA, ALI та інші, але лідером тут є все ж таки Intel.

Саме від чіпсета залежить, який тип процесорів і пам'ять буде підтримувати системна плата, як швидко будуть передаватися дані по її основній магістралі – шині. І чим більше її частота – тим швидше працює комп'ютер (як бачите, не все залежить від частоти процесора).

Наприклад, більшість плат для процесорів Intel линійки 2010 року були засновані на чипсетах серії P55, у 2011 році їх змінив новий чипсет P65.

Тип чіпсету вказується у назві материнської плати. Наприклад, i865PE ASUS P4P800 Deluxe – це плата фірми Asustek на чіпсеті Intel i865PE.

Форм-фактор самої материнської плати. У материнських плат також існує декілька різних форм-факторів. Від них залежить, по-перше, розмір самої плати (повнорозмірна, половинчаста, мікро), а, по-друге, конфігурація так званих портів на ній (*порти* – це спеціальні виводи для підключення до комп’ютера різних зовнішніх пристроїв).

Таких форм-факторів небагато. По-перше, існує вже сильно застарілий тип форм-фактора, так званий AT. У цьому варіанті на самій материнській платі немає колодок портів, а є лише виводи під них, а клавіатурний порт не PS/2, а так званий AT. Під форм-фактор AT потрібний спеціальний корпус, який підтримує такий варіант (вони зараз вже майже не випускаються), а також клавіатура з роз’ємом AT.

Останні два варіанти – це ATX та microATX. ATX – це найпоширеніший стандарт, а microATX створений для спеціальних корпусів, що характеризуються дуже маленькими розмірами.

Кількість слотів (посадкових місць) під плати пам'яті. Зазвичай їх буває два, три або чотири. Жодного особливого значення це не має, бо навіть якщо на материнській платі всього один слот під пам'ять, ви все одно зможете придбати плату пам'яті потрібного об'єму.

Кількість слотів (посадкових місць) для підключення додаткових плат. Існує три основні види слотів: стандартний PCI, застарілий ISA, який на сучасних материнських платах вже не встановоюється, та AGP – слот, який призначений спеціально для відеокарти.

Наприклад, відеокарти виготовляють під роз'єми трьох типів: ISA (довгий роз'єм), PCI (трохи менше) або AGP (короткий). Якщо ви захочете поміняти відеокарту у своєму комп'ютері, слід чітко уявляти собі, які роз'єми присутні на вашій материнській платі та які з них не зайняті іншими пристроями. Звукові карти теж можуть бути під роз'єми трьох типів – ISA, PCI та USB, модеми – для ISA, PCI та зовнішні. Ну й таке інше.

Стандарт ISA вважається застарілим, тому на нових материнських платах таких роз'ємів добре якщо один. Оскільки як PCI з'явився порівняно нещодавно (разом із Pentium), тому на старих комп'ютерах, аж до 486, PCI модем поставити не вдасться. За наявності у комп'ютері сучасного та високошвидкісного AGP-порта (Accelerated Graphic Port) краще купляти саме AGP-картку або PCI-експрес.

Існують материнські плати, на яких мережна карта, модем, відеокарта, так само як і звукова карта, вже є і вбудовані всередині.

IDE-контроллери. Це спеціальні виводи на материнській платі, до яких підключаються різні накопичувальні пристрої, – жорсткі диски, CD-ROM, DVD-ROM, CD-RW і т.і. Зазвичай IDE-контролерів буває два, і на кожен з них можна підключити до двох пристроїв (тобто всього чотири). Проте є материнські плати з чотирма IDE-контролерами, до кожного з яких можна підключити до двох пристроїв (тобто всього вісім).

SATA-контролери. Так звані контролери Serial ATA. Вони з'явилися досить нещодавно і прийшли на зміну IDE. Під ці контролери потрібні спеціальні жорсткі диски – SATA, а не IDE. На момент написання посібника сучасні материнські плати, як правило, містять два види контролерів – IDE та SATA. До IDE підключаються приводи компакт-дисків та IDE-диски, а до SATA – новий вид дисків. Наявність SATA-контролеров є вкрай бажаною.

SCSI-пристрої (*Small Computer Systems Interface*) використовуються для вінчестерів, які дозволяють приблизно на порядок збільшити швидкість читання або запису даних, оскільки всі розряди одного байта розміщуються одночасно на кількох поверхнях.

SCSI–інтерфейс, розроблений для об'єднання на одній шині різних за своїм призначенням пристроїв, таких як тверді диски, накопичувачі на магнітооптичних дисках, приводи CD, DVD, стримери, сканери, принтери тощо. Раніше мав неофіційну назву Shugart Computer Systems Interface на честь творця Алана Ф. Шугарта. SCSI широко застосовується на серверах,

високопродуктивних робочих станціях; RAID-масиви на серверах часто будуються на твердих дисках з SCSI-інтерфейсом.

BIOS. BIOS (Basic Input-Output System – Базова система введення-виведення, точніше – система контролю та управління підключеними до комп'ютера пристроями) –це дуже важлива частина материнської плати, яка являє собою апаратно першу з програм, яка записана у спеціальний постійний запам’ятовуючий пристрій *(ROM – Read Only Memory)* і з якої починає працювати ваш комп'ютер безпосередньо після його включення. BIOS – це перший і найважливіший з містків, що пов'язує між собою «апаратну» і «програмну» частину комп'ютера. Якщо з ним станеться неполадка, то ваш комп'ютер навіть не завантажиться.

У більшості материнських плат з чіпсетами Intel, як правило, встановлена «програмна начинка» BIOS виробництва фірми AWARD Software. Втім, кожна материнська плата має свої особливості, і BIOS плати ASUS дещо відрізняється від встановленого на платі Gigabyte. А тому не занадто дивуйтеся, виявивши у своєму BIOS незнайомі налаштування – замість цього загляньте в інструкцію до материнської плати.

В BIOS закладені основні параметри, необхідні для комп'ютера, щоб правильно розпізнати такі пристрої, як жорсткий диск, на якому зберігається вся ваша інформація, оперативна пам'ять – скільки її, якого вона типу.

1.7. Пам’ять комп’ютера

Пам’ять комп’ютера складається із двох основних частин. Це ***оперативна пам’ять*** (*RAM – Random Access Memory)*, пам'ять з довільним доступом), яка характеризується високою швидкістю роботи та порівняно невеликим об’ємом, а також ***зовнішня пам’ять***, яка має великий об’єм, але характеризується меншою швидкістю доступу до даних.

В **оперативній пам’яті** комп’ютера тимчасово зберігаються дані і команди, які необхідні процесору для виконання ним операцій. Інформація, що знаходиться в оперативній пам’яті, зникає при виключенні комп’ютера (зникнення на модулях пам'яті живлення, навіть короткочасного, призводить до спотворення або повного знищення даних), а інформація, яка міститься у зовнішній пам’яті, зберігається і після виключення живлення.

Сучасні енергозберігаючі режими роботи материнської плати комп'ютера дозволяють переводити його в режим «сну», що значно скорочує рівень споживання комп'ютером електроенергії. Для збереження вмісту оперативної пам'яті в такому випадку, застосовують запис її вмісту в спеціальний файл (наприклад, у системі Windows XP він називається hiberfil.sys).

Доступ до оперативної пам'яті здійснюється набагато швидше, ніж до дискової: «швидкість», вірніше, «час доступу» найсучаснішого жорсткого диску (вінчестера) становить 8-10 мілісекунд (мс). А сучасна оперативна пам'ять має час доступу 3-7 наносекунд (нс). Різниця – в сотні тисяч разів!

Насправді оперативна пам'ять – не найшвидша в комп'ютері: у багато разів швидше її кеш-пам'ять, якою оснащені процесори, жорсткі диски і низка інших пристроїв. Правда, обсяг її невеликий – всього кілька мегабайт – а коштує вона набагато дорожче. Власною графічною, оперативною пам'яттю оснащена також і відеоплата.

Оперативна пам'ять перших комп'ютерів сильно відрізнялася від тієї, з якою ми працюємо сьогодні. Спочатку для зберігання інформації використовувалися електронні лампи, а в 1953 р. з'явилися так звані «магнітні сердечники" – решітка з металевих дротів, на «вузлах» якої було невелике магнітне кільце. Для запису інформації по «рядкам» і «стовпчикам» решітки пропускали електричний розряд. У місці їх перетину виникав спрямований електричний струм, залежно від напрямку якого вміст сердечника тлумачився як 0 або 1.

Така «пам'ять» могла зберігаті в собі всього від 2 до 64 тисяч «машинних слів» (кожне слово включало від 2 до 8 байтів). У 60-і рр. пам'ять на громіздких магнітних сердечниках замінили на модні і компактні транзистори. А в 1969 р. компанія Intel представила першу мікросхему оперативної пам'яті ємністю 1 кілобіт! З цього часу оперативна пам'ять випускається у вигляді мікросхем, зібраних у спеціальні модулі пам'яті. Сьогодні найбільшою популярністю користуються 168-контактні модулі DIMM, кожен з яких може вміщати до 4 Гб оперативної пам'яті.

Мікросхеми звичайної оперативної пам'яті відносяться до типу динамічної оперативної пам'яті (DRAM), а найшвидша пам'ять, що використовується як кеш-пам'ять в процесорах, відносяться до типу статичної оперативної пам'яті (SRAM). Пам'ять динамічного типу дешевше, ніж статичного, і її щільність вище, що дозволяє на тому ж просторі кремнієвої підкладки розміщувати більше комірок пам'яті, але при цьому її швидкодія нижче. Статична, навпаки, більш швидка пам'ять, але вона і дорожче. У зв'язку з цим масову оперативну пам'ять будують на модулях динамічної пам'яті, а пам'ять статичного типу використовується для побудови кеш-пам'яті всередині мікропроцесора.

Серед динамічної пам'яті теж можна виділити кілька видів, але сьогодні в комп'ютерах використовуються лише два: DDR 2 і DDR 3 SDRAM. Абревіатура DDR розшифровується як double data rate – «подвійна швидкість передачі даних»: пам'ять цього типу, як і сучасні процесори, здатна «подвоювати» оригінальну частоту шини пам'яті. Наприклад, пам'ять DDR2-800 працює на частоті шини всього в 400 МГц! На жаль, навіть цієї швидкості сьогодні виявляється недостатньо: нагадаємо, що останні версії чіпсетів під процесори Core i7 підтримують частоту системної шини в 1333 МГц і 1600 МГц. Саме тому вже щосили йде перехід на пам'ять нового типу – швидку DDR 3, яка працює на частоті до 2400 МГц. Частота пам'яті обов'язково вказується в маркуванні: наприклад, DDR3-1600, DDR3-1800.

Зовнішня пам’ять використовується для довготривалого зберігання інформації: програм, даних, тощо. Прикладами пристроїв зовнішньої пам’яті є: жорсткі диски (HDD, хард-диски, «вінчестери», «гвинти»), лазерні компакт-

диски (CD-ROM), магнітні стрічки, дискети та інші носії інформації. Для свого виконання програма завантажується в оперативну пам'ять із зовнішньої пам'яті. Найважливішою характеристикою пам'яті є її об'єм.

Перший жорсткий диск IBM 350 у складі першого серійного комп'ютера IBM 305 RAMAC був розроблений у 1956 році американською фірмою IBM. Накопичувач займав ящик розміром з великий холодильник і мав вагу 971 кг, а загальний об'єм пам'яті (50 тонких дисків діаметром 610 мм) складав близько 5 мільйонів 6-бітових байт (3,75 Мб у перерахунку на 8-бітові байти). Але ж принципи сучасної технології виготовлення жорсткого диска були розроблені у 1973 тією ж фірмою IBM. Новий пристрій, який міг зберігати до 16 мегабайт інформації, мав 30 циліндрів (сукупність смужок з однаковими номерами на всіх магнітних поверхнях) для запису, кожен з яких був розбитий на 30 секторів. Тому він отримав назву 30/30. Відомі гвинтівки «вінчестер» мають калібр 30/30, тому жорсткі диски, за однією з версій, також почали називатися "військовим" словом – «вінчестерами». В літературі зустрічається ще кілька суперечливих даних щодо характеристик диска (30/30), ось деякі з них:

– диск мав характеристики: ємність 30 Мбайт, час доступу 30 мілісекунд;

– накопичувач складався із двох дисків по 30 Мбайт кожен.

За іншою, більш ймовірною версією (Велика енциклопедія Кирила та Мефодія, 2005), він отримав таку назву завдяки м. Вінчестеру (Англія), де проходили перші розробки жорсткого диску філією фірми IBM.

Широке вживання вінчестерів як пристроїв довготривалого зберігання інформації для ПК почалося після випуску у 1980 році фірмою Shugart Technology (нині ця компанія іменується Seagate Technology, Inc.) диску ST506 розміром 5.25 дюймів. Пристрій ємністю 5 Мбайт використовував для підключення до комп'ютера інтерфейсну плату ST506, яка була розроблена у кінці 70-х років компанією Western Digital. Для з'єднання вінчестера з інтерфейсною платою використовувався 34-провідний плоский кабель, до якого можна було підключити два пристрої. Для того щоб диски можна було адресувати, частина кабелю перекручувалася (подібно до кабелю для підключення дисководів). Крім того, для обміну даними з кожним з дисків використовувався окремий 20-провідний плоский кабель. Крупним недоліком інтерфейсу ST506 було покрокове переміщення голівок (один крок на кожну команду переміщення), як це досі відбувається у дисководах для роботи з гнучкими дисками. Більш нова модель – ST412 – забезпечувала можливість буферизованого пошуку (buffered seek), який дозволяє однією командою перемещувати блок магнітних голівок на кілька кроків (наприклад, через весь диск).

Оперативну пам'ять можна уявити собі як пам'ять , яка складається з фізичних комірок, – мінімальних елементів, до яких можна звернутися з програми. Використовується лінійна модель пам'яті з послідовною адресацією. Це означає, що фізичні елементи пам'яті можна представляти розташованими лінійно, причому кожна комірка має свою унікальну адресу – порядковий

номер. В захищеному режимі віртуальної пам’яті використовується плоска логічна модель пам’яті з розбиттям по сторінках (по 4 Kb кожна), при якій логічна адреса змінюється послідовно та неперервно, а фізична адреса визначається за допомогою спеціальних таблиць дескрипторів сторінок, причому сторінки не обов’язково розміщуються послідовно одна за одною, а обираються по мірі звільнення пом’яті іншими програмами у разі багатопрограмної ОС.

В сучасних багатозадачних операційних системах за допомогою сторінкової організації віртуальної пам'яті вирішується проблема фрагментації. Пояснимо це на абстрактному прикладі. Припустимо, що в оперативній пам'яті знаходиться кілька програм, як показано на рисунку. По мірі одночасного їх виконання та завершення в пам'яті звільняються певні ділянки. Наступна нова програма може не вміщуватись у жодну із звільнених ділянок. Раніше ця проблема вирішувалась за допомогою спеціальних системних програм – дефрагментаторів, які переміщували коди програм, що залишились, впритул один до одного, створюючи таким чином неперервною ділянку вільної пам'яті. Така процедура виконувалась досить довгий час (кілька годин), оскільки була пов'язана з перерахунком адрес в машинних командах та даних тих програм, які переміщувались. При сторінковій організації пам'яті частина адреси являє собою тег або індекс, за допомогою якого знаходиться початкова адреса сторінки через спеціальні таблиці дескрипторів сторінок. Дескріптор сторінки являє собою певну структуру даних, яка описує сторінки і містить усі атрибути сторінки, у тому числі її початкову адресу Молодші 12 розрядів віртуальної сторінкової адреси – це внутрішньосторінкова адреса. Початкова адреса сторінки в молодших 12 розрядах містить нулі, а разом із внутрішньосторінковою адресою вони складають фізичні адреси у пам'яті команд або даних.

Таким чином, при віртуальній сторінковій організації пам'яті доступ до кодів команд та даних здійснюється через таблиці дескрипторів сторінок. Це дає можливість операційній системі замість дефрагментації виконати переміщення дескріпторів у таблицях дескріпторів так, щоб тегі або індекси дескріпторів відповідали порядку слідування сторінок одна за одною. Процедура переміщення дескріпторів у таблицях дескріпторів виконується досить швидко і користувач навидь цього не помічає.

Припустимо, що після запуску операційної системи було запущено шість завдань, як показано на рисунку ліворуч (рис. 1а). Тоді дескриптори сторінок також спочатку будуть розташовані у такому ж порядку. Припустимо, що в деякий момент часу три задачі завершили своє виконання і в пам’яті з’явились вільні зони (рис. 11 б). Операційна система перерозподілить дескриптори впритул один до одного і старша частина логічної сторінкової адреси буде змінюватися у цих вільних ділянках послідовно (рис. 11 в). В такому разі деяка програма більшого розміру може розміщуватись по сторінках в різних ділянках пам'яті, але її віртуальна сторінкова адреса буде послідовною і неперервною.

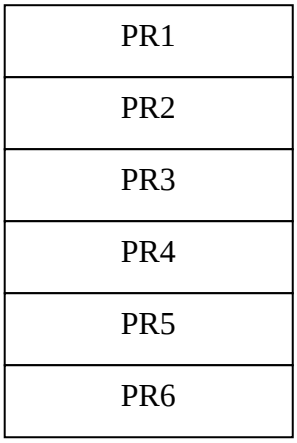


Рис. 11 а. Пам'ять після завантаження шістьох сторінок

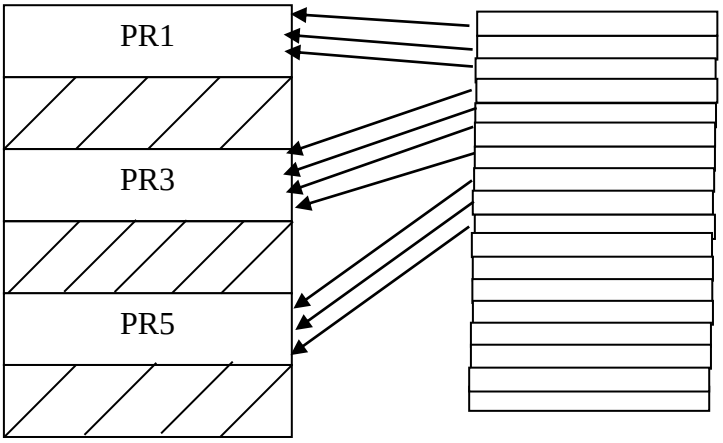


Рис. 11 б. Пам'ять після завершення трьох задач

Рис 11 в. Впорядкування дескріпторів задач

Для ефективної роботи комп'ютера необхідно, щоб операції з даними в оперативній пам'яті виконувалися так само швидко, як працює центральний процесор. Проте виявляється, що чим більше розмір пам'яті, тим повільніше вона працює. Виникає проблема, для розв'язання якої використовується багаторівнева організація пам'яті. При такій організації можна виділити два основні рівні – маленьку, але швидку пам'ять та повільну пам'ять великого розміру. До швидкої пам'яті належать регістри процесора, кеш-пам'ять, а до повільної – оперативна та зовнішня пам'ять.

«Найшвидшою» частиною пам'яті є регістри процесора. Ємність одного регістра складає декілька десятків бітів – двійкових розрядів (наприклад, 32 біта). Далі йде кеш-пам'ять об'ємом від декількох десятків до декількох сотень кілобайт, та часом доступу у десятки разів більше, ніж у регістрів. Оперативна пам'ять має об'єм декілька десятків, сотень і навіть тисяч мегабайт, але час доступу до неї ще раз у десять більше. Кеш-пам'ять є проміжною між оперативною пам'яттю та власно мікропроцесором. Далі йде зовнішня пам'ять об'ємом у десятки, сотні та тисячі гігабайт. Швидкість отримання даних із такої пам'яті у сотні разів менше, ніж з оперативної пам'яті.

Дуже важливий параметр – об'єм жорсткого диска. На старих комп'ютерах були диски розміром 120, 240, 540 мегабайт. Ціна одного мегабайта пам'яті вінчестера значно менше, ніж ціна одного мегабайту оперативної пам'яті, тому сьогодні на більшості персональних комп'ютерів вона має розмір від 80 до 640 гігабайт та навіть вище (кілька терабайт).

Важливою також є швидкодія жорсткого диска: час пошуку інформації на диску повинен бути якомога менше, а швидкість передачі даних – більше. Особливо важливі ці параметри для тих, хто збирається працювати з великими об'ємами даних – базами даних, звуковими файлами та відеофайлами.

Більшість дисків обертаються зі швидкістю 5,4 тис. об/хв., а нові, високопродуктивні, – вже 7,2 (10) тис. На таких швидкостях диски поводять себе як гіроскопи, тому небажано пересувати комп'ютери під час їхньої

роботи). Доступ до даних на таких дисках прискорений, але вони сильніше гріються. Іноді навіть потрібно ставити для них додатковий вентилятор-кулер (від слова cool, що слід розуміти як «охолоджувати»).

Коли виникає необхідність купити та встановити у комп'ютері додатковий вінчестер (або замінити старий), при виборі моделі доводиться враховувати ще й вид інтерфейсу – тобто спосіб підключення диску до материнської плати. Більшість жорстких дисків підключаються через інтерфейс IDE. Але існує також інтерфейс SCSI («сказі») – потужніший та швидкісніший, ніж IDE. Високошвидкісні різновиди цього інтерфейсу можуть називатися, наприклад, Fast SCSI або UltraWide SCSI. Зазвичай комп'ютер комплектується IDE-інтерфейсом.

До речі, серед IDE вінчестерів також є швидкісні. Для них створені вдосконалені види інтерфейсу – Ultra ATA (він же – Ultra DMA). Виграш у швидкодії дисків може бути дво- (Ultra ATA 66) або триразовим (Ultra ATA 100) та навіть вище (Ultra ATA 133). Сучасні вінчестери ще більш швидкісні і мають інтерфейс SATA (serial ATA, SATA-2). Тут не наводиться жодних технічних деталей, вони мало кого з початківців цікавлять. А ось терміни знати варто. І в разі куаівлі це також стане у нагоді, і починаєш трохи розуміти, про що твої друзі розмовляють.

Мабуть, час вінчестерів проходить. На зміну їм йдуть ***твердотільні накопичувачі***, які з часом все ж таки витіснять жорсткі диски. Твердотільні накопичувачі увійшли в наше життя порівняно недавно. Так, компанія OCZ в 2009 р. випустила твердотільний накопичувач OCZ Z-Drive PCI-Express SSD.

Ці недешеві твердотільні накопичувачі мали вражаючу швидкість читання та запису – 500-510 Мбайт/с та 470-480 Мбайт/с. Немає жодного сумніву у тому, що нові SSD будуть потрібні тим користувачам, які не хочуть (і мають для цього грошові можливості) йти на компроміс між швидкістю та ємністю накопичувача. Отже, технологія SSD вже зараз не обмежується швидкостями передачі/запису даних 520/410 Мбайт/с, а далі – більше (Samsung в липні 2013 р. анонсувала SSD з рекордною швидкістю читання – до 3000 Мб в секунду, поставлятися такі SSD будуть в трьох варіантах: з 400 ГБ, 800 ГБ і 1,6 ТБ пам'яті, поставлятися такі SSD будуть в трьох варіантах: з 400 ГБ, 800 ГБ і 1,6 ТБ пам'яті).

У момент появи твердотільних накопичувачів деякі спеціалісти передрікали класичним вінчестерам швидку «смерть». Поки цього, з цілком об'єктивних причин, не сталося. Що ж це за причини?

Справа в тому, що швидкодія сучасного комп'ютера обумовлюється не тільки потужністю процесора, кількістю оперативної пам'яті і пропускною здатністю системної шини. Вона безпосередньо залежить і від швидкості, з якою вінчестер може обмінюватися даними з процесором (якщо говорити більш коректно – від швидкості здійснення операцій введення/виводу). Причому саме останній показник – одне з найбільш «вузьких місць» сучасного ПК.

Обумовлюється це тим, що якщо не вважати DVD-приводу, який використовується далеко не завжди, вінчестер це єдиний пристрій комп'ютера в якому задіяна механіка (кулери теж в розрахунок не беремо). Для того щоб прочитати з диска порцію даних, вінчестера доводитися спочатку перемістити головки в потрібну область, а на це потрібен час і, за комп'ютерними мірками, дуже значний.

Всі знають, що звернення до оперативної пам'яті відбувається набагато швидше, ніж до жорсткого диску. У ній немає механічних частин, і дані зчитуються безпосередньо з регістрів мікросхеми, в яких зберігаються. Приблизно за таким же принципом побудовані і твердотільні накопичувачі. По суті це набір мікросхем, в яких записана збережена інформація і, коли процесору потрібно отримати код будь-якої програми, він просто звертається до тієї з них, в яких вона збережена. Ніяких повільних переміщень головки, розкручування дисків та інших механічних проблем. Звичайно, твердотільні накопичувачі дуже сильно відрізняються від оперативної пам'яті, але все ж перші працюють набагато швидше класичних вінчестерів.

Наведемо цифри. Якщо середній час доступу у класичного вінчестера становить близько 10 мс то, наприклад, твердотіла модель Samsung SSD 470 на ту ж операцію витрачає всього 0.09 мс. На практиці це означає, що в більшості випадків значного збільшення продуктивності комп'ютера простіше домогтися, помінявши його вінчестер на твердотільний накопичувач, а не змінювати процесор на більш потужний.

Так чому ж старі добрі вінчестери все ще встановлюються на ПК? Якщо нові пристрої так гарні, чому все не перейшли на них? Найголовніша перешкода полягає в тому, що поки технології не навчилися робити їх достатнього обсягу. Так, наприклад, самий місткий твердотільний накопичувач, який пропонував досить відомий комп'ютерний супермаркет PET, міг вмістити лише 240 Гб даних. При цьому його ціна становила близько 650 доларів. Ті ж моделі, що не конкурували за вартістю з цілим комп'ютером, вміщували всього лише 60 Гб.

Як бачимо, обсяги твердотільних накопичувачів за нинішніми мірками неприпустимо малі, а ціна на них надзвичайно висока. Так, наприклад, вартість першого терабайтного SSD компанії OCZ наближалась до 4000 євро, що, звичайно, робило новинку цікавою забавою для одиниць. OCZ розуміла це і тому випустила також 500 и 256-гігабайтні SSD.

Наразі, сучасні ціни на «середні» і терабайтні вінчестери (2013 р.) трохи менші (наприклад, SSD Plextor 512 Gb 2.5 SATAIII – 4000 грн; ціновий ряд корпорації Sumsung наступний: 120 Gb SSD – $120, 250 Gb – $200, 500 Gb – $420, 750 Gb – $560 і 1 Tb – $770). Залишилося тільки дочекатися, коли вартість SSD все ж зрівняється з ціною жорстких дисків.

У розробку більш просунутої флеш-технології запису даних приєдналася всесвітньо відома компанія IBM, яка планує вже протягом наступних декількох років вкласти близько мільярда доларів. Частково кошти будуть використані для відкриття 12 дослідницьких центрів по всьому світу для моделювання

систем зберігання даних на основі флеш-пам'яті. Нові центри будуть знаходитися в Китаї, Франції, Німеччини, Індії, Сінгапурі, Південній Америці, Великобританії та США і розпочнуть свою роботу вже до кінця нинішнього року.

Успішно конкуруючі флеш-носії компанія збирається поступово впроваджувати у свої системи зберігання та серверні продукти. Слід зазначити, що не так давно IBM представила публіці новітні пристрої, що мають флеш-накопичувачі з максимальною ємкістю в 24 терабайта. Такі прилади працюють набагато швидше звичайних дисків.

Сьогодні накопичувачі, що використовують магнітні диски, мають дуже демократичну вартість, що і робить їх неймовірно популярними у всьому світі. Незважаючи на це, сучасні реалії змушують компанії, що працюють з величезними потоками даних та інтернет-контенту, переходити на більш ємне і швидке обладнання. IBM дотримується думки, що твердотільні носії, що працюють на флеш-пам'яті – ідеальний вихід з проблемної ситуації. У IBM переконані, що використання флеш-пам'яті може зменшити час відгуку в серверах і системах зберігання даних від мілісекунд до мікросекунд в порівнянні з жорсткими дисками. При цьому відсутність рухомих частин робить цю технологію більш надійною, довговічною і економічною з точки зору живлення.

1.8. Дисководи

Перша дискета діаметром у 200 мм (8″) з відповідним дисководом була представлена фірмою IBM у 1971 році. Сам винахід приписується Алану Шугарту, який працював у кінці 1960-х років в IBM. У 1973 році Алан Шугерт засновує власну фірму Shugart Associates та у 1976 році розробляє дискету діаметром 5,25″. Ці дискети були різної щільності і, відповідно, ємності: на дискети подвійної щільності можна було записати 360 Кб. Правда, існували усілякі хитрщі, які дозволяли записати туди до 720 Кб. А на диски високої щільності поміщалось 1.44 Мбайт.

П’ятидюймові дискети мали значні розміри і були ненадійні. Гнучка обкладинка не захищала магнітну смужку від усіляких пошкоджень, внаслідок чого ці дискети часто збоїли. Тому у 1981 році фірма Sony виводить на ринок дискети діаметром 3,5″ (90 мм) у щільному пластиковому корпусі, для яких були розроблені спеціальні тридюймові дисководи. У першій версії об’єм пам’яті складає 720 Кб (9 секторів на одну магнітну смужку). Пізніша версія вже має об’єм 1440 Кб або 1,44 Мб (18 секторів). Саме цей тип дискети стає стандартом (після того, як IBM використовує його в своєму IBM PC).

І сьогодні сучасні дисководи (FDD – Floppy Disk Drive) розраховані на дискету розміром у 3,5 дюйма. Зазвичай дискета вміщує 1,44 Мб даних (1,44 мільйона символів). Але незабаром FDD-дисководи кануть у минуле.

1.9. CD-ROM, CD-RW, DVD

2 березня 1983 року ознаменувалося сенсацією. У Великій Британії компанії Philips, Sony та Polygram продемонстрували компакт-диск (CD – Compact Disc) діаметром 12 см. На диску була записана Дев'ята симфонія Бетховена. Тривалість запису, яка була віконана лазером і прочитувалась лазерним програвачем, складала 74 хвилини. До рядових користувачів CD дійшли куди пізніше, а в якості штатного пристрою на персональних комп'ютерах вони з'явилися тільки в середині 1990-х років.

Основна маса програмних продуктів випускається сьогодні на компакт-дисках, дуже рідко можна зустріти програми на дискетах. А ігри та усілякі мультимедійні енциклопедії випускають сьогодні вже тільки на CD. Крім того, пристрої CD-ROM (Compact Disc Read-Only Memory) можна використовувати як звичайний CD-плейер – прослуховувати на ньому музичні компакт-диски, навіть якщо у комп'ютері немає власного пристрою для відтворення звуку (звукової карти та динаміків): у CD-ROM-ах найчастіше є прямий вихід на навушники.

CD-ROM-и розрізняються за швидкістю передачі інформації з подвійною, збільшеною вчетверо, збільшеною вшестеро і т. д. швидкістю. Сучасні 36–52-швидкісні CD-ROM пристрої працюють вже досить швидко, хоча до вінчестерів їм далеченько. Компакт-диск вміщує близько 650–700 МБ даних (або близько 70 хвилин звуку).

Дисководи CD-ROM призначені тільки для відтворення інформації і не дозволяють нічого записувати на компакт-диск.

На відміну від магнітних дисків, компакт-диски мають не безліч кільцевих смужок, а **одну – спіральну,** як у грамплатівках. У зв'язку з цим, кутова швидкість обертання диска не постійна. Вона лінійно зменшується в процесі просування лазерної зчитуючої головки до краю диска.

Інформація на магнітооптичних накопичувачах типу CD-R, представляється чергуванням поглиблень і піків. Цей рельєф створюється при виробництві механічним способом. Інформація наноситься вздовж тонких смужок. Читання відбувається шляхом сканування смужок лазерним променем, який по-різному відбивається від поглиблень та піків. Для цього використовується **зчитуюча голівка** з **мікролазером** та **світлодіодом.** Глибина западин на поверхні диска дорівнює чверті довжини хвилі лазерного світла. Якщо у двох послідовних тактах прочитування інформації промінь світла лазерної головки переходить з виступу на дно западини або назад, різниця довжин шляхів світла у цих тактах змінюється на півхвилю, що викликає посилення або послаблення спільно потрапляючих на світлодіод прямого та відбитого від диска світла.

Якщо в послідовних тактах прочитування довжина смужки світла не змінюється, то і стан світлодіода не змінюється. В результаті струм через світлодіод утворює послідовність двійкових електричних сигналів, відповідних поєднанню западин та виступів на смужці. Різна довжина оптичного шляху

променя світла у двох послідовних тактах читання інформації відповідає двійковим одиницям (перепади). Однакова довжина відповідає двійковим нулям.

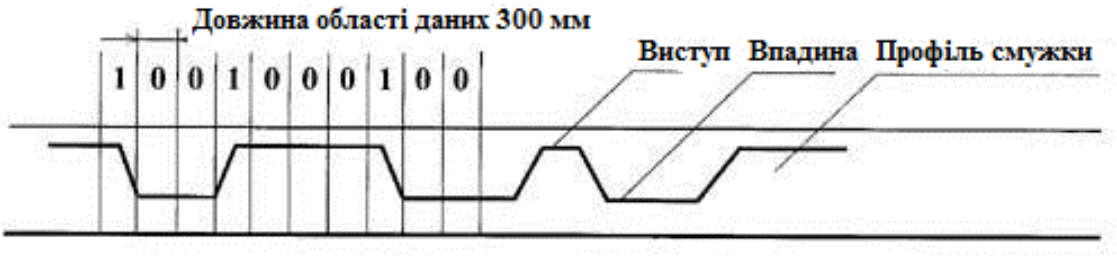


Рис. 1.3. Профіль смужки CD-ROM

Існують накопичувачі, які також пишуть (CD-R, «болванки»). Правда, перезаписати одного разу записаний («пропалений») CD-R-ом диск вже не можна. Вкрай просунуті у цьому відношенні – перезаписуючі накопичувачі (CD-RW – ReWritable), що дозволяють багаторазовий запис на CD.

CD-RW – дещо дорожчі, але незрівнянно корисніші пристрої. На таких дисках, де дозволяється багаторазовий перезапис, застосовується магнітооптичний принцип, в основу якого покладена фізична властивість: коефіцієнт віддзеркалення лазерного променя від по-різному намагнічених ділянок диску з особливим чином нанесеним магнітним покриттям – різний.

CD-RW – зазвичай повільніші пристрої, ніж простий CD-ROM. Наприклад, такий пристрій може записувати з восьмиразовою швидкістю, а прочитувати – з двадцятиразовою. Найсучасніші з CD-RW за швидкістю відтворення вже мало відрізняються від звичайних CD-ROM.

Але для тих, де об’єми даних, які хочеться зберегти, обчислюються гігабайтами (відео, тривимірна графіка, звук), 700 Мб вже явно недостатньо. Недавно виник новий стандарт компакт-дисків та дисководів до них – DVD (Digital Versatile Disc – цифровий багатоцільовий диск; також з англ. Digital Video Disc – цифровий відеодиск). Такі пристрої вміщують на порядок більше інформації, ніж звичайний CD (мінімальна їхня ємність – 4–8 Гб). На такому диску можна тримати повнометражний фільм із високоякісним зображенням та багатоканальним стереозвуком, або десятки звичайних аудіоальбомів, а також величезні мультимедійні енциклопедії. У більшості випадків зараз продаються універсальні накопичувачі, які можуть читати CD та DVD. А деякі з них ще й записують (CD-R, RW, DVD-R і DVD-RW).

1.10. Порти

Порти – це спеціальні пристрої, які забезпечені так званими колодками та виведені, як правило, на задню сторону корпусу і призначені для підключення до комп’ютера різних зовнішніх пристроїв – так званої комп’ютерної периферії. У деяких видах корпусів частина портів може виводитися на лицьову частину корпусу. Є плати, які забезпечені власними спеціалізованими виводами, через

які вони з'єднуються з відповідними зовнішніми пристроями, – відеокарта, звукова карта, – а окремі порти зроблені в комп'ютері для того, щоб до них можна було підключати все, що завгодно.

Порти**,** які зазвичай знаходяться позаду системного блоку, бувають трьох видів. **Паралельні –** для підключення принтерів, сканерів і т. д. (їхні імена в комп'ютері **LPT1, LPT2** та **LPT3**). **Послідовні** (RS-232)**,** або **" Serial ",** через які підключаються миша та модем (їхні імена **COM1, COM2** та **COM3**). **USB (Universal Serial Bus – Універсальна Послідовна Шина) –** для підключення найсучасніших зовнішніх пристроїв (флеш-пам'яті, принтерів та сканерів новітніх моделей, електронних фотоапаратів тощо).

В даний час більшість пристроїв мають стару модифікацію USB 2, проте вже на ринок виходять адаптери та пристрої для нової версії інтерфейсу – USB 3. Швидкість передачі даних у цієї модифікації в 10 разів вище, ніж у USB 2, і досягає 5 Гбіт/с! Правда, це теоретична межа, на практиці швидкість в будь-якому випадку буде нижче. Так, у USB 2 «теоретична» швидкість передачі даних – 480 Мбіт/с (а насправді 60 Мб/с).

На дяких комп'ютерах для підключення миші та клавіатури є порти ще одного застарілого стандарту – **PS/2.**

eSATA – новий швидкісний порт, призначений для підключення зовнішніх жорстких дисків. Перші системні плати з підтримкою eSATA з'явилися на ринку в середині 2007 р., і сьогодні цей інтерфейс вже перетворився на стандарт для підключення швидких накопичувачів.

Існують також особливі **інфрачервоні порти** (IR – infra red), які дозволяють підключити, скажімо, мишу та клавіатуру без дротів. Такий порт спілкується зі своїм підлеглим пристроєм, як телевізор із пультом дистанційного керування. Але бездротові мишки та клавіатури вимагають автономного живлення – батарей.

Деякі мобільні телефони також можуть підключатися до комп'ютера через інфрачервоний порт. Це дозволить зберігати на жорсткому диску комп'ютера телефонну книжку, завантажувати з комп'ютера у свій телефон малюнки, а також дає прямий доступ до Internety та електронної пошти через мобільний телефон.

Інший різновид дистанційного підключення комп'ютера до зовнішнього пристрою – порт на основі радіопередавача (WiFi–порт), більш сучасний, ніж інфрачервоний, але трохи дорожчий. Взагалі, ідея відмінна, яка позволяє позбавитися від гірлянди кабелів, що звисають позаду комп'ютера! Але не всі пристрої вміють у такий спосіб підключатися.

1.11. Переносні запам'ятовуючі пристрої

Як тільки персональний комп'ютер втратив статус престижної та дорогої іграшки, користувачі в усе більш вимогливішій формі почали натякати виробникам на незручності спілкування з ними. Зокрема, для перенесення інформації з комп'ютера на комп'ютер гнучкі диски явно вже не впорувались з

цим завданням. Ті, хто розробляє на своєму комп'ютері складну графіку, звук або відео, створює великі бази даних, використовує для їх збереження та перенесення на інший комп'ютер деякі спеціальні пристрої великої ємності. Окрім згаданих CD та DVD також використовуються переносні вінчестери (будь-якого об'єму), вінчестери із змінним диском (Iomega Zip, Iomega Jazz, SyQuest – об'ємом від 100 Мб до гігабайта й вище), JetFlash-память, а також найдешевші, але й найповільніші пристрої – **стримери** (накопичувачі на магнітній стрічці).

Найпопулярніші з усіх цих пристроїв – **Flash-пам'ять** (**Флеш-карти**) та **переносний диск малих розмірів**, але великої ємності, які підключаються до USB-порту (**Universal Serial Bus – Універсальна Послідовна Шина**). Флеш-пам**'**ять – різновид незалежної напівпровідникової пам'яті, в яку можна перезаписувати інформацію.

Отже, до флеш-пам'яті користувачі пред'являють декілька скромних вимог:

1. Незалежність – тобто не потребувати батарей, несподівана розрядка яких призведе до втрати інформації.
2. Надійність – не втрачати дані під впливом грози, падінні або при попаданні у вологу.
3. Компактність – щоб не розмірковувати, а чи варто тягнути все це із собою.
4. Довговічною – щоб не бігати у магазин кожного місяця за новою, оскільки стара відслужила свій термін.
5. Універсальною – сумісною з безліччю пристроїв, в яких можуть бути потрібні дані.

П'ятнадцять років тому компанія Toshiba розробила технологію незалежної напівпровідникової пам'яті, яку вона назвала ***флеш-пам'яттю***. Мікросхеми, які зберігають дані після відключення живлення, були відомі і раніше (наприклад, BIOS), але з такою пам'яттю було пов'язано багато незручностей: для запису були потрібні спеціальні пристрої-програматори, а щоб стерти інформацію доводилося застосовувати ультрафіолетове опромінювання кристалу. Флеш-пам'ять дозволяє записувати та стирати дані без таких складнощів, завдяки чому має непогану швидкодію та, до того ж, достатньо надійна.

Незабаром чіпи флеш-пам'яті почали вбудовувати у різні пристрої, а на їх основі були створені флеш-карти, за допомогою яких можна транспортувати різні дані великого об'єму (з 2008 року у продажу є флеш-карти об'ємом 8, 16, 32, 64 Гбайт).

Але об'єми флэш-пам'яті зростають з космічною швидкістю. Так, компанія Kingston Technology у 2009 році випустила першу у світі флэш-карту з об'ємом пам'яті в 256 гігабайт, повідомляється в офіційному прес-релізі. Новий USB-накопичувач отримав назву DataTraveler 300. Швидкість запису даних на нову флешку складає 10 мегабайт за секунду, а швидкість читання – 20 мегабайт за секунду.

Новий пристрій поки що доступний лише жителям Європи. Вартість такої флешки складала майже 566 фунтів стерлінгів (більше 930 доларів), пише Engadget. Пристрій працює з операційними системами родини Windows (у тому числі Vista та XP), а також з Mac OS X та Linux.

У комплекті з DataTraveler 300 постачається спеціальне програмне забезпечення Password Traveler для Windows, яке дозволяє захистити дані, що знаходяться на флеш-пам’яті. У прес-релізі також наголошується, що DataTraveler 300 підійде власникам ноутбуків, а також користувачам, які працюють з великими об’ємами даних.

В середині червня 2009 року компанія Kingston Technology також випустила першу у світі флэш-карту з об’ємом пам’яті 128 гігабайт. Вартість накопичувача, який отримав назву DataTraveler 200, складає більше 500 доларів.

Флеш-пам’ять може бути прочитана скільки завгодно разів, але писати в таку пам’ять можна лише обмежену кількість разів (близько 10 тисяч). Причина у тому, що для запису у пам’ять необхідно спочатку стерти ділянку пам’яті, а ділянка може витримати лише обмежене число стирань.

Правда, у кінці 2013 року в Інтернеті з'явилося повідомлення, що вчені з компанії Macronix, розташованої в тайванському місті Сіньчжу, розробили технологію, здатну позбавити флеш-пам'ять від ефекту старіння.

Ефект старіння є одним з недоліків флеш-пам'яті, який поряд з іншими факторами –передусім вартістю – перешкоджає її масовому поширенню. Справа в тому, що постійний вплив на осередки високої напруги зношує їх, і з часом вони втрачають здатність приймати чіткий логічний стан.

Сучасна флеш-пам'ять втрачає здатність надійно зберігати інформацію через 10 тис. циклів перезапису. У випадку з USB-накопичувачем, який також заснований на флеш-пам'яті, це не має значення, зважаючи, як правило, на короткий термін її експлуатації. Однак для SSD-накопичувачів, якими поки в основному оснащуються ноутбуки, це важливо, оскільки термін експлуатації комп'ютера може становити кілька років. Ще більше значення цей показник має для корпоративних систем зберігання даних, з високим навантаженням.

Поставивши собі за мету підвищити життєздатність флеш-пам'яті, вчені звернули увагу на тип пам'яті, що заснована на фазовому стані речовини. У фазової пам'яті осередок змінює свою провідність при нагріванні. Дослідники виявили, що аналогічний нагрів осередку у флеш-пам'яті надає ефект відновлення. Слідом за цим відкриттям вчені модифікували чіп звичайної флеш-пам'яті, додавши в нього крихітні нагрівачі для груп осередків. Нагріваючи осередок до температури 800 °C протягом декількох мілісекунд, вони змогли продовжувати їх життєвий цикл.

За словами представника Macronix Хан-Тіна Люе (Hang-Ting Lue), в лабораторних умовах вони змогли продовжити термін служби флеш-пам'яті з 10 тис. до 100 млн циклів перезапису. Але в компанії стверджують, що теоретична межа може досягати 1 млрд циклів. У дослідників поки не було можливості зрозуміти цю межу, оскільки тестування займе декілька місяців.

При думці про регулярне нагрівання флеш-пам'яті виникає питання про те, наскільки швидко розрядиться акумулятор, якщо таку пам'ять використовувати у ноутбуці або іншому портативному пристрої. Однак Лює стверджує, що турбуватися нема про що, оскільки нагрівати чіпи необхідно досить рідко. Крім того, під час експериментів вчені виявили, що вплив на флеш-пам'ять високої температури збільшує швидкість стирання даних в осередках. Інженери припускають, що використання якогось постійного термального режиму може підвищити продуктивність накопичувачів.

Для того щоб використовувати **великі мобільні диски**, вам знадобиться так званий mobile rack (читається «мобіл рек»). Суть ідеї дуже проста. У корпусі комп'ютера встановлюються реєчки, по яких вінчестер, що упакований у скриньку, в'їжджає всередину і підключається до роз'ємів. Коли потрібно поміняти диск, витягуєте один і вставляєте інший, теж у скринці. Найчастіше змінним роблять другий вінчестер комп'ютера, тоді як перший, головний (де знаходиться операційна система), стоїть постійно.

1.12. Принтери

Принтер це пристрій виведення інформації на папір, його назва утворена від англійського дієслова to print – друкувати. Найстаріші моделі принтерів для персональних комп'ютерів – це матричні (1964 р., фірма Epson). Вони друкують спеціальними голочками, що ударяють по фарбувальній стрічці й, таким чином, малюють деяку комбінацію крапок. Із крапок і складається буква, цифра або малюнок. Матричні принтери дешеві, серед них є 9-, 16- і 24-голкові, і останні навіть дають непогану якість, але працюють вони повільно і дуже гучні.

Лазерні принтери працюють швидко й безшумно (приблизно так само, як ксерокси), дають дуже високу якість друку (як у друкарні). Комп'ютер формує у своїй пам'яті «образ» сторінки тексту і передає його принтеру. Інформація про сторінку проектується за допомогою лазерного променя на барабан, що обертається, зі світлочутливим покриттям, яке змінює свої електричні властивості в залежності від освітленості. Після засвічення на барабан, що знаходиться під електричною напругою, наноситься фарбувальний порошок – тонер, частинки якого налипають на засвічені ділянки поверхні барабана. Принтер за допомогою спеціального гарячого валика протягує папір під барабаном; тонер переноситься на папір і «вплавляється» в нього, залишаючи стійке високоякісне зображення.

Лазерні принтери відносно дорогі, але кожна віддрукована на лазерному принтері сторінка обходиться дуже дешево: однієї заправки принтера вистачає на декілька тисяч сторінок.

Найпоширеніші сьогодні принтери – **струменеві** (jet printer, "струйники"). Струменеві принтери генерують символи у вигляді послідовності чорнильних крапок. Друкуюча головка принтера має крихітні сопла, через які на сторінку вибризкується швидковисихаюче чорнило. Ці принтери вимогливі до якості

паперу, і можуть дати достатньо високу якість друку. Кольорові струменеві принтери створюють кольори, комбінуючи чорнило чотирьох основних кольорів – яскраво-блакитного, пурпурного, жовтого та чорного. Для нас істотним є те, що вони майже не шумлять (для домашнього застосування це вкрай важливо), набагато дешевше лазерних та набагато краще матричних.

Історія створення струменевих та лазерних принтерів почалася у 1933 році з історії всесвітньо відомої компанії Canon, коли два молодих токійських інженери, Горо Йошида та Сабуро Учида, відкрили «Лабораторію точних оптичних приладів», задавшись метою створити японську фотокамеру, яка не поступалася б кращим світовим зразкам. Спершу Йошида та Учида вирішили досконало розібратися у конкурентних перевагах продукції тодішніх лідерів на ринку фототехніки – німецьких фірм Leica та Contax – і закупили всі зразки німецьких камер, що були того часу на японському ринку. Розібравши куплені камери буквально на гвинтики, Йошида та Учида скрупульозно вивчили усі технічні хитрощі німецьких фотоапаратів. Тепер залишалося зробити наступний крок – навчитися робити камери не просто "як у німців", а набагато краще за них.

Не пройшло і року, як Йошида створив прототип першої японської 35-міліметрової камери із затвором. Автор винаходу назвав своє творіння ім'ям тисячерукої буддійської богині милосердя – Kwanon. Нова камера викликала справжній фурор у країні, яка у ті часи готувалася підкорити якщо не всю земну кулю, то хоча б тихоокеанський регіон, і прагнула бути світовим лідером в усьому – від підводних човнів до побутових приладів. Богиня милосердя допомогла своїй "хрещениці", і фотокамера Kwanon стала ще одним видатним досягненням японської технічної думки.

Самий час був виводити новий фотоапарат на світовий ринок. Для цього довелося змінити назву бренду – буддійське ім'я Kwanon перетворилося на співзвучне латинське Canon, таке, що теж має релігійний сенс (так зветься догматичне зведення священних текстів), але, крім того, ще означає і "норма", "правило". Таким чином, вже своєю першою моделлю новоспечена компанія підняла планку якості на максимальну висоту – віднині японська фототехніка повинна була стати еталоном для всього іншого світу.

В умовах підвищеного попиту на фотоапарати Canon "Лабораторія точних оптичних приладів" гостро потребувала додаткових коштів для розвитку. Це змусило засновників компанії зробити її публічною, і в серпні 1937 року була зареєстрована акціонерна компанія Precision Optical Industry Co., Ltd. Але з 1947 року вона почала називатися Canon Camera (останнє слово зникло з назви тільки у 1969 році, коли сфера діяльності Canon вже не обмежувалася фототехнікою).

Серед багатьох технологічних проривів, якими людство зобов'язане компанії Canon, – це винахід струменевого та лазерного принтерів.

Історія струменевого принтера, як це часто буває з революційними винаходами, почалася з випадковості. У 1977 році один із співробітників фірми впустив ввімкнений паяльник на шприц з чорнилом. Під впливом високої

температури на кінчику голки виникла чорнильна бульбашка, яка потім розтеклася тонкою цівкою. Використавши цей випадково виявлений ефект, Canon через чотири роки створила перший у світі струменевий принтер, який "в оригіналі" називався "бульбашковим", – bubble – jet. Це словосполучення зареєстроване як торговельна марка, тому усі інші струменеві принтери використовують іншу назву – ink – jet (буквально "чорнильно-струменевий"). Незабаром на ринку з'явився і перший недорогий "персональний" струменевий принтер BJ – 80 Bubble Jet.

Компанії належить і пріоритет у створенні персональних лазерних принтерів. Ці дослідження співробітники Canon почали ще у 1960-х роках, проте на той час експерименти довелося припинити – японцям ніяк не вдавалося створити портативне джерело лазерного випромінювання. Через десятиліття фахівці Canon дізналися, що американці почали застосовувати на практиці газові лазери. Японці дуже швидко вигадали, як пристосувати газовий лазер до завдань електрографії. У 1975 році прототип лазерного принтера викликав сенсацію на японській Національній комп'ютерній конференції, і першим, хто тут же купив у Canon ліцензію на випуск лазерних принтерів, став директор знаменитої американської компанії Hewlett-Packard Уильям Хьюлетт.

А після того, як фахівці Canon винайшли, як зробити лазерний принтер портативним (за допомогою того самого картріджа, який раніше успішно використовувся у копіювальних пристроях), Hewlett-Packard вчепилася за новинку обома руками. Конкуренти вперше уклали взаємовигідний альянс. Японська сторона взяла на себе "начинку" принтерів, а американська – корпуси, програмне забезпечення та всесвітню дистрибуцію під власною торговельною маркою. І сьогодні, за даними журналу Forbes, обидві компанії контролюють до 70% світового ринку лазерних принтерів.

Із принтерів для кольорового друку у домашніх умовах застосовуються поки що тільки струменеві. Останніми роками з'явилися пристойні та недорогі кольорові струменеві принтери, що друкують у чотири–вісім фарб; якість – близька до фотографічного зображення, особливо коли застосовується спеціальний папір. Проте при якісному кольоровому друці фарба у змінних балончиках (картриджах) закінчується досить швидко, а самі картриджі коштують недешево, що призводить до серйозного збільшення вартості кожного відбитку.

Ще фірмові картріджі оснащені спеціальними захисними чіпами-лічильниками, які не дають використовувати картридж після відпрацювання певного числа сторінок. «Лічильник» на картриджі, правда, можна «обнулити» за допомогою спеціальних програм, що перезарядчікі і роблять. Тому часто докуповують додатково до принтера спеціальну «станцію подачі чорнила» (СНПЧ). Фактично це просто пляшечки з чорнилом, приєднані гнучкими силіконовими шлангами до пристрою, що заміняє оригінальний картридж. У підсумку відбиток на струйниках обходиться приблизно в 10 разів дешевше.

Насправді хитромудрі фірми-виробники свідомо занижують ціни на принтери, щоб стимулювати продажі, а ціни на картриджі, навпаки, завищують.

Отже, купуючи принтер, добре подумайте, під які завдання він вам потрібний. Якщо мета – друк текстів (не дуже часто й помалу), можна обмежитися матричним або чорно-білим струменевим принтером. Якщо ж текстів належить друкувати багато і потрібна висока якість друку, то краще це робити на лазерному принтері. А якщо збираєтеся друкувати кольорові графіки та діаграми, роздруковувати зображення зі свого цифрового фотоапарата, результати своєї дизайнерської або художньої роботи, купуйте кольоровий струменевий принтер (під графіки та діаграми найдешевший, під фотографії – високоякісний).

Альтернатива лазерним принтерам – термічні принтери. Використовуються для отримання кольорового зображення фотографічної якості. Вимагають особливого паперу. Такі принтери придатні для ділової графіки. Професійні кольорові термічні принтери працюють краще, ніж струменеві, а коштують поки ще дорожче. Як, втім, і будь-яке професійне устаткування.

Принтери технології Micro Dry також дають повні фотонатуральні кольори, мають найвищу роздільну здатність. Це новий конкурентоздатний напрям. Передбачається, що вони будуть набагато дешевші за лазерні і струменеві принтери. Розробник – фірма Citizen. Друкує на будь-якому папері або картоні.

Принтери розрізняються за швидкістю друку – від 1 сторінки тексту за п’ять хвилин у матричних принтерах до 10 сторінок за хвилину й вище у лазерних принтерах.

Крім того, у принтерів різна якість друку: роздільча здатність може бути 90, 180, 300, 600, 1200 dpi, тобто крапок на дюйм (dot per inch). Наприклад, роздільна здатність 300 dpi забезпечує цілком професійну (придатну і для друкарні) якість при друці текстів і простих чорно-білих зображень без напівтонів, а 1200 – будь-яких чорно-білих зображень, включаючи напівтонову графіку. Деякі принтери дають різну роздільну здатність по вертикалі та по горизонталі. Тоді пишуть: 300x600 dpi.

Ще одна важлива відмінність – розмір паперу, який у цей принтер можна вставляти. Більшість побутових принтерів розрахована на стандартний лист (формат А4, розмір 210x297 мм). Але є принтери формату А4+ (збільшені у порівнянні з А4), формату А3 (297x420 мм), А3+ і т. д. Є гіганти форматів А2 і навіть А1. Але це вже дуже дороге устаткування, яке, якщо поставити вдома, займе цілу кімнату.

Зазвичай принтери підключаються до паралельного порту (LPT), нові моделі створюються і для USB. Кожен принтер обов’язково має свій **драйвер** – програму, яка здатна переводити (транслювати) стандартні команди друку комп’ютера у спеціальні команди, які потрібні для кожного принтера.

1.13. Миша

Пристрої **миша, джойстик, трекбол** називають **маніпуляторами** – це спеціальні пристрої, які використовуються для управління курсором.

Розглянемо комп'ютерну **мишу** (mouse), за допомогою якої можна малювати на екрані, а також пояснювати комп'ютеру, що взяти і куди віднести, що запустити, а що так залишити. Головне, чого ми чекаємо від миші, – це надійність.

Мишка класичної конструкції має дві кнопки, причому до появи Windows 95 в значній частині програм використовувалася тільки ліва кнопка (а то і зовсім ніяких кнопок – частина старих програм просто не використовували маніпуляторів).

Зараз досить популярними є трикнопкові миші, а також миші з коліщатком замість середньої кнопки (IntelliMouse). За допомогою такого коліщатка можна перегортати екран або запускати програми. Правда, додаткові можливості коліщатка підтримують не всі програми – скажімо, у браузері Internet Explorer та текстовому редакторові Microsoft Word можна перегортати сторінки, а в графічній програмі Photoshop не можна. Які саме операції має виконувати коліщатко або третя кнопка миші, часто можна задати самостійно – за допомогою особливої програми (драйвера миші).

Зустрічаються й більш екзотичні типи мишей – з додатковими кнопками (четвертою, п'ятою), з усілякими там кульками та роликами, миші під ліву руку.

Мишу краще за все переміщувати по спеціальному килимку, але з часом вона також бруднiться і починає працювати гірше. Тоді її доводиться розбирати і чистити: мити з милом важку прогумовану кульку, чистити ролики, протирати світлодіоди. Але зараз такі миші зустрічаються дуже рідко, їх витіснили миші без кульки – **оптичні**, дуже зручні, хоча і дещо дорожчі. І нічого в них не бруднiться.

Окремий різновид – **миші бездротові**. Поширено два основні типи таких пристроїв. Миша «інфрачервона» та «радіоміша». Недолік першої полягає у тому, що вона повинна весь час «бачити» приймач (ІЧ-ПОРТ). Якщо ви ненавмисно покладете на порт книжку або лист паперу, мишка вже не зможе спілкуватися з комп'ютером. Цього недоліку позбавлена миша, яка спілкується з комп'ютером через маленький радіопередавач.

У переносних комп'ютерах (ноутбуках) замість миші часто ставлять трекбол (track ball), пристрій у вигляді кульки та двох кнопок, що розміщені прямо на клавіатурі і такий, що виконує всі функції миші. Заміною миші у ноутбуці може стати і тачпад (touch pad) – окремий мініатюрний екранчик під клавіатурою, по якому ви водите просто пальцем.

У кишенькових моделях (palmtop) миша не використовується зовсім. Проте в деяких з них можна тиснути в екран олівчиком, і цей комп'ютер розуміє, чого від нього хочуть.

Для комп'ютерних художників та дизайнерів дуже корисним буде особливий різновид миші, яка називається **планшет**. По планшету (листу спеціальної конструкції) водять майже справжнім олівчиком, а малюнок з'являється на екрані. Якість планшетного малюнка може бути на порядок

вище, ніж малюнка, який отримується за допомогою миші. Але й ціна планшета також на порядок вища.

1.14. Сканери та камери

Втім, художникові все ж таки звичніше орудувати звичайним олівцем або пензлем. Тоді, щоб перевести свій малюнок у комп'ютерну форму, йому буде потрібен **сканер**. Сканер може перетворювати на комп'ютерні файли будь-які зображення – малюнки, фотографії, слайди. Якщо принтери виводять інформацію з комп'ютера, то сканери, навпаки, переносять інформацію з паперових документів у пам'ять комп'ютера.

Якщо за допомогою сканера вводиться текст, комп'ютер сприймає його як картинку, а не як послідовність символів. Для перетворення такого графічного тексту у звичайний символьний формат використовують програми **оптичного розпізнавання образів**.

Сканери бувають для чорно-білих та кольорових зображень (сьогодні нових чорно-білих в магазинах ви вже не знайдете), а також слайдові. Розрізняються вони, як і принтери, роздільною здатністю (300, 600, 1200 крапок на дюйм) та форматом паперу, який можна відсканувати.

За конструкцією (і, відповідно, за ціною) розрізняють ручні (дешеві, але для серйозних цілей непридатні), які протятують по поверхні документа рукою, планшетні (настільні) та професійні (наприклад, барабанні) сканери. Переважна більшість сканерів відповідає єдиному стандарту TWAIN, який добре знають система Windows та графічні програми.

У числі нових, але дуже перспективних зовнішніх пристроїв можна назвати **цифрові фото-** та **відеокамери**, які відрізняються від звичайних фотоапаратів та відеокамер своєю орієнтацією саме на комп'ютер, на цифрове подання інформації. За наявності цифрової фотокамери та гарного струменевого принтера можна обійтися без звичайного фотоапарату і не здавати плівку на прояв та друк. Не буде потрібний тоді і сканер, якщо ви захочете зробити свій фотоальбом в Internet або записати його на компакт-диск. При цьому можна прямо на вулиці, у лісі або в музеї подивитися на екранчик своєї камери, що у вас вийшло, а що ні, та негайно внести правки. Отже, відсоток вдалих карток зросте надзвичайно.

Фотозображення у цифровій фотокамері зберігається на змінних блоках пам'яті (**флеш-картах**). Є також моделі із записом зображення на магнітний диск, на CD або інший подібний пристрій, але картки пам'яті зручніші.

Цифрові камери розрізняються за багатьма параметрами – якості оптики, збільшенню зображення (zoom), типу флеш-карток, які в них можна використовувати. Але найважливіша відмінність – кількість крапок в одному кадрі: чим їх більше, тим вище якість знімка. Сучасні цифрові апарати мають матрицю розміром 3–11 мегапікселів (мільйонів крапок), їхні знімки можна не тільки переглядати на екрані або друкувати на домашньому принтері, але й

застосовувати в професійних цілях – наприклад, для створення кольорових ілюстрацій поліграфічної якості.

Переведення зображень з “цифровика” у комп’ютер зазвичай достатньо просте: ви підключаєте апарат до комп’ютера через USB-порт, і перекачуєте все, що потрібне, за допомогою програм, які входять у комплект камери. А часто не потрібні і ці програми: нові версії операційної системи Windows (2000, XP та Win 7) вміють працювати з USB-портами (у тому числі з цифровими фотоапаратами) безпосередньо, без програм-посередників.

1.15. Модеми

Модем (слово це означає «**мо**дулятор-**дем**одулятор») дозволить нам підключатися до комп’ютерних мереж найчастіше через телефонну лінію, передавати та приймати дані. Для нас істотним є швидкість передачі даних через модем та стійкість модему на вітчизняних телефонних лініях.

На даний час існують два види модемів: аналогові та цифрові (технологія xDSL). Аналогові модеми популярніші через свою дешевизну і використовуються в основному для виходу в мережу Internet, і лише інколи, через невисоку швидкість передачі даних (до 56 Кбіт/с, Kbps – кілобіт за секунду) – для зв’язку з іншими ПК. Цифрові модеми використовуються для високошвидкісних з’єднань з мережею Internet, або для організації локальної мережі на великих відстанях (xDSL модеми дозволяють передавати та приймати інформацію зі швидкістю до 5 Мбіт/с на відстані 5-7 км.).

Треба також знати, чи сумісний ваш модем зі стандартами Windows. Взагалі, у цій області особливо важливо купувати продукцію відомих фірм (на кшталт US Robotics (вона ж 3Com), Zoom або ZyXel).

Існують ще **факс-модеми**, які об’єднують у собі модем з апаратом факсимільного зв’язку. Користуючись факс-модемом, ви зможете, крім усього іншого, відправляти та приймати факси (за наявності спеціальної програми). Факс-модеми не набагато дорожчі простих модемів, а можливості, як бачимо, ширші. **Факс** – це пристрій факсимільної передачі зображення телефонною мережою. Назва "факс" відбулася від слова "факсиміле" (лат. fac simile – зроби подібне), що означає точне відтворення графічного оригіналу (підпису, документа і так далі) засобами друку.

Модеми мають декілька типів з’єднань з ПК: COM, USB або (для цифрових модемів) за допомогою мережної карти.

Зовнішній (external) модем підключають зазвичай до роз’єму послідовного порту (COM1 або COM2, USB-порт. Модем, з’єднання якого здійснюється через COM-порт, вимагає додаткового джерела (блоку) живлення, а при з’єднанні за допомогою USB-порту потреба у блоці живлення відпадає. xDSL-модеми також вимагають додаткового джерела живлення.

Бувають і **вбудовані**, або **внутрішні (internal)**, **модеми**, вони розміщуються всередині комп’ютера на одному із внутрішніх розйомів материнської плати (PCI або ISA), і не займають порт (а також місця на столі).

Внутрішні модеми дешевші (за рахунок того, що у них немає корпусу та окремого блоку живлення), а за якістю вони нічим не гірші. Але у зовнішніх модемів є усілякі миготливі лампочки, по яких можна контролювати, працює ваш модем або завис, а також кнопка відключення, яка дозволяє скинути (обнулити) такий завислий модем.

1.16. Звукова карта. Аудіо система

Комп'ютерів без звукової карти сьогодні вже не випускають. Але серед старих моделей німих скільки завгодно. Любителі гарного звуку не полінуються і придбають собі будь-який недорогий (або, навпаки, дорогий) пристрій такого типу, і встановлять його у свій комп'ютер (на роз'єм ISA або PCI), завдяки чому звук (у тому числі і з CD-ROM) можна буде прослуховувати через зовнішні стереодинаміки. Крім того, з'явиться можливість записувати у комп'ютер живий звук через мікрофон, обробляти його за допомогою музичних програм та відтворювати. За допомогою деяких спеціалізованих музичних програм можна навіть писати у комп'ютер музику.

Фірма Creative Labs, яка на даний час поки що не втратила своїх лідируючих позицій на ринку пристроїв мультимедіа, почала випускати перші моделі плат, які призначені для відтворення якісного звуку. Ці пристрої називаються просто – Sound Blaster (**аудіоадаптери**), тобто звукова плата. Даний термін, як і у випадку з «ксероксом», прижився настільки, що **саундбластерами** до цих пір називають найрізноманітніші звукові плати, навіть ті, які до Creative Labs не мають анінайменшого відношення.

Як вже згадувалось раніше, у побудові комп'ютерів прийнято для різних важливих напрямів – відео, звук – значну частину навантаження на процесор, чіпсет, пам'ять та інші складові перекладати на спеціалізовані пристрої (плати). Так зроблено у відеокартах – своя пам'ять, свій чіпсет, – так робиться і для звукових плат. Таким чином, звукова плата є як би спеціалізованим комп'ютером для синтезу та обробки звуку. Це може бути досить простий пристрій, а може бути і вкрай складний.

Аудіоадаптер містить у собі два перетворювачі інформації:

– **аналого-цифровий**, який перетворює безперервні (тобто, аналогові) звукові сигнали (мову, музику, шум) у цифровий двійковий код і запише їх на магнітний носій;

– **цифро-аналоговий** – такий, що виконує зворотне перетворення збереженого у цифровому вигляді звуку в аналоговий сигнал, який потім відтворюється за допомогою акустичної системи, синтезатора звуку або навушників.

Звукова карта може комплектуватися динаміками, і тоді ми називаємо її звуковою приставкою. Якщо ж динаміків немає, то для відтворення згодиться будь-який зовнішній підсилювач, навушники або касетний магнітофон.

Є таке поняття – **мультимедійний комп'ютер**. Нічого складного у цьому немає. Якщо будь-хто запитає, то так і скажіть: мовляв, мультимедіа – це коли є CD-ROM (або DVD) та звукова приставка, завдяки чому можна слухати музику, дивитися відеофільми та мультики прямо з компакт-диска. Власне, усі комп'ютери, які випускаються останніми роками, комплектуються CD-ROM-ом та звуковою картою. Але мультимедійними чомусь називають тільки ті, у яких є колонки для відтворення.

Колонки, до речі, теж вважаються комп'ютерним зовнішнім пристроєм, іноді їх навіть вбудовують прямо у монітор (це називається мультимедійний монітор). Але, чесно кажучи, якість звуку в них залишає бажати на краще. Так, більшість доступних за ціною комп'ютерних колонок дуже погано відтворюють низькі частоті («баси»). Ігроманам це, звичайно, не так важливо, а ось для меломанів правильним вибором було б відтворення звуку через хорошу домашню аудіосистему.

Сучасні "аудіофили" віддають перевагу якісному стереозвуку з об'ємним звучанням і глибоким басом, не говорячи вже про ентузіастів, які займаються створенням музики за допомогою своїх персональних комп'ютерів. Для них взагалі обов'язковим елементом домашньої студії є якісна **стереоакустика**, навіть якщо решта всієї ролі покладена на комп'ютер із звуковою платою.

У наші дні на ринку є дуже багато **акустичних систем**, які складаються з двох активних колонок та виконаних за системою 2.1. Подібні системи в народі називаються «пискавками», тому що вони не здатні забезпечити звук високої якості навіть на низькому рівні гучності. Зовсім нещодавно ідеалом у світі комп'ютерних акустичних систем була система **5.1** (п'ять **сателітів** і один **сабвуфер**), але останнім часом виробники акустики розширюють можливості своїх систем, що призвело до появи системи 6.1, а пізніше 7.1 та 8.1.

Потужність колонок вимірюється у ватах. Майже всі виробники пишуть на коробках абсолютно фантастичні цифри: 120 Вт, 200 Вт, і таке інше. Однак позначають ці цифри не реальну, а пікову музичну потужність (PMPO). Реальна ж потужність (RMS) нижче пікової в десятки разів. Так, 100 Вт PMPO будуть відповідати всього 5 «правильним» Вт.

Частотні характеристики – чи не головний показник, який свідчить про якість колонок. Людське вухо (в ідеальному випадку) може сприймати звук в діапазоні від 20 до 20 000 Гц. Більшість малопотужних (до 10 «реальних» Вт) і недорогих колонок здатні забезпечити відтворення звуку в діапазоні 100-8000 Гц. Музичні («меломанські») ж колонки забезпечують якісний звук в діапазоні 55-31000 Гц, і це вже велике досягнення.

1.17. Мережна карта

Якщо вам треба об'єднати пару комп'ютерів у домашню мережу, щоб обмінюватися даними, запускати програми, які є на іншому комп'ютері, але відсутні у вас, разом працювати або грати в ігри, заходити в Internet по одній телефонній лінії, – то слід встановити мережну карту. Мережні карти

ставляться на звичайний слот материнки або на шину PCI. Розрізняються за швидкістю передачі інформації (10 або 100 Мбіт/с) та за типом кабелю, яким ці картки з'єднуються (коаксіальний кабель або вита пара).

Так, не складе великих труднощів об'єднати у мережу старий комп'ютер, який вже не підлягає модернізації, та новий. Або великий комп'ютер та ноутбук. Або ж кинути кабель із сьомого поверху на п'ятий: два комп'ютери, дві картки та 20 метрів кабелю.

1.18. Бездротові з'єднання (Wi-Fi-мережі, Bluetooth)

Популярна система бездротового Ethernet, відома як 802.11b (з підтримкою сімейства протоколів 802.11), або **Wi-Fi**, допоможе вам активно підключитися з будь-якої точки у приміщенні без необхідності підключення дротів, та зробити мережу і засоби, що підключаються до мережі, набагато гнучкішими. А швидкий доступ до Internet з магазину або вестибюля аеропорту може змінити метод, за допомогою якого ви працюєте або граєте у режимі online, коли знаходитеся далеко від домашньої бази.

У передачі даних бездротовою мережею беруть участь три компоненти: радіосигнали, формат даних та структура мережі. Кожен з цих компонентів не залежить від двох інших. У мережну структуру входять адаптери інтерфейсів та базові станції, які передають та приймають радіосигнали.

У бездротовій мережі адаптери на кожному комп'ютері перетворять цифрові дані у радіосигнали, які вони передають на інші мережні пристрої. Вони ж перетворять вхідні радіосигнали від зовнішніх мережних елементів знов у цифрові дані.

Базовий стандарт 802.11 (без індексу "b" на кінці) був прийнятий у 1997 році. Він був зорієнтований на декілька бездротових середовищ: два види радіопередачі та мережі з використанням інфрачервоного випромінювання. Схожий документ, 802.11a, описує бездротові мережі, які працюють на вищих швидкостях та інших радіочастотах. Але на сьогодні найпоширішеною специфікацією є сучасний стандарт 802.11b. Це стандарт де-факто, який використовується практично у кожній Ethernet-мережі.

Бездротова мережа все ще є технологією, яка розвивається, тому вичерпну інформацію про можливості та з встановлення мережі можна знайти у книзі: Джон Росс. Бездротова мережа. -М: НТ Прес, 2007. Книга також запропонує вам деякі ідеї щодо того, як якнайкраще використовувати бездротову мережу після її встановлення, та як розв'язувати питання, які пов'язані з її роботою.

Ще один важливий стандарт бездротового зв'язку – **Bluetooth** – найчастіше зустрічається в ноутбуках. Правда, повільний Bluetooth-канал (стійкий зв'язок між пристроями по ньому можливий лише на відстані до 10 метрів) для передачі великого об'єму даних не годиться. Найчастіше він використовується для зв'язку ноутбука з іншими мобільними пристроями – КПК або мобільним телефоном. Втім, часто Bluetooth-канал задіюється і при роботі в Інтернет. Швидкість передачі даних, безумовно, буде дуже маленькою (порядку 100

кілобіт за секунду), але для отримання пошти та перегляду сторінок вистачить і цього. Нарешті, третя вигода Bluetooth: цей протокол дозволяє вам використовувати бездротові навушники або гарнітуру.

1.19. Плоттери (графічні пристрої)

Ці пристрої застосовуються тільки у певних областях для отримання складних конструкторських креслень, архітектурних планів, географічних та метеорологічних карт, ділових схем, графіків, діаграм тощо. Широке застосування знайшли **плоттери** спільно з програмами систем автоматичного проектування, де частиною результатів роботи програми стає конструкторська або технологічна документація. Незамінні плоттери і при розробках архітектурних проектів.

Ділянка креслення плоттера відповідає форматам А0-А4, хоча є пристрої, які працюють з рулоном та не обмежують довжину креслення, що виводиться (він може мати довжину у кілька метрів). Тобто, розрізняють планшетні та барабанні плоттери.

1. *Планшетні плоттери*, в основному для форматів А2-А3, фіксують аркуш та наносять креслення за допомогою пишучого вузла, який переміщається по двох координатах. Вони забезпечують вищу у порівнянні з барабанними точність друку малюнків та графіків.

2. *Рулонний* (барабанний) плоттер – залишається фактично єдиним видом плоттера, який розвивається, з роликовою подачею листа та пишучим вузлом, що переміщується по одній координаті (по іншій координаті переміщується папір).

Зв’язок з комп’ютером плоттери, як правило, здійснюють через послідовний (COM), паралельний (LPT) або SCSI-интерфейс. Деякі моделі графічних пристроїв оснащуються вбудованим буфером (1 Мбайт та більше).

На даний час стандартом де-факто для планшетних графічних пристроїв є пристрої фірми Hewlett-Packard. Крім того, графічна мова HP-GL (Hewlett-Packard Graphics Language) також стала фактичним стандартом у промисловості.

У плоттерах можуть використовуватися як спеціальні технології (наприклад, електростатичні), так і технології, які добре знайомі по принтерах (термо-, лазерна, LED, струменева). На даний час струменеві пристрої набувають все більшого поширення. Наприклад, плоттери Hewlett-Packard родини DesignJet формату А0 та А1 працюють у 4-5 разів швидше, ніж їхні пір’яні побратими. Використовуючи два струменеві чорнильні картріджи, струменевий плоттер працює з роздільною здатністю не гірше 300 dpi та має два режими: чистовий та ескізний. Алгоритм, який застосовується в ескізному (чорному) режимі, дозволяє майже вдвічі скоротити витрату чорнил.

1.20. Графічний планшет

Графічний планшет – це спеціальний пристрій введення інформації та управління, який забезпечений мишею особливого типу та пером. Використовується бездротова миша, але працює вона за абсолютно іншим принципом, ніж звичайна оптична та бездротова миша. Уся справа тут у графічному планшеті, який являє собоює невеликий пластиковий прямокутник, що приєднується до комп'ютера. По прямокутнику можна водити цією спеціальною мишею, а можна малювати на ньому пластмасовим пером, як звичайним олівцем.

Основна сутність подібного пристрою – у пері. Воно чутливе до натиснення (тобто чим міцніше на нього натиснути, тим товще лінія, що проводиться), та забезпечене додатковою клавішею і «гумкою», яка розташована з верхнього торця. Таким чином, можна малювати на комп'ютері практично так само, як звичайною ручкою або олівцем.

Один з визнаних лідерів в області виробництва графічних планшетів – це фірма Wacom. Вона випускає як дорогі професійні пристрої, так і цілком дешеві варіанти для родини.

1.21. Проекційна техніка

Мультимедіа-проектори увійшли до нашого життя у кінці XX сторіччя, і зараз без них неможливо представити багато сфер людської діяльності. Це навчальний процес, презентації, шоу-бізнес та домашнє кіно. Мультимедіа-проектор дозволяє відтворювати на великому екрані інформацію, яка отримується від найрізноманітніших джерел сигналу: комп'ютера, відеомагнітофона, відеокамери, фотокамери, DVD-програвача, ігрової приставки. Сучасний проектор – найдосконаліша ланка у ланцюжку еволюції проекційного устаткування, початок якій поклали **слайдпроектори**, які дозволяють демонструвати на великому екрані фотографічні діапозитиви. Їм на зміну прийшли так звані **оверхед-проектори**, що проектують зображення з просвічуваних матеріалів великих розмірів. Можливості сучасних мультимедіа-проекторів справді безмежні у порівнянні з їх попередниками.

Зображення у мультимедіа-проекторі формується кількома основними способами: за допомогою **рідкокристалічних панелей (LCD-технологія)** та за допомогою **мікродзеркальних чіпів DMD (DLP-технологія)**. У LCD-проекторах світло від лампи проходить через рідкокристалічну панель, на якій як на звичайній плівці, але за допомогою цифрової електронної схеми, створюється картинка. Світло проходить через панель і об'єктив, і в результаті на екран проектується збільшене у багато разів зображення. У DLP-проекторах світло від лампи відбивається від безлічі керованих електронікою мікродзеркал і також через об'єктив потрапляє на екран.

Основна характеристика мультимедіа-проектора – його яскравість, або світловий потік. Чим сильніший світловий потік, тим більший розмір

зображення можна отримати при заданій освітленості та якості матеріалу екрану. Світловий потік (який вимірюється в ANSI-люменах) залежить від конструкції проектора, якості LCD-панелей, потужності та типу лампи.

Роздільна здатність LCD-панелі або DMD-чіпа – наступний важливий параметр, що впливає на вибір проектора. Більшість панелей та чіпів розробляється з урахуванням стандартної роздільної здатності, яка прийнята для комп'ютерів: 640x480 (VGA), 800x600 (SVGA), 1024x768 (XGA), 1280x1024 (SXGA). Якщо ж роздільна здатність проектованого зображення відрізнятиметься від базової роздільної здатністі проектора (тобто, його LCD-панелі або DMD-чипа), воно буде перераховано при відтворенні за допомогою спеціального алгоритму практично без втрати якості.

Останнім часом почали з'являтися мультимедіа-проектори з LCD-панелями стандарту Wide XGA з роздільною здатністю 1366 x 768, які призначені в основному для перегляду відеозображень. Їх поява обумовлена популярністю «широких» екранів із співвідношенням сторін 16:9, замість традиційного 4:3.

Мультимедіа-проектор – сучасний та високотехнологічний пристрій. Надійність більшості моделей, що випускаються, велика, і користувачеві навряд чи доведеться звертатися до сервісного центру з проханням про ремонт. Єдина деталь проектора, яка замінюється – його лампа. У більшості проекторів використовуються дугові лампи з високою яскравістю та рівнішим, у порівнянні з лампами розжарювання, спектром. Середній термін їхньої служби – 2000 годин роботи. Іноді буває корисно застосовувати функцію економного режиму роботи лампи, який вдвічі продовжує її ресурс.

1.22. Системна шина

В основу архітектури сучасних персональних комп'ютерів покладений магістрально-модульний принцип. Модульний принцип дозволяє споживачеві самому комплектувати потрібну йому конфігурацію комп'ютера та змінювати модернізацію. Модульна організація комп'ютера спирається на магістральний (шинний) принцип обміну інформацією між пристроями.

Системна шина або магістраль включає три багаторозрядні шини (багатодротові керовані електричні лінії):

- шину даних;
- шину адреси;
- шину управління.

Основною функцією системної шини є передача інформації між базовим мікропроцесором та іншими електронними компонентами комп'ютера. По цій шині здійснюється також адресація пристроїв і відбувається обмін спеціальними службовими сигналами. Таким чином, спрощено системну шину можна представити як сукупність керованих сигнальних ліній, які об'єднані за їх призначенням (дані, адреси, управління).

Шина даних. Цією шиною дані передаються між різними пристроями. Наприклад, прочитані з оперативної пам'яті дані можуть бути передані процесору для обробки, а потім оброблені дані можуть бути відправлені назад в оперативну пам'ять для зберігання. Таким чином, дані по шині даних можуть передаватися від пристрою до пристрою у будь-якому напрямку. По цій шині здійснюється, також обмін даними між мікропроцесором та зовнішніми пристроями через спеціальні регістри (порти) пристроїв сполучення з мікропроцесором (контролери введення/виведення). При цьому по шині адреси попередньо передаються адреси (номери портів) цих пристроїв, а звернення до оперативної пам'яті чи до портів зовнішніх пристроїв забезпечується шиною управління. Обмін інформацією здійснюється за принципом рукостиснення, тобто зовнішній пристрій або мікропроцесор очікують готовність до обміну мікропроцесора або зовнішнього пристрою до обміну. Далі, на зовнішній пристрій можуть посилатись певні сигнали управління (наприклад, запуск двигуна принтера, встановлення блоку головок вінчестера на певний циліндр, тощо) для відпрацювання певних дій і тільки після успішного їх виконання відбувається власно обмін інформацією. У відповідності до цього кожний контролер зовнішнього пристрою має три види портів: порт стану, порт управління та порт даних.

Розрядність шини даних визначається розрядністю процесора, тобто кількістю двійкових розрядів, які процесор обробляє за один такт. Розрядність процесорів постійно збільшувалася по мірі розвитку комп'ютерної техніки.

Шина адреси. Вибір пристрою або елементу пам'яті, куди пересилаються або звідки читаються дані по шині даних, здійснює процесор. Кожен пристрій або елемент оперативної пам'яті має свою адресу. Адреса передається по шині адреси, причому сигнали по ній передаються тільки в одному напрямі від процесора до оперативної пам'яті та портів (спеціальних регістрів) зовнішніх пристроїв (односпрямована шина).

Розрядність шини адреси визначає адресний простір процесора, тобто кількість елементів оперативної пам'яті, які можуть мати унікальні адреси. Кількість елементів пам'яті, до яких можна звертатися за їхніми адресами розраховується за формулою: $N = 2^I$, де I – розрядність шини адреси (ширина шини). Наприклад, якщо ширина адресної шини складає 16 біт, і розмір слова пам'яті дорівнює одному байту (мінімальна одиниця даних у пам'яті), то об'єм пам'яті, який можна адресувати, складає 2^{16} = 65536 байтів (64 КБ). Сучасні персональні комп'тери припускають максимальний розмір оперативної пам'яті – від 2^{32} = 4 Гб до 2^{64} = 16 Еб, а простір для портів зовнішніх пристроїв 2^{16} = 64 Кб. Насправді фізично шина адреси та шина даних – це одна і та сама шина, яка використовується за тим чи іншим призначенням за допомогою спеціальних сигналів управління.

Шина управління. По шині управління передаються сигнали синхронізації та сигнали управління, які визначають характер обміну інформацією по магістралі. Сигнали управління визначають, яку операцію – читання або запис інформації – з пам'яті треба виконатии, подають певні

сигнали управління (команди) на зовнішні пристрої, синхронізують обмін інформацією між пристроями тощо.

Якщо розглянути історію перших 16-розрядних процесорів класу x86, 8086, що були розроблені компанією Intel, починаючи з 1978 року, то чіпи того часу працювали на частоті 5, 8 або 10 Мгц і, завдяки 20-розрядній шині адреси, дозволяли адресувати 1 Мб оперативної пам'яті (2^{20} = 1 Мб).

На той час популярними були 8-бітові комп'ютери, тому Intel розробила інший чіп – 8088, який був апаратно та програмно сумісний з 8086, але оснащений лише 8-розрядною шиною даних.

У 1982 році компанія Intel представила процесор 80286, який був сумісний з обома попередніми моделями, але використовував «ширшу», 24-розрядну, шину адреси. Цей процесор дозволяв адресувати 16 Мб оперативної пам'яті.

Чотири роки потому, у 1986 році, Intel випустила процесор 80386 DX, у якого обидві шини (шина даних та шина адреси) вже були 32-розрядними.

1.23. Кнопки та клавіші

На корпусі комп'ютера є клавіша або кнопка ввімкнення живлення. Зазвичай на ній написано Power (живлення) або 1-0 (вкл. – викл.). У старих моделях її любили заховати в незручне місце – наприклад, збоку на корпусі або навіть ззаду, там, де роз'йоми. Нині виробники комп'ютерів ставлять вимикачі на передній панелі процесорного блоку.

Сучасні корпуси можуть мати два вимикачі живлення: один на лицьовій панелі, а другий ззаду, на блоці живлення. На передній панелі будь-якого комп'ютера є також кнопка Reset (перезавантаження), а в старих моделях буває ще й Turbo (для збільшення або зменшення швидкості роботи комп'ютера) та відповідні їм індикатори.

На моніторі теж є кнопка ввімкнення живлення, але у багатьох моделях при вимкненні комп'ютера вимикається і монітор, тому користуватися нею не доводиться. Крім того, монітори мають спеціальні органи управління яскравістю, контрастом, кольоровістю – все таке саме, як у телевізорі, тільки краще.

3-дюймовий дисковод розташований на звичайному комп'ютері спереду (на ноутбуці – збоку) і виглядає як проріз, куди вставляється дискета. 3"-дискети роміщені у пластмасових корпусах, магнітний шар захищений від випадкового дотику металевою платівкою, яка сама зсувається у той момент, коли дискета вставляєтеться у дисковод. На лицьовій стороні дискети неодмінно що-небудь написано, тоді як на зворотній, якщо щось і написане, то значно менше.

Вставляти 3"-дискету треба металевою пластинкою вперед і лицьовою стороною догори. Якщо дискета встановлена вірно, кнопочка, розташована на дисководі біля прорізу, вискочить уперед. Якщо ви вставляєте дискету неправильно, то вона чинитиме опір. Тільки не треба застосовувати силу! Щоб дістати дискету, треба натиснути кнопку, і вона наполовину висунеться сама.

Дисковод CD-ROM має зазвичай кнопочку для відкриття та закриття, гніздо для підключення навушників, регулятор гучності звуку у навушниках та маленьку лампочку-індикатор (горить тільки тоді, коли у накопичувачі є компакт-диск, і до нього здійснюється звернення – якась програма його читає). У CD-R та CD-RW може бути ще індикатор іншого кольору, який спалахує, коли йде процес запису.

Компакт-диск кладуть у висувний лоток етикеткою догори, а дзеркальною поверхнею донизу і закривають, натиснувши кнопку або трохи штовхнувши лоток (тільки обережно: не всі дисководи правильно розуміють таке рукоприкладство).

Клавіатура комп'ютера – це пристрій для введення символьної інформації у комп'ютер та подачі управляючих сигналів. Містить стандартний набір клавіш як у друкарської машинки та деякі додаткові клавіші – управляючі та функціональні клавіші, клавіші управління курсором та малу цифрову клавіатуру.

На сучасній клавіатурі є зазвичай близько сотні клавіш та 3-4 лампочки (точніше, світлодіоди). У ноутбуків клавіатура має дещо інший вигляд і клавіш менше, а у кишенькових комп'ютерів може й зовсім не бути клавіатури. Бувають інші варіанти розташування клавіш і на клавіатурах звичайних комп'ютерів. Ну й, звичайно, існує маса різновидів "наворочених" клавіатур, з усілякими додатковими кнопками довільної форми та будь-якого іншого призначення.

Основна частина клавіатури – з буквами, цифрами та розділовими знаками – розташована по центру «кнопкової дошки» (так буквально перекладається українською англійське слово keyboard). Найпоширеніша сьогодні клавіатура з розкладкою клавіш **QWERTY** (читається "кверті"), названа так по клавішах, розташованих у верхньому лівому ряду алфавітно-цифрової частини клавіатури. Зазвичай на русифікованій клавіатурі зверху кожної клавіші написаний символ латинського алфавіту (та українські – "ї", "є"), а нижче (іноді – іншим кольором) – російського алфавіту.

Стрілками «донизу», «вгору», «ліворуч» та «праворуч» помічені клавіші управління курсором – тією мерехтливою рискою, що переміщується по екрану, відзначаючи те місце, «де ви зараз знаходитеся». У деякому розумінні можна сказати, що курсор – це і є ви, це та активна точка, яка діє за вас на екрані комп'ютера. Клавіші управління курсором розташовані внизу, правіше за основну клавіатуру, і продубльовані на правій, цифровій клавіатурі (на цифрах 2, 4, 6 і 8).

У нижньому ряду знаходиться найпомітніша (тому що найдовша) клавіша – пробіл або пропуск. Вище за неї – справа та зліва – дві довгі клавіші Shift, які призначені для переведення регістру (тобто переходу з прописних букв на заголовні). А щоб весь час друкувати заголовними буквами, можна натиснути клавішу Caps Lock (при цьому засвітиться індикатор Caps Lock – «фіксатор» верхнього регістра).

Нагорі розташовано 12 функціональних клавіш – від F1 до F12, – які призначені для виконання деяких спеціальних операцій, що найчастіше використовуються (для кожної програми або гри вони мають своє особливе призначення). Але, наприклад, клавіша F1 майже в усіх програмах робить одне й те саме – викликає довідку про програму.

Крім того, є низка спеціальних клавіш: Esc («ескейп» – втекти, відміна будь-чого – клавіша, яка дуже часто використовується); одна або дві клавіші Enter (введення, підтвердження, переміщення на початок наступного рядка – клавіша, яка найчастіше використовується), Tab (табуляція) – для встановлення курсору у розмічені позиції.

На сучасних клавіатурах завжди є такі пари найважливіших управляючих клавіш – Ctrl (контроль) та Alt (альтернатива). Ці клавіші (а також клавіша Shift) частіше за усе використовується не самостійно, а у сполученні з деякими іншими клавішами – для швидкого виклику команд. Права та ліва клавіші Ctrl та Alt можуть викликати різні дії (але, як правило, ці дії всеж-таки однакові).

Також по дві клавіші Del або Delete («деліт» – вилучити), Ins або Insert («інсерт» – вставити), Home («додому», перехід на початок тексту або рядка), End («у кінець»), PgUp (перегортання на сторінку догори) та Page Down або PgDn (перегортання на сторінку донизу), з їх допомогою можна переглядати текст. Клавіша Backspace («назад») на багатьох клавіатурах не довга, а звичайного розміру, і не завжди там пишуть слово – просто малюють стрілку вліво (←). Але не слід її плутати з відповідною клавішею управління курсором: це абсолютно різні клавіші! Backspace завжди знаходиться праворуч у верхньому ряду, над клавішею Enter (це може не стосуватися до якихось незвичайних екзотичних клавіатур, а також ноутбуків). Використовується, як правило, текстовими редакторами для стирання символу перед курсором та переміщення курсору на одну позицію ліворуч.

У правій частині клавіатури розташовується група клавіш із цифрами та знаками арифметичних дій – цифрова клавіатура. Коли була натиснута клавіша NumLock («фіксатор цифрової клавіатури») та горить відповідний індикатор, тоді з цифрової клавіатури вводяться цифри, а коли не натиснута, вона дублює дію стрілок управління курсором та клавіш Home, End, PgUp і PgDn. Сірі клавіші на цифровій клавіатурі (знак ділення «/», зірочка або знак множення «*», мінус «–», плюс «+») призначені для введення знаків арифметичних дій (якщо була натиснута клавіша NumLock), а якщо не була натиснута, то можуть виконувати і деякі спеціальні функції – у кожній програмі свої. На деяких сучасних клавіатурах ці клавіші зовсім не сірі, а такі ж білі або чорні, як і всі інші, але за звичкою їх іноді ще називають сірими. Називають їх і по-іншому – з приставкою Num (цифрова) або Dop (додаткова). Наприклад, Num+ – це плюс на цифровій клавіатурі, а Num5 – це ще й п’ятірка.

Клавіша Print Screen («друкувати екран») у режимі DOS дозволяє миттєво надрукувати все, що ви зараз бачите на екрані. У режимі Windows ця клавіша діє інакше: у поєднанні із клавішею Ctrl (Alt) вміст повного (активного вікна) екрану копіюється у вигляді картинки у буфер проміжного зберігання в

оперативній пам’яті, в якусь «кишеню», з якої картинку можна у будь-який момент дістати.

Клавіша Pause («пауза», найчастіше там написана інша назва – Break) у режимі DOS призупиняє роботу комп’ютера доти, доки не буде натиснута будь-яка інша клавіша. А у режимі Windows нічого не призупиняє, і навіщо вона потрібна, незрозуміло.

У нових моделях клавіатур введені ще й додаткові клавіші зі своїм специфічним призначенням, як, наприклад, пара клавіш Windows (на них намальований логотип Windows). Розташовуються вони зазвичай між клавішами Alt та Ctrl. При натисненні цієї клавіші викликається головне меню Windows.

Дуже корисною буває, особливо у текстових редакторах (наприклад Microsoft Word) ще одна клавіша, на якій зображена маленька табличка та стрілка курсора. Цією клавішею викликається так зване контекстне меню, через що вона називається Context.

Якщо натиснути клавішу (букву, цифру, інший символ) і потримати її близько секунди, то комп’ютер почне повторювати її (наприклад, ууууууу). Отже, натискати клавіши треба досить уривчасто. В системі Windows час, після якого починається це ууууууу, можна змінити – зменшити, або, навпаки, збільшити.

Дуже часто у комп’ютерному житті потрібно натискати не одну клавішу, а комбінацію клавіш. Наприклад, у більшості програм-редакторів – текстових, графічних, музичних та інших – для запису на жорсткий диск результатів нашої праці ми користуємось клавіатурною комбінацією CTRL-S (іноді пишуть Ctrl+S). При цьому клавіші Ctrl і S не потрібно натискати одночасно. Досить натиснути спершу клавішу Ctrl, а потім, не відпускаючи її, клавішу S. Текст ваш або малюнок піде записуватися на диск, а клавіші можна буде вже відпустити. Так само треба поступати, коли нам треба буде у текстовому редакторі набрати замість прописної букви заголовну. Натиснемо спершу клавішу Shift, а потім, не відпускаючи її, потрібну нам букву.

Бувають комбінації ще складніші – трьома пальцями (мається на увазі, наприклад, Ctrl-Alt-Del, Ctrl-Shift-Esc або Ctrl-Alt-1) і навіть із чотирьох (Ctrl-Alt-Shift-S). Клавіатура містить вбудований **мікроконтролер** (місцевий пристрій управління), який виконує такі функції:

– послідовно опитує клавіші, прочитуючи введений сигнал та виробляючи двійковий **скан-код** клавіші;

– управляє світловими індикаторами клавіатури;

– проводить внутрішню діагностику несправностей;

– здійснює взаємодію з центральним процесором через порти введення-виведення клавіатури та механізм переривання.

Клавіатура має власний **вбудований буфер** – проміжну пам’ять малого розміру, куди поміщаються введені символи. В разі переповнення буфера натиснення клавіші супроводжуватиметься звуковим сигналом – це означає, що

символ не введений (відкинутий). Але переповнення буферу практично неможливо, оскільки його розмір розрахований на максимальну швидкість роботи машиністки. Роботу клавіатури підтримують спеціальні програми, які "зашиті" у BIOS (низький рівень), а також драйвер (програма обслуговування більш високого рівня – рівня операційної системи) клавіатури, який забезпечує можливість введення українських або російських букв, управління швидкістю роботи клавіатури тощо.

Підводячи підсумки цього розділу, треба сказати, що разом із розвитком обчислювальних систем, стрімко розвиваються також інші галузі цифрового світу. На сьогоднішній день серед різноманіття цифрових пристроїв можна зустріти деякі пристрої, які ще кілька років тому не мали ані найменшого відношення до персональних комп'ютерів.

До таких пристроїв можна віднести цифрові фотоапарати. Фотографії з цифрових фотокамер дуже легко можна перенести на комп'ютер за наявністю USB-кабеля.

За допомогою комп'ютера можна завантажувати різні картинки або мелодії у сучасні мобільні телефони.

Контрольні запитання для самоперевірки

1. Які основні принципи полягли в основу моделі комп'ютера фон Неймана?
2. Які види пам'яті існують у комп'ютері?
3. Що являє собою кеш-пам'ять?
4. Що являє собою системна шина?
5. Які групи команд є у комп'тері?
6. Що являє собою форм-фактор процесора?
7. Які контролери прийшли на зміну IDE-контроллерам?
8. Які порти існують у комп'ютері для під'єднання зовнішніх пристроїв?
9. Які три компоненти використовуються для бездротових з'єднань?
10. Яке призначення модему?
11. Що являє собою системна шина та яке її призначення?
12. Які існують види зовнішньої пам'яті?
13. Які існують графічні цифрові пристрої?
14. Які види клавіш присутні на клавіатурі?

2. Класифікація комп’ютерів

2.1. Класифікація комп’ютерів за поколіннями

Дотепер прийнято виділяти п’ять поколінь обчислювальної техніки.

До **першого покоління** відносять машини, що були побудовані на електронних лампах. До цієї групи входять машини, які були створені у період, що починається з електронної обчислювальної машини "EDSAC" і закінчується приблизно у кінці п’ятидесятих років. Ці машини коштували дуже дорого, займали величезні площі, були не зовсім надійні у роботі, мали маленьку швидкість обробки інформації і зберігали дуже мало даних. Створювалися вони в одиничних примірниках, і використовувалися, головним чином, для військових та наукових цілей. Як типові приклади машин першого покоління можна згадати американські комп’ютери UNIVAC, IBM-701, IBM-704, а також радянські машини БЕСМ та М-20. Типова швидкість обробки даних для машин першого покоління складала 5-30 тисяч арифметичних операцій за секунду.

До **другого покоління** відносять машини, які були побудовані на транзисторних елементах у період з кінця п’ятидесятих і до середини шестидесятих років. У цих машин значно зменшилися вартість і габарити, виросли надійність, швидкість роботи та об’єм інформації, що у ній зберігався. Типові представники машин другого покоління – PDP-8, IBM-7094, CDC-6600 (США), ATLAS (Великобританія), БЕСМ-4, М-220, Мінськ-32, БЕСМ-6 (СРСР). Швидкість обробки даних у машин другого покоління зросла до 1 мільйона операцій за секунду.

Машини **третього покоління** виконані на так званих **інтегральних схемах**, які скорочено позначають ІС. Інтегральна схема є микромініатюрною електричною схемою певного функціонального призначення, яка за допомогою спеціальної технології розміщується на дуже маленькій кремнієвій (або будь-якій іншій відповідній за властивостями) платівці – "основі". Площа такої схеми порядку одного квадратного міліметра, але за своїми функціональними можливостями інтегральна схема еквівалентна сотням і тисячам транзисторних елементів. Через дуже маленькі розміри та товщину інтегральну схему іноді називають **мікросхемою**, а також **чіпом** (chip – тонкий шматочок). Завдяки переходу від транзисторів до інтегральних схем змінилися вартість, розмір, надійність, швидкість та розмір пам’яті машин.

Окрім переходу на нову елементну базу, важливо відзначити також, що машини почали випускатися **родинами**. Машини, що входять до родини, мають однакову логічну структуру, одні й ті ж способи обробки інформації, але різні параметри вартості, швидкості та об’єму даних, що в них зберігаються. Це дозволяє здійснювати широкий обмін програмами та даними між різними користувачами без внесення до програм істотних змін. Машини третього покоління з’явилися в середині шеситдесятих років. Це були машини родини IBM/360. Популярність цих машин виявилася настільки великою, що в усьому світі їх сталі копіювати, або випускати схожими за функціональними

можливостями та співпадаючими за способами кодування та обробки інформації. Причому програми, які були підготовлені для виконання на машинах IBM, успішно виконувалися на їхніх аналогах, так само як і програми, які були створені для виконання на аналогах, могли бути виконані на машинах IBM. Такі моделі машин прийнято називати **програмно-сумісними**.

Починаючи з третього покоління, обчислювальні машини стають повсюдно доступними, і широко використовуються для вирішення самих найрізноманітніших завдань. Характерним для цього часу є колективне використання машин, оскільки вони все ще достатньо дорогі, займають великі площі, і вимагають складного та дорогого обслуговування. Правда, доступ до машини вже організовується з пристроїв, які використовувались індивідуально – терміналів (terminal – кінцевий пункт) та знаходились на деякій відстані від основного устаткування машини, іноді навіть на робочих місцях користувачів. До складу терміналу, як правило, входять клавіатура (сукупність клавіш у різних пристроях), що використовувалась для набору даних та виконання простих операцій з управління роботою комп'ютера, та дисплей (display – показувати, демонструвати) або монітор. Як випливає з опису функцій терміналів, клавіатури та дисплею, ці пристрої відносяться до групи пристроїв введення-виведення. Носіями початкової інформації все ще є перфокарти та перфострічки, хоча вже значний об'єм інформації розташовується на магнітних носіях – дисках і стрічках. Швидкість обробки інформації у машин третього покоління сягала декількох мільйонів операцій за секунду.

Четверте покоління комп'ютерів з'явилося у першій половині семидесятих років після переходу від звичайних інтегральних схем до схем з більшою щільністю монтажу **– великих інтегральних схем (ВІС).** Якщо звичайні інтегральні схеми еквівалентні тисячам транзисторних елементів, то великі інтегральні схеми замінюють уже десятки та сотні тисяч таких елементів. На тлі цього переходу відбувся розподіл єдиного до цієї пори потоку розвитку засобів обчислювальної техніки на дві гілки. Одна гілка продовжувала стару тенденцію розвитку машин по лінії нарощування потужності та надійності, а також по лінії колективного використання обчислювальних потужностей. Серед них слід згадати родину машин IBM/370, а також модель IBM 196, швидкість якої досягла 15 мільйонів операцій за секунду.

Відмінна риса четвертого покоління – наявність в одній машині кількох (зазвичай 2-6, іноді до кількох сотень і навіть тисяч) центральних, головних пристроїв обробки інформації – процесорів, які можуть дублювати один одного або незалежним чином виконувати обчислення. Така структура дозволила різко підвищити надійність машин, а також швидкість обчислень. Інша важлива особливість – поява могутніх засобів, які забезпечують роботу комп'ютерних мереж. Це дозволило згодом створювати та розвивати на їхній основі глобальні, всесвітні комп'ютерні мережі.

Друга гілка розвитку засобів обчислювальної техніки виявилася спрямованою на мікромініатюризацію та персоналізацію засобів обробки

даних. Своїм народженням цей напрям зобов'язаний появі у 1971 році першого **мікропроцесора** Intel 4004 (від назви фірми-виробника INTegrated ELectronics – об'єднана електроніка). Мікропроцесором вважається процесор, який реалізований на одній або декількох інтегральних схемах, без втрати функціональних властивостей звичайних процесорів, тобто пристроїв, що забезпечують усі необхідні операції по обробці даних. Для мікропроцесорів введена окрема класифікація, за якою Intel 4004 відноситься до першого покоління мікропроцесорів.

Останнім на сьогоднішній день вважається п'яте покоління комп'ютерів. Про проект створення машин цього покоління, який був розрахований на десять років, оголосили на початку восьмидесятих років японські розробники. За ними в цю стратегічну гонку втягнулися вчені США, СРСР та деяких країн Західної Європи. Було заявлено, що на початок дев'яностих років буде створено принципово інше за стилем обробки інформації та взаємодії з користувачем покоління машин. Якщо раніше людина ретельно та детально формулювала машині послідовність дій з обробки інформації, то тепер машина, згідно з поставленою перед нею метою, повинна самостійно скласти план дій та виконати їх. Такий спосіб розв'язання завдань прийнято називати логічним програмуванням. Крім того, планувалося ввести спілкування з машиною на рівні природної мови. Проте розв'язати повністю весь комплекс завдань проекту до сих пір не вдалося. Хоча є вражаючі досягнення по кожному з напрямів проекту, виникли певні фінансові та технічні труднощі.

2.2. Класифікація комп'ютерів за застосуванням

Класифікація за застосуванням включає такі групи: мікропроцесори, мікрокомп'ютери (МІКРО-ЕОМ), мінікомп'ютери (МІНІ-ЕОМ), універсальні комп'ютери (універсальні ЕОМ) та суперкомп'ютери (супер-ЕОМ).

Мікропроцесорами є програмовані інтегральні схеми, які вбудовуються у будь-який окремий пристрій, механізм (автомобіль, металорізальний верстат, крилату ракету) з метою автоматизації управління або оптимізації роботи механізму. Якщо до мікропроцесора підключити інші необхідні для комп'ютера пристрої (пам'ять, пристрої введення-виведення), вийде МІКРО-ЕОМ, або мікрокомп'ютер. Обидві ці назви практично витиснені з побуту популярнішими назвами: персональна ЕОМ та персональний комп'ютер. За визначенням виходить, що персональний комп'ютер експлуатується, як правило, однією людиною або відносно невеликим колективом фахівців для розв'язання своїх професійних завдань.

2.2.1. Основні області застосування комп'ютерів

Застосування методів та засобів інформатики можливо в усіх тих областях людської діяльності, де існує принципова можливість (і необхідність) реєстрації та обробки інформації. З цього приводу існує чудовий вислів:

"Застосування обчислювальних машин обмежене тільки рамками наших фантазій". Зараз важко назвати таку сферу діяльності людини, в якій не використовують або не намагаються застосувати сучасні інформаційні технології. Нижче наведені приклади тільки найзначніших областей застосування засобів обробки даних.

1. Військова справа, наприклад, системи протиракетної оборони, космічні системи.
2. Моделювання фізичних явищ та дослідження побудованих моделей.
3. Обробка конкретних експериментальних даних при проведенні математичних, фізичних, хімічних, біологічних та інших досліджень.
4. Розв'язання завдань метеопрогнозу.
5. Автоматизовані робочі місця (АРМ) фахівця.
6. Системи автоматичного проектування, які забезпечують підтримку роботи інженера-конструктора та істотно підвищують продуктивність його праці.
7. Управління роботою окремих верстатів (верстати із числовим програмним управлінням), роботи, робототехнічні лінії, цехи та заводи-автомати.
8. Автоматизовані системи планування та управління виробництвом, починаючи з окремих підприємств, і закінчуючи управлінням цілими галузями (транспорт, авіація тощо).
9. Отримання зображень внутрішніх частин непрозорих тіл, зокрема в медицині – комп'ютерна томографія, і на виробництві – контроль якості, що не руйнує виробів.
10. Системи масового обслуговування та інформаційно-довідкові системи. Наприклад, системи резервування та продажу авіаквитків.
11. Бази даних правової інформації (швидкий доступ до нормативних актів, указів та ухвал уряду, статей Кримінального та інших кодексів), криміналістичні бази даних, які зберігають відомості про злочинців тощо.
12. Банківські та біржові комп'ютерні системи.
13. Підготовка різних документів, звітів та інших друкарських матеріалів.
14. Комп'ютерна верстка та підготовка до видання газет, журналів, книг.
15. Аранжування музичних творів, світломузика.
16. Комп'ютерний дизайн пристроїв, що розробляються, приміщень.
17. Комп'ютерний підбір зачісок, моделей одягу.
18. Комп'ютерна мультиплікація та анімація.
19. Машинний переклад із різних природних мов.
20. Лінгвістика, розшифровка невідомих мов.
21. Криптографія – шифрування та розшифрування документів, доступ до яких повинен бути обмежений.
22. Навчальні, тестуючі та контролюючі програми.

Необхідно все-таки згадати ще про одну, специфічну область "застосування" інформаційних технологій. Практично одночасно з появою персональних комп'ютерів та зростанням популярності комп'ютерних мереж

з’явилися програми, які були названі **комп’ютерними вірусами**. Основною метою розробки таких програм можна вважати нанесення шкоди апаратним засобам, програмам або даним конкурентів. Важливою відмінністю вірусів є їхня здатність до самоpосповсюдження, що дозволяє вірусам за невеликий проміжок часу "заразити" велику кількість комп’ютерів і завдати їм максимальної шкоди.

Детальніше слід обговорити такі відносно нові інформаційні поняття та технології, як **гіпертекст**, **мультимедіа,** інформаційні послуги у мережах.

2.2.2. Гіпертекст

При роботі з різного роду словниками та довідниками достатньо часто доводиться стикатися із ситуацією, коли у визначенні або перекладі будь-якого слова, терміну зустрічається посилання "див .", що направляє читача за отриманням додаткової інформації в якесь інше місце. Наприклад, у словнику іноземних слів наведено таке визначення терміну "інформація": "**Інформація**" [< лат.; див. Інформувати] – 1) повідомлення; 2) відомості, зберігання, що є об’єктом, переробки та передачі. Зустрівши таке посилання (див. Інформувати), доводиться, перегортаючи безліч сторінок довідника, шукати термін, на який дано посилання. Технологія гіпертексту значно полегшує, а точніше повністю автоматизує отримання додаткової інформації, яка пов’язана з такого роду посиланнями. Читачеві достатньо тільки певним чином вказати на обране посилання, усе інше виконає спеціальна програма.

Гіпертекстом є текст з посиланнями, що читається за допомогою спеціальної програми, яка автоматично знаходить у комп’ютері пов’язану з обраним посиланням додаткову інформацію, і виводить її на екран дисплея.

Посилання у гіпертексті прийнято називати ділянками. Ділянками можуть бути символи, які зустрічаються у тексті, поняття, словосполучення, зображення. Як правило, ділянки відрізняються від решти тексту яскравішим або інтенсивнішим кольором. Вибір будь-якої ділянки у гіпертексті автоматично призводить до виконання якої-небудь дії, наприклад, пошуку виділеного у ділянці визначення терміну. Це визначення може у свою чергу містити довільну кількість нових гіпертекстових посилань – полів. З полем може бути пов’язане виконання деякої додаткової програми, програвання фрагмента аудіо- або відеозапису. Після виконання пов’язаних із полем дій можна повернутися у початкове місце або перейти далі за новим посиланням, що зустрілося у викликаному фрагменті гіпертексту.

2.2.3. Мультимедіа

Спочатку комп’ютери вміли "працювати" тільки з числами. Трохи пізніше вони "навчилися" працювати з текстами та графікою. І лише в останньому десятилітті XX століття комп’ютер "засвоїв" звук та рухоме зображення. Нові можливості комп’ютера отримали назву **мультимедіа** (multimedia – множинне

середовище, тобто середовище, яке складається з кількох компонентів різної природи).

Мультимедіа – це сукупність апаратних та програмних засобів, які забезпечують створення звукових та візуальних ефектів, а також вплив людини на хід виконання програми, що передбачає їх створення.

Яскравим прикладом застосування мультимедійних можливостей є різні енциклопедії, в яких виведення тексту тієї чи іншої статті супроводжується показом пов'язаних з текстом зображень, фрагментів кінофільмів, синхронним озвучуванням тексту, що виводиться, тощо. Мультимедіа широко застосовується у навчальних, пізнавальних, ігрових програмах. Експерименти, що проводилися над великими групами осіб, які навчалися, показали, що в пам'яті залишається 25% почутого матеріалу. Якщо матеріал сприймається зором, то запам'ятовується 1/3 побаченого. У разі комбінованої дії на зір і слух частка засвоєного матеріалу підвищується до 50%. А якщо навчання організоване при діалоговому, **інтерактивному** (interaction – взаємодія) спілкуванні особи, яка навчається, та мультимедійних навчальних програм, засвоюється до 75% матеріалу. Ці спостереження свідчать про величезні перспективи застосування мультимедійних технологій в області навчання.

2.2.4. Комп'ютерні мережі

Ідея з'єднання один з одним кількох комп'ютерів для обміну інформацією між ними з'явилася майже одночасно зі створенням перших комп'ютерів. Таке з'єднання за аналогією з телефонним почали називати **мережею ЕОМ**, **комп'ютерною мережею**, **інформаційною мережею, мережею передачі даних** або просто мережею. Тобто, комп'ютерною мережею називається об'єднання двох або більше обчислювальних машин та окремих пристроїв спеціальними засобами зв'язку, за допомогою яких можна здійснювати обмін інформацією між будь-якими включеними у мережу комп'ютерами.

Розрізняють **локальні** та **глобальні мережі**. Локальні мережі об'єднують декілька десятків або сотень близько розташованих машин. Таке об'єднання дозволяє організовувати обмін інформацією між машинами, зберігати тільки по одному примірнику програм та даних та спільно їх використовувати, спільно експлуатувати дороге устаткування, об'єднувати обчислювальні потужності кількох машин та для розв'язання складних завдань. Проте найбільш вражаючі можливості виникають при об'єднанні машин у глобальні мережі, коли між собою з'єднуються сотні тисяч та мільйони машин, які знаходяться на різних материках. Найвідомішим прикладом глобальних мереж є мережа **Internet** (Internet – INTERNational NET – міжнародна мережа), яка по суті справи є як би "мережею мереж". Вона об'єднує безліч різнорідних локальних та регіональних мереж, має планетарний масштаб.

Локальні та глобальні мережі дуже широко використовуються в найрізноманітніших областях людської діяльності. Вже є такі сфери, робота яких сильно залежить від наявності або відсутності виходів у глобальні мережі – це

біржі, банки, крупні бібліотеки, метеослужба, управління транспортними комунікаціями, газопроводами. Це такі сфери, успішна робота яких безпосередньо залежить від своєчасності обробки дуже великих масивів інформації, які передаються на великі відстані. Користувачам, що мають вихід у комп'ютерні мережі, надаються різноманітні інформаційні послуги. Найпопулярнішими з них є **електронна пошта** та **Всесвітня павутина**.

Контрольні запитання для самоперевірки

1. Яка відмінна риса комп'ютерів четвертого покоління?
2. Що являє собою гіпертекст?
3. Що являє собою мультимедіа?
4. Чим розрізняються локальні та глобальні комп'ютерні мережі?

3. Поняття інформатики та інформації

3.1. Що таке інформатика?

Термін "**інформатика"** (франц. informatique) походить від французьких слів *information* (інформація) і *automatique* (автоматика) і дослівно означає "**інформаційна автоматика"**. Широко поширений також англомовний варіант цього терміну – "**Computer science"**, що означає буквально "**комп'ютерна наука".**

Ще років 40 тому у більшості наших сучасників будь-який натяк на "інформатику" або "обчислювальну машину" викликав інстинктивну реакцію, в якій виявлялася серйозна заклопотаність. Перелік найбільш поширених думок з приводу інформатики можна було б звести до кількох основних тем. Тема неминучого панування машини над людиною ("з цими машинами ми скоро будемо не потрібні"). Тема картотек та відомостей ("ми стали просто номерами"); тема помилки ("з тих пір, як нараховує машина, у мене все не так").

Щоб дати відповідь на наведені вище думки або побоювання, спробуємо показати, з одного боку, можливості, які відкриваються інформатикою, а з іншого – її межі. Так що ж являє собою інформатика? Якщо звернутися до найранішіх енциклопедичних словників, то там ми не виявимо статті "інформатика". Наприклад, у новому виданні ВРЕ в статті "інформатика" згадується: "Дисципліна, що вивчає структуру і загальні властивості наукової інформації, а також закономірності її створення, перетворення та використання в різних сферах людської діяльності". Таке визначення слова "інформатика" виходить з його вживання тільки в області, яка пов'язана з документалістикою та інформаційно-пошуковими системами. Якщо ж звернутися до "громадської думки", то можна отримати, ймовірно, два типи визначень: "Автоматична обробка інформації" та "Наука про обчислювальні машини".

Навіть прийшовши до компромісу типу "наука про обробку інформації за допомогою обчислювальних машин", потрібно визначати два нові поняття: "інформація" та "обчислювальна машина". Двозначність цього визначення є наслідком глибшого дуалізму, який був характерною рисою інформатики з самого її початку. Говорити про інформацію – це означає акцентувати увагу на людських аспектах інформатики; сказати "наука про обчислювальні машини" – означає наполягати на важливості машини. Це балансування між двома полюсами – машиною та людиною – розкриває принципову двоїстість.

Що можна взяти з цих визначень? Перш за все, що для людини, яка охоча зрозуміти, що таке інформатика, ці два слова – "інформація" та "ЕОМ" – є, звичайно, ключовими словами. Головна проблема інформатики полягає у необхідності переходу від людських понять до комп'ютерного подання. Методи програмування, що розглядаються далі, у деякому розумінні є не що інше, як засоби, які призначені для полегшення цього роз'яснення.

Вперше термін «інформатика» виник на початку 60-х рр. XX в. у Франції для виділення області знань, що пов'язана з автоматизованою обробкою інформації за допомогою електронно-обчислювальних машин.

Інформатика – це наукова та прикладна область знань, яка вивчає закони, методи та способи накопичення, обробки і передачі інформації за допомогою комп'ютерних та інших технічних засобів.

Інформатика вивчає властивості, структуру та функції інформаційних систем, а також інформаційні процеси, що відбуваються у них. Під інформаційною системою розуміють систему для організації, збереження та перетворення інформації. Переважна більшість сучасних інформаційних систем є автоматизованими.

Інформатика тісно пов'язана з кібернетикою, але не тотожна їй. Кібернетика вивчає загальні закономірності процесів управління складними системами в різних областях людської діяльності незалежно від наявності або відсутності комп'ютерів. Інформатика ж вивчає загальні властивості тільки конкретних інформаційних систем. Інформатику можна розглядати як науку і як технологію.

Інформатика як наука об'єднує групу дисциплін, що займаються вивченням різних аспектів властивостей інформації в інформаційних процесах, а також застосуванням алгоритмічних, математичних та програмних засобів для її обробки за допомогою комп'ютерів.

Інформатика як технологія включає систему процедур комп'ютерного перетворення інформації з метою її формування, зберігання, обробки, розповсюдження та використання.

Основними рисами сучасної (нової) інформаційної технології є:

– засоби взаємодії людини з апаратними і програмними засобами (дружній програмний та апаратний інтерфейси);

– наскрізна інформаційна підтримка всіх етапів розв'язання задачі на основі інтегрованої бази даних;

– можливість колективного розв'язання завдань на основі інформаційних мереж та систем телекомунікацій;

– безпаперова технологія, за якою основним носієм інформації є не паперовий, а електронний документ.

Як видно з цього списку, в інформатиці особлива увага приділяється питанням взаємодії. Для цього навіть є спеціальне поняття – ***інтерфейс***. Методи та засоби взаємодії людини з апаратними і програмними засобами називають ***інтерфейсом користувача***. Відповідно, існують ***апаратні інтерфейси***, ***програмні інтерфейси*** та ***апаратно-програмні інтерфейси.***

Основним завданням інформатики є систематизація прийомів та методів роботи з апаратними та програмними засобами обчислювальної техніки. ***Мета*** систематизації полягає у виділенні, впровадженні та розвитку передових, найефективніших технологій, в автоматизації етапів роботи з даними, а також у методичному забезпеченні нових технологічних досліджень.

Інформатика – практична наука. Її досягнення повинні проходити підтвердження практикою і прийматися в тих випадках, коли вони відповідають критерію підвищення ефективності. У складі основного завдання інформатики сьогодні можна виділити такі напрями для практичних застосувань:

– архітектура обчислювальних систем (прийоми та методи побудови систем, які призначені для автоматичної обробки даних);

– інтерфейси обчислювальних систем (прийоми та методи управління апаратним та програмним забезпеченням);

– програмування (прийоми, методи та засоби розробки комп'ютерних програм);

– перетворення даних (прийоми та методи перетворення структур даних);

– захист інформації (розробка методів та засобів захисту даних);

– автоматизація (функціонування програмно-апаратних засобів без участі людини).

На всіх етапах технічного забезпечення інформаційних процесів для інформатики ключовим поняттям є **ефективність**. Для апаратних засобів під ефективністю розуміють відношення продуктивності устаткування до його вартості (з урахуванням вартості експлуатації та обслуговування). Для програмного забезпечення під ефективністю розуміють продуктивність осіб (користувачів), що працюють з ними. У програмуванні під ефективністю розуміють об'єм програмного коду, що створюється програмістами в одиницю часу. В інформатиці все жорстко орієнтовано на ефективність. Питання, як реалізувати ту чи іншу операцію, для інформатики є важливим, але не основним. Основним же є питання, як реалізувати дану операцію ефективно.

3.2.Що таке інформація?

Слово "**інформація**" походить від латинського слова informatio – роз'яснення, виклад. В енциклопедичному словнику цей термін визначається як "… відомості, які передаються одними людьми іншим людям усним, письмовим або будь-яким іншим способом (наприклад, за допомогою умовних сигналів, з використанням технічних засобів тощо)…". Проте таке трактування цього поняття, строго кажучи, не є визначенням. Більш того, виявляється, що точно, строго науково визначити поняття "інформація" взагалі неможливо.

Річ у тім, що поняття "інформація" належить до так званих первинних, невизначених понять. У математиці існує група понять, дати строге визначення яким у принципі неможливо. До них належить поняття "множина", "крапка" та деякі інші. Будь-яка спроба яким-небудь чином визначити їх зведеться до використання синонімів. Наприклад, для терміну "**інформація**" синонімами, які часто використовується є терміни "**відомості**" та "**дані**". У таких випадках поняття вводиться шляхом його пояснення, яке спирається на інтуїцію, здоровий глузд або побутове застосування терміну.

Під ***інформацією*** зазвичай розуміють відображення в людській свідомості знань та фактів (відомостей, даних), що використовуються або зустрічаються в різних областях людської діяльності.

Як знання, так і окремі систематизовані та не систематизовані факти, відомості або дані передаються за допомогою повідомлень. У якості прикладів ***повідомлень*** можна навести такі: "підозрюваному на вигляд 40-45 років"; "швидкість вітру не перевищує 120 кілометрів на годину" тощо. Повідомлення можуть передаватися між людьми за допомогою найрізноманітніших способів. Проте будь-яке повідомлення завжди матеріальне, тобто, представлено деякою матеріальною субстанцією – каменем, глиною, папером, магнітною плівкою, електромагнітними або акустичними коливаннями, молекулами або атомами речовини тощо.

Повідомлення **–** це матеріальна форма інформації. Фактично всіма прикладами, які можна трактувати як інформацію (малюнки, запахи, звуки, паперові тексти, фотографії), є приклади повідомлень, що несуть людині деяку важливу або не важливу для неї інформацію. Тому інформація – це нематеріальний сенс, витягнутий людиною з повідомлення.

Відповідність між інформацією та повідомленням, за допомогою якого вона передається, не є взаємно однозначною. Одна і та сама інформація може передаватися за допомогою різних повідомлень. Наприклад, повідомлення про затримку потяга, що передане на вокзалі різними мовами, – одна й та ж інформація, а повідомлення різні. Та навпаки, одне й те саме повідомлення може нести різну інформацію. Повідомлення "над всією Іспанією безхмарне небо" було сприйняте більшістю радіослухачів як частина метеозведення. І лише дуже небагато посвячених знали, що це умовний сигнал на початок фашистського заколоту в Іспанії у 1936 році.

Можливість витягувати різну інформацію з одного й того ж повідомлення є наслідком того, що різні одержувачі інформації по-різному сприймають, трактують, інтерпретують повідомлення. А у тому випадку, коли повідомлення передається на невідомій слухачеві мові або ж невідомим чином закодованим (зашифроване повідомлення), його одержувач взагалі не зможе витягувати з такого повідомлення ніякої інформації.

Таким чином, виникає питання про спосіб, якості та повноту отримання інформації з повідомлення. Зазвичай спосіб виявлення сенсу повідомлення є загальноприйнятим (природні мови), результатом домовленості між відправником та одержувачем повідомлення (шифровані повідомлення) або ж він може бути вказаний заздалегідь їм обом (азбука Морзе). У загальному випадку говорять, що задається **правило** або група правил інтерпретації повідомлень. Вирішальним чинником для отримання інформації з повідомлення є знання мови повідомлення або способу його кодування, тобто сукупності **правил інтерпретації повідомлення**, тлумачення його сенсу. Як наочні приклади правил інтерпретації повідомлень можна вказати жаргони, природні, професійні та наукові мови, криптографічні шифри і т.і.

Виходячи з розглянутих прикладів та аналогій, групу правил, що використовується для тлумачення сенсу, інтерпретації повідомлень, тобто для отримання з повідомлення корисної для одержувача інформації, часто називають **мовою інтерпретації повідомлень**. Як правило, повідомлення записуються, або передаються за допомогою деякої послідовності знаків –букв або звуків письмової або розмовної мови, спеціально підібраних значків: крапок і тире, математичних, хімічних символів тощо.

Набір знаків, які використовуються для формування та передачі повідомлень, прийнято називати абеткою мови інтерпретації повідомлень. Абетка, як правило, задається прямим перерахуванням усіх знаків, що до нього належать. Наприклад, набором знаків (цифр) {0,1,2,3,4,5,6,7,8,9} є **абетка**, яка зазвичай використовується для запису чисел. Людство за свою історію вигадало безліч різних наборів знаків, які можуть бути використані для запису або передачі повідомлень. Прикладами можуть служити абетки природних мов, набір знаків азбуки Брайля (для сліпих), азбука Морзе.

Оскільки у загальному випадку для формування повідомлення можуть використовуватися не тільки букви абеток природних мов, прийнято говорити, що повідомлення кодується тим або іншим набором знаків, а саму абетку або набір знаків іноді називають ще і кодом.

Як було відмічено вище, одна і та сама інформація може бути передана за допомогою різних повідомлень, причому ця різниця зазвичай виявляється у різному виборі мови або коду, за допомогою якої воно формується. Наприклад, міжнародний морський сигнал лиха SOS можна передати повідомленням англійською мовою "Save our souls", аналогічним повідомленням українською мовою – "Врятуйте наші душі", за допомогою азбуки Морзе та багатьма іншими способами. Але в усіх випадках, на будь-яких мовах і за будь-яких способів кодування сенс усіх повідомлень один і той самий – поданий сигнал лиха. Цей важливий момент можна трактувати по-іншому, а саме можна вважати, що існує тільки одне повідомлення, але застосовуються різні способи його запису, кодування.

З безлічі можливих способів кодування повідомлень найважливішим для інформатики окремим випадком є кодування двійковими наборами знаків, тобто наборами, що складаються всього із двох різних знаків, символів, цифр. Саме до таких наборів відноситься набір {є отвір, немає отвору}, який використовувався для запису інформації на перфокартах у згаданих вище прикладах, та двійковий код – набір знаків, абетка {0,1}, яка застосовується при зберіганні інформації у пам’яті комп’ютера, через що її часто називають ***машинним кодом***.

Не дивлячись на істотну різницю між поняттями "повідомлення" та "інформація", відповідно до сталої практики усної та письмової мови у подальшому викладі практично скрізь використовується термін "інформація", хоча за контекстом треба було б використовувати термін "повідомлення".

Основними операціями, що виконуються над інформацією (точніше кажучи, над повідомленнями), є: ***збирання, обмін, зберігання та обробка***.

Збирання інформації – це діяльність людини або технічного пристрою, у ході якої людина або пристрій отримує необхідні відомості.

Під **обміном** інформацією розуміється процес передачі інформації між різними об'єктами – між людиною та людиною, між людиною та технічним пристроєм, між різними технічними пристроями. Важливо розуміти, що прийняту інформацію одержувач може використовувати неодноразово.

Зберігання інформації – це процес підтримки інформації у вигляді, який забезпечує її передачу у потрібному вигляді та у потрібний час.

Обробка інформації – перетворення інформації з одного вигляду в інший, здійснюється за строгими формальними правилами. По суті – це в основному скорочення кількості інформації. З практичної точки зору важливо пам'ятати, що обробка інформації, автоматична чи ні, ніколи не додає інформацію, вона полягає в тому, що витягує цікаву інформацію з тієї, яка міститься у даних. Природа операцій, що здійснюються в ручній або машинній обробці інформації, розрізняється не принципово. Їх розрізняє по суті тільки порядок величин тієї кількості інформації, яка може бути оброблена, і швидкість, з якою виконується обробка.

Інформацією також називають будь-який критерій вибору серед елементів деякої множини, тобто будь-який критерій, що дозволяє скоротити множину, в якій розшукується відповідь на деяке конкретне питання. Якщо, наприклад, завдання ставиться так: "Яке слово щонайдовше в українській мові?", то корисною інформацією будуть вислови: «довжина розшукуваного слова більше 15» та «розшукуване слово є прикметником». Ці відомості дозволяють скоротити пошук спочатку до множини слів довжиною більше 15, потім до підмножини попередньої множини, яка сформована виключно з прикметників.

3.3. Інформаційна модель

Отримуючи інформацію з навколишнього середовища, людина певним чином її просіває, відкидаючи неістотну, випадкову інформацію, і залишаючи тільки важливу для вирішення нею завдання. Будь-який даний об'єкт або явище незалежно від його матеріальності або ідеальності має деякі характерні для нього риси, властивості, якості. Наприклад, тіло як матеріальний об'єкт має геометричні розміри (довжину, ширину, висоту), вагу, колір тощо. А історичні події характеризуються датою та місцем, де вони відбулися. Такі характерні, невід'ємні риси, властивості, якості прийнято називати ***атрибутами*** об'єктів, явищ. Взагалі кажучи, об'єкт або явище може мати дуже велику кількість атрибутів (десятки, сотні тисяч і більше). І далеко не всі з них істотні, важливі для даного завдання. Неусвідомлено обробляючи отриману інформацію, ми, як правило, звертаємо увагу тільки на те, що нас цікавить, а решту всіх атрибутів, які не мають значення для вирішення завдання, відкидаємо. Тобто, *замість реального об'єкта* в нашій свідомості *формується його образ* – ***модель***, що

містить тільки потрібну в даній ситуації інформацію, і при цьому позбавлена всіх несуттєвих подробиць.

Таким чином, **моделлю** називається матеріальний або ідеальний образ деякої сукупності реальних об'єктів або явищ, який за певних обставин використовується як замінник або представник початкових об'єктів або явищ. Це образ, який отримується за допомогою концентрації уваги тільки на деяких найважливіших з погляду вирішення завдання атрибутах даних предметів або явищ, та відкиданням усіх їх несуттєвих властивостей.

При розвяанні завдань у різних областях діяльності доводиться будувати різні моделі. В інформатиці розглядаються в основному **інформаційні** та **математичні моделі**. Розглянемо, наприклад, **інформаційну модель особи**. На кожного співробітника будь-якого підприємства або установи у відділі кадрів заводиться особиста справа, в якій, зокрема, знаходиться особистий листок з обліку кадрів. У цьому документі відображаються такі атрибути співробітників, як прізвище, ім'я та по батькові, дата народження, освіта, домашня адреса тощо. А такі атрибути, як колір очей, вага, в особистому листку не вказуються. Можна вважати, що цей документ є інформаційною моделлю особи співробітника установи. У цій моделі відбиті тільки значущі для відділу кадрів атрибути співробітників. Якщо ж розглянути ситуацію, коли створюється інформаційна модель особистості злочинця, який розшукується органами правопорядку, то не значущі раніше атрибути – колір очей і вага – тепер можуть стати суттєвими, а, скажемо, освіта та точна дата народження – несуттєвими.

Один і той самий об'єкт, одне і те саме явище, що розглядається з різних точок зору, можуть мати різні інформаційні моделі.

Поняття **математичної моделі** дуже близько до поняття інформаційної моделі, і багато фахівців розглядають математичну модель як специфічний, частковий випадок інформаційної моделі. Характерною межею математичної моделі є необхідність залучення математичних співвідношень, рівнянь, обмежень для адекватного опису даних явищ або зв'язків між об'єктами. Наприклад, відбулася дорожньо-транспортна пригода. Необхідно визначити винуватця аварії. У деяких випадках може допомогти вимірювання довжини гальмівного шляху, по якому, з урахуванням стану дорожнього покриття, погодних умов та деяких інших чинників, можна за допомогою спеціальних математичних співвідношень визначити швидкості машин, що брали участь у події. Будується математична модель ситуації, що включає такі атрибути, як довжина гальмівного шляху, вага та габарити машин, стан дорожнього покриття, спеціальні коефіцієнти, що враховують погодні умови, та математичні співвідношення, що зв'язують між собою всі дані величини. Виконавши необхідні математичні розрахунки, можна розв'язати поставлену задачу та з великою часткою впевненості визначити винуватця аварії.

Відступ від несуттєвих деталей, про який йшла мова вище, прийнято називати ***абстрагуванням***. Таким чином, абстрагування є одним із найважливіших інструментів при побудові моделі будь-якої предметної області.

Як видно з попереднього обговорення, при абстрагуванні здійснюється певне огрублення реальної дійсності. Проте концентрація уваги на найважливіших аспектах, атрибутах дозволяє виявити визначальні властивості, закономірності та, отже, зрозуміти суть об'єкта, що вивчається, явища. Наявність адекватної моделі, тобто моделі, яка вірно відображає найважливіші особливості реальних об'єктів або явищ, дозволяє спрогнозувати поведінку об'єкта у тій чи іншій ситуації, описати процес розвитку явища у часі, вчасно отримати та використати потрібну інформацію. Наприклад, дізнатися ймовірний характер погоди, і на підставі отриманої інформації провести відповідну підготовку. Побудова моделі є найважливішою, найскладнішою та найбільш творчим етапом при вивченні будь-яких об'єктів або явищ. Розробка моделі у більшості випадків вимагає об'єднаних зусиль висококваліфікованих фахівців конкретної предметної області, а також фахівців в області інформатики.

Контрольні запитання для самоперевірки

1. Що вивчає інформатика?
2. Які ви знаєте приклади повідомлень?
3. Які операції виконуються над інформацією?
4. Чим інформаційна модель відрізняється від математичної?
5. Що прийнято називати абстрагуванням?

4. Дані

Дані – діалектична складова частини інформації. Вони є зареєстрованими сигналами. При цьому фізичний метод реєстрації може бути будь-яким: механічне переміщення фізичних тіл, зміна їхньої форми або параметрів якості поверхні, зміна електричних, магнітних, оптичних характеристик, хімічного складу та (або) характеру хімічних зв'язків, зміна стану електронної системи та багато чого іншого. Відповідно до методу реєстрації дані можуть зберігатися та транспортуватися на носіях різних видів.

4.1. Носії даних

Найпоширенішим носієм даних, хоча й не найекономічнішим, мабуть, є папір. На папері дані реєструються шляхом зміни оптичних характеристик його поверхні. Зміна оптичних властивостей (зміна коефіцієнта відображення поверхні у певному діапазоні довжин хвиль) використовується також у пристроях, що здійснюють запис лазерним променем на пластмасових носіях із відображаючим покриттям (CD-ROM). Як носії, що використовують зміну магнітних властивостей, можна назвати магнітні стрічки та диски. Реєстрація даних шляхом зміни хімічного складу поверхневих речовин носія широко використовується у фотографії. Від цих властивостей носія нерідко залежать такі властивості інформації, як повнота, доступність та достовірність. Наприклад, ми можемо розраховувати на те, що у базі даних, яка розміщується на компакт-диску, простіше забезпечити повноту інформації, ніж в аналогічній за призначенням базі даних, яка розміщена на гнучкому магнітному диску, оскільки у першому випадку щільність запису даних на одиниці довжини магнітної смужки набагато вища. Для звичайного споживача доступність інформації у книзі помітно вища, ніж тієї ж інформації на компакт-диску, оскільки не всі споживачі володіють необхідним устаткуванням.

І нарешті, відомо, що візуальний ефект від перегляду слайду через проектор набагато більший, ніж від перегляду аналогічної ілюстрації, яка надрукована на папері, оскільки діапазон сигналів яскравості у світлі, що проходить, на два-три порядки більше, ніж у відбитому на папері.

Завдання перетворення даних з метою зміни носія належить до одного з найважливіших завдань інформатики. У структурі вартості обчислювальних систем пристрої для введення та виведення даних, що працюють із носіями інформації, складають до половини вартості апаратних засобів.

4.2. Операції з даними

У ході інформаційного процесу дані перетворюються з одного вигляду в інший за допомогою методів (алгоритмів). Обробка даних включає безліч різних операцій. По мірі розвитку науково-технічного прогресу трудовитрати на обробку даних неухильно зростають. Перш за все це пов'язано з постійним

ускладненням умов управління виробництвом та суспільством. Інший чинник, який також викликає загальне збільшення об'ємів даних, що оброблюються, також пов'язаний з науково-технічним прогресом, а саме з швидкими темпами появи та впровадження нових носіїв даних, засобів їх зберігання та доставки. У структурі можливих операцій із даними можна виділити такі:

– *збір даних* – накопичення інформації з метою забезпечення достатньої повноти для прийняття рішень;

– *формалізація даних* – приведення даних, що надходять з різних джерел, до однакової форми, щоб зробити їх сумісними між собою, тобто підвищити їх рівень доступності;

– *фільтрація даних* – відсіювання «зайвих» даних, в яких немає необхідності для прийняття рішень. При цьому повинен зменшуватися рівень «шуму», а достовірність та адекватність даних повинні зростати;

– *сортування даних* – впорядкування даних за заданою ознакою з метою зручності використання. Це підвищує доступність інформації;

– *архівація даних* – організація зберігання даних у зручній та досяжній формі. Це потрібно для зниження економічних витрат при зберіганні даних та підвищує загальну надійність інформаційного процесу у цілому;

– *захист даних* – комплекс заходів, що спрямовані на запобігання втрат, відтворення та модифікації даних;

– *транспортування даних* – прийом та передача (доставка та постачання) даних між віддаленими учасниками інформаційного процесу. При цьому джерело даних в інформатиці прийнято називати сервером, а споживача – клієнтом;

– *перетворення даних* – перетворення даних з однієї форми в іншу або з однієї структури в іншу.

Перетворення даних часто пов'язано із зміною типу носія: наприклад, книги можна зберігати у звичайній паперовій формі, але можна використовувати для цього як електронну форму, так і мікрофотоплівку. Необхідність у багаторазовому перетворенні даних виникає також при їх транспортуванні, особливо якщо воно здійснюється засобами, які не призначені для транспортування того чи іншого виду даних. Як приклад можна згадати, що для транспортування цифрових потоків даних по каналах телефонних мереж (які спочатку були орієнтовані тільки на передачу аналогових сигналів у вузькому діапазоні частот) необхідне перетворення цифрових даних у дещо схоже на звукові сигнали, чим і займаються спеціальні пристрої – телефонні модеми.

4.3. Кодування даних

Для автоматизації роботи з даними, що належать до різних типів, дуже важливо уніфікувати їхню форму представлення – для цього зазвичай використовується прийом кодування, тобто подання даних одного типу через

дані іншого типу. Природні людські мови – це не що інше, як системи кодування понять для виразу думок за допомогою мови до мов близько дотичної азбуки (системи кодування компонентів мови за допомогою графічних символів). Така сама проблема універсального засобу кодування досить успішно реалізується в окремих галузях техніки, науки та культури. Як приклади можна навести систему запису математичних виразів, телеграфну азбуку, морську азбуку прапорця, систему Брайля для сліпих та багато чого ще.

4.3.1. Кодування даних двійковим кодом

Своя система, яку ми докладніше розглядатимемо далі, існує і в обчислювальній техніці – вона називається ***двійковим кодуванням*** і грунтується на представленні даних послідовністю всього двох знаків: 0 та 1. Ці знаки називаються двійковими цифрами, англійською – **bi**nary digi**t** або, скорочено, **bit** (біт).

Одним бітом можуть бути виражені два поняття: 0 або 1 (так чи ні, чорне або біле, істина або брехня і т. і.). Якщо кількість бітів збільшити до двох, то вже можна виражати чотири різні поняття: 00 01 10 11

Трьома бітами можна закодувати вісім різних значень:

000 001 010 011 100 101 110 111

Збільшуючи на одиницю кількість розрядів у системі двійкового кодування, ми збільшуємо вдвічі кількість значень, яка може бути виражена у даній системі, тобто загальна формула має вигляд $N = 2^m$, де: N – кількість незалежних кодованих значень; m – розрядність двійкового кодування, що прийнята у даній системі.

4.3.2. Кодування текстових даних

Якщо кожному символу абетки поставити у відповідальність певне ціле число (наприклад, порядковий номер), то за допомогою двійкового коду можна кодувати і текстову інформацію. Вісім двійкових розрядів досить для кодування 256 різних символів. Цього вистачить, щоб виразити різними комбінаціями з восьми бітів усі символи англійської та російської мов, як малі, так і прописні, а також розділові знаки, символи основних арифметичних дій та деякі загальноприйняті спеціальні символи, наприклад символ «§».

Технічно це виглядає дуже просто, проте завжди існували достатньо вагомі організаційні складнощі. У перші роки розвитку обчислювальної техніки вони були пов’язані з відсутністю необхідних стандартів, а на даний час викликані, навпаки, надлишком суперечливих стандартів, що діють одночасно. Для того щоб весь світ однаково кодував текстові дані, потрібні єдині таблиці кодування, а це поки-що неможливо через суперечності між символами національних абеток, а також суперечності корпоративного характеру.

Для англійської мови, що захопила де-факто нішу міжнародного засобу спілкування, суперечності вже зняті. Інститут стандартизації США (***ANSI*** –

American National Standard Institute) ввів у дію систему кодування ***ASCII*** (*American Standard Code for Information Interchange – стандартний код інформаційного обміну США*). У системі ASCII закріплено дві таблиці кодування – базова та розширена. Базова таблиця закріплює значення коду від 0 до 127, а розширена належить до символів із значеннями від 128 до 255 (у тому числі великі та маленьки літери української та російської абеток).

Перші 32 коди базової таблиці, починаючи з нульового, віддано виробникам апаратних засобів (у першу чергу виробникам комп'ютерів та друкуючих пристроїв). У цій області розміщуються так звані *управляючі коди*, яким не відповідають жодні символи мов, і, відповідно, ці коди не виводяться ані на екран, ані на пристрої друку, але ними можна управляти тим, як проводиться виведення інших даних. Починаючи з коду 32 по код 127 розміщені коди символів англійської абетки, цифр, арифметичних дій та деяких допоміжних символів.

Аналогічні системи кодування текстових даних були розроблені і в інших країнах. Так, наприклад, у колишньому СРСР у цій області діяла система кодування **КОІ-7** (*код обміну інформацією, семизначний*). Проте підтримка виробників устаткування та програм вивела американський код ASCII на рівень міжнародного стандарту, і національним системам кодування довелося «відступити» у другу, розширену частину системи кодування, що відповідає значенням кодів з 128 по 255.

Відсутність єдиного стандарту у цій області призвела до множини кодувань. Тільки у Росії можна вказати три стандарти кодування, що діють, та ще два застарілих. Так, наприклад, кодування символів російської мови, відоме як кодування *Windows-1251*, було введене «ззовні» – компанією *Microsoft*, але, враховуючи широке розповсюдження операційних систем та інших продуктів цієї компанії в Росії, вона глибоко закріпилася і знайшла широке розповсюдження.

Інше поширене кодування носить назву **КОІ-8** (*код обміну інформацією, восьмизначний*). Її походження – дії Ради Економічної Взаємодопомоги держав Східної Європи. Сьогодні кодування КОІ-8 має широке розповсюдження у комп'ютерних мережах Росії та у російському секторі та Internet. Міжнародний стандарт, в якому передбачено кодування символів російської абетки, носить назву кодування **ISO** (*International Standard Organization – Міжнародний інститут стандартизації*).

4.3.3. Універсальна система кодування текстових даних

Якщо проаналізувати організаційні труднощі, які пов'язані зі створенням єдиної системи кодування текстових даних, то можна прийти до висновку, що вони викликані обмеженим набором кодів (256). У той же час очевидно, що якщо, наприклад, кодувати символи не восьмирозрядними двійковими числами, а числами з більшою кількістю розрядів, то й діапазон можливих значень кодів стане набагато більший. Така система, заснована на 16-розрядному кодуванні

символів, отримала назву ***універсальної – UNICODE***. Шістнадцять розрядів дозволяють забезпечити унікальні коди для 65536 різних символів – цього цілком достатньо для розміщення в одній таблиці символів більшості мов планети.

Не дивлячись на тривіальну очевидність такого підходу, простий механічний перехід на дану систему довгий час стримувався через недостатні ресурси засобів обчислювальної техніки (у системі кодування UNICODE всі текстові документи автоматично стають вдвічі довшими). У другій половині 90-х років технічні засоби досягли необхідного рівня забезпеченості ресурсами, і сьогодні ми спостерігаємо поступовий переклад документів та програмних засобів на універсальну систему кодування. Для індивідуальних користувачів це ще більше додало турбот на погодження документів, що виконані в різних системах кодування, з програмними засобами, але це треба розуміти як труднощі перехідного періоду.

4.3.4. Кодування графічних даних

Якщо розглянути за допомогою збільшувального скла чорно-біле графічне зображення, яке надруковане у газеті або книзі, то можна побачити, що воно складається з найдрібніших крапок, що утворюють характерний узор, який зветься ***растром***, здавна прийнятий у поліграфії.

Оскільки лінійні координати та індивідуальні властивості кожної крапки (яскравість) можна виразити за допомогою цілих чисел, то можна сказати, що растрове кодування дозволяє використовувати двійковий код для представлення графічних даних. Загальноприйнятим на сьогоднішній день вважається представлення чорно-білих ілюстрацій у вигляді комбінації крапок з 256 градаціями сірого кольору, і, таким чином, для кодування яскравості будь-якої крапки звичайно достатньо восьмирозрядного двійкового числа.

Для кодування кольорових графічних зображень застосовується ***принцип декомпозиції*** довільного кольору на основні складові. В якості таких складових використовують три основні кольори: червоний (*Red, R*), зелений (*Green, G*) та синій (*Blue, B*). На практиці вважається (хоча теоретично це не зовсім так), що будь-який колір, який бачить людське око, можна отримати шляхом механічного змішування цих трьох основних кольорів. Така система кодування називається системою ***RGB*** за першими літерами назв основних кольорів.

Якщо для кодування яскравості кожної з основних складових використовувати по 256 значень (вісім двійкових розрядів), як це прийнято для півтонових чорно-білих зображень, то на кодування кольору однієї крапки треба витратити 24 розряди. При цьому система кодування забезпечує однозначне визначення 16,5 млн різних кольорів, що насправді близько до чутливості людського ока. Режим представлення кольорової графіки з використанням 24 двійкових розрядів називається повнокольоровим (*True Color*).

Кожному з основних кольорів можна поставити у відповідність додатковий колір, тобто колір, який доповнює основний колір до білого. Неважко відмітити, що для будь-якого з основних кольорів додатковим буде колір, що утворений сумою пари решти основних кольорів. Відповідно, додатковими кольорами є: блакитний (*Cyan, C*), пурпурний (*Magenta, M*) та жовтий (*Yellow, Y*). Принцип декомпозиції довільного кольору на складові компоненти можна застосовувати не тільки для основних кольорів, але й для додаткових, тобто будь-який колір можна представити у вигляді суми блакитної, пурпурної та жовтої складової. Такий метод кодування кольору прийнятий у поліграфії, але в поліграфії використовується ще й четверта фарба – чорна (*Black, K*). Тому дана система кодування позначається чотирма буквами *CMYK* (чорний колір позначається буквою *K*, тому що буква *B* вже зайнята синім кольором), і для представлення кольорової графіки у цій системі треба мати 32 двійкових розряди. Такий режим теж називається повнокольоровим (*True Color*).

Якщо зменшити кількість двійкових розрядів, що використовуються для кодування кольору кожної крапки, то можна скоротити об'єм даних, але при цьому діапазон кодованих кольорів помітно скорочується. Кодування кольорової графіки 16-розрядними двійковими числами називається режимом *High Color*.

В разі кодування інформації про колір за допомогою восьми битів даних можна передати тільки 256 кольорових відтінків. Такий метод кодування кольору називається ***індексним***. Сенс назви у тому, що, оскільки 256 значень абсолютно недостатньо, щоб передати весь діапазон кольорів, доступний людському оку, код кожної крапки растру виражає не колір сам по собі, а тільки його номер (індекс) в якійсь довідковій таблиці, яка зветься ***палітрою***. Зрозуміло, що ця палітра повинна додаватися до графічних даних – без неї не можна користуватися методами відтворення інформації на екрані або папері (тобто, скористатися, звичайно, можна, але через неповноту даних отримана інформація не буде адекватною: листя на деревах може виявитися червоним, а небо – зеленим).

4.3.5. Кодування звукової інформації

Прийоми та методи роботи із звуковою інформацією прийшли в обчислювальну техніку пізніше за усе. До того ж, на відміну від числових, текстових та графічних даних, у звукозаписі не було такої ж тривалої та перевіреної історії кодування. В результаті методи кодування звукової інформації двійковим кодом далекі від стандартизації. Безліч окремих компаній розробили свої корпоративні стандарти, але якщо говорити узагальнено, то можна виділити два основні напрями.

Метод ***FM*** *(Frequency Modulation)* грунтується на тому, що теоретично будь-який складний звук можна розкласти на послідовність простих гармонійних сигналів різних частот, кожна з яких є правильною синусоїдою, а отже, може бути описана числовими параметрами, тобто кодом. У природі

звукові сигнали мають безперервний спектр, тобто є аналоговими. Їх розкладання у гармонійні ряди та представлення у вигляді дискретних цифрових сигналів виконують спеціальні пристрої – ***аналогово-цифрові перетворювачі (АЦП)***. Зворотне перетворення для відтворення звуку, закодованого числовим кодом, виконують ***цифро-аналогові перетворювачі (ЦАП).*** При таких перетвореннях неминучі втрати інформації, які пов'язані з методом кодування, тому якість звукозапису зазвичай виходить не цілком задовільною, і відповідає якості звучання простих електромузичних інструментів із забарвленням, яке є характерним для електронної музики. У той же час, даний метод кодування забезпечує вельми компактний код, і тому він знайшов застосування ще у ті роки, коли ресурси засобів обчислювальної техніки були явно недостатні.

Метод ***таблично-хвильового*** *(Wave-Table)* синтезу краще відповідає сучасному рівню розвитку техніки. Якщо говорити спрощено, то можна сказати, що десь у заздалегідь підготовлених таблицях зберігаються зразки звуків для безлічі різних музичних інструментів (хоча не тільки для них). У техніці такі зразки називають ***семплами***.

Числові коди виражають тип інструменту, номер його моделі, висоту тону, тривалість та інтенсивність звуку, динаміку його зміни, деякі параметри середовища, в якому відбувається звучання, а також інші параметри, які характеризують особливості звуку. Оскільки як зразки використовуються «реальні» звуки, то якість звуку, що отриманий в результаті синтезу, виходить дуже високою і наближається до якості звучання реальних музичних інструментів.

4.4. Основні структури даних

Робота з великими наборами даних автоматизується простіше, коли дані впорядковані, тобто утворюють задану структуру. Існує три основні типи структур даних: ***лінійна, ієрархічна*** та ***таблична***. Їх можна розглянути на прикладі звичайної книги. Якщо розібрати книгу на окремі листи і перемішати їх, книга втратить своє призначення. Вона як і раніше представлятиме набір даних, але підібрати адекватний метод для отримання з неї інформації дуже непросто. (Ще гірше справа йтиме, якщо із книги вирізати кожну букву окремо, – у цьому випадку навряд чи взагалі знайдеться адекватний метод для її читання.)

Якщо ж зібрати всі листи книги у правильній послідовності, ми отримаємо просту структуру даних – ***лінійну***. Таку книгу вже можна читати, хоча для пошуку потрібних даних її доведеться прочитати підряд, починаючи із самого початку, що не завжди зручно.

Для швидкого пошуку даних існує ***ієрархічна структура***. Так, наприклад, книги розбивають на частини, розділи, підрозділи, параграфи. Елементи структури нижчого рівня входять в елементи структури більш високого рівня: розділи складаються із глав, глави з параграфів і так далі

Для великих масивів пошук даних в ієрархічній структурі набагато простіший, ніж у лінійній, проте і тут необхідна навігація, яка пов’язана з необхідністю перегляду. На практиці завдання спрощують тим, що у більшості книг є допоміжна **перехресна таблиця**, яка поєднує елементи ієрархічної структури з елементами лінійної структури, тобто що пов’язує розділи та параграфи з номерами сторінок. У книгах із простою ієрархічною структурою, що розраховані на послідовне читання, цю таблицю прийнято називати ***стислим змістом***, а у книгах зі складною структурою, що припускає вибіркове читання, її називають ***змістом***.

Контрольні запитання для самоперевірки

1. Яка діяла система кодування у країнах СНД?
2. Який принцип кодування в універсальній системі UNICODE.
3. Який принцип застосовується для **декомпозиції** довільного кольору?
4. Який метод синтезу звуку краще відповідає сучасному рівню розвитку техніки?

5. Файли та файлова структура

Існує безліч систем подання даних. З однією з них, яка прийнята в інформатиці та обчислювальній техніці – двійковим кодом, ми вже познайомилися вище. Найменшою одиницею такого уявлення є **біт** (*двійковий розряд*) – мінімальна одиниця інформації.

Сукупність двійкових розрядів, які відображають числові або інші дані, утворює якийсь бітовий малюнок. Практика показує, що з бітовим уявленням зручніше працювати, якщо цей малюнок має регулярну (постійну, впорядковану) форму. На даний час такі форми використовують групи з восьми бітів, які називаються ***байтами***.

Поняття про байт як групу взаємозв'язаних бітів з'явилося разом з першими зразками електронної обчислювальної техніки. Довгий час воно було ***машиннозалежним***, тобто для різних обчислювальних машин довжина байта була різною. Тільки у кінці 60-х років поняття байта стало універсальним та ***машиннонезалежним***. Вище ми бачили, що в багатьох випадках доцільно використовувати не восьмирозрядне кодування, а 16-розрядне, 24-розрядне, 32-розрядне. Група з 16 взаємозв'язаних біт (двох взаємозв'язаних байтів) в інформатиці називається ***словом***. Відповідно, групи із чотирьох взаємозв'язаних байтів (32 розряди) називаються ***подвійним словом***, а групи з восьми байтів (64 розряди) – ***збільшеним учетверо словом***.

5.1. Одиниці вимірювання даних

Існує багато різних систем та одиниць вимірювання даних. Кожна наукова дисципліна і кожна область людської діяльності може використовувати свої, найзручніші або традиційно сталі одиниці. В інформатиці для вимірювання даних використовують той факт, що різні типи даних мають універсальне двійкове подання і тому вводять свої одиниці даних, які засновані на цьому представленні.

Як вже наголошувалося, біт є мінімальною неподільною одиницею інформації. Таким чином, біт – це кількість інформації, що міститься в одному розряді двійкового числа. Тепер можна сказати, що якесь умовне двійкове значення 10011010 складається з восьми бітів, оскільки включає вісім заповнених одиницями та нулями розрядів.

Послідовність з восьми бітів, як ми уже наголошували, називають байтом. Саме байт є найспоживанішою величиною у комп'ютерній індустрії, оскільки ця одиниця володіє значно більшою "інформаційною ємністю", ніж біт. Наступна за розміром одиниця інформації – це кілобайт, яка містить 1024 байт. Чому саме 1024, адже ми звикли, що приставка «кіло» означає зазвичай рівно тисячу яких-небудь одиниць? Розгадка криється у тому, що для підрахунку об'єму інформації використовується двійкова система числення. Дійсно, число 1000 не є ступенем двійки і виглядає "круглим" лише у десятковій системі числення, у двійковій системі воно не просто відображається "непривабливо",

але ще і незручно для здійснення математичних операцій. Тому вчені-кібернетики вирішили використовувати для підрахунку великих обсягів інформації значення не 1000, а коефіцієнт 2^{10}, тобто 1024.

Оскільки одним байтом, як правило, кодується один символ текстової інформації, то для текстових документів розмір у байтах відповідає лексичному об'єму у символах (поки що винятком є розглянуте вище універсальне двобайтове кодування UNICODE).

Проте комунікаційна лінія, яка передає 1 Кбіт/с, насправді передає 1000 бит/с, тому що ці швидкості не є ступенем двійки. У кілобайтах вимірюють порівняно невеликі об'єми даних. Умовно можна вважати, що одна сторінка неформатованого машинописного тексту складає приблизно 2 Кбайт. Більш крупні одиниці вимірювання даних утворюються додаванням префіксів мега-, гіга-, тера-, пета-; у ще більш крупних одиницях поки що немає практичної потреби.

1 Мбайт = 1024 Кбайт = 2^{20} байт; 1 Гбайт = 1024 Мбайт = 2^{30} байт

1 Тбайт = 1024 Гбайт = 2^{40} байт; 1 Пбайт = 1024 Тбайт = 2^{50} байт

5.2. Одиниці зберігання даних

При зберіганні даних вирішуються дві проблеми: як зберегати дані у найкомпактнішому вигляді, та як забезпечити до них зручний та швидкий доступ (якщо доступ не забезпечений, то це не зберігання). Для забезпечення доступу необхідно, щоб дані мали впорядковану структуру і при цьому, як ми вже знаємо, утворюється «паразитне навантаження» у вигляді адресних даних. Без них не можна забезпечити доступ до потрібних елементів даних, які входять у структуру.

Оскільки адресні дані теж мають розмір і теж підлягають зберіганню, зберігати дані у вигляді дрібних одиниць, таких як байти, незручно. Їх незручно зберігати і у більш крупних одиницях (кілобайтах, мегабайтах і т.ін.), оскільки неповне заповнення однієї одиниці зберігання призводить до неефективності зберігання.

Одним із засобів організації збору, зберігання та введення/виведення даних є використання поняття під назвою *файл*. За одиницю зберігання даних прийнятий об'єкт змінної довжини, який називається ***файлом***. *Файл – це послідовність довільного числа байтів у певному форматі, що має унікальне власне ім'я і розміщується на зовнішньому носії (диску)*. Зазвичай в окремому файлі зберігають дані, які належать до одного типу. У цьому випадку тип даних визначає тип файлу. Найпростіше уявити собі файл у вигляді безрозмірного канцелярського досьє, в яке можна за бажанням додавати вміст або здобувати його звідти. Оскільки у визначенні файлу немає обмежень на розмір, можна уявити собі файл, який має 0 байтів (порожній файл), та файл, що має будь-яку кількість байтів.

У визначенні файлу особлива увага приділяється імені. Воно фактично несе у собі адресні дані, без яких дані, що зберігаються у файлі, не стануть

інформацією через відсутність методу доступу до них. Окрім функцій, що пов'язані з адресацією, ім'я файлу може зберігати і відомості про тип даних. Наприклад, файл може бути вкладеним каталогом (особливий тип файлу), тобто дозволяється створювати деревоподібну структуру каталогів, яка визначає маршрут доступу до файла. Для автоматичних засобів роботи з даними це важливо, оскільки по імені файлу вони можуть автоматично визначити адекватний метод доступу до інформації у файлі.

У байтах, кілобайтах, мегабайтах та гігабайтах вимірюється не тільки об'єм будь-якої фактичної інформації (наприклад, однієї сторінки шкільного твору), але й, так би мовити, "потенційна ємність" деяких пристроїв комп'ютера. Тому, якщо основна функціональна характеристика будь-якого апаратного елементу ПК позначається у мегабайтах, то це не означає, що такий елемент доверху наповнений деякими даними – у подібних випадках мається на увазі, що цей пристрій потенційно може вміщувати у собі вказаний в його властивостях об'єм інформації, та ані байтом більше.

5.3. Поняття про файлову структуру

Вимога унікальності імені файлу очевидна – без цього неможливо гарантувати однозначність доступу до даних. У засобах обчислювальної техніки вимога унікальності імені забезпечується автоматично – створити файл з іменем, тотожним тому, що вже існує, користувач не може.

Збереження файлів організовується в ієрархічній структурі, яка у даному випадку називається файловою структурою. Вершиною такої структури слугує ім'я носія, на якому зберігаються файли. Вважається, що крім файлів є ще й каталоги (папки). Справа у тому, що папка на моніторі відображається у вигляді спеціального значка тільки тому, що так налаштована програма перегляду файлів. Насправді папка – це теж файл, який зберігає список дескрипторів (елементів опису) вхідних в нього файлів і підкаталогів. Але ці тонкощі користувач може і не знати (хоча в деяких ситуаціях це може стати в нагоді). Ми також будемо міркувати, що файли групують у каталоги (папки, теки), всередині якіх могут буті створені вкладені каталоги (папки, теки).

Каталогом є спеціальний файл з елементами строго визначеної структури (дескрипторами файлів), в яких зберігається інформація про файли, що відносяться до даного каталогу. Дескриптор може вказувати через атрибут типу на інший (вкладений) каталог, створюючи при цьому дерево каталогів. Шлях доступу до файлу починається з імені пристрою та включає всі імена каталогів, через які він проходить. Як розподільник використовується символ «\» (лівий або зворотний слеш). Після імені носія додатково ще ставиться двокрапка. Унікальність імені файлу забезпечується тим, що повним іменем файлу вважається власне ім'я файлу разом із вказівкою повного маршруту доступу до нього. Зрозуміло, що у цьому випадку на одному носії не може бути двох файлів із тотожними повними іменами. Приклад запису повного імені файлу:

<ім'я носія>:\<ім'я каталога-1>\...\<ім'я каталога-N>\< ім'я файлу>

Ось приклад запису двох файлів, які мають однакове власне ім’я і розміщені на одному носієві, але які відрізняються маршрутом доступу до нього, тобто повним іменем. Для наочності імена каталогів надруковані прописними буквами.

C:\МАТЕМАТИКА\ДОСЛІДЖЕННЯ\результати досліджень

C:\ФІЗИКА\ДОСЛІДЖЕННЯ\результати досліджень

З точки зору фізичного розміщення данні на дисках розташовуються послідовним кодом побітно на магнітних поверхнях по концентричних колових смужках, які нумеруються. Смужки розбиваються на сектори. Сектори, як і смужки, мають свої номери. Сукупність магнітних смужок з однаковими номерами створюють циліндри. Схема розміщення інформації на диску представлена на рис. 5.1.

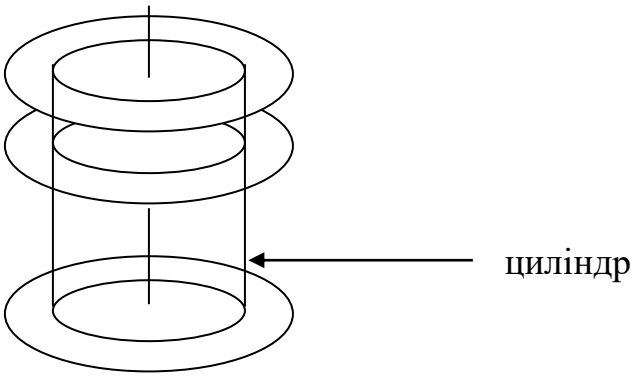


Рис 5.1. Розміщення інформації на диску

Для доступу до даних на диску існує блок магнітних голівок, які перемішуються радіально вздовж поверхонь по циліндрах за командами контролера диску. Команди контролера формуються при отриманні команд введення/виведення з основного процесора. Структура сектора має три частини: голова, власно інформаційна частина та хвіст. В голові записується номер сектора, а у хвості – контрольна сума. Під час розбиття диска по секторах у голову записується номер сектора, в інформаційну частину записується стандартний код, а у хвіст – контрольна сума. Якщо стандартне значення контрольної суми у секторі під час розмітки відрізняється від того, що обчислюється при розмітці, то такий сектор вважається пошкодженим. При розмітці диска виконується читання кожного його сектора та формування контрольної суми.

голова	інформаційна частина	хвіст

Таким чином, фізична адреса початку файлу складається з власно ідентифікатора пристрою, номера поверхні, номера циліндра та номера сектора. Розмір інформаційної частини сектора, який прийнято фірмою Microsoft, складає 512 байтів. Доступ до файлів може бути послідовний та за прямою

адресою. При звичайному зверненні використовується послідовний доступ. Як вже було зазначено вище, папки або каталоги – це спеціальні файли, які містять інформацію про файли або інші папки (каталоги). Папки являють собою таблиці із елементів стандартної структури, в яких розміщується ім’я файлу, його розширення, дата та час його створення або останньої модифікації, розмір, тип файлу (папка, звичайний файл, системний, тощо) та логічний номер початкового кластера (логічна адреса початку файлу). У подальшому ОС перераховує логічну адресу на фізичну. Кластери являють собою сукупність секторів. Чим більший розмір кластеру, тим більший адресний простір на диску може бути доступним. Тому диски великого розміру мають більший розмір кластеру у порівнянні з дисками меншого розміру. З іншого боку, якщо кластер не повністю заповнений, то його решта не може бути використана. Тому для кожного диску в залежності від розміру обирається свій розмір кластеру.

Треба зазначити, що перші дискові пристрої не встигали прочитати номер сектору та вчасно його обробити. Тому, якщо послідовне розташування секторів співпадало з їх нумерацією виникав ефект пропуску наступного сектору та зайвий оборот диску. Для усунення цього недоліку застосовувався метод чередування (inerleaving) при нумерації секторів. Так, якщо на магнітній смужці розміщено 9 секторів, то у такому випадку послідовність нумерації секторів буде через два сектори і матиме наступний вигляд – 1 4 7 2 5 8 3 6 9. Сучасні дискові пристрої та висока швидкість роботи процесорів дозволяють немерувати сектори послідовно без чередування.

Доступ до файлу здійснюється за допомогою системних звернень (функцій): створення, знищення, відкриття, введення, виведення та закриття файлів. Усі інші функції є похідними щодо цих основних. Перед тим, як прочитати файл с диска, треба виконати його відкриття. Процес відкриття файлу дозволяє ОС знайти файл на диску та передати у пам’ять інформацію, яка необхідна для доступу до цього файлу. Пошук здійснюється шляхом лінійного пошуку відповідного дескриптора за ім’ям файлу у зазначеному каталозі, який попередньо копіюється з диска у пам’ять і являє собою таблицю фіксованого розміру.

Після відкриття файлу до нього можна мати доступ. При цьому у пам’яті створюється структура даних, що пов’язана з цим файлом, яка містить такі параметри:

1. Вказівник, який саме відкритий файл треба читати (логічна адреса).
2. Вказівник на буфер користувача у пам’яті, де треба розмістити дані.
3. Кількість байтів, що читаються або записуються.
4. Системний буфер, через який буде відбуватися доступ до файлу.

При відкритті файлу такий системний виклик розміщує в системній області потрібні дані – ім'я файлу, його розширення, дату та час створення або останньої модифікації, тип файлу, його розмір та адресу початкового кластеру. Процес відкриття полягає файлу у перепису відповідного каталогу у пам'ять та пошук у ньому дескриптора файлу за його ім'ям. У разі читання файлу він повертає фактичну кількість прочитаних байтів. Ця кількість може бути менше

тієї кількості, що запитувалась спочатку (наприклад неможливо прочитати 2000 байтів з файлу довжиною 1000 байтів).

З кожним відкритим файлом пов'язаний покажчик, який вказує, який запис буде читатись наступним. Після виконання команди read (у мові Pascal) вказівник збільшується на кількість прочитаних байтів, тому послідовні команди read послідовно читають блоки даних з файлу. Коли програма закінчила читання файлу, вона може закрити його та повідомити ОС, що вона більше не буде використовувати цей файл. При цьому ОС звільняє простір пам'яті таблиці, де зберігалась інформація про цей файл.

Системна команда ОС введення читає черговий запис певного файлу та розміщує його в оперативній пам'ятті (буфері), починаючи з певної адреси. Для того щоб виконати цю операцію, системна команда повинна отримати відомості про те, який файл треба читати та куди у пам'яті розмістити запис.

Основна системна команда виведення записує логічний запис з пам'яті буферу у файл. Послідовні команди write виконують послідовні записи у файл.

При закритті файлу на диск записується останній неповнісю заповнений буфер користувача та оновлюється дескріптор файлу. Справа у тому, що програмісти у своїх програмах створюють власні буфери пам'яті для роботи з файлами – буфери користувача, які за своїм розміром, як правило, більші за системні буфери. Через це фізичний запис буфера у файл відбувається лише тоді, коли буфер користувача повністю заповнений.

Для того щоб зрозуміти, як виконуються системні команди введення/виведення, треба вивчити, як організуються та зберігаються файли. Головне питання – це розподіл дискової пам'яті по блоках. Таким блоком є кластер, який складається з кількох послідовних секторів. Для доступу до кожного такого кластера файла в ОС MS DOS на диску створюється спеціальна зв'язуюча таблиця, що складається з послідовних елементів, які пов'язані з кластерами і які вказують на наступний за номером кластер файла. Така структура організації файлів має назву FAT (File Access Table) – таблиця доступу до файлів. Останній кластер помічений у відповідному елементі FAT спеціальним кодом, який вказує на кінець файлу. Структура файлів та сукупність системних функцій, які забезпечують роботу з файлами, складають файлову систему. Файлова система є одним з основних компонентів операційної системи. Існують різновиди файлової системи FAT: FAT12, FAT16 та FAT32 в залежності від розмірів елементів FAT (12 біт, 16 біт та 32 біт відповідно). В ОС Windows використовується файлова система NTFS, яка вже використовує більш складну структуру про файли у вигляді спеціальної бази даних. Треба додати, що розбиття диска на сектори здійснюється при ініціюванні (розмітці) дисків, тобто при виконанні спеціальної програми розмітки, яка до того ж створює початковий (кореневий) каталог, а також таблицю FAT у двох примірниках – для надійності як важливу інформацію. Кореневий каталог, на відміну від інших, які можуть створюватись користувачем, не може бути знищеним, оскільки для нього виділяється дисковий простір при розмітці. Решта дискового простору виділяється власно під файли. Крім того, на нульовому секторі першої магнітної смужки записується програма-завантажувач

ОС (так званий BOOT-сектор). Оскільки файли на дисках періодично створюються, знищуються та модифікуються, то кластери, які можуть бути відведені під файл, не обов'язково повинні бути слідкувати один за одним по порядку.

Інформація у файлах, які використовуються в ЕОМ, може бути як статичною, так і динамічною. Так, чисельна, символьна та логічна інформація є статичною – її значення не пов'язано з часом. Файли програм для виконання (розширення .COM, .EXE), текстові файли (розширення .DOC, .TXT), а також файли текстів програм на мовах програмування (PAS, C, PRG, ASM, тощо) та пакетів програм (.BAT) є статичними. Але, аудіо-інформація має динамічний характер – існує тільки в режимі реального часу і не може бути зупинена для докладного вивчення. Якщо змінювати мірило часу, то аудіоінформація викривлюється, що може бути використано для створення звукових ефектів.

Відеоінформація. Буває як статичною, так і динамічною. Статична відеоінформація включає в себе текст, малюнки, графіки, креслення, таблиці і т.і. Малюнки розподіляються на плоскі – двовимірні та об'ємні – трьохвимірні.

Динамічна інформація – це відео-, мульт- та слайд-фільми. В їхній основі лежить послідовне відображення на екрані у реальному часі окремих кадрів у відповідності зі сценарієм. Динамічна інформація використовується або для передачі рухливих зображень (анімація), або для послідовної демонстрації окремих кадрів (слайд-фільми). Анімаційні фільми демонструють таким чином, щоб зоровий апарат людини не міг фіксувати окремих кадрів (для якісної анімації кадри мають змінюватись порядка 70 разів/сек.). При демонстрації слайд-фільмів кожен кадр експонується на екрані стільки часу, скільки необхідно для сприйняття його людиною (зазвичай від 30 сек. До 1 хвил.). Слайд-фільми можна віднести до статичної відеоінформації.

В обчислювальній техниці існує два способи подання графічних зображень: *матричний (растровий)* та *векторний.* Матричні формати (bitmap) добре підходять для зображень зі складними гамами кольорів, відтінків та форм, таких як фотографії, малюнки, відскановані дані. Векторні формати більше пристосовані до креслень та зображень з простими формами, тінями та кольором.

В матричних форматах зображення подається прямокутною матрицею крапок – *пікселів* (picture element), положення яких в матриці відповідає координатам крапок на екрані. Окрім координат кожен піксел характеризується своїм кольором, кольором фону або градацією яскравості. Кількість бітів, що виділяються для вказівки кольору піксела, змінюється в залежності від формату. У високоякісних зображеннях кольори піксела описують 24 бітами, що дає біля 16 млн. кольорів. Основним недоліком матричної (растрової) графіки є великий обсяг пам'яті, який необхідний для збереження зображення, через що для опису зображень використовують різні форми стиснення даних. На даний час існує велика кількість форматів графічних файлів, які розрізняються алгоритмами стиснення та способами подання матричних зображень (табл. 5.1).

Табл. 5.1. Формати матричних зображень файлів

Позначка	Повна назва
BMP	Windows та OS\2 Bitmap
GIF	Graphics Interchange Format
PCX	PC Paintbrush File Format
JPEG	Joint Photographic Experts Group
TIFF	Tagged Image File Format
PNG	Portable Network Graphics

Векторне уявлення, на відміну від матричної графіки, визначає опис зображення не пікселами, а кривими – сплайнами. Сплайн – це гладка крива, що проходить через дві або більше опорні точки, що керують формою сплайна. У векторній графіці для сплайнів використовуються спеціальні криві. Головна перевага векторної графіки – опис об'єкту є простим і займає мало пам'яті. Крім того, векторна графіка у порівнянні з матричною має такі переваги: 1) простота зміни мірила зображення без погіршення його якості; 2) незалежність розміру пам'яті для збереження зображення від обраної кольорової моделі.

Недоліком векторних зображень є їхня штучність, яка полягає в тому, що будь-яке зображення необхідно розбити на кінцеву множину примітивів, що її складають. Як і для матричної графіки, існує кілька форматів графічних векторних файлів (табл. 5.2).

Табл. 5.2. Формати векторних зображень файлів

Позначка	Повна назва
DXF	Drawing Interchange Format
CDR	Corel Drawing
HPGL	Hewlett-Paccard Graphics Language
PS	PostScript
SVG	Scalable Vector Graphics
VSD	Microsoft Visio format

Матрична та векторна графіка існують не окремо один від одного. Графічні формати, що дозволяють поєднувати матричний та векторний опис зображення, називають *метафайлами*. Метафайли забезпечують достатню компактність файлів із збереженням високої якості зображення. Типи таких файлів наведені нижче.

Табл. 5.3. Формати метафайлів

Позначка	Повна назва
EPS	Encapsulated PostScript
WMF	Windows Metafile
CGM	Computer Graphic Metafile

Розглянуті форми подання статичної відеоінформації використовуються для окремих кадрів, що являють собою анімаційні фільми. Для збереження

анімаційних фільмів застосовуються різні методи стиснення інформації, більшість з яких стандартизовано.

Аудіоінформація. Поняття аудіо пов'язано із звуками, які сприймає наше вухо. Частоти аудіо сигналів лежать у діапазоні від 15 гц до 20 кгц, а сигнали за своєю природою є неперервними (аналоговими). Перш ніж бути поданою аудіоінформація має бути перетворена у цифрову (оцифрована). Для цього значення звукових сигналів, що взяті через малі проміжки часу, за допомогою аналогово-цифрового перетворювача (АЦП) переводяться у двійковий код. Зворотнє перетворення здійснюється цифро-аналоговим перетворювачем (ЦАП). Чим частіше виконуються виборки звукового сигналу, тим вище якість відновлення початкового сигналу, але тим більше пам'яті вимагається для збереження оцифрованого звуку. Цифровий еквівалент аудіо сигналу зазвичай зберігається у вигляді файлів, причому широко застосовуються різні методи стиснення такої інформації. Як правило, до методів стиснення аудіо інформації ставиться вимога можливості відновлення неперервного сигналу без відчутного погіршення його якості. На даний час розповсюджена ціла низка форматів для збереження аудіоінформації. Наведемо деякі з них. Зауважимо, що файли з розширенням AVI використовуються для відеофільмів і поєднують у собі як відео зображення, так і звук.

Табл. 5.4. Формати аудіо-файлів

Позначка	**Повна назва**
AVI	Audio Video Interleave
WAF	WAVeform Extension
MIDI	Musical Instrument Digital Interface
AIF	Audio Interchange Format
MPEG	Motion Picture Expert Group Audio
RA	Real Audio

Контрольні запитання для самоперевірки

1. Що являє собою файл, які існують види файлів?
2. В яких одиницях вимірюється розмір файлів?
3. Що являє собою каталог (папка, тека)?
4. Яке призначення кореневого каталогу?
5. З чого складається фізична адреса файла на диску?
6. Що являє собою кластер і на що впливає його розмір?
7. Які основні операції виконуються над файлами?
8. Що відбувається при розмітці диску?
9. Яка інформація міститься у структурі даних після відкриття файлів?
10. Що відбувається при закритті файла?
10. Які існують основні форми подання зображень?
12. Чим відрізняються між собою формати файлів для подання зображень?

6. Системи числення

6.1. Загальні уявлення

Почнемо з деяких загальних міркувань стосовно поняття *число*. Можна вважати, що будь-яке число має ***значення*** (вміст) та ***форму подання***. Значення числа задає його відношення до значень інших чисел («більше», «менше», «дорівнює», «не дорівнює») і, отже, порядок розташування чисел на числовій вісі. Форма подання, як випливає з назви, визначає порядок запису числа за допомогою призначених для цього знаків. При цьому значення числа є інваріантом, тобто не залежить від способу його подання. Це означає також, що число з одним і тим самим значенням може бути записано по-різному, тобто відсутня взаємно однозначна відповідність між поданням числа та його значенням. У зв'язку з цим виникають питання, по-перше, про форми подання чисел, і, по-друге, про можливість та способи переходу від однієї форми до іншої. Спосіб подання числа визначається системою числення.

Різні системи числення відрізняються не лише базовим набором знаків, але й основними концепціями, які лежать в їх основі. Взяти, наприклад, систему числення, яка використовувалася древніми римлянами: вона досить важка для сприйняття, в ній дуже складно виконати обчислення і неможливо представити 0. Дана система незручна навіть для людини, не кажучи вже про те, аби навчити комп'ютер «розуміти» її.

Системою числення називають сукупність правил запису чисел за допомогою заданого набору спеціальних знаків (цифр). Для запису числа використовуються спеціальні символи – цифри. Кількість цифр називають **основою** системи числення. Так, у двійковій системі числення всього дві цифри: 0 та 1. У десятковій системі використовуються добре знайомі нам десять арабських цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. У шістнадцятковій системі числення використовуються десять арабських цифр, а як останні шість – перші букви латинської абетки: A, B, C, D, E, F.

Існують позиційні та непозиційні системи числення.

Людьми використовувалися різні способи запису чисел, які можна об'єднати в декілька груп : *унарна; непозиційні; позиційні.*

Унарна – це система числення, в якій для запису чисел використовується тільки один знак, - **I** ("паличка"). Наступне число виходить з попереднього додаванням нової **I**; їх кількість (сума) дорівнює самому числу. Саме така система застосовується для початкового навчання рахунку дітей (можна згадати "рахункові палички"). Використання унарної системи виявляється важливим педагогічним прийомом для введення дітей у світ чисел і дій з ними. Але, як ми побачимо надалі, унарна система важлива також в теоретичному відношенні, оскільки в ній число представляється найбільш простим способом і, отже, прості операції з ним. Крім того, саме унарна система визначає значення цілого числа кількістю одиниць, що містяться у ньому. Для запису числа в унарній системі надалі використовуватимемо позначення ***Z1.***

У *непозиційних системах* вага цифри (тобто той внесок, який вона додає до значення числа) не залежить від її позиції у запису числа. Так, у римській системі числення у числі XXXII (тридцять два) вага цифри X у будь-якій позиції завжди дорівнює десяти.

У ***позиційній системі*** числення значущість кожної цифри визначається її положенням (позицією) у записі числа. Номер позиції відлічується справа, від молодшої цифри цілого числа (відповідає нульовому ступеню основи). Звична для нас десяткова система є позиційною. Її основою є число 10. У числі 1923, наприклад, перша справа цифра відображає кількість одиниць (3), друга – кількість десятків (2), третя – кількість сотень (9), а права вказує кількість тисяч (1). Запис 1923 можна вважати скороченим записом виразу: $1*10^3+9*10^2+2*10^1+3*10^0$, де 0, 1, 2, 3 – показники ступеня, а 10^3 – 10^0 – ваги розрядів числа.

Аналогічно можна представити десятковий дріб, наприклад 3,14, тільки показники ступеня для дробової частини будуть від'ємними: $3*10^0+1*10^{-1}+4*10^{-2}$.

Запис чисел у будь-якій позиційній системі числення з основою q означає скорочений запис виразу

$$a_{n-1} q^{n-1} + a_{n-2} q^{n-2} + ... + a_1 q^1 + a_0 q^0 + a_{-1} q^{-1} + ... + a_{-m} q^{-m},$$

де a_i – цифри системи числення; n і m – кількість цілих і дробових цифр відповідно.

Як бачимо, існує багато способів запису чисел цифровими знаками. Але система числення, яка призначена для практичного використання, повинна забезпечувати:

- можливість подання будь-якого числа у заданому діапазоні чисел;
- однозначність подання;
- стислість та простоту запису чисел;
- легкість сприйняття системи;
- простоту та зручність виконання арифметичних операцій у ній.

6.2. Римська система числення

Розглянемо, як приклад, непозиційну римську систему числення. Базис цієї системи числення утворюють великі латинські літери:

Q={**I-1,V-5, X-10, L-50, C-100, D-500, M-1000**}.

Сім цих особливих знаків діляться на дві групи.

1. Чотири знаки **десяткових розрядів**

I = 1, X = 10, C = 100, M = 1000.

2. Три знаки ***половин*** **десяткових розрядів**

V = 5, L = 50, D = 500.

6.2.1. Читання римських цифр

Цілі позитивні числа можна записувати повторенням римських цифр. Щоб дізнатіся, що означає число, записане римськими цифрами, треба скористатися двома правилами.

Правило додавання римських цифр. Якщо всі цифри в числі по значенням не зростають, якщо вважати зліва направо, то вони *додаються*.

Приклади.

1. II = 2, VI = 6, XI = 11 — правильно.
2. IV = 6, XL = 60 — неправильно.

Правило віднімання римських цифр.

1. Спочатку в усіх парах, де менша цифра стоїть перед більшою, віднімається менша цифра з більшої.
2. Потім отримані результати разом з незайманими цифрами підпадають під правило додавання і додаються.

Приклади.

1. IV = 4, XIV = 14, XXIX = 29 – правильно.
2. IVX = 6, IXX = 1 – неправильно.

Алгоритм обчислення значення римських чисел.

1. Число проглядається зліва направо по одній цифрі.
2. Якщо поточна цифра більше або дорівнює наступній, то вона додається до значення числа.
3. Якщо поточна цифра менша за наступну, то з наступної цифри віднімається поточна, а результат додається до значення числа.
4. Якщо цифра остання, то вона додається до значення числа.

6.2.2. Запис римських цифр

У попередньому пункті римські числа розшифровувалися. Розглянемо алгоритм, по якому числа записуються римськими цифрами.

Алгоритм запису римських цифр.

1. Число записується зліва направо максимально можливими цифрами.
2. Але якщо в п. 1 виходить 4 однакових десяткових знаків підряд, то
3. тоді вони замінюються цим десятковим знаком і наступним половинним. (Наприклад, 40=XXXX=XL).
4. Але якщо при заміні в п. 3 десятковий знак виявляється між двома *однаковими половинними*, то ці 3 знаки замінюються цим *десятковим* знаком і *наступним десятковим*. (Наприклад, 9=VIIII=VIV=IX).

З цього алгоритму, як наслідок, можна зробити наступні висновки:

- цифри **I, X, C** і **M** можуть слідувати поспіль не більше трьох разів кожна;
- цифри **V, L** і **D** можуть використовуватися в запису числа не більше одного разу.

Приклади.

1. 3 = III; 6 = VI; 11 = XI; 22 = XXII.

2. 4 = IV, а не IIII; 45 = XLV, а не XXXXV.
3. 9 = IX, а не VIIII або VIV; 19 = XIX, а не XVIIII або XVIV.
4. 1549 = MDXLIX.

В якості демонстрації випишемо всі одиниці, десятки і сотні, записані в римській системі числення:

I	**II**	**III**	**IV**	**V**	**VI**	**VII**	**VIII**	**IX**	**X**
X	**XX**	**XXX**	**XL**	**L**	**LX**	**LXX**	**LXXX**	**XC**	**C**
C	**CC**	**CCC**	**CD**	**D**	**DC**	**DCC**	**DCCC**	**CM**	**M**

Запис чисел в такій системі громіздкий і незручний, але ще більше незручним виявляється виконання в ній навіть найпростіших арифметичних операцій. Відсутність нуля і знаків для чисел більше **M** не дозволяють римськими цифрами записати будь-яке число (хоч би натуральне). З указаних причин тепер римська система використовується лише для нумерації, при написанні століття, місяця і дати. Наприклад, максимальне натуральне число, яке можна записати, – це 3999. Використовуючи алгоритм запису римських цифр, отримаємо по кроках максимальне число:

MMM → MMMDCCCC → MMMDCD → MMMCM → MMMCMLXXXX → MMMCMLXL → MMMCMXC → MMMCMXCVIIII → MMMCMXCVIV → MMMCMXCIX.

6.3. Системи числення, що використовуються в обчислювальній техніці

Безумовно, одним з основних напрямів вживання комп’ютерів були і залишаються усілякі числення. Обробка числової інформації ведеться і при розв’язанні задач, які на перший погляд не пов’язані з якимись розрахунками, наприклад, при використанні комп’ютерної графіки або звуку. У зв’язку з цим постає питання про вибір оптимального подання чисел у комп’ютері.

Електронно-обчислювальна машина тому й називається "електронною", що фактично працює на електроенергії. Якщо припустити, що комп’ютер складається з набору якихось умовних логічних елементів, то у кожен момент часу ці елементи можуть перебувати тільки в одному з двох стійких станах: "є струм" чи немає струму". Прийнявши стан "є струм" за "одиницю", а стан "немає струму" – за "нуль", ми тим самим прийдемо до очевидного висновку: будь-які дані можна помістити в комп’ютер у вигляді послідовності одиниць та нулів, а на фізичному рівні – у формі низки електронних імпульсів.

Залишилося тільки домовитися про те, як звести всю різноманітність інформації, яка оточує нас, до одиниць та нулів. Саме тут нам на допомогу приходить математика, оскільки наведена вище схема подання інформації чудово вкладається у найпростішу для технічної реалізації, так звану, двійкову систему числення. Для запам’ятовування двійкових чисел можна використовувати пристрої, які мають два стійкі стани. Один із цих станів можна вважати «нулем», а інший – «одиницею». Простим пристроєм з двома станами є звичайна електрична лампочка! Вона може бути або увімкнена, або вимкнена.

Зрозуміло, людині зручніше працювати із звичною десятковою системою числення, але технічна реалізація такого комп'ютера виявилася б набагато складнішою. В кінці 1950-х років в СРСР під керівництвом Н. П. Бруснецова була розроблена серія обчислювальних машин «Сетунь» на основі трійкової арифметики. У цій арифметиці замість бітів використовувалися «тріти», а для їх технічної реалізації застосовувалися трьохстабільні пристрої. Комп'ютери «Сетунь» та «Сетунь 70» успішно застосовувалися для розв'язку різних завдань, виявилось, що вони мають низку переваг перед двійковими комп'ютерами, але широкого поширення так і не знайшли.

У сучасній обчислювальній техніці застосовується двійкова система числення, а програмісти для скорочення громіздкого двійкового запису застосовують вісімкову та шістнадцяткову системи числення, тобто системи з основою «цілий позитивний ступінь двійки». Виняткове використання двійкової системи пов'язане з низкою її особливостей:

1. Відносна простота технічної реалізації за допомогою пристроїв (тригерів), що мають два стійкі стани або, наприклад, наявність або відсутність заряду в конденсаторі. Для реалізації *p-ічної* системи знадобилися б пристрої з *p* стійкими станами, які є складнішими при реалізації та, до того ж, менш надійні.

2. Відносна простота реалізації двійкової арифметики.

3. Простота реалізації логічних операцій, тобто операцій над значеннями «істина» та «хиба». Відповідний розділ математики називається ***булевою логікою***. Булева логіка передбачає, що окрім «істини» та «хиби» нічого більш немає. Двійкова система у цьому випадку підходить ідеально. Звичайно, можна розглядати і багатозначну логіку, яка, можливо, більш зрозуміло описує реалії навколишнього світу (коли вислів може бути і не «хибою», і не «істиною»), але вона складніша для реалізації в автоматичних обчислювальних пристроях.

4. Арифметичні операції можна формалізувати в термінах булевої логіки, а булеві логічні операції порівняно легко реалізуються цифровими електронними пристроями. Одним з недоліків двійкової системи – її громіздкість. Порівняйте, скільки символів знадобиться для запису одного й того ж числа у десятковій та у двійковій системах: 2003_{10} (або $(2003)_{10}$) (або $(11111010011)_2$). У першому випадку знадобилися всього 4 розряди – проти одинадцяти в другому!

Двійкова система, зручна для комп'ютерів, а для людини – незручна через її громіздкість та незвичний запис.

Переведення чисел з десяткової системи у двійкову та навпаки виконують програми комп'ютера. Проте, щоб професійно використовувати комп'ютер, слід навчитися розуміти слово комп'ютера. Для цього використовується вісімкова та шістнадцяткова системи.

Числа у цих системах читаються майже так само легко, як десяткові та вимагають відповідно у три (вісімкова) та у чотири (шістнадцяткова) рази менше розрядів, ніж у двійковій системі (адже числа 8 та 16 – відповідно, третя та четверта ступені числа 2). Вісімкова система числення набула широко

використовувалася у програмуванні у 1950-70-і рр. Дотепер практично повністю витиснена шістнадцятковою системою числення.

Запис з дужками (або без дужок) та нижнім індексом у тексті програми незручний, тому використовуються наступні позначки при записі чисел для різних систем числення:

1. Запис шістнадцяткового числа починається з 0x або $0, або закінчується символом «h». Якщо перша цифра шістнадцяткового числа – символи A-F, то перед таким числом обов'язково потрібно написати 0, аби компілятор зрозумів, що перед ним число, а не ідентифікатор, наприклад, 0DEADh.

Таким чином, записи 0x1234, $01234 і 01234h представляють число $(1234)_{16}$.

2. Десяткові числа можуть записуватися без змін або закінчуватися постфіксом «d». Наприклад, 1234 та 1234d представляють число $(1234)_{10}$.

3. Двійкові цифри повинні закінчуватися постфіксом «b», наприклад, 1100b – це $(1100)_2$.

Найчастіше шістнадцяткові числа в різних системах програмування записуються у вигляді «0x...», двійкові – з постфіксом «...b», а десяткові – без змін. Основу системи числення краще вказувати явно, тому що деякі мови програмування (особливо асемблери) розглядають число без префіксів як десяткове, а інші – як шістнадцяткове.

6.4. Перетворення між системами числення

Оскільки одне і те саме число може бути записане в різних системах числення (СЧ), постає питання про перетворення представлення числа з однієї системи (p) в іншу (q) – позначимо таке перетворення Zp → Zq. Теоретично можливо зробити його при будь-яких q та p. Проте подібне пряме перетворення буде ускладнено тим, що доведеться виконувати операції за правилами арифметики недесяткових систем числення. З цієї причини зручнішими з практичної точки зору виявляються варіанти перетворення з проміжним переведенням Zp → Zr → Zq з основою r, для якого арифметичні операції виконати легко. Такими зручними основами є r = 1 та r = 10, тобто перетворення здійснюється через унарну або десяткову систему числення. Такий метод перетворення називається ***розрахунковим.***

6.4.1. Перетворення цілих чисел за схемою Zp → Z1 → Zq

Ідея алгоритму перетворення проста: покладемо початкове значення Zq := 0; з числа Zp віднімемо 1 за правилами віднімання системи p, тобто Zp:= Zp - 1 та додамо її до Zq за правилами додавання системи q, тобто Zq:= Zq + 1; повторюватимемо цю послідовність дій, поки не досягнемо Zp = 0. Правила додавання 1 та віднімання 1 можуть бути записані таким чином:

Для системи *p*	Для системи *q*
(p–1) – 1 = (p–2)	0 + 1 = 1
(p–2) – 1 = (p–3)	1 + 1 = 2
…	…
1 – 1 = 0	(q–2) + 1 = (q–1)
0 – 1 = 1 (p–1)	(q–1) + 1 = 1 0

Приклад. Виконати перетворення $22_3 \rightarrow Z_6$. Послідовність дій і проміжні результати для наочності представимо у вигляді таблиці:

Крок	0	1	2	3	4	5	6	7	8
Z_3 - 1	22	21	20	12	11	10	2	1	0
Z_6 + 1	0	1	2	3	4	5	10	11	12

Отже, $22_3 = 12_6$.

Розглянутий алгоритм перетворення може бути легко реалізований програмним шляхом, зокрема машиною Тьюринга.

6.4.2. Перетворення цілих чисел за схемою $Z_p \rightarrow Z_{10} \rightarrow Z_q$

Очевидно, що перша і друга частина перетворення не повязані одна з одною, що дає підставу розглядати їх окремо.

Алгоритми перетворення $\mathbf{Z_{10} \rightarrow Z_q}$ витікають з наступних міркувань. Нехай задано число A у десятковій позиційній системі числення, і його необхідно перевести у нову систему з основою **q**, тобто перетворити до вигляду

$$A = a_{n-1}q^{n-1} + a_{n-2}q^{n-2} + \ldots + a_1q^1 + a_0q^0 \qquad (\mathbf{6.1})$$

де $a_i = 0 \div (q\text{-}1)$, q – основа нової системи числення.

З формули (6.1) безпосередньо витікають два способи перетворення цілих чисел з 10-ої системи числення у систему з довільною основою ***q***.

Перший спосіб полягає у тому, що вираз (6.1) можна записати у вигляді $A = A_1q + a_0$, де $A_1 = a_{n-1}q^{n-2} + a_{n-2}q^{n-3} + \ldots + a_2q + a_1$ – ціла частина від ділення A на q; a_0 – залишок від ділення A на q, яка є цифрою молодшого розряду шуканого числа, збережені в символах старої системи числення.

При діленні числа A_1 на q тим же способом отримаємо залишок a_1 і так далі. Іншими словами, вираз (6.1) записується за схемою Горнера:

$$A = (\ldots((a_{n-1}q + a_{n-2})q + a_{n-3})q + \ldots + a_1)q + a_0,$$

після чого його права частина послідовно ділиться на основу нової системи числення q.

Таким чином, в результаті низки ділень даного числа на основу нової системи числення q знаходимо коефіцієнти $a_0, a_1, a_2, \ldots, a_{n-1}$:

$A = A1 * q + a_0;$

$A1 = A2 * q + a_1;$

. . . . . .

An-2 = An-1 * q + a_{n-2};

An-1 = An-1 * q + a_{n-1};

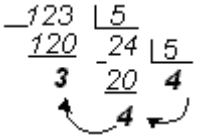
При цьому розподіл продовжується доти, доки не будуть виконані співвідношення: $A_{n-1} < q$; $A_n = 0$.

Розглянемо основні правила перетворення цілих чисел з однієї системи числення в іншу.

Для перетворення цілого числа X_{10} із системи числення з основою $p = 10$ у систему числення з основою q використовується правило ділення:

1. Число X_{10} ділиться на нову основу q, яке представлене у десятковій СЧ.
2. Отриманий від першого ділення залишок є кількісним еквівалентом молодшої цифри числа з основою q.
3. Результат ділення знову ділиться на основу q. При цьлму отримуємо новий залишок, кількісний еквівалент якого дорівнює наступній цифрі числа з основою q.
4. Ділення відбувається доти, доки результат не буде менше за дільник. Остання частка дасть кількісний еквівалент старшої цифри числа з основою q.
5. Після завершення ділення перевести кількісні еквіваленти у цифри в q-тій системі числення та записати їх справа наліво у зворотному порядку щодо їх оримання.

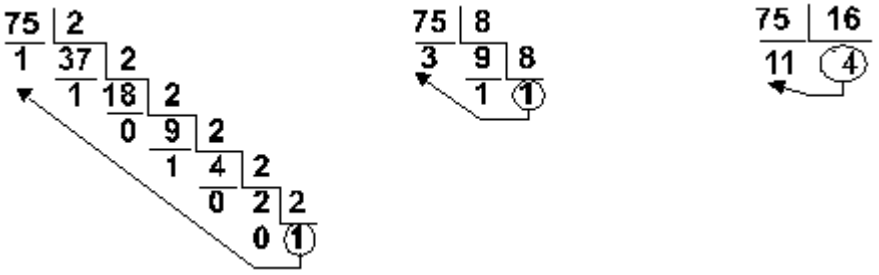
Приклад 1. Виконати перетворення $123_{10} \rightarrow A_5$.



Залишки від ділення та результат останнього цілочисельного ділення утворюють зворотний порядок цифр нового числа. Отже, $123_{10} = 443_5$.

Необхідно зауважити, що отримане число не можна читати "чотириста сорок три", оскільки десятки, сотні, тисячі та інші подібні позначення чисел відносяться тільки до десятковій системі числення. Читати число слід простим перерахуванням його цифр із зазначенням системи числення ("число чотири, чотири, три в пятеричній системі числення").

Приклад 2. Перетворити число 75 із десяткової системи у двійкову, вісімкову та шістнадцяткову методом “ділення кутом (драбинкою)”:



В результаті отримаємо: $75_{10} = 1001011_2 = 113_8 = 4B_{16}$.

Алгоритм зворотного перетворення цілого числа з р-кової системи у десяткову виглядає так само, як і при загальному перетворенні, тобто необхідно організувати цикл ділення на 10, але 10 має бути подана у тій системі численні, з якої відбувається перетворення.

Приклад 3. Перевести задані числа у десяткову систему числення. Для перетворення заданих чисел скористаємося виразом (6.1):

a) $\mathbf{731.5_8=7*8^2+3*8^1+1*8^0+5*8^{-1}=473.625_{10}}$;

b) $\mathbf{110011.01_2 =1*2^5+1*2^4+1*2^1+1*2^0+1*2^{-2}=51.25_{10}}$;

c) $\mathbf{A7F.\ D_{16}=10*16^2+\ 7*16^1+15*16^0+13*16^{-1}=2687.8125_{10}}$.

Другий спосіб випливає з виразу (6.1), якщо дії проводяться у відповідності з наступним алгоритмом:

1. Визначити **n-1** – максимальний покажчик ступеня у поданні числа за формою (6.1) для основи **q**.

2. Цілочисельно розділити число ($\mathbf{A_{10}}$) на основу нової системи числення в ступені **n - 1** (тобто $\mathbf{q^{n-1}}$) і знайти залишок від ділення; результат ділення визначить першу цифру числа $\mathbf{A_{n-1}}$;

3. Залишок від ділення целочисленно розділити на $\mathbf{q^{n-2}}$, результат ділення прийняти за другу цифру нового числа; знайти залишок; продовжувати цю послідовність дій, поки показник ступеня **q** не досягне значення 0.

Такий спосіб перетворення числа ще називають "**підрахунок обміну**", оскільки він аналогічний підрахунку наявності та обміну грошей в автоматі. Якщо необхідно визначити, які монети віддати в обмін при 42 копійках, то першою буде обрана монета в 25 копійок, оскільки це найбільша монета, що менше 42. Якщо відняти 25 з 42, то необхідно ще 17 копійок. Тепер вибираємо монету в 10 копійок і віднімаємо з 17, потім з 7 віднімаємо 5 копійок, залишається ще 2 копійки, які необхідно додати.

Подібним чином, якщо є, наприклад, число 83_{10}, яке потрібно перевести у двійкове, то спочатку обирається найбільша ступінь двійки, яка не перевершує 83_{10}. В даному випадку це $2^6 = 64_{10}$. Розміщуємо 1 в стовпчик, який відповідає 2^6. Віднімаємо 64_{10} з 83_{10} і отримуємо 19_{10}: $2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 1$

Нам не потрібно використовувати $2^5 = 32_{10}$ (воно більше 19_{10}), тому у цьому стовпчику розміщуємо 0 і т.д.:

1	0	1	0	0	1	1
2^6	2^5	2^4	2^3	2^2	2^1	1^0,

таким чином, $83_{10} = 1010011_2$.

Продемонструємо дію алгоритму у загальному випадку на тому ж прикладі, що був розглянутий вище (Приклад 1). Визначити *n-1* можна або шляхом підбору ($5^0 = 1 <123$; $5^1 = 5 <123$; $5^2 = 25 <123$; $5^3 = 125 > 123$, отже, *n–1=2*), або логарифмуванням з залишенням цілої частини логарифму ($\log_5 123 = 2{,}99$, тобто *n-1 = 2*).

Далі (*r* – залишок від ділення):

$$\mathbf{a_2 = 123\ div\ 5^2 = 4;}$$
$$\mathbf{r_1 = 123\ mod\ 5^2 = 23;}$$
$$\mathbf{(i = 2 - 1 = 1);}$$
$$\mathbf{a_1 = 23\ div\ 5^1 = 4;}$$

$$\mathbf{r_0 = a_0 = 23 \bmod 5^1 = 3;}$$
$$\mathbf{(i) = 0.}$$

Тобто, $123_{10} = 443_5$.

Алгоритм зворотного перетворення цілого числа з р-кової системи у десяткову виглядає так само, як і при загальному перетворенні, тобто необхідно організувати цикл ділення на 10, але 10 має бути подана у тій системі численні, з якої відбувається перетворення. Тому для зворотного перетворення цілого числа за схемою $\mathbf{Zp \rightarrow Z_{10}}$ краще скористатися виразом (6.1): необхідно Zp представити у формі многочлена і виконати всі операції за правилами десяткової арифметики.

Приклад 3. Виконати зворотне перетворення $443_5 \rightarrow Z_{10}$.

$443_5 = 4\cdot5^2 + 4\cdot5^1 + 3\cdot5^0 = 4\cdot25 + 4\cdot5 + 3\cdot1 = 123_{10}$.

Приклад 4. Перевести задані числа у десяткову систему числення. Для перетворення чисел скористаємося виразом (6.1):

a) $\mathbf{731.5_8 = 7*8^2+3*8^1+1*8^0+5*8^{-1}=473.625_{10}}$;

b) $\mathbf{110011.01_2 = 1*2^5+1*2^4+1*2^1+1*2^0+1*2^{-2}=51.25_{10}}$;

c) $\mathbf{A7F.\ D_{16}=10*16^2+\ 7*16^1+15*16^0+13*16^{-1}=2687.8125_{10}}$.

Необхідно ще раз підкреслити, що наведеними алгоритмами зручно користуватися коли потрібно перевести число з десяткової системи в якусь іншу або навпаки. Вони працюють і для перетворення між будь-якими іншими системами числення, але перетворення буде ускладнено тим, що всі арифметичні операції необхідно здійснювати за правилами вихідної (в перших алгоритмах) або кінцевої (в останньому алгоритмі) системи числення. З цієї причини перетворення, наприклад, $Z_3 \rightarrow Z_8$ простіше здійснити через проміжне перетворення до 10-ї системі $Z_3 \rightarrow Z_{10} \rightarrow Z_8$.

Ситуація значно спрощується, якщо основи систем числення виявляються пов'язаними співвідношенням $\mathbf{p = q^r}$, де *r* – ціле число (природно, більше 1).

6.5. Перетворення між двійковою, 8-ковою та 16-ковою системами

Найбільшу цифру вісімкового розряду (7_8) можна представити двійковою тріадою – тризначним двійковим числом (111_2), а найбільшу цифру шістнадцяткового (F_{16}) – двійковою тетрадою – чотиризначним (1111_2) двійковим числом. Перетворення вісімкових та шістнадцяткових чисел у двійкову систему дуже просте. Для того щоб перетворити вісімкове представлення числа у двійкове, досить замінити кожну його цифру двійковою тріадою, наприклад: $3.14_8 = 011.001100_2$.

Для перетворення шістнадцяткового числа у двійкове слід кожну шістнадцяткову цифру замінити двійковою тетрадою: $A.BC_{16} = 1010.1011\ 1100_2$.

Таблиця 6.1. Представлення чисел в системах числення з основами 10, 2, 8, 16

Десяткова	Двійкова	8-кова	16-кова
0	0000	00	0
1	0001	01	1
2	0010	02	2
3	0011	03	3
4	0100	04	4
5	0101	05	5
6	0110	06	6
7	0111	07	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Нескладно вивести правило зворотного перетворення двійкових чисел у вісімкові бо шістнадцяткові. Для цього ціле двійкове число розбивається на тріади (тетради), починаючи з молодшого двійкового розряду, що розміщений праворуч. Якщо двійкове число не ціле, а дріб, розбиття проводиться ліворуч та праворуч від дробової крапки:

$$\underbrace{001}_{1}\underbrace{101}_{5}\underbrace{111}_{7}\underbrace{001}_{1}.\underbrace{110}_{6}\underbrace{100}_{4} = 1571.64_8$$

$$\underbrace{0111}_{7}\underbrace{1111}_{F}\underbrace{1011}_{B}.\underbrace{1001}_{9}\underbrace{1100}_{C} = 7FB.9C_{16}$$

Для перетворення чисел між системами з основами 2, 8, 10 і 16 можна використовувати табл. 6.1.

6.6. Перетворення правильної дробі

Дійсне число у загальному випадку містить цілу та дробову частину і завжди можна представити у вигляді суми цілого числа та правильного дробу. Оскільки проблема перетворення натуральних чисел в різних системах числення вже була розглянута, можна обмежити розгляд тільки алгоритмами перетворення правильних дробів. Нагадаємо, що правильний дріб – це дробове число, у якого ціла частина дорівнює нулю.

Введемо наступні позначки: правильний дріб у вихідній системі числення **p** будемо записувати у вигляді **0,Yp**, а дріб у системі **q** – **0, Yq**, а перетворення – у вигляді **0, Yp → 0,Yq**.

Послідовність міркувань дуже нагадує проведену раніше для натуральних чисел. Зокрема, це стосується рекомендації здійснювати перетворення через проміжний перехід до 10-ї системі, щоб уникнути необхідності робити обчислення у "незвичних" системах відліку, тобто **0,Yp → 0,Y$_{10}$ → 0,Yq**. Це, у свою чергу, розбиває задачу на дві складові: перетворення **0,Yp → 0,Y$_{10}$** та **0,Y$_{10}$ → 0,Yq**, кожне з яких може розглядатися незалежно. Загалом алгоритм переведення **0,Yp → 0,Y$_{10}$** виводиться шляхом наступних міркувань.

Нехай правильний дріб *A*, який заданий у десятковій позиційній системі числення, необхідно перевести у нову систему числення *q*, тобто перетворити до виду

$$A = a_{-1}q^{-1} + a_{-2}q^{-2} + \ ... \ + a_{-k+1}q^{-k+1} + a_{-k}q^{-k}. \qquad \textbf{(6.2)}$$

Якщо, аналогічно перетворенню цілих чисел, розділити обидві частини виразу (6.2) на q^{-1}, тобто помножити на ***q***, то отримаємо $A \times q = a_{-1} + A1$, де $A_1 = a_{-2}q^{-1} + a_{-3}q^{-2} + \ ... \ + a_{-k}q^{-k+1}$ – дріб після множення; a_{-1} – ціла частина результату. Отримана при цьому цифра цілої частини результату і буде першою цифрою шуканого числа. Помноживши дробову частину результату *A1* знову на *q*, будемо мати значення

$A1 \times q = a_{-2} + A2$, де *A2*– дріб після множення (нового); a_{-2} – наступна цифра шуканого числа. Звідси випливає, при перетворенні вираз 6.2) записується за схемою Горнера

$$A = q^{-1}(a_{-1} + q^{-1}(a_{-k} + ... + q^{-1}(a_{-k+1} + q^{-1} \cdot a_{-k})...)).$$

Перемножуючи цей вираз послідовно ***k*** разів на основу *q*, отримаємо перетворене число у новій системі числення із ***k*** цифр.

На відміну від цілих чисел, точне перетворення можливо не для всіх правильних дробів. Похибка при цьому становить 1/2 молодшого розряду числа у новій системі.

Для перетворення правильного дробу X_{10} з десяткової системи числення у систему числення з основою *q* використовується **правило множення**:

1. Дане число множиться на нову основу *q*, яке представлено у 10-й системі числення. Ціла частина значення виразу, яка при цьому утворилася, є кількісним еквівалентом першої шуканої цифри дробу з основою *q*.

2. Дробова частина отриманого значення знов множиться на основу *q*, знов ціла частина буде кількісним еквівалентом наступної цифри числа з основою *q*.

3. Множення дробової частини поточного значення на основу *q* триває доти, доки дробова частина не стане рівною 0, або не буде знайдений період дробу, або буде досягнута потрібна точність обчислень.

Приклад 1. Перетворити число 0,35 з десяткової системи у 2-, 8- та 16-кову системи:

у двійкову	у вісімкову	у шістнадцяткову
0, | 35	0, | 35	0, | 35
| 2	| 8	| 16
0, | 70	2, | 80	5, | 60
| 2	| 8	| 16
1, | 40	6, | 40	9, | 60
| 2	| 8	
0, | 80	3, | 20	
| 2	| 8	

1, | 60 1, | 60
| 2_____
1, | 20
| 2_____
0, | 40
| 2_____
0, | 80
| 2_____
1, | 60

Відповідь: $0,35_{10} = 0,01011001_2 = 0,2631_8 = 0,59_{16}$

Приклад 2. Перетворити число 0.11_{10} в 5-ічну СЧ:

0.11*5=**0**| 0.55, 0.55*5=**2**| 0.75, 0.75*5=**3**| 0,75; таким чином, $0.11_{10}=0.02(3)_5$.

Як видно з обчислень, після чергового множення отримана дробова частина, що вже зустрічалася. Тому при наступному множенні її на основу 5 буде отримана цифра, яка належить періоду дробу. Таким образом, $0.11_{10}=0.02\,(3)_5$.

Приклад 3. Перетворити число 0.31_{10} у двійкову СЧ:

0.31*2=**0**| 0.62, 0.62*2=**1**| 0.24, 0.24*2=**0**| 0.48, 0.48*2=**0**| 0.96, 0.96*2=**1**|0.92, 0.92*2=...

Зупинимо процес множення, оскільки необхідна точність вже досягнута. Таким чином, 0.3110 ~ 0.010011111.

Але, як вже відзначалося, виконувати розрахунки у системі числення з основою p≠10 складно. Тому, для перетворення правильного дробу числа Xp з системи числення з основою p≠10 у число Yq з основою q≠10 можна користуватися схемою:

$Xp \rightarrow Z_{10} \rightarrow Yq$.

Зауваження. При перетворенні правильного дробу з *p*-ічної СЧ у СЧ з основою *q*, коли процес стає досить довгим, зупинити його слід, коли точність числа у СЧ з основою *q* відповідатиме точності числа в *p*-ічній системі. При цьому похибка числа не перевищує половину ваги молодшого розряду, тобто $\frac{1}{2} q^{m}$, де *m* – вага молодшого розряду. Таким чином, процес перетворення слід завершити, коли

$$\frac{1}{2} p^{k} \approx \frac{1}{2} q^{m}$$

де ***k*** – вага молодшого розряду у представленні числа у десятковій системі, ***m*** – кількість розрядів у представленні числа в СЧ з основою *q*, при цьому ***m*** слід обирати за умови:

$$|m| >= |k|/\log_p q$$

Так, для Прикладу 3, p=10, k=-2, q=2, тому за *m* можна взяти число

$$|m| = 7 >= |-2|/\lg 2 \approx 6.643, \text{ тобто } m = -7.$$

Перетворення **0,Yp → 0,Y10**, як і у разі натуральних чисел, зводиться до обчислення значення формули (6.2) у десятковій системі числення. Наприклад,

$$\mathbf{0{,}011_2 = 0\cdot 2^{-1} + 1\cdot 2^{-2} + 1\cdot 2^{-3} = 0{,}25 + 0{,}125 = 0{,}375_{10}}$$

При перетворенні неправильних дробів необхідно окремо перевести цілу та дробову частини числа за наведеними вище правилами перетворення, і записати у новій системі числення, залишивши незмінним положення коми.

Насправді машинний алгоритм перетворення неправильного десяткового дрібу, який використовується при введенні дійсних чисел (наприклад, оператором READ у мові PASCAL), пов'язаний з перетворенням його у формат дійсних даних, але про це далі.

6.7. Швидкий алгоритм перетворення

Один із швидких алгоритмів перетворення з двійкової системи числення у десяткову був запропонований **Соденом** у 1953 році.

З метою скорочення кількості дій спочатку переведемо число з двійкової системи у вісімкову. Для цього розбиваємо справа наліво його цифри на "трійки" (остання "трійка" насправді може бути "двійкою" або навіть однією цифрою) і переводимо їх у вісімкову систему. Наприклад, $(10001010)_2=(010.001.010)_2=(212)_8$.

При перетворенні з вісімкової системи у десяткову записуємо задане у вісімковій системі число, потім на *i -тому* кроці подвоюємо перші *i* цифр, використовуючи десяткову арифметику, і віднімаємо отриманий результат від перших (*i+1*) цифр за правилами десяткової арифметики. Якщо задане число складається з *n* цифр, то цей процес закінчується через (*n - 1*) кроків. Перетворення здійснюється виходячи з основного алгоритму шляхом ділення на 10, тобто (8 + 2). Тому подвоєна старша цифра зсувається на один розряд праворуч і віднімається від числа. З метою запобігання можливих помилок вводять розділову точку, щоб виділити подвоєні цифри.

Приклад 1. Перевести число 212_8 до десяткового вигляду

```
 2.12
   4
 17.2
  3 4
 13 8,  тобто 212₈ = 138₁₀.
```

Алгоритм перетворення з десяткової системи числення у двійкову, запропонований **Розье** у 1962 році, майже такий самий. Спочатку число, з метою скорочення кількості дій, перекладається у вісімковий запис.

Записуємо задане у десятковій системі число. На *i-тому* кроці, використовуючи вісімкову арифметику, подвоюємо перші *i* цифр і додаємо, також за допомогою вісімкової арифметики, ці подвоєні цифри до перших (*i+1*) цифр. Якщо задано число з *n* цифр, то цей процес закінчиться через (*n - 1*) кроків. Перетворення здійснюється виходячи з основного алгоритму шляхом ділення на 8, тобто (10 – 2). Тому подвоєна старша цифра зсувається на один розряд праворуч і додається до числа.

Приклад 2. Перевести число 138_{10} до вісімкового виду.

_138
 2
_158
 3 2
212, тобто $138_{10} = 212_8$.

Таким чином, для переходу від вісімкового представлення до десяткового треба виконувати віднімання; аналогічним чином при переході від десяткового представлення до вісімкового необхідно складати. При виконанні обчислень використовується нова основа.

Розглянемо ще один спосіб швидкого перетворення у двійкове подання великих десяткових чисел. Такий спосіб був придуманий французьким математиком **Лежандром** [12].

Нехай, наприклад, дано число 12345678. Ділимо його на 64, отримуємо залишок 14 і частку 192901. Це число ділимо знову на 64, отримуємо в залишку 5 і частку 3014. Нарешті, 3014, поділене на 64, дає в залишку 6 і частку 47. Але 64 у двійковій системі є 1000000, тому всі залишки і останню частку (<64) записуємо в двійковому вигляді по шість цифр (справа наліво). Початкове число запишеться у двійковій системі наступним чином:

47 6 5 14
101111 000110 000101 001110

Зауважимо, що в остаточне двійкове число включаються всі залишки, а на останньому кроці і залишок і частка. Частка від ділення записується перед залишком. Якщо отримана частка або залишок мають менше 6 розрядів у двійковому поданні (так як $64_{10} = 1000000_2$), то до нього додаються незначущі нулі.

6.8. Перетворення нормалізованих чисел

Дійсне число *X* може бути представлене в двох формах – природній і нормалізованій. У *природній* формі у *X* є ціла і дробова частини, між якими розміщується розподільник (кома або крапка), наприклад, *123,4567*. Проте такий запис незручний для занадто великих або, навпаки, занадто малих чисел. Крім того, використання такої форми (вона називається також «*представлення числа з фіксованою комою*») в комп'ютері викликало б зниження точності обчислень через необхідність приведення у відповідність розрядів оброблюваних чисел і пов'язаних з цим округлень або могло б породити ситуацію, звану *переповнюванням*, коли старший розряд числа не уміщається у відведеній розрядній сітці. З указаних причин дійсні числа в комп'ютері представляються в *нормалізованому* вигляді (інша назва – «*представлення числа з плаваючою крапкою*»), головною перевагою якої є автоматичне масштабування числа на кожному етапі обробки. З одного боку це забезпечує максимально можливу точність обчислень, а з іншої – позбавляє від необхідності вживати заходи по запобіганню переповнення (за винятком досить екзотичних ситуацій з виходом числа за відведену розрядну сітку). По суті, це

універсальна форма запису усіх чисел, окрім визначених як «тип: цілі» (наприклад, Integer, Word або Byte у мові PASCAL).

Спочатку познайомимося з необхідними поняттями стосовно 10-вої системи числення.

Число X_{10} називається нормалізованим, якщо воно представлене у вигляді

$$X_{10} = \pm M_{10} \cdot 10^{\pm k10}$$

У цьому записі M_{10} називається *мантисою* нормалізованого числа; значення мантиси лежать в інтервалі $0{,}1<=M_{10}<1$. **k10** називається *порядком* нормалізованого числа – це ціле позитивне десяткове число. Приклади:

$$-1234_{10} = -0{,}1234 \cdot 10^4; \quad 0{,}03456_{10} = 0{,}3456 \cdot 10^{-1}.$$

Поняття нормалізованого числа слід відрізняти від поняття числа в *нормальній формі*; ця форма досить часто використовується при записі чисел в математиці, фізиці, технічних дисциплінах і відрізняється від нормалізованого представлення тим, що мантиса лежить в інтервалі $1<=M_{10}<10$, наприклад $1{,}38 \cdot 10^{-23}$.

При нормалізації відбувається розчленовування «складових» числа з виділенням знака числа, мантиси, знака порядку і порядку – як буде показано нижче, це створює певні зручності при зберіганні та обробці чисел у комп'ютері.

Аналогічно нормалізації десяткового числа можна в нормалізованій формі представити і число в довільній системі числення p:

$$X_p = \pm M_p \cdot p^{\pm kp} \qquad (6.1)$$

При цьому значення мантиси лежать в інтервалі $p^{-1} \le M_p < 1$ (тобто перша значуща цифра мантиси завжди ненульова), а показник ступеня представляється в системі $p(k_p)$. Наприклад, для $p = 2$:

$$X_2 = -101{,}01_2 = -0{,}10101_2 \cdot 2^{11(2)}$$

Мантиса розташовується в проміжку $0{,}1_2 \le M_2 < 1$, що відповідає десятковому інтервалу $0{,}5_{10} \le M_{10} < 1$.

Подібно завданню про перетворення цілих і дробових чисел, можна поставити завдання про перетворення представлення числа в нормалізованій формі в системі числення p до нормалізованого поданням в системі q. Практичне значення такого перетворення полягає у тому, що, як було сказано, в комп'ютері всі дійсні числа зберігаються і обробляються в нормалізованому двійковому поданні і, отже, при їх введенні здійснюється перетворення $X_{10} \to X_2$, а при виведені – зворотнє перетворення $X_2 \to X_{10}$. Однак перш, ніж обговорювати таке перетворення, необхідно розглянути, як виконується перетворення дійсного числа з природної форми до нормалізованого виду.

При нормалізації розрізняються ситуації Xp>1 і Xp< p - 1. У першому випадку для нормалізації необхідно переміщувати розподільник розрядів вліво по числу доти, доки не зникне ціла частина числа, але перша цифра після роздільника буде ненульова. Кожне переміщення розподільника на 1 розряд

вліво еквівалентне діленню числа на p і, щоб число не мінялося, покажчик повинен збільшуватись на 1 при кожному переміщенні.

Якщо позначити цю операцію $N_{\leftarrow}$ (будемо називати її «*нормалізація вліво*»), то $N_{\leftarrow}[(123{,}45)_{10}] = 0{,}12345_{10}\cdot 10^{3}$; $N_{\leftarrow}[(23{,}4\cdot 10^{5})_{10}] = 0{,}234_{10}\cdot 10^{7}$; $N_{\leftarrow}[(1212{,}2)_{3}] = 0{,}12122_{3}\cdot 3^{11}$. Аналогічно можна ввести операцію «нормалізація праворуч» ($\mathbf{N}_{\rightarrow}$), що забезпечує нормалізацію чисел менших p-1; очевидно, такі числа необхідно множити на p з одночасним зменшенням покажчика на 1 доти, доки перша цифра після розподільника стане ненульовою. наприклад, $N_{\rightarrow}[(0{,}000101\cdot 2^{-101})_{2}] = 0{,}101\cdot 2^{-1000}$; $N_{\rightarrow}[(0{,}000987)_{10}] = 0{,}987_{10}\cdot 10^{-3}$. Загальний алгоритм нормалізації можна відобразити у вигляді блок-схеми на рис. 6.1.

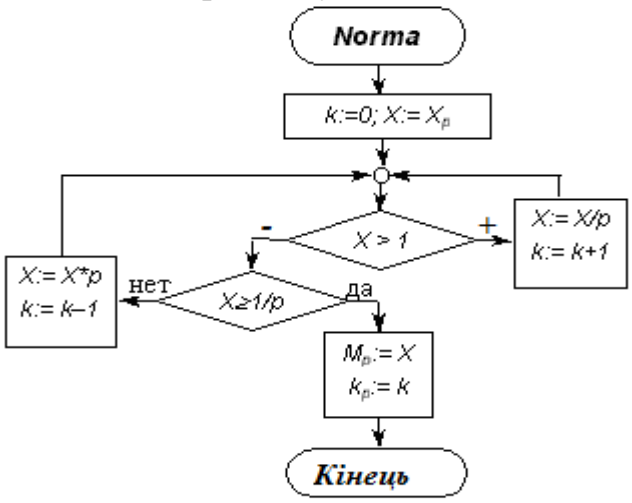


Рис. 6.1. Алгоритм нормализації

При практичній реалізації цього алгоритму не слід забувати, що зміна значення k_p на 1 повинна робитися за правилами арифметики системи числення p. Надалі при необхідності проведення нормалізації в ході яких-небудь перетворень просто посилатимемося на наведений алгоритм як готовий модуль.

Повернемося до задачі перекладу нормалізованого числа з однієї системи числення в іншу. Нехай, існує число $Xp = \pm Mp \cdot p^{\pm Kp}$, для якого необхідно знайти відповідне йому $Xq = \pm Mq \cdot q^{\pm Kq}$. Представляється досить очевидним, що перетворення не торкнеться знаків мантиси і покажчика ступеню. Таким чином, для здійснення перетворення необхідно встановити відповідність (Mp, k_p) і (Mq, k_q). Воно виходить досить просто, виходячи з того, що $Xp = Xq$, звідки слідує:

$$M_q = M_p \frac{p^{k_p}}{q^{k_q}} \qquad \textbf{(6.3)}$$

З (6.3) витікає, що для здійснення перетворення можна Mp помножити на p^{Kp}, тобто перейти до природної форми числа в системі p, перевести його в систему q, а потім нормалізувати (q^{Kq}). Проте в такому варіанті дій втрачається точність числа і можливе переповнення на проміжних етапах перетворення. Щоб уникнути цього необхідно контролювати множення (чи ділення) на p і нормалізацію за основою q. При цьому, оскільки усі операції виконуються за

правилами арифметики в системі p, то будуть отримані не $(M_q,\ k_q)$ в остаточному варіанті, а їх представлення в системі p – позначимо їх $(M_q)_p$ і $(k_q)_p$, які потім треба буде перевести в систему q.

Розрізняються також ситуації $k_p \geq 0$ і $k_p < 0$ – в першому випадку необхідно множити початкове і проміжні значення мантиси на p , і для нормалізації ділити на q, в другому – навпаки. Кожного разу при множенні або діленні на p покажчик k_p мінятиме своє значення на 1; продовжувати дії слід доти, доки не виконається умова $k_p = 0$. Алгоритм дій для ситуації $k_p \geq 0$ представлений на рис. 6.2.

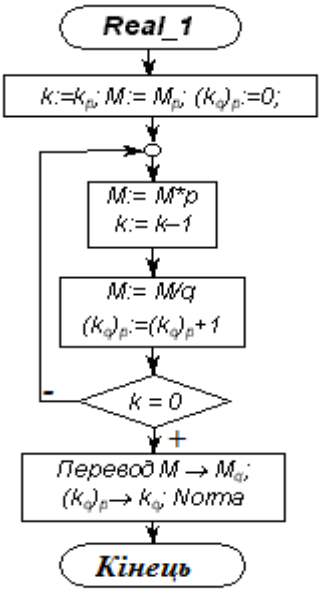


Рис. 6.2. Алгоритм перетворення $X_{10} \rightarrow X_2$

Приклад 1. Виконати перетворення $X_{10}=16{,}5_{10} \rightarrow X_2$.

Перетворення можна здійснити окремо для цілої і дробової частини, а потім їх об'єднати – для нас цей результат послужить еталоном для перевірки нового алгоритму. Легко бачити, що $16_{10} = 10000_2$, а $0{,}5_{10} = 0{,}1_2$; отже,

$$16{,}5_{10} = 10000{,}1_2 = (0{,}100001 \cdot 2^{101})_2.$$

Алгоритм *Real_1* починає функціонувати після нормалізації вихідного числа; для цієї мети можна скористатися алгоритмом *Norma*; в результаті початковими значеннями будуть $M_{10} = 0{,}165;\ k_{10} = 2$. Результати операцій будемо заносити в таблицю:

Крок	***Дія***	$\boldsymbol{M=(M_2)_{10}}$	$\boldsymbol{k_{10}}$	$\boldsymbol{(k_2)_{10}}$
0		*0,165*	*2*	*0*
1	$M:=M \cdot 10;\ k_{10}:= k_{10}-1$	*1,65*	*1*	*0*
2	$M:=M/2;\ k_2:= k_2+1$	*0,825*	*1*	*1*
3	$M:=M \cdot 10;\ k_{10}:= k_{10}-1$	*8,25*	*0*	*1*
4	*Т.к.* $k_{10}= 0,\ M \rightarrow M_2$	$1000{,}01_2$		*1*
5	*Нормализация по q=2*	$0{,}100001_2$		*5*
6	$(k_2)_{10} \rightarrow k_2$	$0{,}100001_2$		*101*

остаточно маємо: $X_2 = (0{,}100001 \cdot 2^{101})_2$.

Подібним же буде алгоритм перетворення $X_{10} \rightarrow X_2$ і при $k_p < 0$.

Послідовність дій при зворотному перетворенні $X_2 \rightarrow X_{10}$ протилежна тільки що розглянутим діям; для $k_p \geq 0$ вона представлена у вигляді блок-схеми на рис. 6.3. Нормалізація в кінці (після $k = 0$) проводиться за необхідністю.

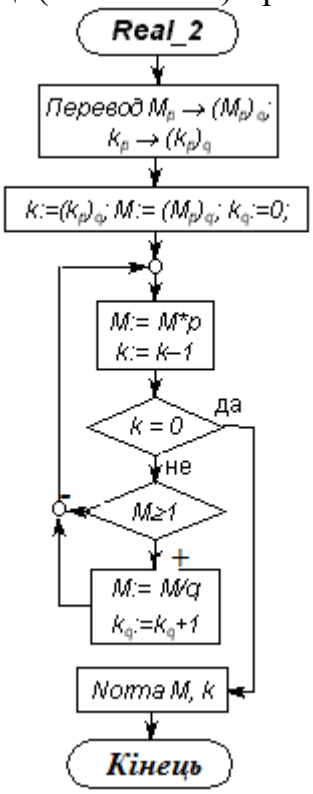


Рис. 6.3. Алгоритм перетворення $X_2 \rightarrow X_{10}$

Приклад 2. Виконати перетворення: $X_2 = (0{,}11 \cdot 2^{110})_2 \rightarrow X_{10}$

Для контролю результату: $0{,}11_2 = 0{,}75_{10}$; $(2^{110})_2 = (2^6)_{10} = 64$; отже, $(0{,}11 \cdot 2^{110})_2 = 0{,}75 \cdot 64 = 48_{10}$. Для перетворення скористаємося побудованим алгоритмом Real_2. (рис. 6.3). Проміжні результати знову будемо заносити в таблицю:

Крок	***Дія***	***$M=(M_2)_{10}$***	***$k=(k_2)_{10}$***	***k_{10}***
0	*Переведення*	*0,75*	*6*	*0*
1	*M:=M·2; k = k-1*	*1,5*	*5*	*0*
2	*M:=M/10; k_{10}= k_{10}+1*	*0,15*	*5*	*1*
3	*M:=M·2; k = k-1*	*0,3*	*4*	*1*
4	*M:=M·2; k = k-1*	*0,6*	*3*	*1*
5	*M:=M·2; k = k-1*	*1,2*	*2*	*1*
6	*M:=M/10; k_{10}= k_{10}+1*	*0,12*	*2*	*2*
7	*M:=M·2; k = k-1*	*0,24*	*1*	*2*
8	*M:=M·2; k = k-1*	*0,48*	*0*	*2*
9	*Т.к. k = 0, M = M_{10}*	*0,48*	*0*	*2*

Таким чином, отримуємо: $(0{,}11\cdot 2^{110})_2=0{,}48\cdot 10^2=48$.

Як і в попередньому перетворенні, алгоритм у випадку $k_p<0$ буде відрізнятися тільки тим, що число в процесі перекладу необхідно ділити на q і множити на p.

6.9. Кодовані позиційні системи числення

Це такі системи, в яких цифри однієї системи числення кодуються за допомогою цифр іншої системи, а число A в загальному вигляді записується таким чином:

$$(a_{n,k}p^n + a_{n-1,k}p^{n-1} + .. + a_{0,k}p^0)P^k + (a_{n,k-1}p^n + .. + a_{0,k-1}p^0)P^{k-1} + .. + (a_{n,0}p^n + .. + a_{0,0}p^0)P^0$$

де p – основа системи числення, символами якої кодуються числа; P – основа вхідної системи числення.

При побудові кодованих позиційних систем замість ваги розрядів можна вибрати як члени геометричної прогресії, так і довільні числа. У першому випадку системи називаються кодованими системами числення з ***природною вагою розрядів***, а в другому – з ***штучною вагою розрядів***.

Прикладом системи числення з природною вагою розрядів може бути двійково-десяткова система з вагою 8-4-2-1. У ній кожна цифра десяткового числа кодується двійковою тетрадою. Наприклад, десяткове число 1593 в цій двійково-десятковій системі матиме вигляд: 0001 0101 1001 0011.

Прикладом системи числення з штучною вагою розрядів може бути двійково-десяткова система з вагою 2-4-2-1. Десяткове число 1593 в цій системі відповідно матиме вигляд: 0001 1011 1111 0011.

Десяткова Цифра	Код 8-4-2-1	Код 2-4-2-1	Код 4-2-2-1	Рефлексний код
0	0000	0000	0000	0000
1	0001	0001	0001	0001
2	0010	0010	0010	0011
3	0011	0011	0011	0010
4	0100	0100	0110	0110
5	0101	1011	1001	0111
6	0110	1100	1100	0101
7	0111	1101	1101	0100
8	1000	1110	1110	1100
9	1001	1111	1111	1000

Неважко помітити, що даний код є самодоповненим, тобто в цій системі десяткове число 9 кодується як 1111, що дозволяє найбільш раціонально будувати десяткові лічильники та арифметичні схеми. Код називається

самодоповненим, якщо двійкові коди будь-яких двох десяткових цифр, які доповнюють один одного до 9 (тобто, якщо їх десяткова сума дорівнює 9: $a^i_{10}+a^k_{10}=9$), доповнюють один одного до $15_{10} =1111_2$ (тобто їх сума $a^i_2+a^k_2=1111$).

У таблиці наведені десяткові цифри, представлені в двійково-десяткових кодах (ДДК). Двійково-десяткові коди володіють певною надмірністю, оскільки для кодування десяткових цифр використовується лише 10 комбінацій з 16. Найбільш поширеним є ДДК "8-4-2-1", перевагою якого можна вважати простоту і зручність перетворення чисел з десяткової системи числення в двійково-десяткову і зворотнє перетворення, а також зведення операцій десяткової арифметики до виконання операцій за правилами двійкової арифметики.

Властивість самодоповнення мають ДДК "2-4-2-1" і "4-2-2-1", в яких сума вагів дорівнює 9. Ці коди мають важливу властивість: для того щоб для – будь-якої цифри отримати її доповнення до 9, досить взяти її зворотний код. Це дозволяє спростити десяткову арифметику так само, як і застосування додаткового коду спрощує двійкову арифметику. Замість віднімання будь-якого числа досить зробити додавання з числом, яке є доповненням до 9 початкового числа плюс 1.

Наприклад, 85-16 = 69; доповнення до 99 числа 16 = 83, потім додали1 (83+1=84), 85+84 = 1| 69. У двійково-десятковому кодуванні “4-2-2-1” маємо: 85+84 = 1| 69 = 11101001 + 11100110 = 1| 11001111 = 69. Одиниця (1|), що виходить за розрядну сітку, не враховується.

Рефлексивний код ДДК (**код Грея**) має властивість, при якому двом сусіднім десятковим цифрам в його представленні відповідають кодові комбінації, які відрізняються тільки в одному двійковому розряді. Це властивість ефективно використовується при побудові вимірювальних перетворювачів (датчиків), особливо в механічних шифраторах кута повороту валу. При переході з будь-якого його стану до наступного змінюється лише один розряд.

Наведемо приклад застосування коду Грея для чотирьохрозрядного двійкового коду. З таблиці видно, шо за виключенням старшого розряду код є симетричним, або рефлексивним відносно середини таблиці. Така особливість дозволяє використовувати його при кодуванні станів автоматів. Для обрання потрібної кількості станів достатньо симетрично вилучити парну кількість рядків відносно середини даної таблиці.

Десяткове значення	Двійкове значення	Код Грея
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111

6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Так, якщо вилучити рядки з номерами 7,8 або 6, 7, 8, 9, то у такому випадку сусідні коди будуть відрізнятись тільки в одному розряді. Єдине, що треба враховувати, що кількість кодів при цьому завжди має бути парною.

6.10. Поняття економічності системи числення

Число в системі числення *p* з *k* розрядами, очевидно, матиме найбільше значення у тому випадку, якщо усі цифри числа виявляться максимальними, тобто рівними *p - 1*. Тоді

$$(Z_p)^{max} = \underbrace{<p-1> \ldots <p-1>}_{k \text{ цифр}} = p^k - 1$$

Кількість розрядів числа при переході від однієї системи числення до іншої в загальному випадку змінюється. Очевидно, якщо $p = q^{\sigma}$ (σ – не обов'язково ціле), то $(Z_p)^{max} = p^k - 1 = q^{\sigma k} - 1$. Тобто, кількість розрядів числа в системах числення *p* і *q* будуть відрізнятися в σ раз. очевидно співвідношення:

$$\sigma = \frac{\log p}{\log q}$$

При цьому основа логарифму ніякого значення не має, оскільки σ визначається відношенням логарифмів. Порівняємо кількість цифр у числі 99_{10} і його представлення у двійковій системі числення: $99_{10} = 1100011_2$; тобто двійковий запис вимагає 7 цифр замість 2 у десятковому числі. σ **= ln(10)/ln(2) = 3,322**; отже, кількість цифр у десятковому поданні потрібно помножити на 3,322 і округлити в більшу сторону: $2 \cdot 3{,}322 = 6{,}644 \approx 7$.

Введемо поняття економічності представлення числа в системі числення.

Під ***економічністю*** системи числення будемо розуміти ту кількість чисел, яку можна записати в цій системі за допомогою певної кількості цифр.

В даному випадку йдеться не про кількість розрядів, а про загальну кількість поєднань цифр, які інтерпретуються як різні числа. Пояснимо це на прикладі: нехай в нашому розпорядженні є 12 цифр. Ми можемо розбити їх на 6 груп по 2 цифри ("0" і "1") і отримати шестирозрядне двійкове число; загальна кількість таких чисел, як вже неодноразово обговорювалося, дорівнює 2^6. Можна розбити задану кількість цифр на 4 групи по три цифри і скористатися трійковою системою числення – в цьому випадку загальна кількість різних їх

поєднань складе 3^4. Аналогічно можна зробити інше розбиття; при цьому число груп визначить розрядність числа, а кількість цифр в групі – основа системи числення. Результати різного розбиття можна проілюструвати таблицею:

Основа системи числення (*p*)	***2***	***3***	***4***	***6***	***12***
Розрядність числа (*k*)	***6***	***4***	***3***	***2***	***1***
Загальна кількість різних чисел (*N*)	$2^6 = 64$	$3^4 = 81$	$4^3 = 64$	$6^2 = 36$	$12^1 = 12$

З наведених оцінок видно, що найбільш економічною виявляється трійкова система числення, причому, результат буде таким самим, якщо досліджувати випадки з іншою початковою кількістю цифр.

Точне розташування максимуму економічності може бути встановлене шляхом наступних міркувань. Нехай є n знаків для запису чисел, а основа системи числення p. Тоді кількість розрядів числа $k = n/p$, а загальна кількість чисел (N), які можуть бути складені, рівна:

$$N = p^{\frac{n}{p}}$$

Якщо вважати $N(p)$ безперервною функцією, то можна знайти таке значення p_m, при якому N приймає максимальне значення. Функція має вигляд, представлений на малюнку 6.4.

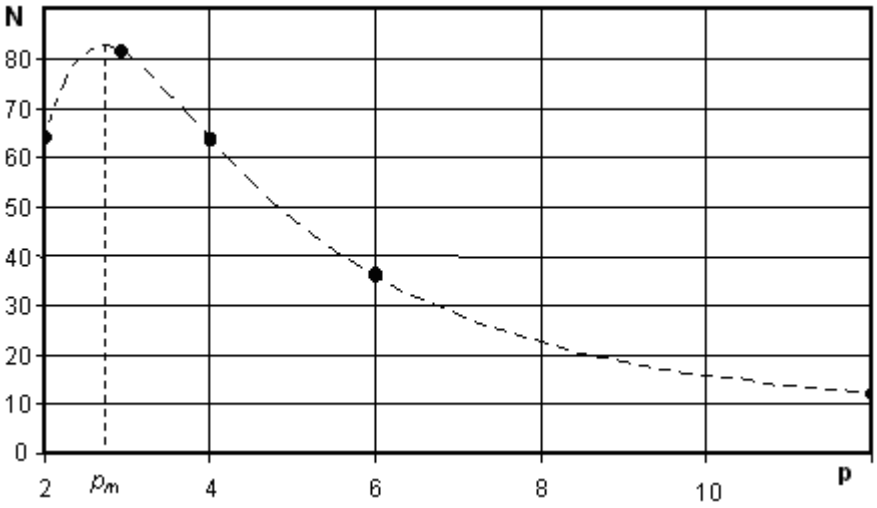


Рис. 6.4. Залежність кількості чисел від основи системи числення при використанні 12 можливих цифр для запису чисел

Для знаходження положення максимуму потрібно знайти похідну функції $N(p)$, прирівняти її до нуля і розв‘язати отримане рівняння щодо p.

$$\frac{dN}{dp} = -\frac{n}{p^2} p^{\frac{n}{p}} \ln p + \frac{n}{p} p^{\frac{n}{p}-1} = n p^{\frac{n}{p}-2} (1 - \ln p)$$

Прирівнюючи отриманий вираз до нуля, отримуємо $ln(p) = 1$, або $p_m = e$, де e = 2,71828… – основа натурального логарифму. Найближче до ***e*** ціле число, вочевидь, 3 – з цієї причини трійкова система числення виявляється самою економічною для представлення чисел.

Трійкова система числення – позиційна система числення з основою 3 існує в двох варіантах: несиметрична (цифри 0,1,2) і симетрична (цифри −1,0,1).

Симетрична система дозволяє зображувати від’ємні числа, не використовуючи окремий знак мінуса.

Десяткова система	−3	−2	−1	0	1	2	3	4	5	6	7
Трійкова несиметрична	−10	−2	−1	0	1	2	10	11	12	20	21
Трійкова симетрична	1̶0	1̶1	1̶	0	1	1̶1̶	10	11	1̶1̶1̶	1̶10	1̶11

Прикладом подання чисел у трійковій системі числення може служити запис у цій системі цілих позитивних чисел:

десяткове число……..……0,1,2,3,4,5,1**0**
трійкове число……0,1,1̶1̶,10,11,1̶1̶1̶,101

Наявність додатної і від'ємної цифр дозволяє безпосередньо представляти як додатні, так і від'ємні числа. При цьому немає необхідності в спеціальному розряді знаку і не потрібно вводити додатковий (чи зворотний) код для виконання арифметичних операцій з від’єними числами. Усі дії над числами, які представлені в трійковій системі числення з цифрами 0, 1, - 1, виконуються природно з урахуванням знаків чисел. Знак числа визначається знаком старшої значущої цифри числа: якщо вона додатня, то і число додатне, якщо віж’ємне, то число від'ємне. Для зміни знаку числа потрібно змінити знаки усіх його цифр (тобто інвертувати його код). Наприклад, 10̶1̶ =8, 1̶01 = -8.

Іншим корисним наслідком симетричного розташування значень цифр є відсутність проблеми округлення чисел: абсолютна величина частини числа, представлена відкиданими молодшими цифрами, ніколи не перевершує половини абсолютної величини частини числа, відповідній молодшій значущій цифрі молодшого з розрядів, що зберігаються. Отже, в результаті відкидання молодших цифр числа виходить найкраще при цій кількості цифр, що залишилися, наближення цього числа, і округлення не потрібно.

Вся річ у тому, що класична двійкова система числення має ряд принципових недоліків, головними з яких є: проблема представлення від’ємних чисел і "нульова" надмірність класичного двійкового способу представлення чисел.

Щоб здолати вказані недоліки двійкової системи, вже на етапі зародження комп'ютерної ери були виконані ряд проектів і зроблено декілька цікавих математичних відкриттів, пов'язаних з системами числення. Мабуть, найцікавішим проектом в цьому відношенні є трійковий комп'ютер "Сетунь", який був розроблений в 60-х роках в Московському університеті під керівництвом Н.П. Бруснецова (У 1970 р. Н. П. Бруснецов побудував другу електронну трійкову ЕОМ (комп'ютер) «Сетунь-70»).

Використання у ньому так званої трійкової симетричної системи числення для представлення чисел уперше в історії комп'ютерів поставило знак

рівності між від’ємними і додатними числами, дозволивши відмовитися від різних "хитрощів" (зворотний та додатковий код), що використовується для представлення від’ємних чисел. Ця обставина, а також використання "трійкової логіки" при створенні програм привело до створення архітектури, яка і була втілена в моделі "Сетунь". Саме "Сетунь" є найбільш яскравим історичним прикладом, що підтверджує вплив системи числення на архітектуру комп'ютера!

Проте широкого поширення вона не отримала, оскільки все ж елементна база комп'ютерів, як того часу, так і сучасних, залишається двійковою, а бітове представлення трійкової системи навіть довше, ніж двійковою. Тому перевага все ж віддається двійковій системі, оскільки за економічністю вона виявляється другою за трійковою, а технічно вона реалізується набагато простіше за інших. Таким чином, простота технічних рішень виявляється не єдиним аргументом на користь застосування двійкової системи в комп'ютерах.

Контрольні запитання для самоперевірки

1. Чим викликано обрання двійкової системи числення у комп’ютерній техніці?
2. Що є основою числа?
3. Яка система числення є позиційною?
4. Як перетворити число з однієї позиційної системи числення в іншу?
5. Як перетворити число з двійкової системи числення у вісімкову, шістнадцяткову та навпвки?
6. Як перетворити дробову частину з десяткової системи числення у двійкову?
7. Чому неможливо точне перетворення дробової з десяткової системи числення у двійкову?

7. Арифметичні операції в позиційних системах числення

Розглянемо основні арифметичні операції: додавання, віднімання, множення та ділення. Правила виконання цих операцій у десятковій системі добре відомі – це додавання, віднімання, множення стовпчиком та ділення кутом. Ці правила застосовні й до всіх інших позиційних систем числення. Тільки таблицями додавання та множення треба користуватися виключно для кожної системи.

7.1. Додавання

Таблиці додавання легко скласти, використовуючи Правило Рахунку.

Додавання у двійковій системі

+	0	1
0	0	1
1	1	10

Додавання у вісімковій системі

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	10
2	2	3	4	5	6	7	10	11
3	3	4	5	6	7	10	11	12
4	4	5	6	7	10	11	12	13
5	5	6	7	10	11	12	13	14
6	6	7	10	11	12	13	14	15
7	7	10	11	12	13	14	15	16

Додавання у шістнадцятковій системі

+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

При додаванні цифри підсумовуються по розрядах, і якщо при цьому виникає надлишок, то він переноситься ліворуч.

Приклад 1. Розглянемо додавання та віднімання чисел у різних системах числення. Переглянемо всі можливі варіанти знаків операндів.

1. Додатне + додатне

Варіанти результатів: 1) знаковий розряд не міняється перенос відсутній (результат додатній, переповнення немає) (результат додатній , є переповнення)

приклад: 47+53=100

000101111
+ 000110101
001100100 =100

Переповнення немає – знак не змінюється.

62+67=129

000111110
+ 001000011
010000001 = 129

Є переповнення – у знакових розрядах 01

2. Додатне + від'ємне (від'ємне + додатне)

Ч1 + (2^n – Ч2) (2^n – Ч1) + Ч2

Наприклад 53 – 47 = 6 -53 + 47 = - 6

000110101	111001011
111010001	000101111
000000110	111111010
c=1	c=0

В цих двох останніх випадках переповнення немає, оскільки операнди з різними знаками.

3. Від'ємне + від'ємне

1) а) є перенос, але знаковий розряд не змінюється, тобто результат є від'ємним.

(-47)+(-53)=-100 (2^n – Ч1) + (2^n – Ч2)

111010001 (-47)
111001011 (-53)
111011100 (-100) у знакових розрядах 11
c=1

б) є перенос, але знак змінюється, тобто маємо переповнення

(-62)+(-67)=-129

111000010 (-62)
110111101 (-67)
101111111 (-129) у знакових розрядах 10
c=0

7.2. Віднімання

Для від'ємних чисел було введено додатковий код числа, який являє собою значення 2^n – D, де n – розрядність даних у комп'ютері, а D – число. Зараз найбільш розповсюдженими є 32-розрядні комп'ютери. У комп'ютері віднімання замінюється на додавання з додатковим кодом, тобто A – B = A + (2^n – B) = 2^n + A - B. Якщо A > B, то виникає перенос у старшому розряді (2^n) і результат буде додатній (тобто матимемо прямий код числа), якщо A < B, то переносу не буде і результат буде від'ємний у додатковому коді. В наведеному прикладі для знаку відведені два розряди, де старший розряд зберігає власно знак, а молодший вказує на переповнення, але докладніше про це далі.

У комп'ютерній техніці застосовуються три форми запису (кодування) цілих чисел зі знаком: ***прямий код, обернений код, додатковий код*** (детально буде описано в розділі 8. Подання чисел у комп'ютері).

7.3. Множення

Розглянемо приклад множення двох двійкових чисел. Множення будемо виконувати так само, як и при множенні десяткових чисел. Тобто множення розпочнемо з молодшої цифри множника і будемо накопичувати часткову суму.

```
          10100111
          11001010
         ---------
         101001110
       10100111
     10100111
    10100111
   ---------------
   1000001111000110
```

При виконанні множення багатозначних чисел у різних позиційних системах числення, можна використовувати звичайний алгоритм множення чисел у стовпчик, але при цьому результати множення та додавання однозначних чисел необхідно запозичувати з відповідних таблиць множення та додавання.

Множення у двійковій системі

*	0	1
0	0	0
1	0	1

Множення у вісімковій системі

*	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	10	12	14	16
3	0	3	6	11	14	17	22	25
4	0	4	10	14	20	24	30	34
5	0	5	12	17	24	31	36	43
6	0	6	14	22	30	36	44	52
7	0	7	16	25	34	43	52	61

Зважаючи на надзвичайну простоту таблиці множення у двійковій системі, множення зводиться лише до зсувів множника та додавань.

Приклад 1. Перемножимо числа 5 та 6.

Десяткова: $5_{10} * 6_{10}$ **Двійкова:** $101_2 * 110_2$ **Вісімкова:** $5_8 * 6_8$

```
  5
×  6
 --
 30
```

```
  101
×  110
 ----
  101
 101
 -----
 11110
```

```
  5
×  6
 --
 36
```

Відповідь: $5*6 = 30_{10} = 11110_2 = 36_8$.

Перевірка. Перетворимо отримані результати до десяткового вигляду:
$11110_2 = 2^4 + 2^3 + 2^2 + 2^1 = 30$;
$36_8 = 3 \cdot 8^1 + 6 \cdot 8^0 = 30$.

Приклад 2. Перемножимо числа 115 та **51.**

Десяткова: 11$5_{10}$ * 51_{10} **Двійкова:** 1110011_2 * 1100011_2 **Вісімкова:** 163_8 * 63_8

```
  115          1110011          163
×  51        ×  110011        ×  63
 ----         -------          ----
  115          1110011          531
 575          1110011          1262
-----        1110011          -----
 5865       -------------      13351
            1011011101001
```

Відповідь: $115*51 = 5865_{10} = 1011011101001_2 = 13351_8$.

Перевірка. Перетворимо отримані добутки до десяткового вигляду:
$1011011101001_2 = 2^{12} + 2^{10} + 2^9 + 2^7 + 2^6 + 2^5 + 2^3 + 2^0 = 5865$;
$13351_8 = 1*8^4 + 3*8^3 + 3*8^2 + 5*8^1 + 1*8^0 = 5865$.

В багатьох комп'ютерах множення проводиться як послідовність додавань та зсувів. Для цього в АЛУ є регістр, так званий накопичуючий суматор, який до початку виконання операції містить нуль. У процесі виконання операції у цьому регістрі по черзі розміщуються множене та результати проміжних додавань, а після закінчення операції – остаточний результат.

Інший регістр АЛУ, що бере участь у виконанні цієї операції, спочатку містить множник. Потім по мірі виконання додавань число, що міститься у ньому, зсувається праворуч, поки не досягне нульового значення.

Для ілюстрації помножимо 110011_2 на 101101_2.

Накопичуючий суматор		**Множник**	
000000000000	**|**	**101101**	
+ 110011	**|**		
000000110011	**|**	**001011**	**Зсув на два розряди праворуч**
+ 110011	**|**		
000011111111	**|**	**000101**	**Зсув на один розряд праворуч**
+ 110011	**|**		
001010010111	**|**	**000001**	**Зсув на два розряди праворуч**
+ 110011	**|**		
100011110111	**|**	**000000**	

З цього прикладу видно, що для реалізації операції множення можуть застосовуватися схеми множення із молодших розрядів або із старших розрядів другого множника, а також із зсувом часткових сум або першого множника (див. приклади). Комбінація цих параметрів множення дають 4 варіанти алгоритмів множення.

Нижче наведені приклади для різних схем множення. Для спрощення у прикладах наведені восьмірозрядні операнди.

1. Множення з молодших розрядів другого множника та зсувом ліворуч першого множника. Перший множник подано у другому рядку на кожному кроці. Починаючи з другого кроку цей множник зсувається ліворуч на один двійковий розряд та додається до загальної суми, якщо чергова цифра другого множника, що висувається – 1, інакше нічого не додається. Результат у вигляді часткової суми подається третім рядком на кожному кроці.

На цій схемі перший рядок відповідає другому множнику, який зсувається праворуч на один розряд. Другий рядок відповідає першому множнику, який зсувається ліворуч. У третьому рядку наводиться часткова сума. Якщо черговий висунутий розряд другого множника – 0, то часткова сума не змінюється, якщо – 1, то до часткової суми додається зсунутий перший множник.

	11001010
00000000	10100111
00000000	00000000

Початковий стан

	01100101	0
00000000	10100111	
00000000	00000000	

Перший крок

	00110010	1
00000001	01001110	
00000001	01001110	

Другий крок

	00011001	0
00000010	10011100	
00000001	01001110	

Третій крок

	00001100	1
00000101	00111000	
00000110	10000110	

Четвертий крок

	00000110	0
00001010	01110000	
00000110	10000110	

П’ятий крок

	00000011	0
00010100	11100000	
00000110	10000110	

Шостий крок

	00000001	1
00101001	11000000	
00110000	01000110	

Сьомий крок

	00000000	1
01010011	10000000	
10000011	11000110	

Восьмий крок

2. Множення зі старших розрядів другого множника та зсувом першого множника на кожному кроці на один розряд праворуч. За цією схемою множення другий множник, що подається на схемі першим рядком, зсувається ліворуч на один двійковий розряд на кожному кроці. Перший множник при цьому зсувається праворуч на один двійковий розряд та додається до загальної суми, де накопичується результат, якщо чергова цифра першого множника, що аналізується при зсуві – 1, то до часткової суми додається зсунутий перший множник, інакше результат не змінюється. Зауважимо, що за даною схемою зсуваються обидва множника: другий – ліворуч, а перший праворуч і початково зсунутий на n-1=7 розрядів ліворуч, оскільки кількість зсувів на один менше за n.

	11001010
10100111	00000000
00000000	00000000

Початковий стан

	1	10010100
01010011		10000000
01010011		10000000

Перший крок

	1	00101000
00101001		11000000
01111101		01000000

Другий крок

	0	01010000
00010100		11100000
01111101		01000000

Третій крок

	0	10100000
00001010		01110000
01111101		01000000

Четвертий крок

	1	01000000
00000101		00111000
10000010		01111000

П’ятий крок

	0	10000000
00000010		10011100
10000010		01111000

Шостий крок

	1	00000000
00000001		01001110
10000011		11000110

Сьомий крок

	0	00000000
00000000		10100111
10000011		11000110

Восьмий крок

3. Множення з молодших розрядів другого множника та зсувом часткових сум праворуч. Перший множник подається на кожному кроці другим рядком попередньо зсунутим ліворуч на. 7 розрядів, оскільки цей алгоритм віддзеркалює перший алгоритм, але із зворотним формуванням часткових сум і перший операнд на першому кроці не зсувається. Другий множник подано у першому рядку. На кожному кроці при його зсуві праворуч показується чергова двійкова цифра. Часткова сума у третьому рядку перед виконанням попередньо зсувається праворуч на один розряд. На першому кроці праворуч зсувається початкове нульове значення цієї суми.

	11001010	
01010011	10000000	
00000000	00000000	

Початковий стан

	01100101	0
01010011	10000000	
00000000	00000000	

Перший крок

	00110010	1
01010011	10000000	
01010011	10000000	

Другий крок

	00011001	0
01010011	10000000	
00101001	11000000	

Третій крок

	00001100	1
01010011	10000000	
01101000	01100000	

Четвертий крок

	00000110	0
01010011	10000000	
00110100	00110000	

П’ятий крок

	00000011	0
01010011	10000000	
00011010	00011000	

Шостий крок

	00000001	1
01010011	10000000	
01100000	10001100	

Сьомий крок

	00000000	1
01010011	10000000	
10000011	11000110	

Восьмий крок

4. Множення зі старших розрядів другого множника та зсувом часткової суми ліворуч. За цією схемою на кожному кроці здійснюється зсув другого множника на один розряд ліворуч та додавання до часткової суми, якщо чергова висунута двійкова цифра – 1, інакше нічого не додається. Часткова сума, що подана третім рядком зсувається на кожному кроці на один розряд ліворуч. Перший раз зсувається нульове значення.

	11001010
00000000	10100111
00000000	00000000

Початковий стан

1	10010100
00000000	10100111
00000000	10100111

Перший крок

1	00101000
00000000	10100111
00000001	11110101

Другий крок

0	01010000
00000000	10100111
00000011	11101010

Третій крок

0	10100000
00000000	10100111
00000111	11010100

Четвертий крок

1	01000000
00000000	10100111
00010000	01001111

П’ятий крок

0	10000000
00000000	10100111
00100000	10011110

Шостий крок

1	00000000
00000000	10100111
01000001	11100011

Сьомий крок

0	00000000
00000000	10100111
10000011	11000110

Восьмий крок

Як бачимо, усі перелічені алгоритми дають однаковий результат.

5. Прискорені методи множення. Алгоритм Бута. Алгоритм множення Бута, який дозволяє перемножити на дві двійкові цифри був розроблений Ендрю Дональдом Бутом в 1951 р. при проведенні досліджень кристалографії в

коледжі Birkbeck в Bloomsbury (Лондон). Бут користувався настільними обчислювачами, які виконували операцію зсуву швидше, ніж операцію додавання, і створив алгоритм для збільшення швидкості їх роботи.

В основі алгоритму Бута лежить наступне співвідношення, яке характерне для послідовностей двійкових цифр:

$$2^m + 2^{m-1} + \ldots + 2^k = 2^{m+1} - 2^k,$$

де *m* і *k* – номери крайніх розрядів в групі з послідовних одиниць.

Це означає, що за наявності у множнику груп з декількох одиниць (комбінацій виду 011,110), послідовне додавання до суми часткових добутків (СЧД) множеного з наростаючою вагою (від 2^k до 2^m) можна замінити відніманням з СЧД множеного з вагою 2^k і збільшенням до СЧД множеного з вагою 2^{m+1}. Наприклад, $011110 = 2^5 - 2^1$.

Як бачимо, алгоритм припускає три операції: зсув, додавання і віднімання. Окрім скорочення числа додавань (віднімань) у нього є ще одна перевага – він в рівній мірі застосований до чисел без знаку і зі знаком.

Алгоритм Бута зводиться до того, що перекодує множник з системи (0, 1) в надмірну систему (-1,0,1), через що його часто називають перекодуванням Бута (Booth recoding). У записі множника в новій системі 1 означає додавання множеного до суми часткових додатків, -1 – віднімання множеного, і 0 не передбачає ніяких дій. В усіх випадках після чергової ітерації робиться зсув множеного ліворуч або суми часткових додатків праворуч.

Реалізація алгоритму припускає послідовний у напрямі справа наліво аналіз пар розрядів множника – поточного b_i, і попередього b_{i-1} - $(b_i b_{i-1})$.

Для молодшого розряду множника (i-0) вважається, що попередній розряд дорівнює 0, тобто має місце пара $b_0 0$. На кожному кроці *i (i = 0,1,., n-1)* аналізується поточна комбінація $b_i (b_i b_{i-1})$.

Комбінація 10 означає початок ланцюжка послідовних одиниць, і в цьому випадку робиться віднімання множеного з СЧД.

Комбінація 01 відповідає завершенню ланцюжка одиниць, і тут множене додається до СЧД.

Комбінація 00 свідчить про відсутність ланцюжка одиниць, а 11 – про знаходження всередині такого ланцюжка. В обох випадках ніякі арифметичні операції не робляться. По закінченню описаних дій здійснюється зсув множеного ліворуч або суми часткових додатків праворуч, і цикл повторюється для наступної пари розрядів множника.

Описану процедуру розглянемо на прикладах (використовується варіант із зсувом множеного ліворуч). У наведених прикладах операція віднімання виконується шляхом додавання з множником, який береться з протилежним знаком і подається у додатковому коді. Нагадаємо, що для подовження коду до потрібного числа розрядів в додаткові позиції ліворуч заноситься значення знакового розряду.

Розглянемо приклад множення двох чисел 6*3 = 18 (у двійковому вигляді 0110*0011 = 00010010). Після перекодування Бута множник (0,0,1,1) набуває вигляду(0,1,0,-1). На початку сума часткових додатків приймається рівною

нулю – 00000000. Вважається, що молодшому розряду множника передував 0. Подальший процес пояснює рис. 7.1.

A B	0110 00110	0110 00110	0110 00110	0110 00110
	00000000 +11111010	11111010	11111010 + 011000	00010010
P	11111010	11111010	00010010	00010010
	Початок низки одиниць – (-A)	Всередини низки одиниць	Кінець низки одиниць – (+A)	Поза низкою одиниць

Рис. 7.1. Приклад множення (6*3) відповідно до алгоритму Бута

Наведемо приклад множення -4*3 = -12 (у двійковому вигляді 1100*0011 = 11110100, - 4(0100) представлено в додатковому коді як 1100). Процес обчислення ілюструє рис. 7.2.

A B	1100 00110	1100 00110	1100 00110	1100 00110
	00000000 +00000100	00000100	00000100 +11110000	11110100
P	00000100	00000100	11110100	11110100
	Початок низки одиниць – (-A)	Всередини низки одиниць	Кінець низки одиниць – (+A)	Поза низкою одиниць

Рис. 7.2. Приклад множення (-4*3) відповідно до алгоритму Бута

При найбільш сприятливому поєднанні цифр множника кількість додавань дорівнює n/2, де n – кількість розрядів множника.

Наведемо ще один приклад множення за покращеним алгоритмом Бута, в якому використовуються ті ж самі множники, що й у попередніх прикладах за класичними схемами множення. При цьому другий множник зсувається одразу на два розряди праворуч.

	10100111
00000000	11001010

Початковий стан

	00000010	10
00001100	10100000	
00011001	01000000	
000111101	11000110	

Перший крок

	00101001	11
00000000	11001010	
11111111	00110110	
11111111	00110110	

Другий крок

	00001010	10
00000011	00101000	
00000110	01010000	
00000101	10000110	

Третій крок

	00000000	10
00110010	10000000	
01100101	00000000	
10000011	11000110	

Четвертий крок

У першому рядку показаний другий множник, який на кожному кроці зсувається на 2 розряди праворуч. На першому кроці до початкової нульової суми додається додатковий код першого множника, а у старший розряд другого множника додається одиниця переносу, тобто виконується множення на (4 - 1). На наступних кроках при зсуві другого множника на два розряди висувається однаковий код – 10, тому попередньо зсунутий на два розряди ліворуч перший множник зсувається ще на один розряд ліворуч (множення на 10) і додається до проміжного результату. Якщо другий множник буде складатися з низки одиниць, то то перший перенос у старші розряди обнулить одиниці у другому множнику.

Таким чином бачимо, що для виконання множення за алгоритмом Бута потрібно виконати вдвічі менше кроків, ніж за звичайними алгоритмами порозрядного множення, але за рахунок аналізу двох двійкових цифр програмні витрати на один такий крок збільшуються. Результат множення повністю співпадає з попередніми результатами за чотирма схемами множення.

7.4. Ділення

Приклад. Розділимо число 30 на число 6.
Десяткова: $30_{10} : 6_{10}$ **Двійкова:** $11110_2 : 110_2$ **Вісімкова:** $36_8 : 6_8$

```
 30 | 6
 30 | 5
  0
```

```
 11110 | 110
 110   | 101
  110
  110
    0
```

```
 36 | 6
 36 | 5
  0
```

Відповідь: $30 : 6 = 5_{10} = 101_2 = 5_8$.

Ділення в будь-якій позиційній системі числення виконується за тими самими правилами, що й ділення кутом у десятковій системі. У двійковій системі ділення виконується особливо просто, адже чергова цифра частки може бути тільки або нулем або одиницею. Машинне ділення організовано так само,

як і звичайне десяткове ділення. Тобто, починаючи із старших розрядів числа діленого віднімається дільник. Якщо результат віднімання додатній, то чергова цифра результату ділення, починаючи зі старшої цифри, дорівнює 1, інакше 0 і повторювати ці дії, зсуваючи дільник праворуч на один розряд, доки останній розряд дільника не досягне останнього розряду діленого. Якщо при черговому відніманні дільника результат буде від'ємний, то треба відновити від'ємний залишок. Від'ємний залишок можна і не відновлювати, якщо на наступному кроці віднімання замінити на додавання. Припустимо, що і-му кроці маємо від'ємний результат. Тобто, ми віднімали 2^i*D, де D – це дільник. При відновленні від'ємного залишку виконується $+2^i$*D а на черговому кроці віднімання $-2^{(i-1)}$*D, тобто фактично виконується додавання $+2^{(i-1)}$*D. Звідси існує два алгоритми ділення: з відновленням від'ємного залишку та без його відновлення.

При діленні з відновленням від'ємного залишку алгоритм складається із таких кроків:

1) до старшої частини діленого додається додатковий код дільника,

2) додатковий код дільника зсувається арифметично, тобто з розмноженням знакового розряду, праворуч на один розряд відносно діленого,

3) якщо результат попередньої дії додатній, то чергова цифра частки ділення буде 1, інакше перехід на крок 5,

4) зсув прямого та додаткового коду дільника праворуч на один розряд та перехід на крок 7,

5) якщо результат від'ємний, то чергова цифра частки - 0,

6) додавання прямого коду до від'ємного залишку та перехід на крок 4.

7) Якщо кількість повторів вказаної послідовності не перевищує u – v (u – кількість значущих цифр діленого, v – кількість значущих цифр дільника), то перейти на крок 1, інакше кінець операції. Вказані дії повторюються циклічно.

При діленні без відновлення від'ємного залишку замість відновлення від'ємного залишку виконується додавання прямого коду зсунутого праворуч дільника до від'ємного залишку алгоритм складається із таких кроків:

1) встановити змінну **код** = додатковий код дільника,

2) до старшої частини діленого додається **код**, 2) прямий та додатковий коди дільника зсуваються арифметично праворуч на один розряд відносно діленого,

3) якщо результат додатній, то чергова цифра частки ділення буде 1 та **код =** додатковий код дільника,

4) перейти на крок 6,

5) якщо результат від'ємний, то чергова цифра частки - 0 та **код** = прямий код дільника

6) Якщо кількість повторів вказаної послідовності не перевищує u – v (u – кількість значущих цифр діленого, v – кількість цифр значущих дільника), то перейти на крок 2, інакше кінець операції..

Алгоритм машинного ділення (як і множення) у відповідних машинних командах реалізований як мікропрограма (низка мікрокоманд).

Приклад машинного ділення за алгоритмом з відновленням від'ємного залишку. Для прикладу візьмемо результат попереднього множення та один з множників. В результаті ділення отримаємо другий множник.

```
 1000001111000110
-11001010
 1011100111000110          від'ємний залишок
+11001010                  відновлення
 1000001111000110
 -11001010
 0001111011000110          чергова цифра результату - 1
  -11001010
 1110110001000110          від'ємний залишок
  +11001010                відновлення
 0001111011000110          чергова цифра результату - 0
    -11001010
 0000010110000110          чергова цифра результату - 1
      -11001010
 1111100011100110          від'ємний залишок
      +11001010            відновлення
 0000010110000110          чергова цифра результату – 0
       -11001010
 1111110100110110
       +11001010            відновлення
 0000010110000110          чергова цифра результату – 0
        -11001010
 0000001001011110          чергова цифра результату - 1
         -11001010
 0000000011001010          чергова цифра результату - 1
            -11001010      чергова цифра результату - 1
```

Як бачимо, в результаті ділення отримали значення другого множника з попереднього прикладу. Насправді при діленні дільник зберігається як додатній у прямому коді та як від'ємний у додатковому коді для реалізації цього машинного алгоритму. Нижче наведено приклад ділення без відновлення від'ємного залишку.

```
011001010 – прямий код дільника,   100110110 – додатковий код дільника
01000001111000110  - ділене
01000001111000110
100110110
11011100111000110          від'ємний залишок
0011001010                 додавання
00001111011000110          чергова цифра результату - 1
11100110110
11110110001000110          від'ємний залишок , чергова цифра результату - 0
000011001010
00000010110000110          чергова цифра результату - 1
1111100110110
11111100011100110          від'ємний залишок , чергова цифра результату - 0
00000011001010
11111111100110110          від'ємний залишок , чергова цифра результату – 0
000000011001010
```

00000001001011110 чергова цифра результату - 1
1111111100110110
00000100011001010 чергова цифра результату - 1
11111111100110110
00000000000000000 чергова цифра результату – 1, нульовий залишок

В обох випадках маємо однаковий результат 10100111.

Контрольні запитання для самоперевірки

1. Коли виникає переповнення при додаванні?
2. Коли виникає переповнення при відніманні?
3. Які існують схеми множення?
4. Які існують методи ділення?
5. В якому вигляді подається дільник при машинному виконанні ділення?
6. Самостійно сформулюйте схему ділення з відновленням від’ємного залишку.
7. Самостійно сформулюйте схему ділення без відновлення від’ємного залишку.

8. Подання чисел у комп’ютері

При проектуванні ЕОМ створенні інструментального та прикладного програмного забезпечення розробникам доводиться вирішувати питання про уявлення в ЕОМ числових даних. Для розв’язання більшості прикладних завдань зазвичай досить використовувати цілі та дійсні числа. Запис цілочисельних даних в запам’ятовуючі пристрої ЕОМ не викликає жодних труднощів: число переводиться у двійкову систему і записується у прямому коді. Діапазон чисел, що представляються, у цьому випадку обмежується кількістю виділених для запису розрядів. Для дійсних даних зазвичай використовуються дві форми запису: число з фіксованою крапкою та число з плаваючою крапкою.

8.1. Поняття машинного слова

Числова інформація була першим видом інформації, яку почали обробляти на ЕОМ, і довгий час вона залишалася єдиним видом. Тому не дивно, що у сучасному комп’ютері існує велика різноманітність типів і представлень чисел. Передусім, це цілі та дійсні числа, які за своєю сутністю та за уявленням в машині розрізняються дуже істотно. Цілі числа, у свою чергу, діляться на числа із знаком та без знаку, що мають вже не такі істотні відмінності. Нарешті, дійсні числа мають два способи подання - з фіксованою та з плаваючою крапкою (комою), щоправда, перший спосіб зараз являє в основному історичний інтерес.

Мінімальна порція інформації, яка адресується до пам’яті, називається **коміркою** (елементом) пам’яті. Але що означає «адресувати», щоб розібратися з цим спершу подивимося, як вказана довжина елементу пам’яті у бітах для деяких обчислювальних машин:

1. М-20 - 45 біт. 2. БЭСМ-6 - 48 біт.
3. IBM - 8 біт(байт). 4. HoneyWell - 9 біт.

Основна перевага елементів пам’яті, як вже відзначалося вище, у тому, що ми можемо їх адресувати, тобто будь-який елемент пам’яті має свій унікальний номер, за яким комп’ютер завжди може безпосередньо за адресою звернутися до вмісту комірки. Біт такої властивості не має. Звичайно, якщо нам потрібно звернутися до 2057-го біта в комп’ютері IBM, то ми звертаємося до 257 байта і виділяємо у нього 1 біт (такі засоби у комп’ютера є), але як ви бачите адресуємо ми саме байт, а не біт.

Різними типами елементарних даних є символи, цілі числа, дійсні числа та логічні дані. Логічний та фізичний рівні їх подання визначаються конструктивними особливостями оперативної пам’яті комп’ютера. Зокрема, оскільки пам’ять комп’ютера має байтову структуру, до неї прив’язується представлення будь-яких даних. Тому, точнішим є твердження, що для подання

значень елементарних даних у пам’яті комп’ютера використовується машинне слово; цей термін в інформатиці застосовується у двох значеннях:

<u>Машинне слово</u>* - *(1) сукупність двійкових елементів, що обробляється як єдине ціле в пристроях та пам’яті комп’ютера;

(2) дані, що містяться в одному елементі пам’яті комп’ютера.

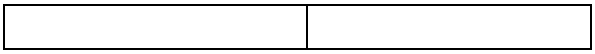
Інформація в ЕОМ подається бітами. Відповідно, будь-яка інформація перед використанням в ЕОМ повинна бути якимось чином закодована. Закодовану інформацію, яку може обробляти процесор певними порціями, у програмуванні називають даними.

Основними даними, з якими може працювати процесор, є байт та машинне слово (розмір машинного слова кратний розміру байта, для РС-сумістних комп’ютерів – це 16 біт), або подвійне слово (32 біт). Як правило, у слові може бути розміщене ціле число, а в байті – один символ тексту.

Розряди машинного слова нумеруються справа наліво, від 0 до 15. При цьому розряд із номером 0 (нульовий розряд) називається молодшим (у англомовній літературі LSB-біт, Least Significant Bit – найменш значущий біт), а розряд із номером 15 – старшим (MSB-біт, Most Significant Bit – найбільш значущий біт). Такий спосіб запису слова співпадає із записом числа у двійковій системі числення. Числа можуть подаватися у комп’ютері із знаком або без знаку. При цьому старший розряд слова вважається знаковим. Він дорівнює "1" для від’ємних чисел і "0" – для додатніх.

Розглянемо послідовно байти, які об’єднуються у більш крупні структури (машинні слова) для РС- сумістних обчислювальних систем.

У комп’ютері **Intel *86** слово (word) складається з 2 байт. Послідовність дотримання байт нумерується справа наліво (спочатку йде байт з номером 0, потім 1, 2, ..).



1 0

Відзначимо "екзотичну" особливість подання чисел в даних комп’ютерах. Числа розміром у слово та подвійне слово зберігаються у пам’яті в перевернутому вигляді. Якщо на число відведено слово пам’яті, то старші (ліві) 8 бітів зберігаються у другому байті слова, тобто за старшою адресою, а молодші (праві) 8 бітів – у першому байті, тобто за молодшою адресою. В термінах шістнадцяткової системи: перші дві цифри числа зберігаються у другому байті слова, а дві останні – у першому байті. Наприклад, число 299 = 012Bh (h означає hexadecimal – шістнадцятковий) повинно зберігатися в пам’яті так (А – адреса слова):

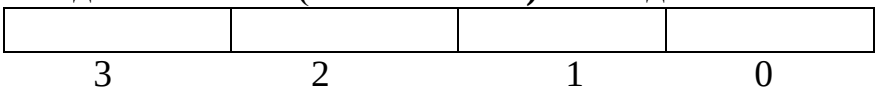
A	A+1
2B	**01**

Для чого це зроблено? Як відомо, додавання та віднімання багатозначних чисел ми починаємо з дій над молодшими цифрами. З іншого боку, перші

моделі ПК (з процесором 8080) були 8-розрядними, в них за один раз можна було прочитати з пам’яті лише один байт. Оскільки у цих умовах багатозначне число не можна прочитати з пам’яті одразу, то в першу чергу доводиться читати байт, де знаходяться молодші цифри числа, а для цього треба, аби такий байт зберігався у пам’яті першим, тобто за молодшо. адресою. З цієї причини у перших моделях ПК і з’явилося "перегорнуте" подання чисел. У наступних моделях, де вже можна було одразу рахувати з пам’яті все число, заради збереження наступності, щоб раніше створені програми могли виконуватися на нових ПК, зберегли це "перегорнуте" подання. Тому на малюнку слова така дивна нумерація байтів. Тобто, старша частина числа записується у другий байт.

Далі на малюнках числа під байтами означають порядок їх запису в пам’ять.

Подвійне слово (Double Word) – складається з 4 байт



Для вирішення більшості прикладних завдань зазвичай досить використати цілі та ійсні числа. Діапазон чисел, що представляються в цьому випадку, обмежується кількістю виділених для запису розрядів. Для дійсних даних зазвичай використовуються дві форми запису. Одна з них дістала назву ***форми з фіксованою комою*** *(крапкою)* або природної (натуральної) форми. Інше подання чисел – ***форма з плаваючою комою*** (*крапкою*), яка має і іншу назву: нормальний, експонентний або так званий науковий запис.

8.2. Подання цілих чисел у комп’ютері

Інформація в ЕОМ подається по бітах. Відповідно, будь-яка інформація перед використанням в ЕОМ повинна бути якимось чином закодована. Закодовану інформацію, яку може обробляти процесор певними порціями, у програмуванні називають даними.

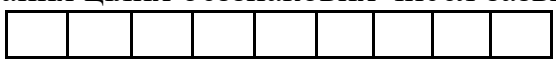
8.2.1. Цілі числа без знаку

Цілі числа без знаку зазвичай займають у пам’яті один, два або чотири байти, і приймають в однобайтовому форматі значення від 00000000_2 до 11111111_2, а у двобайтовому форматі – від 00000000 00000000_2 до 11111111 11111111_2.

Діапазони значень цілих чисел без знаку

Формат числа в байтах	Діапазон	
	Запис з порядком	Звичайний запис
1	0 ... 2^8–1	0 ... 255
2	0 ... 2^{16}–1	0 ... 65535

Форма для подання цілих беззнакових чисел зазвичай мають такий вигляд:



n-1 n-2 n-3 ... ... 2 1 0 ← номер разряда

Так, до беззнакових типів у мові PASCAL відносяться типи BYTE і WORD. Формат машинного подання чисел типу BYTE(а) і WORD(б) наведений на рис. 8.1.

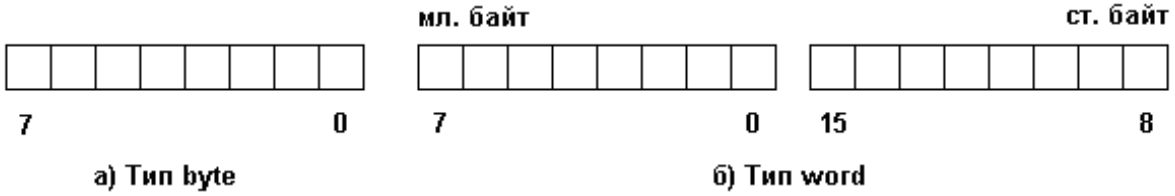
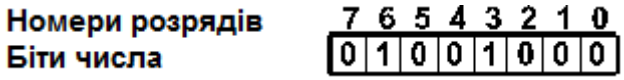


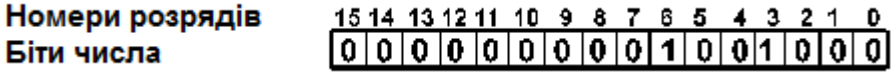
Рис. 8.1. Формат машинного подання беззнакових чисел

Приклади:

а) число $72_{10} = 1001000_2$ в однобайтовому форматі:



б) це ж число у двобайтовому форматі:



Не менш просто визначити мінімальне та максимальне значення чисел для N- розрядного беззнакового цілого: мінімальне складається з одних нулів, тобто при будь-якому N дорівнює нулю; максимальне, навпаки, утворено самими одиницями. Для обчислення максимально припустимого значення зазвичай використовують формулу Max = 2^N - 1.

У зв’язку з обговоренням граничних значень певний інтерес представляє проблема переходу через ці значення. Якщо взяти найпростіший, 8-розрядний випадок, то це будуть дії 255+1 і 0-1. Розглянемо їх детальніше.

Для того щоб отримати правильний результат, досить уявити собі, що при виконанні операції ліворуч існує ще один додатковий (дев’ятий) знаковий розряд. Тоді у двійковому виді додавання 255+1 виглядає таким чином:

```
  0 1111 1111
+ 0 0000 0001
  --------------
  1 0000 0000
```

Тепер, відкинувши неіснуючий додатковий розряд, отримуємо дещо дивний, але такий, що дійсно має місце на практиці результат: 255 + 1 = 0.

Часто в літературі наводиться ще один спосіб пояснення даної не дуже природної арифметики. Говорять, що при N розрядах арифметика "виконується за модулем" 2^N, тобто при N=8 маємо: (255+1) mod 256 = 256 mod 256 = 0.

Аналогічно, хоча і трохи складніше, можна отримати результат для іншого "граничного переходу" 0-1. Для цього доведеться додатково припускати можливість позики з дев’ятого розряду:

```
  1 0000 0000
- 0 0000 0001
  --------------
  0 1111 1111
```

Звідси 0 - 1 = 255.

Якщо тепер уважно подивитися на отримані результати, то можна помітити, що при послідовному збільшенні на одиницю ми доходимо до максимального значення і повертаємося до мінімального значення. При відніманні одиниці виходить зворотна картина.

Обговорювана нами проблема виходу за відведену розрядну сітку машини займає важливе місце при реалізації комп’ютерної арифметики і називається ***переповненням***. Як ми бачили, ситуація ця не зовсім нормальна і для отримання достовірних результатів її слід уникати. Положення ускладнюється тим, що для процесора описані результати не є чимось "загрозливим", і він "спокійно" продовжує обчислення. Єдиний нюанс полягає у тому, що сам факт переповнювання завжди фіксується шляхом установки в одиницю спеціального керуючого біта, який у подальшому програма має можливість проаналізувати. Простіше кажучи, процесор "знає" про переповнення, але надає програмному забезпеченню право прийняти рішення реагувати на нього або проігнорувати.

8.2.2. Цілі числа зі знаком

Для алгебраїчного представлення чисел, тобто для представлення чисел з урахуванням їх знака, в обчислювальних машинах використовуються спеціальні коди: зворотний або додатковий.

Ці коди дозволяють замінити незручну для комп'ютера операцію віднімання на операцію додавання з від’ємним числом. Додатковий код

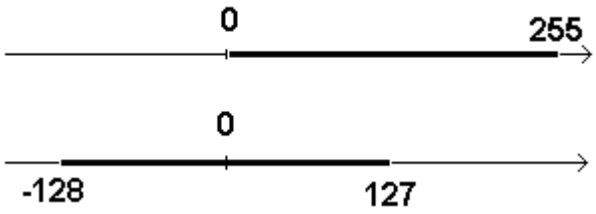
забезпечує більш швидке виконання операцій, тому в комп'ютері застосовується частіше саме він.

Цілі числа зі знаком зазвичай займають у пам’яті комп’ютера один, два або чотири байти, при цьому самий лівий (старший) розряд містить інформацію про знак числа. Знак “плюс” кодується нулем, а “мінус” – одиницею.

Діапазони значень цілих чисел зі знаком

Формат числа в байтах	Діапазон	
	Запис із порядком	Звичайний запис
1	$-2^7 \dots 2^7-1$	-128..127
2	$-2^{15} \dots 2^{15}-1$	-32768..32767
4	$-2^{31} \dots 2^{31}-1$	-2147483648..2147483647

Якщо порівняти числа зі знаком та без нього, то загальна кількість їх значень однакова, але їх діапазони зсунуті вздовж числової вісі:



Розглянемо особливості запису цілих чисел із знаком на прикладі однобайтового формату, при якому для знаку відводиться один розряд, а для цифр абсолютної величини – сім розрядів. У загальному випадку додатні числа представляються в діапазоні від 0 до 2^{n-1}-1. Усі інші конфігурації бітів відображають від’єднані числа в діапазоні від - 2^{n-1} до - 1. Таким чином, можна сказати, що число X може зберігатися в n розрядах пам’яті, якщо його значення знаходиться в діапазоні:

$- 2^{n-1} \le X \le 2^{n-1} - 1$.

У комп’ютерній техніці застосовуються три форми запису (кодування) цілих чисел зі знаком: ***прямий код, обернений код, додатковий код.***

Останні дві форми застосовуються особливо широко, оскільки дозволяють спростити конструкцію арифметико-логічного пристрою комп’ютера шляхом заміни різноманітних арифметичних операцій операцією додавання.

Для кодування чисел із знаком було введено кодування прямим кодом (для додатних даних), та кодування додатковим кодом (для від’ємних даних). При цьому старший розряд використовується як знаковий. Значення 0 – вказує на

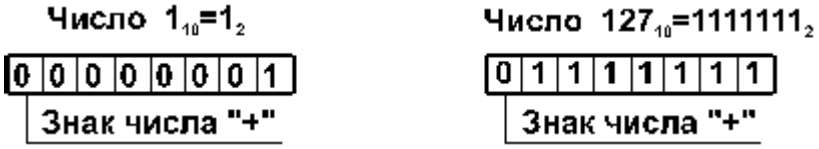
додатне число, а 1 – на від'ємне число. Далі після цього розряду розміщується власно код числа.

Для від'ємних чисел було введено додатковий код числа, який являє собою значення 2^n – D, де n – розрядність даних у комп'ютері, а D – число. Зараз найбільш розповсюдженими є 32-розрядні комп'ютери. У комп'ютері віднімання замінюється на додавання з додатковим кодом, тобто A – B = A + (2^n – B) = 2^n + A - B. Якщо A > B, то виникає перенос у старшому розряді (2^n) і результат буде додатній (тобто матимемо прямий код числа), якщо A < B, то переносу не буде і результат буде від'ємний у додатковому коді.

Для спрощення перетворення числа у додатковий код було введено обернений код числа, який являє собою значення (2^n-1) - D, де (2^n-1) являє собою двійкове число з n одиниць. Таким чином, усі розряди прямого коду при цьому інвертуються (нулі перетворюються в одиниці, а одиниці – у нулі). Обернений код відрізняється від додаткового на одиницю у молодшому розряді. Тому для отримання додаткового коду числа треба до його оберненого коду додати одиницю у молодший розряд.

Емпіричне правило (отримание дослідним шляхом): для отримання додаткового коду від'ємного двійкового числа необхідно усі символи цього числа інвертувати, окрім останньої (молодшої) одиниці і низки нулів, які за нею йдуть.

Додатні числа у прямому, оберненому та додатковому кодах відображаються однаково – двійковими кодами із цифрою 0 у знаковому розряді. Наприклад:



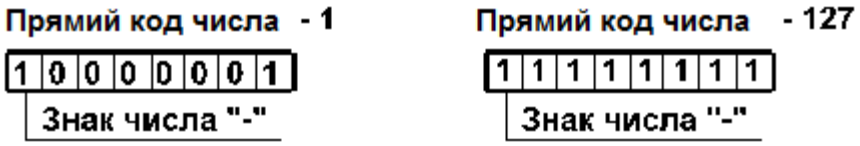
Від'ємні числа в прямому, оберненому та додатковому кодах мають різне зображення.

1. Прямий код. Прямий код цілого двійкового числа **X**, представленого в n-розрядній сітці визначається так :

Xпр = { X, якщо **X >= 0;** **A+|X|,** якщо **X < 0 },**

де **A** – величина, рівна вазі старшого розряду сітки, тобто **A = 2^{n-1}** – для цілих чисел і 2^0 =1 – для правильних дробів. Це означає, що прямі коди чисел +X та -X відрізняються тільки знаковими розрядами.

У знаковому розряді поміщується цифра 1, а у розряди цифрової частини числа – двійковий код його абсолютної величини. Наприклад:



На практиці при побудові прямого коду цілих чисел в знаковий розряд (крайній лівий) заноситься 0, якщо число додатне і 1, якщо число від'ємне, а знаковий розряд відділяється від цифрових, наприклад, вертикальною рискою |, при цьому усі бракуючі старші розряди заповнюються нулями.

Наприклад, представити задані числа в прямому коді, якщо довжина розрядної сітки **n** рівна **8**:

а) **X = 11101102**, тоді **Хпр = 0|1110110**;

b) **X = -1010102**, тоді **Хпр = 1|0010101**.

При алгебраїчному додаванні в цьому коді чисел з різними знаками доводиться визначати більше за модулем число, робити віднімання модулів чисел і привласнювати різниці знак більшого за модулем числа. Іншим недоліком прямого коду є наявність двох представлень нуля:

додатнний нуль: **+0пр = 0|0000...00**;

від'ємний нуль: **-0пр = 1|0000...00.**

Проте операцію віднімання (алгебраїчне додавання) можна звести до операції простого арифметичного додавання за допомогою оберненого і додаткового кодів, а також проводити обробку знакових розрядів за тими ж правилами, що і числових розрядів. Їх особливості полягають у тому, що для додатних чисел вони співпадають з прямим кодом, а для від'ємних чисел вони є доповненнями деяких чисел.

2.Обернений код. Обернений код двійкового цілого числа **X**, що представляється в n-розрядній сітці, визначається так

Хобр = { **X**, якщо **X >= 0;** **B-|X|**, якщо **X < 0 }**,

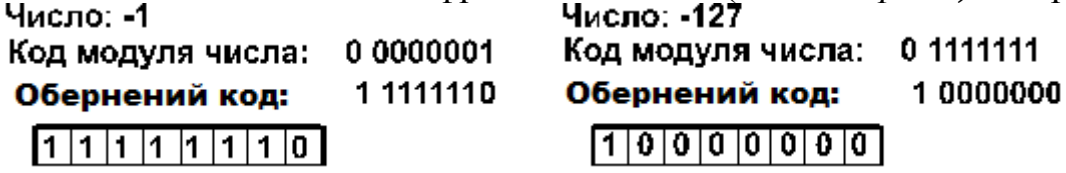
де **B** – величина найбільшого числа без знаку, що розміщується в n-разрядній сітці, тобто $\mathbf{B = 2^{n-1}}$ – для целих чисел і $\mathbf{B = 2-2^{-(n-1)}}$ – для правильних дробів.

Наприклад, представити задані числа в оберненому коді, якщо довжина розрядної сітки **n=8**:

а) **X= 111012**, тоді **Хобр = Хпр = 0|0011101**;

b) **X= -1000012**, тоді **Хпр = 1|0100001, Хобр = 1|1011110**.

На практиці обернений код одержується інвертуванням усіх цифр двійкового коду абсолютної величини числа, включаючи розряд знаку: нулі замінюються одиницями, а одиниці – нулями. Або у загальному випадку – ***оберненним кодом*** числа в системі з основою *p* називається число у цій системі, що отримується заміною цифри у кожному розряді числа на його доповнення до максимальної цифри в цій системі (тобто до *p – 1*). Наприклад:



Для числа -127_{10} обернений код буде 872_{10}.

Неважко помітити, що в оберненому коді також існує два представлення нуля:

+ 0обр = 0|0000...00 і **- 0обр = 1|1111...11**.

При алгебраїчному додаванні двох двійкових чисел, які представлені в обереному коді, робиться арифметичне підсумовування цих кодів, включаючи знакові розряди. При виникненні переносу з розряду знаку одиниця переносу додається до молодшого розряду суми кодів (таке перенесення називається **коловим** або **циклічним**). В результаті виходить сума в обереному коді. Для отримання результату в звичайному виді треба скористатися правилом зворотним правилу побудови обереного коду.

Наприклад, знайти суму чисел A=-11101_2 і B=-10011_2 в зворотному коді, якщо розрядна сітка має довжину n=8.

Побудуємо спочатку зворотні коди чисел A і B:
Апр=1|0011101, Аобр=1|1100010; Впр=1|0010011, Вобр=1|1101100 інайдемо їх суму,

Аобр= 1|1100010
+ Вобр= 1|110110 0
перенесення -> 11|1001110
+ 1
Собр= 1|1001111

Знайдемо результат в звичайному вигляді: Спр = 1|0110000, тоді
C = –110000_2.

На відміну від прямого коду, в оберненому коді не можна відкидати нулі після знакового розряду в цілій частині від'ємного числа, а також нулі у кінці дробової частини.

3. Додатковий код. Основним недоліком алгебраїчного додавання з використанням обернених кодів є наявность циклічного перенесення, для чого в процесорі ЕОМ потрібні додаткові витрати часу і спеціальні засоби схемотехніки. Від цього недоліку вільний спосіб виконання алгебраїчного складання в додатковому коді.

Ідея побудови додаткового коду досить проста. Додатковий код можна уявити собі як відображення деякого діапазону, що включає додатні та від'ємні цілі числа, на інший діапазон, що містить тільки додатні цілі числа. Розглянемо код доповнення одного байта.

Один байт може містити числа в діапазоні від 0 до 255. Код доповнення замінює цей діапазон іншим – від -128 до 127. Діапазон від 0 до 127 відображається сам на себе, а від'ємним числам надається діапазон від 128 до 255: числу -1 відповідає число 255 (0xFF), -2 – 254 (0xFE), -50 буде представлено як – 206 і так далі.

Зверніть увагу на те, що самий старший біт від'ємного числа завжди встановлюється в 1 – так можна визначити, що число від'ємне.

Додатковий код може бути розширений до 2 байтів (від 0 до 65535). Він охоплюватиме діапазон чисел від -32768 до 32767. Якщо додатковий код розширити до 4 байтів, то отримаємо діапазон від -2147483648 до 2147483647. У багатьох мовах програмування саме цей діапазон використовується для цілочисельного типу даних (Integer).

Наприклад, перетворимо числа 4, - 4, 386, - 8000 і 45000 в додатковий код, вважаючи, що цільовий діапазон – 16 біт (2 байти).

Передусім, з'ясуємо, скільки чисел може поміститися в 16 розрядів. Це дуже просто: піднесемо 2 до степеня 16. $2^{16} = 65536$, що відповідає діапазону від 0 до 65535. Тепер визначимо межі: 65536/2 = 32768. Отже, ми отримали діапазон від -32768 до 32767 (0 – це додатне число!).

Перше число, 4, потрапляє в діапазон > 0, 32767 <, тому воно відображається само на себе – в цільовому діапазоні це буде теж число 4. Число -4 – від'ємнене, воно знаходиться в діапазоні <-32768, 0>. У цільовому діапазоні воно буде представлено як 65536 - 4 = 65532. Число 386 залишиться само собою. Число -8000 – від'ємнене, в результаті відображення виходить 65536 - 8 000 = 57536 – це і буде число -8000 в додатковому коді.

І, нарешті, число 45000 не може бути представлено в додатковому коді, оскільки воно виходить за межі діапазону 16-розрядної сітки для додатного числа.

Приналежність до інтервалу кодів додатних або від'ємних чисел видно за станом старшого біта – у кодів додатних чисел його значення "0", від'ємних – "1". Це нагадує представлення зі знаком, але не є таким, оскільки використовується інший принцип кодування. Його застосування дозволяє замінити віднімання чисел їх підсумовуванням в додатковому коді, так само як і в оберненому коді. Переконаємося в цьому трохи пізніше після того, як обговоримо спосіб побудови додаткового коду цілих чисел.

Доповненням (*D*) *k*-розрядного цілого числа *Z* в системі числення *p* називається величина $\boldsymbol{D(Zp, k) = p^k - Z}$.

Або ж, цю формулу можна представити в іншому вигляді: $D(Z_p, k) = ((p^k - 1) - Z) + 1$. Число $p^k - 1$, як нам відомо, складається з *k* найбільших в цій системі числення цифр $(p - 1)$, наприклад, 9999_{10}, FFF_{16} або 1111111_2. Тому $(p^k - 1) - Z$ можна отримати шляхом доповнення до $p - 1$ кожної цифри числа *Z* і наступним додаванням до останнього розряду 1.

Наприклад, побудувати доповнення числа 278_{10}. В даному випадку $p = 10, k = 3$.

$D(278_{10}, 3) = \{<9-2><9-7><9-8>\}+1$, тобто $721+1=722$.

Важливою властивістю доповнення є те, що його сума з початковим числом в заданій розрядній сітці дорівнюватиме 0. У розглянутому прикладі:

722 + 278 = 1 000 = 0.

В розряді тисяч 1 повинна бути відкинута, оскільки вона виходить за відведену розрядну сітку.

Таким чином, додатковий код двійкового цілого числа X, що представлене в n - розрядній сітці, можна визначити як

Хдоп = { **X**, якщо **X >= 0**; **C-|X|**, якщо **X < 0** },

де C - величина рівна вазі розряду, що йде за старшим розрядом в n-розрядній сітці, тобто $C=2^n$ – для цілих чисел і C=2 – для правильних дробів.

Для цілих чисел додатковий код дозволяє використовувати числа в діапазоні $-2^{n-1} <= X <= 2^{n-1} - 1$.

Як видно з визначення, для X>=0 Хпр = Хобр = Хдоп. Для від'ємних двійкових цілих чисел справедливе співвідношення:
Хобр=2^n-1 - |X| = C - |X| - 1, звідси витікає, що **Хдоп = Хобр + 1.**

Наприклад, представити задане число **X** в додатковому коді, якщо розрядна сітка має довжину **n=8**:

X = - 1001102, оскільки **X<0,** то для отримання **Хдоп** здійснимий наступні перетворення:

Хпр = 1|0100110; Хобр = 1|1011001; Хдоп = 1|1011001 + 1 = 1|1011010, таким чином, **Хдоп = 1|1011010**.

Емпіричне правило: для отримання додаткового коду від'ємного числа у будь-якій системі числення необхідно скануючи усі символи цього числа зліва неаправо оставити всі нулі до першої ненульової цифри, цю цифру замінити на доповнення на одиницю більше, решту цифр замінити на доповнення.
Наприклад, побудувати доповнення числа 27800_{10}; *278доп = {<9-2><9-7><9-8+1><0><0>}*, тобто *72200.*

Перевагою додаткового коду є **єдине представлення нуля**, дійсно:

+ 0доп = 0|0000...00 и **- 0доп = 1|1111...11**

\+ **1**

10|0000...00,

оскільки в розрядній сітці представлення числа немає розряду лівіше за знаковий, тому крайня ліворуч 1 відкидається, а у знаковому розряді буде 0, тобто

- 0доп = 0|0000...00 = + **0доп**.

Ще однією перевагою додаткового коду є просте правило визначення парного (чи непарного) числа. Число є парним, якщо <останній розряд> = 0, інакше – непарним. Запис числа в додатковому коді це правило не порушує.

При алгебраїческому додаванні двох двійкових чисел, що представлені додатковим кодом, виконується арифметичне додавання цих кодів, включаючи розряди знаків. При виникненні переносу з розряду знаку одиниця переносу відкидається (ігнорується). В результаті виходить алгебраїчна сума в додатковому коді.

Цікавим є таке спостереження: якщо число 01111111 (при n=8) збільшити на одиницю, то вийде 10000000, що означає: 127+1 = -128! Схожа ситуація виникає, якщо скласти -67 та -71.

-67	1000011
-71	1000111
-137	(1)0001010

перенос (у дужках) не враховується, тому результат дорівнює 10_{10} (1010_2). Тобто має місце переповнення, при цьому знаковий розряд змінюється на протилежний.

Щоб виявити таку ситуацію у комп'ютері, для знакового розряду було відведено два розряди. Тоді його значення для двох попередніх випадків дорівнювало би 01 та 10 відповідно (**01** при додаванні додатних та **10** при

додаванні від'ємних операндів). За відсутністю переповнення їхнє значення для додатного результату завжди дорівнює 00, а для від'ємного – 11. Таким чином, переповнення фіксується у комп'ютері, коли знакові розряди різні. Старший знаковий розряд зберігає знак, а молодший – вказує на переповнення.

Отже, безліч елементів типу byte можна уявляти собі у вигляді згорнутого у кільце відрізка [-128,127]. Принципово нічого не змінюється і для типів даних, для подання яких потрібно більше байт – збільшується лише довжина відрізка, яка вирізається з дійсної вісі перед згортанням його у кільце.

Наприклад:

(-1) + (-127) = 11111111 + 10000001 = 10000000 = -128.

Зазвичай від'ємні десяткові числа при введенні у комп'ютер автоматично перетворяться програмою-драйвером у двійковий, обернений і далі у двійковий додатковий код, і у такому вигляді зберігаються, переміщуються та використовуються в операціях. При виведенні таких чисел з комп'ютера відбувається зворотне перетворення у від'ємні десяткові числа.

Наприклад, для подання цілих чисел із знаком у мові програмування PASCAL визначені наступні типи SHORTINT, INTEGER, LONGINT. У наведених типах числа зберігаються у **додатковому коді.** Формат машинного подання чисел типу SHORTINT (а), INTEGER (б), LONGINT (в) наведений на рис 8.2, де s- знаковий розряд числа. Для додатних чисел s=0, для від'ємних s=1.

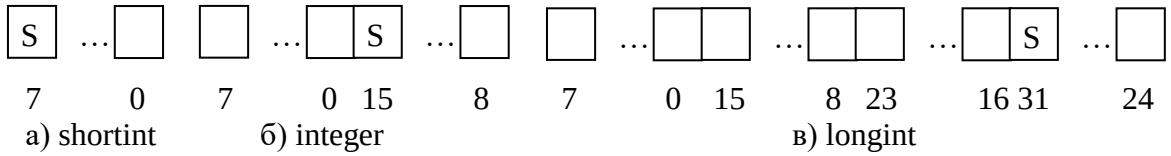


Рис. 8.2. Формат машинного подання чисел зі знаком

Усі типи значень цілих чисел в PASCAL приведені у наступній таблиці:

Тип	Диапазон значений	Машинне представлення
shortint	-128..127	8 бит із знаком
integer	-32768..32767	16 бит со знаком
longint	-147483648..2147483647	32 бита із знаком
byte	0..255	8 бит без знака
word	0..65535	16 бит без знака

comp	-2^63+1..2^63-1	64 бита із знаком

Особливе положення в PASCAL займає числовий тип даних **comp** (від англ. Compound – складений), який трактується як дійсне число без експоненціальної та дробової частин. Фактично, comp – це «велике» ціле число із знаком, що зберігає 19…20 значущих десяткових цифр. У той же час числовий тип даних comp у виразах повністю сумісний з іншими дійсними типами: над ним визначені усі дійсні операції, він може використовуватися як аргумент математичних функцій.

Слід також відмітити, що у мові Object Pascal цілий тип Integer вже визначений як 4-байтний знаковий тип. У багатьох компіляторах тип З int також реалізований як 4-байтний.

8.3. Числа з фіксованою крапкою

У формі представлення з фіксованою крапкою числа зображуються у вигляді послідовності цифр з постійним для всіх чисел становищем крапки, що відокремлює цілу частину від дробової .

Наприклад, нехай числа представлені в десятковій системі числення і мають п'ять розрядів в цілій частині числа (до крапки) і п'ять у дробовій частині (після крапки). Числа, записані в таку розрядну сітку, мають вигляд +00654.32100; +00000.00325.

Ця форма найбільш проста, природна, але має невеликий діапазон представлення чисел і тому найчастіше неприйнятна при обчисленнях.

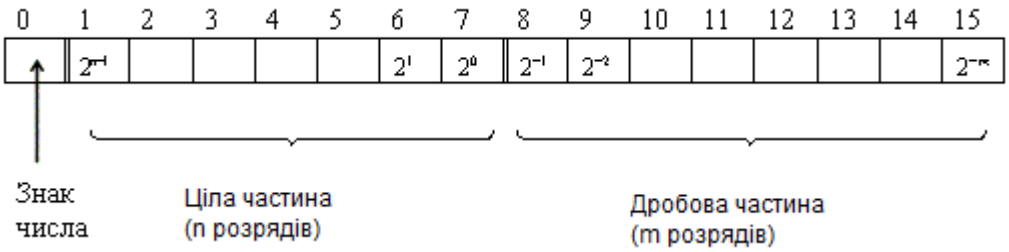
Форма запису числа з фіксованою крапкою використовувалася в основному на ранніх етапах розвитку обчислювальної техніки. Запис числа з фіксованою крапкою зазвичай має знаковий і цифровий розряди. Фіксована крапка означає, що на етапі конструювання ЕОМ було визначено, скільки і які розряди машинного слова відведені під зображення цілої і дробової частин числа. Крапка в розрядній сітці може бути зафіксована, в принципі, після будь-якого розряду.

Зазвичай в пам'яті ЕОМ числа з фіксованою крапкою зберігаються в трьох форматах:

- півслово – це зазвичай 16 біт або 2 байта;
- слово – 32 біта або 4 байта;
- подвійне слово – 64 біта або 8 байтів.

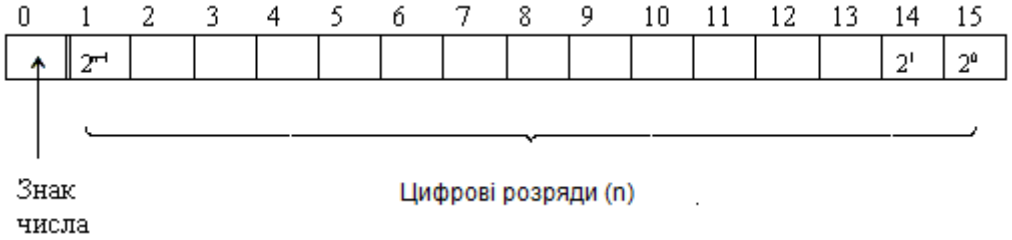
Від’ємні числа з фіксованою крапкою записуються в розрядну сітку в додаткових кодах, які утворюються додаванням одиниці до молодшого розряду оберненого коду.

Приклад. Комірка з цілою та дробовою частиною.



Як окремий випадок числа з фіксованою крапкою, може бути розглянута запис цілого числа (у цьому випадку всі розряди, окрім знакового, використовуються для запису цілої частини).

Приклад. Комірка із записом цілого числа.



До переваг використання чисел з фіксованою крапкою відносяться простота виконання арифметичних операцій та висока точність зображення чисел. До недоліків – невеликий діапазон подання чисел.

8.4. Числа з плаваючою крапкою

Дійсними числами у програмуванні називають правильні та неправильні десяткові дроби. У записі таких чисел замість коми, що відокремлює цілу частина числа від дробу, використовують «десяткову крапку»: 156.273_{10}.

Комп'ютерна (або «машинна») арифметика відрізняється від «справжньої». Безліч дійсних чисел нескінченна і безперервна (вона включає будь-які числові значення). Для зберігання чисел у комп'ютерах використовуються елементи пам'яті, які містять кінцеву кількість двійкових розрядів. Кінцевим числом розрядів неможливо представити будь-яке число (наприклад, періодичний десятковий дріб 0,33...), тому множина дійсних чисел, які можна використовувати в комп'ютерах, є кінцевою і «дискретною» (тобто деякі числа не можна записати у пам'ять комп'ютера). Отже, комп'ютерна арифметика неминуче є наближеною: арифметичні операції з дійсними числами виконуються приблизно, з деякою похибкою, яка називається похибкою машинної арифметики. Похибка залежить від системи числення.

Для зручності зображення чисел, які приймають значення з достатньо широкого діапазону (тобто, як дуже маленьких, так і дуже великих), використовується форма запису чисел з порядком основи системи числення. Наприклад, десяткове число 1.25 можна в цій формі представити так:

$1.25*10^0 = 0.125*10^1 = 0.0125*10^2 = ...$, або так:

$12.5*10^{-1} = 125.0*10^{-2} = 1250.0*10^{-3} = \ldots$.

Таке зручне подання дійсних чисел не довелося спеціально вигадувати. У математиці вже існував відповідний спосіб запису, який заснований на тому факті, що будь-яке число X в позиційній однорідній системі числення з основою P можна записати як $X = \pm MP^{\pm K}$, де M називають ***мантисою*** (від лат. – надбавка) числа X, K – ***порядком***, а $P^{\pm K}$ – ***характеристикою*** цього числа (наприклад, якщо довжину екватора Землі записати як $X = 0.4*10^8$ м, то $M = 0.4$, $P = 10$, $K = 8$). Інакше порядок числа називають також ***експонентою***, а мантисою також називають дробову частину десяткового логарифма.

Такий спосіб запису чисел називається поданням з плаваючою крапкою. Положення крапки визначається значенням порядку P. Із зміною порядку крапка переміщується (плаває) ліворуч або праворуч.

Якщо “плаваюча” крапка розташована в мантисі перед першою значущою цифрою, то при фіксованій кількості розрядів, які відведені під мантису, забезпечується запис максимальної кількості значущих цифр числа, тобто максимальна точність представлення числа в комп’ютері.

З цього виходить, що мантиса повинна бути правильним дробом, перша цифра якого відмінна від нуля: M із [0.1, 1). Тому, для встановлення однозначності при записі чисел прийнята нормалізована форма запису числа. Мантиса нормалізованого числа може змінюватися в діапазоні: $1/P \le |M| < 1$. Таким чином, в нормалізованих числах цифра після крапки повинна бути значущою.

Мантису і порядок q-ічного числа прийнято записувати в системі з основою q, а сама основа – у десятковій системі.

В машинній арифметиці прийнято, що значення двійкової мантиси має бути у діапазоні $1.0 \le M < 2$. це зручно з точки зору подання значень дійсних чисел при виведенні, коли першою є обов’язково значуща цифра, далі йде дробова крапка і решта цифр мантиси та вказівкою порядку. До того ж, якщо порядок числа дорівнює нулю, то таке число у такому вигляді відповідає нульовому значенню порядка.

Приклади нормалізованого подання:

Десяткова система	Двійкова система
$753.15 = 0.75315*10^3$;	$-101.01 = -0.10101*2^{11}$ (порядок $11_2 = 3_{10}$)
$-0.000034 = -0.34*10^{-4}$;	$-0.000011 = 0.11*2^{-100}$ (порядок $-100_2 = -4_{10}$)

Дійсні числа в комп’ютерах різних типів записуються по-різному. При цьому комп’ютер зазвичай надає програмістові можливість вибору з кількох числових форматів, які найкраще відповідають для конкретного завдання – з використанням чотирьох, шести, восьми або десяти байтів. Як приклад наведемо характеристики форматів дійсних чисел, які використовуються IBM-сумісними персональними комп’ютерами:

Формати дійсних чисел	Розмір в байтах	Зразковий діапазон абсолютних значень	Кількість значущих десяткових цифр
Одинарний	4	$10^{-45} \ldots 10^{38}$	7 або 8
Дійсний	6	$10^{-39} \ldots 10^{38}$	11 або 12
Подвійний	8	$10^{-324} \cdot 10^{308}$	15 або 16
Розширений	10	$10^{-4932} \ldots 10^{4932}$	19 або 20

З цієї таблиці видно, що форма подання чисел з плаваючою крапкою дозволяє записувати числа з високою точністю та з дуже широким діапазоном.

8.5. Внутрішні формати дійсних даних

У 1937 році ***Конрадом Цузе*** (Німеччина) для збільшення діапазону чисел, що подані в арифметиці двійкових чисел, а також для підвищення точності цього подання було запропоновано представлення чисел у плаваючій, нормалізованій формі – число x представляється у вигляді: $\boldsymbol{x = m \times p^k}$, де m – **мантиса числа**, k – цілий **порядок числа**, $\boldsymbol{p^{-1} \le |m| < 1}$.

Якщо з n розрядів, що відводяться під зображення чисел, m двійкових розрядів відвести під мантису, k – під порядок, один розряд – під знак числа і один розряд – під знак порядку (наприклад, 0 – плюс, 1 – мінус), то діапазон уявних у формі з плаваючою комою чисел різко збільшується ($m + k + 2 = n$):

$$-(0.1\ldots1)_2 \times (10)_2^{+(1\ldots1)_2} \le x \le +(0.1\ldots1)_2 \times (10)_2^{+(1\ldots1)_2}$$

(багато крапок відповідає m одиницям мантиси та k одиницям порядку).

Числа, менші нижньої межі додатних чисел і більші верхньої межі від'ємних чисел, вважаються рівними нулю, не розрізняються між собою. Числа, більше верхньої межі додатних чисел, покладаються рівними додатній нескінченності (менші за нижню межу від'ємних – від'ємної нескінченності). Порівняння двох різних за величиною чисел в арифметиці з обмеженою розрядністю може через це призводити до невірного результату, як і порівняння двох рівних в таких *системах* чисел з математичної точки зору.

Таке подання дуже зручно для зберігання в ЕОМ, оскільки насправді необхідно зберігати не саме число, а його знак, мантису, порядок та знак порядку, і усі операції з числами зводяться до операцій з цими об'єктами. Операції додавання та віднімання з цими об'єктами прості: порівняння знаків, вирівнювання порядків, додавання або віднімання мантис, нормалізація, тобто зрештою зводяться до операцій зсуву, порівняння розрядів, що досить просто реалізовуються. Для множення та ділення чисел іхні порядки відповідно додаються або віднімаються, а мантиси перемножуються або діляться. При цьому результат нормалізується.

Таким чином, запис чисел у форматі з плаваючою точкою є дуже ефективним засобом подання дуже великих і дуже малих дійсних чисел за умови, що вони містять обмежену кількість значущих цифр, і, отже, не усі дійсні числа можуть бути представлені у пам'яті. Звичайне число використовуваних при обчисленнях значущих цифр таке, що для більшості завдань помилки округлення дуже малі.

Суттєво, що двійкова мантиса завжди розпочинається з одиниці ($M \geq 1$). Тому у багатьох ЕОМ ця одиниця не записується в ОЗУ, що дозволяє зберегти ще один додатковий розряд мантиси (так звана **прихована одиниця)**.

Арифметика чисел з плаваючою крапкою виявляється помітно складніше, ніж з фіксованою крапкою. Наприклад, щоб скласти два числа з плаваючою крапкою, необхідно заздалегідь привести їх до представлення, коли обидва порядки рівні; таку процедуру прийнято називати вирівнюванням порядків. Крім того, в результаті обчислень нормалізація часто порушується, а значить необхідно її відновлювати. Проте, обчислювальні машини з усім цим чудово вміють автоматично впоратися, і саме такий спосіб обчислень лежить в основі роботи сучасних комп'ютерів.

Зауважимо, що для процесорів Intel усі операції з плаваючою крапкою винесені в спеціальний блок, який прийнято називати математичним співпроцесором (до 486 моделі співпроцесор конструктивно був окремою мікросхемою).

Таким чином, ми бачимо, що при використанні методу подання дійсних чисел з плаваючою крапкою фактично зберігається два числа: мантиса і порядок. Розрядність першої частини визначає **точність обчислень**, а другої - **діапазон подання чисел**.

Для IBM PC мантиса зберігається у прямому коді, а для зберігання порядку використовується так званий зсув: до значення порядку додається деяка константа.

Оскільки код дійсного числа у комп'ютері є приблизним представником багатьох чисел з інтервалу, то і результати обчислень також будуть свідомо неточними, такими, що містять неминучу похибку. У цьому полягає *головна особливість обробки дійсних чисел у комп'ютері – вона завжди ведеться з похибкою*.

Разом з поняттям найбільшого дійсного числа (через обмеженість розрядної сітки) з'являється поняття найменшого числа або ***машинного нуля***. Наприклад, в типі *Real* мови *PASCAL* будь-яке десяткове число, за модулем менше $2{,}3\ 10^{-39}$ виявляється машинним нулем, тобто вважається рівним 0 при збереженні і в операціях з ним. Таким чином, математичне поняття «0» як точне значення числа в комп'ютерному уявленні замінюється поняттям «Машинний нуль» як значення числа менше деякої певної величини.

Формат для подання чисел з плаваючою крапкою містить одне або два поля фіксованої довжини для знаків. Кількість позицій для значущих цифр різна в різних ЕОМ, але існує, проте, загальний формат наведений на рис. 8.3 а). Відповідно до цього запису формат дійсного числа містить у загальному випадку

поля мантиси, порядку та знаків мантиси і порядку. Проте, найчастіше, замість порядку використовується характеристика, що дорівнює збільшенню до порядку такого зміщення, щоб характеристика була завжди додатною. Формат подання дійсних чисел такий, як на рис. 8.3 б.

При представленні чисел з плаваючою крапкою порядок може бути додатнім або від'ємним, але обов'язково цілим числом. Мантиса ж являє собою додатній або від'ємний правильний дріб, причому $P^{-1} <= M < 1$. Це співвідношення називають ***умовою нормалізації мантисси***.

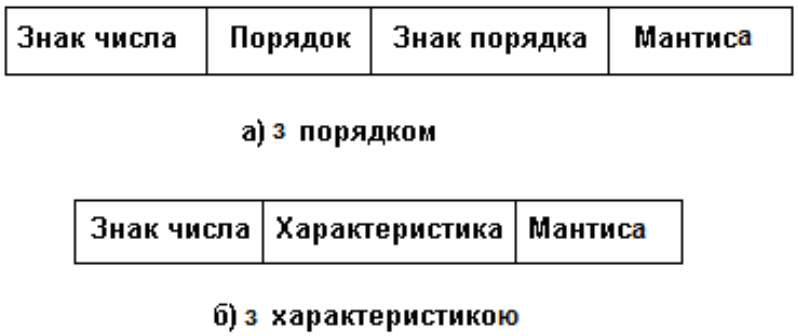


Рис. 8.3. Формат подання дійсних чисел

Нормалізоване представлення мантиси дозволяє раціональніше використати розрядну сітку (у старших розрядах мантиси не можуть знаходитися нульові цифри) і виключає неоднозначність запису усього числа.

У ЕОМ, де використовується подання чисел у формі з плаваючою крапкою, арифметичні операції виконуються як над мантисами, так і над їх порядками. При цьому часто потрібно виконувати операцію нормалізації чисел (виконання умови $P^{-1} <= M < 1$). Тому процесори (співпроцесор) з плаваючою крапкою є складнішими і менш швидкодіючими, ніж процесори з фіксованою крапкою. У свою чергу, арифметичний співпроцесор у порівнянні з програмною реалізацією обробки дійсних чисел оскільки дійсних чисел значно (у десятки разів) прискорює обчислення, які пов'язані з дійсними числами.

Формат даних з плаваючою крапкою, що використався в ЕОМ 1-го і 2-го поколінь, включав чотири поля, а саме: поле мантиси і її знаку, а також два додаткові поля для порядку і його знаку. Робота з числами у формі з плаваючою крапкою в ЕОМ, в яких був відсутній спеціалізований процесор, реалізовувався за допомогою бібліотечних функцій.

Хоча діапазон подання чисел у формі з плаваючою крапкою набагато ширше за діапазон подання чисел з фіксованою крапкою, проте у процесі обчислень результат обчислень може виходить за межі діапазону. Якщо результат обчислень перевищує Xмах, то такий випадок називають **переповнюванням порядку** і обчислення при цьому зупиняють. Якщо результат обчислень менше Xmin (**антипереповнення**), то це сприймається як нульовий результат.

В ході виконання арифметичних операцій робиться нормалізація проміжних і кінцевого значень, що полягає в зсуві мантиси праворуч або ліворуч з одночасною зміною порядку, що еквівалентно зміщенню роздільника десяткових розрядів (десяткової крапки) - саме з цієї причини така форма подання числа дістала назву «***з плаваючою крапкою».*** Як і у разі цілих чисел, для кодів дійсних чисел існує поняття *переповнювання*, проте виникає воно не після заповнення розрядної сітки мантиси – це призводить лише до нормалізації числа, а при заповненні усіх розрядів порядку. Так, для розміщення числа в 32-х бітах (2 біта – знак порядку і мантиси, 6 біт – порядок, 24 біта – мантиса), очевидно

$$|X_2|^{max} = 0{,}11111111111111111111111_2 \cdot 2^{111111} \approx 2{,}147_{10} \cdot 10^9$$

При цьому точність обробки складе 7 десяткових розрядів (24/3.322=7.22). При $|X_2| > |X_2|_{max}$ виникне переповнення, тобто операція стане некоректною.

Наведений метод нормалізації є класичним методом, при якому результат нормалізації представляється у вигляді правильного дробу, тобто з одиницею після точки і нулем у цілій частині числа.

У IBM PC нормалізована мантиса використовується по-іншому: старший біт міститься зліва від точки. Іншими словами нормалізована мантиса в IBM PC належить до інтервалу 1 <= M < 2. У пам’яті машини для даних типу real, single, double старший біт не зберігається, тобто є "прихованим" і використовується для збільшення точності у форматах single та double або для зберігання знаку у форматі real. Для додатніх і від’ємних чисел нормалізована мантиса у пам’яті подається у прямому коді.

Для спрощення операцій над порядками їх зводять до операцій над цілими додатніми числами, використовуючи подання чисел із зміщеним порядком (зміщення на $\mathbf{2^n}$). Наприклад, при семи розрядах, які відводяться на представлення порядку (**n = 6** плюс знаковий розряд), $\mathbf{2^n = 64}$. Тоді порядки діапазону - $\mathbf{2^n \le P \le 2^n}$ - 1, тобто **- 64 ≤ P ≤ 63** будуть мати обмеження $\mathbf{0 \le P_{см} = (P + 2^n) \le 2^{n+1} - 1}$, що відповідає $\mathbf{0 \le P_{см} \le 127}$.

Зміщені порядки називають ***характеристиками***. Оскільки усі характеристики є додатними, то це спрощує виконання деяких дій над ними, наприклад, порівняння. Для цього віднімання однієї характеристики з другої можна зробити без попереднього аналізу їх знаків.

Так, наприклад, у процесорі Intel на апаратному рівні прийнята угода, що порядок Рсм (характеристика) визначається у форматі дійсного числа шляхом додавання константи, що умовно називається її ***фіксованим зміщенням***, до порядку Р: $\mathbf{P_{см} = P + <фіксоване\ зміщення>}$.

Тому від’ємні порядки будуть представлені величиною $P_{см}$ < [фіксоване зміщення], а додатні - $P_{см}$ > [фіксоване зміщення].

Таким чином, для подання дійсних чисел у пам’яті ЕОМ порядок Р дійсного числа представляється у вигляді характеристики шляхом додавання зміщення. Звичайне **Зміщення** – це число, що дорівнює половині максимально можливого числа, яке може поміститися в поле **Характеристика**. Тобто, максимально можливе число необхідно зсунути праворуч, щоб отримати

зміщення. Або ж для різних форматів можна скористатися загальною формулою:

$$X = 2^{(n-1)} + k + P,$$ де:

– n - кількість біт, що відведені для характеристики,

– P - порядок числа,

– k - поправочний коефіцієнт фірми IBM, рівний +1 для real, і - 1 для форматів single, double, extended.

Розмір простору, який ПК використовує для зберігання **Характеристики** і **Мантиси**, встановлений стандартом **IEEE 724–1985** (Institute of Electrical and Electronics Engineers, Standard for floating point numbers) і підтримується усією сучасною комп'ютерною архітектурою. Звичайно це базові дійсні величини типу float, double, long double (наприклад для C/C++), та single, double, extended (для Pascal). Зазвичай їх називають: Короткий, Довгий, Розширений. Тип Real визначений у стандартному Pascal.

Для кожного з трьох можливих форматів дійсних чисел зміщення має різне, але фіксоване для конкретного формату значення, яке залежить від кількості розрядів, що відводяться під характеристику (таблиця. 8.1).

У таблиці показані діапазони значень характеристик і істинних порядків *p* дійсних чисел, що відповідають їм. Відмітимо, що нульовому порядку дійсного числа у короткому форматі відповідає значення характеристики дорівнює 127, якому у двійковому представлені відповідає значення 0111 1111.

Таблиця 8.1. Формат дійсних чисел

Формат	**Короткий**	**Довгий**	**Розширений**
Довжина числа	32	64	80
Розмірність мантиси M	23	52	64
Діапазон значень	10^{-}	$10^{-308}…10^{+308}$	$10^{-4932}…10^{+4932}$
Розмірність	8	11	15
Значення фіксов.	+ 127	+ 1023	+16383
Діапазон характеристик	0...255	0...2047	0...32767
Діапазон порядків *p*	-126...+127	-1022...+1023	-16382...+16383

Від'ємному порядку *p*, наприклад, - 1, відповідатиме характеристика *p* = - 1+127=126. У двійковому виді їй відповідає значення 01111110. Додатньому порядку *p*, наприклад, +1, відповідатиме характеристика 1+127=128, у двійковому виді їй відповідає значення 1000 0000. Тобто, усі додатні порядки мають у двійковому уявленні характеристики старший біт, який дорівнює одиниці, а від'ємні порядки – нулю.

Формули для обчислення характеристики для дійсних чисел у PASCAL та кількість біт, що необхідні для її зберігання, наведені у таблиці 8.2.

Таблиця 8.2. Характеристики для дійсних чисел в PASCAL

Тип	Кількість біт на характеристику	Характеристика	Розмір фіксов. зміщення
real	8	$x = 2^7 + 1 + P$	+129
single	8	$x = 2^7 – 1 + P$	+127
double	11	$x = 2^{10} – 1 + P$	+1023
extended	15	$x = 2^{14} – 1 + P$	+16383

Для кожного з чотирьох можливих форматів дійсних чисел зміщення має різне, але фіксоване для конкретного формату значення, яке залежить від кількості розрядів, що відводяться під мантису та порядок. Для персонального комп'ютера IBM PC сумарна кількість байтів, діапазони припустимих значень чисел дійсних типів, а також кількість значущих цифр після крапки у поданні чисел наведена у таблиці 8.3 (Help Delphi – 5).

Таблиця 8.3. Формат дійсних чисел

Тип	Діапазон значень	Значущі цифри	Розмір в байтах
real	2.9*10^(-39)..1.7*10^38	11 або 12	6
single	1.5*10^(-45)..3.4*10^38	7 або 8	4
double	5.0*10^(-324)..1.7*10^308	15 або 16	8
extended	3.6*10^(-4951)..1.1*10^4932	19 або 20	10

Оскільки нормалізоване дійсне число завжди має цілу одиничну частину (виключаючи подання нуля), то при його поданні у пам'яті з'являється можливість вважати перший розряд дійсного числа одиничним за замовчанням та враховувати його наявність тільки на апаратному рівні. Це дає можливість збільшити точність дійсних чисел, оскільки з'являється зайвий розряд, який можна використати для подання мантиси числа. Але це справедливо, як говорилося вище, тільки для real, single, double форматів дійсних чисел Intel. Розширений формат, як внутрішній формат подання числа будь-якого типу у співпроцесорі, містить цілу одиничну частину дійсного числа в явному вигляді. І навіть більше – усі операції з числами співпроцесор виконує над числами тільки у форматі з розширеною точністю –

extended. У цьому форматі зберігається і "зайвий" біт цілої частини нормалізованого числа.

Наведемо алгоритм формування машинного подання дійсного числа у пам'яті ЕОМ, що складається з наступних пунктів:

1. Число подається у двійковому коді.

2. Двійкове число нормалізується. При цьому для чисел, більше одиниці, плаваюча крапка переноситься ліворуч, визначаючи додатній порядок. Для чисел, менших одиниці, крапка переноситься праворуч, визначаючи від'ємний порядок.

3. За формулою з таблиці 8.2 з урахуванням типу дійсного числа визначається характеристика.

4. У відведене у пам'яті поле відповідно до типу числа записуються мантиса, характеристика та знак числа. При цьому необхідно відмітити наступне:

– для чисел типу real характеристика зберігається у молодшому байті пам'яті, для чисел типу single, double, extended – у старших байтах;

– знак числа знаходиться завжди у старшому біті старшого байта;

– мантиса завжди зберігається у прямому коді;

– ціла частина мантиси (для нормалізованого числа завжди 1) для чисел типу real, single, double не зберігається (є прихованою). Для чисел типу extended усі розряди мантиси зберігаються у пам'яті ЕОМ.

Машинне подання даних типу REAL. Формат машинного подання даних типу REAL наступний:

	мол. байт					ст. байт	
а:	**7… 0**	**15..8**	**23..16**	**31..24**	**39..32**	**47..40**	
	х....х	**м....м**	**м....м**	**м....м**	**м....м**	**s м...м**	
б:	**7 0**	**-32 -39**	**-24 -31**	**-16 -23**	**-8 -15**	**-1 -7**	де:

– а - номери разрядів пам'яті,
– б - показники ступенів розрядів характеристики і мантиси,
– s - знаковий розряд числа,
– м - нормалізована мантиса,
– х - характеристика числа.

Наведемо приклади.

1. Десяткове число 15.375;

у двійковій системі числення 1111.011;

результат нормализації $1.111011*2^3$; p=3.

Враховуючи відкидання неявної одиниці та зсув порядку, отримуємо: s=0; $x=2^7+1+3=2^7+2^2=132$;

у двійковій системі числення х=10000100; м=1110110...0;

машинне подання числа:

10000100 00000000 00000000 00000000 00000000 01110110

2. Десяткове число -25.75;

нормализована мантиса: 1.10011100...0; p=4.

машинне подання числа:

10000101 00000000 00000000 00000000 00000000 11001110

3. Число 0.0; Машинне подання числа:

00000000 00000000 00000000 00000000 00000000 00000000

Машинне подання даних типу SINGLE. Формат машинного подання даних типу SINGLE наступний:

мол. байт ст. байт

7 .. 0	**15.. 8**	**23**	**22..16**	**31**	**30..24**		- номери разрядів пам'яті
м....м	**м...м**	**х**	**м...м**	**s**	**х...х**		
-16	**-23**	**-8**	**-15**	**0**	**-1**	**-7**	- покажчики ступенів розрядів мантиси і характеристики, де:
0	**7**	**1**					

s - знаковий розряд, х - характеристика числа, м - нормалізована мантиса.

Наприклад:

1. Число -15.375;

у двійковій системі числення – 1111.011;

нормалізоване двійкове число – $1.111011*2^3$; p=3.

Враховуючи відкидання прихованої одиниці та зсув порядку, отримуємо: s=1; $x=2^7–1+3=128–1+3=130$;

у двійковій системі числення х=10000010; м=1110110...0;

машинне подання числа у форматі SINGLE:

00000000 00000000 01110110 11000001

2. Число - 0.1875;

у двійковій системі числення - 0.0011;

нормалізоване двійкове число $-1.1*2^{(-3)}$; p=-3.

Враховуючи відкидання прихованої одиниці та зсув порядку, отримуємо: s=1;

$x=2^7–1–3=2^7–2^2$;

у двійковій системі числення х=01111100; м=100...0;

машинне подання числа у форматі SINGLE:

00000000 00000000 01000000 10111110

Машинне подання даних типу DOUBLE. Формат машинного подання даних типу DOUBLE наступний:

мол.байт ст.байт

7 .. 0	**15..8**	**23..16**	**31..24**	**39..32**	**47..40**	**51..48**	**52..49**	**63**	**62..56**
м...м	**м...м**	**м...м**	**м...м**	**м...м**	**м...м**	**м..м**	**х..х**	**s**	**х..х**
-44-51	**-37-43**	**-29-36**	**-21-28**	**-13-20**	**-5-12**	**-1- 4**	**7..4**	**0**	**3..1**,

де:

- верхній рядок цифр від 0 до 63 - номери розрядів пам'яті;
- нижній рядок цифр від - 50 до - 1 - показники ступенів розрядів мантиси, від 0 до 10 - розрядів характеристики;
- s - знаковий розряд числа;
- м - нормалізована мантиса;

−x - характеристика числа ($x=2^{10}$ - 1 + p, де p - порядок нормалізованого числа).

Наведемо декілька прикладів.

1. Число 15.375;

у двійковій системі числення 1111.011;

результат нормализації $1.111011*2^3$; p=3.

Враховуючи відкидання прихованої одиниці та зсув порядку, отримуємо: s=0; $x=2^{10}–1+3=1024 –1+3 =1026$;

у двійковій системі числення x=10000000010; m=1110110...0;

машинне подання числа у форматі DOUBLE:

с 0 розряду 00000000 00000000 00000000 00000000 по 31 розряд

с 32 розряду 00000000 11000000 00101110 01000000 по 63 розряд

2. Десяткове число 0.375;

у двійковій системі числення 0.011;

результат нормалізации $1.1*2^{(-2)}$; p=-2.

Враховуючи відкидання прихованої ї одиниці та зсув порядку, отримуємо: s = 0; $x = 2^{10}–2^1–2^0 = 2^{10} – 3 = 1021$;

у двійковій системі числення x=01111111101; m=100...0;

машинне подання числа у форматі DOUBLE:

с 0 раозряду 00000000 00000000 00000000 00000000 по 31 розряд

с 32 розряду 00000000 00000000 11011000 00111111 по 63 розряд

Машинне подання даних типу EXTENDED. Цей формат є основним робочим форматом для даних співпроцесора. Формат машинного подання даних типу EXTENDED наступний:

мол.байт ст. байт

7.. 0.	**15.. 8**	**23..16**	**31..24**	**39..32**	**47..40**	**55..48**	**63..56**	**71..64**	**79..72**
м..м	**м..м**	**м..м**	**м..м**	**м..м**	**м..м**	**м..м**	**м..м**	**х..х**	**s х..х**
56-63	**48-55**	**40-47**	**32-39**	**24-31**	**16-23**	**-8-15**	**0 -7**	**7..0**	**14..8**

де:

−верхній рядок цифр - номери розрядів пам’яті;

−нижній рядок цифр - покажчики ступенів розрядів мантиси і характеристики;

−s - знаковий розряд числа;

−м - нормалізована мантиса;

−x - характеристика числа.

Наприклад:

1. Число -15.375; у двійковій системі числення -1111.011;

після нормалізації $-1.111011*2^3$; p=3.

Враховуючи відкидання прихованої одиниці та зсув порядку, отримуємо: s=1; $x=2^{14}-1+3=2^{14}+2^1=16386$;

у двійковій системі числення x=100000000000010; м=11110110...0

(у мантисі одиниця стоїть зліва від коми не відкидається).

Машинне представлення цього числа у форматі EXTENDED :

0..0 0..0 0..0 0..0 0..0 0..0 0..0 11110110 00000010 11000000

2. Число 1.0; аналогічні викладення дають
нормалізовану мантису: 1.0...0;
машинне подання числа 1.0:
0..0 0..0 0..0 0..0 0..0 0..0 0..0 10000000 11111111 00111111

8.6. Виконання арифметичнх дій над нормалізованими числами

Перед початком виконання арифметичної дії операнди операції розміщуються у відповідних регістрах співпроцесора. Співпроцесор має власні процесор мантис та процесор порядків.

8.6.1. Додавання та віднімання

При додаванні та відніманні спочатку проводиться підготовча операція, яку називають вирівнюванням порядків.

В процесі вирівнювання порядків мантиса числа з меншим порядком зсувається у своєму регістрі праворуч на кількість розрядів, що дорівнює значенню різниці порядків операндів. Після кожного зсуву порядок збільшується на одиницю. В результаті вирівнювання порядків однойменні розряди чисел виявляються розташованими у відповідних розрядах обох регістрів, після чого мантиси додаються або віднімаються.

У разі потреби отриманий результат нормалізується шляхом зсуву мантиси результату праворуч на один розряд (при додаванні чисел з однаковими знаками та відніманні з різними знаками), а порядок збільшується на одиницю. Якщо виконується додаванні чисел із різними знаками та відніманні з однаковими знаками, то мантиса результату може виявитись менше одиниці У такому випадку для нормалізації числа необхідно зсунути мантису ліворуч на таку кількість розрядів, щоб вона стала нормалізованою, а порядок при цьому зменшити на величину, яка дорівнює кількості зсувів.

Приклад 1. Скласти двійкові нормалізовані числа $0.10111 \cdot 2^{-1}$ та $0.11011 \cdot 2^{10}$. Різниця порядків доданків тут рівна трьом, тому перед додаванням мантиса першого числа зсувається на три розряди праворуч:

$$\begin{array}{ll} & 0.0010111 \cdot 2^{10} \\ + & \underline{1.1011 \quad \cdot 2^{10}} \\ & 1.1101111 \cdot 2^{10} \end{array}$$

Приклад 2. Виконати віднімання двійкових нормалізованих чисел $0.10101 \cdot 2^{10}$ та $0.11101 \cdot 2^{11}$. Різниця порядків між першим та другим операндами дорівнює одиниці, тому перед відніманням мантиса другого числа зсувається на один розряд праворуч:

$$\begin{array}{ll} & 1.0101 \cdot 2^{10} \\ - & \underline{0.11101 \cdot 2^{10}} \\ & 0.01101 \cdot 2^{10} \end{array}$$

Результат вийшов ненормалізованим, тому його мантиса зсувається ліворуч на два розряди з відповідним зменшенням порядку на дві одиниці: $1.101 \bullet 2^0$.

8.6.2. Множення

При множенні двох нормалізованих чисел їх порядки додаються, а мантиси перемножуються.

Приклад 3. Виконати множення двійкових нормалізованих чисел:

$(0.11101 \bullet 2^{101}) \bullet (0.1001 \bullet 2^{11}) = (0.11101 \bullet 0.1001) \bullet 2^{(101+11)} = 0.100000101 \bullet 2^{1000}$.

Результат множення мантис може не вкладатися у діапазон подання $1 \le M < 2$, або бути більшим за 2, оскільки обидва множника мають нормалізовані мантиси, добуток яких може перевищувати 2.

Приклад 4. Виконати множення двійкових нормалізованих чисел:

$(1.11011 \bullet 2^{10}) \bullet (1.10111 \bullet 2^7) = 11.0010101101 \bullet 2^{17}$

Після нормалізації результат має вигляд:

$1.10010101101 \bullet 2^{18}$

Таким чином, після множення можлива нормалізація із зсувом мантиси на один розряд праворуч та додаванням одиниці до порядку.

8.6.3. Ділення

При діленні двох нормалізованих чисел з порядку діленого віднімається порядок дільника, а мантиса діленого ділиться на мантису дільника. Потім у разі потреби отриманий результат нормалізується.

Приклад 5. Виконати ділення двійкових нормалізованих чисел:

$0.1111 \bullet 2^{100} : 0.101 \bullet 2^{11} = (0.1111 : 0.101) \bullet 2^{(100-11)} = 1.1 \bullet 2^1 = 0.11 \bullet 2^{10}$.

Використання подання чисел з плаваючою крапкою істотно ускладнює схему арифметико-логічного пристрою.

Результат ділення мантис може не вкладатися у діапазон подання $1 \le M < 2$, або бути меншим за 1, коли мантиса діленого менше за мантису дільника.

Приклад 6. Виконати ділення двійкових нормалізованих чисел:

$1.000010 \bullet 2^6 : 1.011 \bullet 2^3 = (1.000010 : 1.011) \bullet 2^{(6-3)} = 0.110 \bullet 2^3$

Після нормалізації результат має вигляд:

$1.10 \bullet 2^2$

Таким чином, після ділення можлива нормалізація із зсувом мантиси на один розряд ліворуч та відніманням одиниці з порядку. В усіх наведених прикладах наводиться суто значення двійкового порядку без нульового зміщення, яке узагальнено дорівнює $2^{j-1} - 1$, де j – кількікість розрядів під характеристику. Тоді, наприклад, для формату EXTENDED нульове зміщення становитиме 3FFFh, оскільки для характеристики виділено 15 розрядів. Нульове зміщення складатиме $2^{14} - 1 = 11111111111111b = 3FFFh$, як про це вже було наголошено вище.

Виключення складають формати SINGLE та REAL, де нульове значення характеристики дорівнює 2^7 -1 = 1111111b = 7Fh, але у форматі це значення зсунуто на один розряд праворуч, через що маємо запис 3F8h.

8.7. Перетворення цілих та дійсних даних при введенні/виведенні у внутрішнє уявлення

У мовах програмування є оператори введення даних (READ у мові Pascal) та виведення (WRITE у мові Pascal), які реалізуються з використанням системних підпрограм. Розглянемо окремо виконання цих підпрограм для цілих та дійсних даних.

8.7.1. Перетворення цілих даних при введенні у внутрішнє подання

Обробка вхідних цілих даних виконується, починаючи зі сташих цифр, оскільки надходять вони у комп'ютер у вигляді рядка саме так. Для спрощення будемо розглядати додатні данні, адже в разі від'ємних даних треба зробити тільки їх подання у додатковому коді. Для перетворення використовується наступний алгоритм.

1) введемо змінну SUM=0, де буде розміщено машинне уявлення цілого числа,

2) перейти на крок 4,

3) помножити SUM на 10,

4) ввести та перевірити код чергової цифри, починаючи зі старшої,

5) перевірити коректність символу цифри, та в разі її коректності виділити значення цифри і додати його до SUM,

6) якщо число, яке вводиться не закінчилось, то перейти на початок циклу – крок 3, інакше кінець перетворення.

Припустимо, треба ввести десяткову константу 273. Після виконання першого циклу SUM = 2. Після виконання другого циклу SUM = 2•10 + 7 = 27 у двійковому коді. Після виконання третього циклу SUM = 27•10 + 3 = 273 двійковому коді і буде містити остаточне значення числа у двійковому коді 273. Крок 2 введений для того, щоб перший раз не виконувати множення нульового значення SUM на 10.

8.7.2. Перетворення цілих даних із внутрішнього подання у зовнішнє при виведенні

При виведенні двійковий код числа треба ділити на 10, виділяючи цифри десяткового числа, починаючи з молодшої та перетворюючи її у символьний вигляд. Отже, алгоритм перетворення можна сформулювати таким чином.

1) розділити двійковий код числа на 10, залишок буде являти собою двійковий код чергової десяткової цифри, починаючи з молодшої, помістити символ цифри у буфер для виведення.

2) якщо результат ділення ≥ 10, то результат ділення = числу і перейти на крок 1.

3) якщо результат ділення < 10, то це буде код останньої старшої цифри,

4) помістити символ цифри у буфер для виведення та закінчити перетворення.

8.7.3. Перетворення дійсних даних при введенні у внутрішнє подання

Для перетворення дійсного числа із зовнішнього символьного подання у внутрішній формат скористаємось алгоритмом перетворення цілих даних. Ознакою дійсного числа є десяткова крапка або наявність експоненціальної частини або обидва чинника.

За наявності десяткової крапки організується лічильник для підрахунку кількості цифр після десяткової крапки и продовжується перетворення числа у двійковий код як цілого. До цього лічильника за наявності експоненціальної частини потім треба додати значення десяткового ступеня з експоненціальної частини. Далі ціле число перетворюється у формат дійсного. У співпроцесорі комп'ютера є спеціальна команда завантаження цілого у регістри співпроцесора з перетворенням у формат EXTENDED – FILD, а раніше це виконувалось за допомогою спеціальної підпрограми. В залежності від знаку значення лічильника десяткового ступеня циклічно виконувати ділення на 10 у форматі дійсного числа (додатнє значення десяткового ступеня) або множення на 10 у форматі дійсного числа (від'ємне значення десяткового ступеня).

Кількість множень (ділень) виконується у відповідності абсолютного значення десяткового ступеня. Більш формально як це було зроблено для цілих даних алгоритм виглядає доволі складно, тому для поглибленого засвоєння можна спробувати самостійно це зробити.

Приклад перетворення при введенні дійсного числа.

Нехай вводиться число 12.643. У наведеному прикладі перший рядок показує значення мантиси та двійкового порядку цього числа для формату EXTENDED.

Число	Порядок	Мантиса
12.643	4	1100101001001001101110100101111000110101001111110111
12643	14	1100010110001100000000000000000000000000000000000000
1264.3	11	1001111000001001100110011001100110011001100110011001
126.43	7	1111110011011100001010001111010111000010100011110101
12.643	4	1100101001001001101110100101111000110101001111110111

Як бачимо, після введення усе число було сприйнято як ціле 12643 і при перетворенні його у формат EXTENDED маємо порядок 14 та відповідну

мантису. Кількість цифр після десяткової крапки дорівнює 3. Далі в циклі тричі виконується ділення на 10 у форматі EXTENDED. Двійковий порядок зменшується до 11, далі до 7 і нарешті до 4. При цьому початкове значення у першому рядку, яке було оброблено транслятором з иови Асемблер, співпадає з кінцевим результатом роботи нашого алгоритму.

8.7.4. Перетворення дійсних даних із внутрішнього подання у зовнішнє при виведенні

Перетворення дійсного числа із внутрішнього подання у зовнішнє для виведення полягає у визначенні десяткового порядку числа з його перетворенням у десятковий вигляд. Для виконання такого перетворення небхідно число у форматі дійсного перетворити у вигляді десяткової мантиси $1 \le M < 10$ та експоненціальної частини. Звідсі цикл перетворення зводиться до наступних кроків.

Шляхом множення або ділення на 10 (у дійсному форматі EXTENDED) довести дійсне число до значення, що відповідає вигляду десяткової мантиси. Кількість множень та ділень рахувати в окремому лічильнику, який вказуватиме на десятковий ступінь нашого числа для підготовки його у зовнішній вигляд для виведення. При множенні на 10 маємо від'ємний десятковий ступінь, а при діленні – додатній.

Далі, з отриманої мантиси виділяється ціла частина, яка буде першою десятковою цифрою мантиси, за якою має бути десяткова крапка. Дії з множення на 10 дробової частини, що залишилась, та виділення чергової десяткової цифри повторюються стільки разів, скільки десяткових цифр має бути для конкретного формату дійсних даних (SINGLE, REAL, DOUBLE, EXTENDED). По закінченню цього циклу з літерою **E** з урахуванням знаку виводиться значення десяткового порядку з лічильника порядку.

Приклад перетворення при виведенні дійсного числа.

Припустимо, ми маємо дійсне число 135.72, машинний еквівалент якого у форматі EXTENDED має вигляд 400687B851EB851EB852. Приводимо його у формат з нормалізованою десятковою мантисою. Двічи поділивши число на 10 маємо у форматі EXTENDED значення числа 1.3572 3FFFADB8BAC710CB295E. Отже виділили першу цифру десяткової мантиси, це 1. Далі, виділяємо наступні цифри числа множенням на 10. Порядок виділення цифр та наступного множення дробової частини, що залишилась, на 10 показаний нижче

3FFFADB8BAC710CB295E	1.3572	виділення першої цифри
3FFDB6E2EB1C432CA57A	0.3572	
4000E49BA5E353F7CED9	3.572	виділення чергової цифри
3FFE926E978D4FDF3B65	0.572	
4001B70A3D70A3D70A3E	5.72	виділення чергової цифри
3FFEB851EB851EB851EC	0.72	
4001E666666666666666	7.2	виділення чергової цифри

3FFCCCCCCCCCCCCCCCCD 0.2
40008000000000000000 2.0 виділення чергової цифри

Отже, для виведення підготовлено дійсне число у десятковому вигляді: 1.3572E+2 – це число 135.72. Десятковий додатний порядок відповідає кількості ділень на 10, щоб отримати десяткову мантису. Похибка перетворення тут відсутня, оскільки округлення дає саме такий результат.

8.8. Округлення і точність представлення чисел в ЕОМ

8.8.1. Похибки наближених чисел

Число a називають ***наближеним числом*** точного числа A, якщо воно за певним критерієм не дуже відрізняється від числа A і замінює його при обчисленнях.

Якщо $a < A$, то кажуть, що число a є наближеним значенням числа A з нестачею; якщо $a > A$ – наближеним значенням з надлишком.

Різниця між точним числом A і його наближеним значенням a називають ***похибкою***. Якщо $a < A$, то $A - a > 0$; якщо $a > A$, то $A - a < 0$.

Абсолютною похибкою наближеного числа a називається абсолютна величина різниці між відповідним точним числом A і наближеним числом a, тобто

$$\Delta = |A - a| \qquad (8.1)$$

При цьому можливі два випадки:

1. Число A відомо, тоді абсолютна похибка Δ визначається за формулою (8.1);

2. Число A не відоме, що на практиці має місце частише за все, тобто неможливо визначити Δ за формулою (8.1). В цьому випадку корисно замість невідомої теоретичної абсолютної похибки Δ розглядати її оцінку зверху. Так, вимірюючи довжину будь-якого відрізка милимитровой лінійкою, ми маємо право затверджувати, що похибка виміру не перевищує 1 мм, точніше 0.5 мм, тому що завжди можна бачити, як більш точно округлити знайдене число.

Граничною абсолютною похибкою наближеного числа a називається будь-яке число Δ^*, що перевищує абсолютну похибку (або у крайньому випадку дорівнює їй):

$$\Delta = |A - a| \le \Delta^* \qquad (8.2)$$

Абсолютна похибка наближеного числа a дає можливість установити границі, у яких лежить точне число A. Значення точного числа A завжди задовольняє нерівності

$$a - \Delta^* \le A \le a + \Delta^*, \qquad (8.3)$$

де $a - \Delta^*$ – наближення з нестачею; $a + \Delta^*$ – наближення з надлишком.

Надалі співвідношення (8.3) будемо записувати так:

$$A = a \pm \Delta^* \qquad (8.4)$$

Наприклад, число $a = 45.3$ одержано округленням деякого числа, точне значення якого невідоме. Визначити граничну абсолютну похибку числа a.

Користуючись відомими правилами округлення чисел, можна сказати, что абсолютна похибка не перевищує 0.05, тобто гранична абсолютна похибка дорівнює $\Delta^* = 0.05$, тоді згідно (8.4) $A = 45.3 \pm 0.05$.

Абсолютна і гранична абсолютна похибки не дають можливості зробити висновок про те, добре чі погано проведено вимірювання значень реальних даних.

Твердження, що деяка величина вимірювана з точністю до 1 м, ще не дає можливості судити про точність виконаного виміру. Якщо з такою точністю вимірювана відстань між двома точками, що дорівнює 100 км, то це досить точний вимір з похибкою 1/100000 = 0.001% вимірюваної величини. Якщо ж з такою точністю вимірили ширину або довжину кімнати, то цінність такого виміру невелика.

Для того щоб визначити якість проведених вимірювань, необхідно визначити, яку долю складає абсолютна (гранична абсолютна) похибка від значення величини, що вимірюється. Тобто, для характеристики точності наближених чисел вводиться поняття відносної (граничної відносної) похибки.

Відносною похибкою δ наближеного числа a називається відношення абсолютної похибки цього числа до модуля відповідного точного числа:

$$\delta = \frac{\Delta}{|A|}, \text{ де } A \neq 0. \qquad (8.5)$$

Ця величина використовується для характеристики точності вимірювання даних на одиницю вимірювання. Так само вводиться поняття граничної відносної похибки.

Граничною відносною похибкою δ^* даного наближеного числа a називається будь-яке число, що перевищує відносну похибку цього числа, тобто $\delta \le \delta^*$, тоді $\delta = \frac{\Delta}{|A|} \le \delta^*$, де $A \neq 0$ звідси $\Delta \le |A|\delta^*$, тому за граничну абсолютну похибку можна прийняти величину $\Delta^* = |A|\delta^*$.

У свою чергу за граничну відносну похибку можна прийняти величину $\delta^* = \frac{\Delta^*}{|A|}$

На практиці за δ приймають величину $\delta = \frac{\Delta}{|a|}$, де $a \neq 0$.

Інформацію про те, що число a є наближеним значенням числа A з граничною відносною похибкою δ^* записують так: $A = a(1 \pm \delta^*)$.

На відміну від абсолютної похибки, що найчастіше буває величиною розмірною, відносна похибка, як відношення двох іменованих чисел однієї і тієї ж розмірності, завжди є величина безрозмірна. На практиці величини δ і δ^* часто задають у відсотках.

Відзначимо, що наближене число часто вважають тим більше точним, чим більше десяткових знаків воно має. Для того щоб перевірити, так це чи ні, порівняємо відносні похибки хоча б таких двох наближених чисел, знайдених у результаті округлення більш точних чисел: 0.000001 і 98.1. Може показатися,

що перше число визначене надзвичайно точно, а друге – дуже приблизно. У дійсності ж обоє наближених числа при округленні визначаються з точністю до половини одиниці останнього їхнього розряду, і абсолютна похибка для першого числа дорівнює $\Delta_1 = 0.0000005$, а для другого $\Delta_2 = 0.05$. Але відносна похибка для першого числа $\delta_1 = 0.0000005/0.000001 = 0.5 = 50\%$,
а для другого $\delta_2 = 0.05/98.1 = 0.00051 = 0.051\%$,
тобто друге число визначене майже в тисячу разів точніше.

Наприклад, нехай абсолютна похибка наближеного числа $a = 28.4$ дорівнює $\Delta = 0.1$. Визначити граничну відносну похибку цього наближеного числа і записати оцінку його точного значення.

Знайдемо $\delta = \frac{0.1}{28.4} \approx 0.003521126760563$. Тоді $\delta^* = 0.0036 = 0.36\%$ і $A = 28.4(1 \pm 0{,}36\%)$.

Відносна точність наближеного числа визначається тільки кількістю його значущих цифр, і положення крапки при цьому ніякого значення не має.

Наприклад, числа 743.8; 0.007438; 7438 мають одинакову відносну точність, оскільки для них δ = 0.05/743.8 = 0.0000005/0.007438 = 0.5/7438 = 0.000067 = 0.0067%.

Значущою цифрою наближеного числа a називається будь-яка цифра в його десятковому поданні, починаючи з першої ліворуч, яка відмінна від нуля, або нуль, якщо він міститься між значущими цифрами або розташований в кінці числа (справа) і є представником збереженого разряду точності.

Всі інші нулі, що входять до зображення наближеного числа і слугують лише для позначення його розрядів, не є значущими. Зокрема нулі, що знаходяться лівіше першої відмінної від 0 цифри, не є значущими цифрами.

Наприклад, дани наближені числа 0.001405 і 5.0300. Визначити кількість значущих цифр в цих числах.

За правилом значущої цифри маємо відповідно в першому числі 4 і в другому 5 значущих цифр.

8.8.2. Округлення і точність представлення чисел

Існування проблем точності для обчислювальних засобів викликане, передусім, наближеним характером послідовностей операцій, кінцівкою розрядної сітки, похибками округлення і неточністю початкової числової інформації. Тому результати обчислень повинні супроводжуватися оцінкою похибки обчислень, які представлені у вигляді сумарної (повної) похибки від усіх джерел похибок або окремо по кожному джерелу.

У загальному випадку ступінь близькості наближеного й істинного результатів залежить від ряду факторів, що впливають на процес обчислень. Тому при аналізі точності засобів обчислювальної техніки виділяють ***повну похибк*****у** результату та її компоненти: ***методичну*** (похибка методу, або мтематичної моделі), ***трансформовану*** (внесену, успадковану, неусувну) і ***арифметичну похибки***.

Методична похибка є зазвичай похибкою математичної моделі деякого процесу або ж відповідає абстрактної математичної задачі (наприклад, обчислення визначеного інтеграла). Однак для цих алгоритмів характерна наближеність відповідності між реальними процесами та їх математичними описами (обчислення визначеного інтеграла зводиться до обчислення суми площ прямокутників або трапецій). Похибка, яка породжена наближеним характером вихідного алгоритму, називається ***похибкою алгоритмізації.***

Трансформована похибка (похибка спотворення) наближених обчислень виникає, в першу чергу, через похибки у вихідних даних (наприклад, через неточне їх вимірювання або обчислення на попередніх етапах). Може виявитися, що операнди не можна точно подати кінцевою кількістю цифр, наприклад, у випадку ірраціональних чисел (наприклад, квадратний корінь з 2, число π, e тощо.). Крім того, дроби, кінцеві в одній системі числення, можуть виявитися нескінченними в іншій системі.

Арифметична похибка обумовлена обмеженою точністю виконання арифметичних операцій, яка, у свою чергу, є наслідком виконання операцій округлення і кінцевої довжини представлення чисел. Похибки округлення можуть накопичуватися і при поганій обумовленості задачі можуть призвести до дуже великих похибок.

Тому повну похибка обчислень отримують як композицію складових абсолютних похибок: $\Delta = \Delta_M + \Delta_T + \Delta_A$.

Округлення в комп'ютерній арифметиці – операція, яка чисельним значенням операнда певної довжини (далі точному операнду) ставить у відповідність деяке наближене його значення, що має меншу довжину. З чисельного аналізу відомі два основних підходи до трактування змісту цієї операції: ***округлення з нестачею*** та ***округлення з надлишком***.

Якщо між точним операндом A і результатом округлення $\underline{A}$ має місце співвідношення $\underline{A} \le A$, то говорять про ***округлення з нестачею***, а якщо $\underline{A} \ge A$, то говорять про ***округлення з надлишком***. Однак ці підходи ще не визначають алгоритмів реалізації округлення стосовно до двійкового кодування операндів та використаним ними форматам. Далі для визначеності будемо розглядати формат з фіксованою перед старшим розрядом комою, що має n розрядів для A і m розрядів для $\underline{A}$, $n > m$. Найбільше поширення для цього формату отримали наступні алгоритми округлення:

А) відкидання молодших розрядів;

Б) установка молодшого розряду, що залишився в форматі;

В) додавання одиниці до старшого розряду, що опинився за форматом.

Алгоритм-А – відкидання молодших розрядів (відкидання "хвоста"), найпростіший з реалізації, проте він вносить похибку Δ в абсолютну величину A. Значення цієї похибки визначається межами $0 \ge \Delta \ge -(2^{-m}-2^{-n})$, тобто, такий алгоритм є округленням з недоліком, середня похибка округлення становить

$$\Delta_{ср} = 2^{-1}(2^{-n}-2^{-m}) = -2^{-1}(2^{-m}-2^{-n}) = -2^{-m-1}(1-2^{-(n-m)}).$$

Таким чином, максимальна за абсолютним значенням похибка тут не перевищує одиниці молодшого розряду і залишається в форматі, а середня – одиниці старшого розряду, що виходить за формат округлення.

Приклад. Нехай задано два числа: A = 0.11011111, B = 0.10110101, округлити їх до n=6. $\underline{A}$ = 0.110111, $\underline{B}$ = 0.101101.

Алгоритм-Б – установка молодшого розряду, що залишився в форматі, в той стан, який відповідає логічній сумі з цифрою, що виходить за формат. Алгоритм-Б найчастіше використовується, коли округлення проводиться на один розряд, тобто, коли *n-m = 1*. У такому випадку похибка округлення знаходиться в межах $-2^{-m} \le \Delta \le 2^{-m}$, а її середнє значення дорівнює нулю. Якщо ж n-m> 1, то похибка округлення знаходиться в межах

$-2^{-m}(1-2^{-(n-m)}) \le \Delta \le 2^{-m-1}$, а її середнє значення становить

$\Delta_{ср}=2^{-1}(2^{-m-1}-2^{-m}(1-2^{-(n-m)}))=-2^{-m-1}(1-2^{-(n-m)})+2^{-m-2}=-2^{-m-1}(1-2^{-(n-m)}-2^{-1}) = -2^{-m-1}(2^{-1}-2^{-(n-m)})$, що дещо менше, ніж середня похибка за Алгоритмом А.

Приклад. Нехай задано три числа: A = 0.11011111, B = 0.10110101, C = 0.10111011 округлити їх до n=6. $\underline{A}$ = 0.110111, $\underline{B}$ = 0.101101, $\underline{C}$ = 0.101111.

Алгоритм-В – додаток одиниці до старшого розряду, що опинився за форматом, визначає наступні межі похибок округлення:

$-2^{-m}(1-2^{-(n-m)+1}) \le \Delta \le 2^{-m-1}$ із середнім її значенням $\Delta_{ср} = 2^{-n-1}$.

Важливою перевагою цього алгоритму є незалежність середньої похибки від *m*. Якщо n-m = 1, то Алгоритм-Б дає або нульову, або додатню похибку, і в цьому випадку може вважатися округленням з надлишком.

Приклад. Нехай задано три числа: A = 0.11011111, B = 0.10110101, C = 0.10111011 округлити їх до n=6. $\underline{A}$ = 0.111000, $\underline{B}$ = 0.101101, $\underline{C}$ = 0.101111.

При використанні будь-якого алгоритму абсолютна величина похибки округлення не перевищує одиниці (тобто, половини ваги) молодшого залишеного в форматі розряду. Однак виконання операцій над операндами, що отримані шляхом округлення, може призвезти до результатів, похибки яких більше половини ваги молодшого в форматі розряду. Наприклад, додавання двох операндів, які були округлені Алгоритмами-Б,-В, вже може призвезти до похибки суми, що дорівнює одиниці молодшого розряду. Виконання послідовності операцій з такими операндами може призвезти до накопичення великої трансформованої похибки. Це наочно видно на наступному прикладі в десятковій арифметиці.

Нехай необхідно підсумувати чотири числа: 0.116, 0.126, 0.136, 0.146. Їх сума дорівнює 0.524, а округлена до двох розрядів після коми сума дорівнює 0.52. Якщо ж підсумовувані числа округлити відразу до двох розрядів після коми, то отримаємо: 0.12, 0.13, 0.14, 0.15. Їх сума дорівнює 0.54. Тобто, трансформована похибка підсумовування цих чотирьох опреандів склала дві одиниці молодшого в форматі розряду. Зрозуміло, що чим довше ланцюжок сумарних округлених операндів, тим сильніше ефект накопичення похибок.

Точність представлення числа характеризується абсолютною і відносною похибками. Відповідно до вище викладеного далі можна визначити

абсолютну похибку операнда як різницю між його точним значенням A та його машинним поданням A_M, тобто, $\Delta = A - A_M$.

При перетворені чисел з однієї системи числення в іншу неминуче виникають похибки, оцінити які неважко, якщо відомі істинні значення вхідних чисел.

Відповідно до $-1 < [A]_{фm} < 1$ ($[A]_{фm}$ – машинне зображення числа для форми подання з фіксованою крапкою подаються в комп'ютері у вигляді $A_q = [A]K_A$, де масштабний коефіцієнт K_A обирають так, щоб абсолютне значення машинного зображення числа A в системі числення з основою q = 2 було завжди менше 1:

$$A_q = K_A[a_{-1}q^{-1} + a_{-2}q^{-2} + \dots + a_{-n}q^{-n} + \dots]$$

Оскільки довжина розрядної сітки дорівнює n двійковим розрядам після крапки, то абсолютна похибка перетворення десяткового числа в систему з основою q буде

$$\Delta[A] = a_{-(n+1)}q^{-(n+1)} + \dots + a_{-(n+s)}q^{-(n+s)} + \dots = \sum_{I=-(n+1)}^{-\infty} a^i q^i$$

Якщо q=2, то при $a_i = 1$ максимальне значення цієї похибки

$$\Delta[A]_{max} = \sum_{I=-(n+1)}^{-\infty} 1*2^i = 2^{-n}\sum_{I=-1}^{-\infty} 2^i = 2^{-n} \quad \textbf{(8.1)}$$

З (8.1) випливає, що максимальна похибка перетворення десяткових даних у двійкову не перевищуватиме одиниці молодшого розряду розрядної сітки машини. Мінімальна похибка перетворення дорівнює нулю.

Усереднена абсолютна похибка переведення чисел в двійкову систему числення $\Delta[A] = (0 + 2^{-n})/2 = 2^{-(n+1)}$.

Таким чином, абсолютна похибка для обчислювальних систем, що використовують форму представлення чисел з фіксованою крапкою, постійна і дорівнює половині ваги молодшого розряду (тобто $2^{-(n+1)} = 0.5*2^{-n}$).

Для представлення чисел у формі з фіксованою крапкою абсолютне значення машинного зображення числа

$$2^{-n} <= |[A]_ф| <= 1 - 2^{-n}$$

Оскільки сам операнд може змінюватися в межах від $A_{min} = 2^{-n}$ до $A_{max} = 1 - 2^{-n}$, то і відносна похибка подання чисел залежить від кількісного еквівалента чисел і може змінюватися в межах.

$$\delta [A]_{ф\ min} = \Delta[A] / [A]_{ф\ max} = 2^{-(n+1)} / (1-2^{-n}) \quad \textbf{(8.2)}$$

Для ЕОМ, як правило, n = 16 ÷ 64, тому $1 >= 2^{-n}$, звідки $\delta [A]_{ф\ min} \approx 2^{-n-1}$

Аналогічно, для максимального значення:

$$\delta [A]_{ф\ max} = \Delta[A]_{ф\ max} / [A]_{ф\ min} = 2^{-(n+1)}/2^{-n} = 2^{-1} = 0.5 \quad \textbf{(8.3)}$$

З (8.3) видно, що похибки подання малих чисел у формі з фіксованою комою можуть бути дуже значними (до 50%).

Для ЕОМ, що використовують форму представлення чисел з **плаваючою крапкою**, абсолютна похибка визначається як $\Delta = \Delta_M 2^P$, де Δ_M – абсолютна похибка мантиси, обумовлена так само, як і для з фіксованою крапкою; P – порядок числа. У свою чергу порядок знаходиться в межах $-(2^k-1) \le P \le (2^k-1)$. Тому мінімальна абсолютна похибка буде при найбільшому від'ємному порядку, а максимальна – при найбільшому додатньому порядку:

$$\Delta_{min} = 2^{-(n+1)} \cdot 2^{-(2^k-1)} = 2^{-(n+2^k)}, \quad \Delta_{max} = 2^{-(n+1)} \cdot 2^{2^k-1} = 2^{2^k-n-2}.$$

Відносна похибка представлення чисел з плаваючою крапкою визначається за загальним правилом

$\delta = |\Delta/A| = (\Delta_M 2^p)/(M(A)2^p) = \Delta_M/M(A)$, где *M(A)* – мантиса операнду *A*, тобто, відносна похибка не залежить від порядку *P* і знаходиться в межах

$\delta_{min} = |\Delta_M/M_{max}| = 2^{-n-1}/(1-2^{-n}) \approx 2^{-n-1}$, $\delta_{max} = |\Delta_M/M_{min}| = 2^{-n-1}/2^{-1} = 2^{-n}$.

Це означає, що в машинах з плаваючою точкою, на відміну від машин з фіксованою точкою, відносна похибка відображення чисел у в усьому діапазоні представлення практично постійна і для чисел з нормалізованою мантисою залежить від кількості розрядів мантиси: чим їх більше, тим менша похибка подання.

Контрольні запитання для самоперевірки

1. Які існують формати дійсних даних?
2. На що впливає розмір ділянки під мантису?
3. На що впливає розмір ділянки під характеристику?
4. Яка нормалізація виникає при додаванні та відніманні дійсних даних?
5. Яка нормалізація виникає при множенні дійсних даних?
6. Яка нормалізація виникає при діленні дійсних даних?
7. Самостійно сформулюйте алгоритм перетворення цілого десяткового числа із зовнішньої форми у внутрішнє подання.
8. Самостійно сформулюйте алгоритм перетворення внутрішнього подання цілого числа у вигляді цілого десяткового числа для виведення.
9. Самостійно сформулюйте алгоритм перетворення дійсного десяткового числа із зовнішньої форми у внутрішнє подання при введенні.
10. Самостійно сформулюйте алгоритм перетворення дійсного десяткового числа із внутрішньої форми у зовнішню при виведенні.
11. Які існують похібки представлення даних?
12. Які існують методи округлення?
13. Що являє собою відносна похибка і як вона відображає данні?
14. Як обраховується похибка при перетворення даних?
15. Як обраховується поибка при машинному поданні даних?

9. Поняття алгоритму

У попередніх розділах було показано структуру та призначення пристроїв, з яких складається комп'ютер, спробували з'ясувати, як подається інформація в ЕОМ. Але нами не було розглянуте питання, яким чином на ЕОМ вдається вирішувати різні складні завдання, якщо вона вміє виконувати лише прості операції?

Головна особливість усіх обчислень комп'ютера полягає у тому, що в основі його роботи лежить програмний принцип управління. Це означає, що для розв'язання як найпростішого, так і найскладнішого завдання програмісту необхідно використовувати перелік інструкцій або команд, виконуючи які крок за кроком ЕОМ видасть необхідний результат.

Таким чином, для того щоб розв'язати задачу на ЕОМ, її необхідно спочатку, так би мовити, алгоритмізувати. Саме алгоритмічний принцип і лежить в основі роботи всіх ЕОМ.

Не треба думати, що вивчення та побудова алгоритмів знадобиться лише тим хто обере професію програміста. Вміння виділяти алгоритмічну сутність явища та будувати алгоритми – дуже важливо для людини будь-якої професії.

9.1. Що являє собою алгоритм?

Побудова інформаційної моделі є перший, але не єдиний етап вивчення або використання в практичних цілях даного об'єкту, явища. Після побудови інформаційної або математичної моделі майже завжди доводиться виконувати відповідно до моделі обробку конкретної інформації (даних). Усвідомлена обробка інформації до останнього часу відбувалася в основному в мозку людини або ж застосовувалися достатньо прості пристосування – пальці на руках, камінчики, рахівниці, арифмометри, логарифмічні лінійки тощо. Проте схему обробки інформації, послідовність дій, які необхідно виконати, людина або запам'ятовувала, або записувала на папері для довготривалого зберігання.

У повсякденному житті людині часто доводиться застосовувати різні алгоритми. Спрощуючи, можна сказати, що значна частина життя полягає у вивченні, складанні та виконанні різноманітних інтуїтивних алгоритмів (тобто зрозумілих з досвіду).

Алгоритм – це заздалегідь визначений, точний припис дискретного (покрокового) процесу розв'язання тієї чи іншої задачі, що починається певним чином, і який призводить до результату за кінцеву кількість кроків.

Людина часто стикається з необхідністю розв'язання типових завдань, тобто завдань, що виникають систематично. Для кожного з них людство придумало метод (а іноді декілька методів) розв'язання. Цим методам навчають дитину батьки, педагоги в яслах, дитячому садку, вчителі у школі, друзі та знайомі. Як правильно їсти манну кашу, як перейти вулицю, як вирішити

квадратне рівняння, як добитися перемоги у тенісному поєдинку або взаємності в улюбленої людини – для розв'язання кожного з цих завдань є свої алгоритми.

Рецепт розв'язання задачі створюється для певного **виконавця**. Це може бути людина (наприклад, вартовий, який охороняє військовий об'єкт та діє згідно зі Статутом караульної служби), механізм (наприклад, музичний автомат, що виконує мелодію, яка записана на перфокартах) або електронний пристрій (комп'ютер). Можливості будь-якого виконавця обмежені набором тих дій, які він може виконувати. Може виявитись, що даний виконавець в принципі не може розв'язати поставлену задачу, тобто для нього неможливо скласти алгоритм. Наприклад, музичний автомат не зможе обчислити добуток двох чисел. Іноді складно визначити, що виконавець в принципі не зможе розв'язати дану задачу.

Класичним прикладом нездійсненного завдання є геометричне завдання про трисекцію кута (тобто про його ділення на три рівні кути) за допомогою циркуля та лінійки без поділок. Виконавцем у цьому завданні є комбінація з циркуля і лінійки. Проте за допомогою лінійки та транспортира те ж саме завдання розв'язується легко. Проблеми існування алгоритмів розв'язує наука – теорія алгоритмів. Вона ж досліджує їх властивості.

Виконавцем є комп'ютер. Надалі йтиметься про алгоритми, що складаються для комп'ютера. Зазвичай таку послідовність дій, яку необхідно виконати комп'ютеру над початковими даними, щоб досягти поставленої мети, прийнято називати алгоритмом. Відзначимо, що наведене визначення поняття алгоритму є не строгим. Його можна вважати швидше поясненням на рівні побутового використання терміну.

Таким чином, опис послідовності дій, достатньо визначений для того, щоб його можна було виконати за допомогою деякого виконавця (зазвичай автоматичного пристрою) називають ***алгоритмом.*** Зазвичай цю послідовність записують (кодують) за допомогою деяких формальних позначок. При цьому формальна система, призначена для запису алгоритмів, називається ***алгоритмічною мовою***, сам текст алгоритму – ***програмою***, а процес його створення – ***програмуванням***.

Поняття алгоритму виникло значно раніше, ніж з'явилися ЕОМ. Виникнення терміну "***алгоритм***" пов'язують з іменем великого узбецького математика IX століття Аль Хорезмі, який дав визначення правил виконання основних арифметичних операцій. У європейських країнах його ім'я Algorithmi трансформувалося в слово "алгорифм", а потім вже в "алгоритм". Надалі цей термін почали використовувати для позначення будь-якої сукупності правил, що визначають послідовність дій, виконання яких призводить к досягненню поставленої мети, але широкого поширення це поняття набуло в епоху автоматів та роботів. Є декілька загалом схожих пояснень поняття алгоритм, які акцентують увагу на різних аспектах цього поняття. Для більшої повноти сприйняття поняття "алгоритм" наведемо ще два його пояснення, які достатньо часто використовуються.

В інформатиці під **алгоритмом** розуміється строга, кінцева система правил, інструкцій для виконавця, яка визначає деяку послідовність дій, і після кінцевого числа кроків призводить до досягнення поставленої мети. **Алгоритм** є опис способу вирішення задачі, досягнення мети, а власне розв'язання задачі або виконання дій з даного способу є **виконанавцем алгоритму**.

9.2. Що являє собою "Виконавець алгоритму"?

Як видно з визначення алгоритму, важливим моментом в останніх поясненнях є використання ще одного поняття – "***виконавець алгоритму***". У загальному випадку виконувати алгоритми може не тільки людина. Тварини, комахи і навіть рослини в процесі своєї життєдіяльності виконують певні алгоритми. В принципі, доручити виконання алгоритму можна і неживим механізмам так і пристроям.

Якщо провести більш менш уважний аналіз, то виявиться, що переважну більшість своїх дій людина виконує за певними алгоритмами, іноді навіть не усвідомлюючи цього. За певними рецептами готуються ті чи інші кулінарні вироби, за певними схемами здійснюється пошиття одягу, виплавка сталі, вирощується зерно, виконуються лабораторні роботи на заняттях з фізики, хімії, біології, розв'язуються математичні завдання. Різні довідники значною мірою є збірками алгоритмів, які є способами розв'язання тих чи інших завдань, що розроблені за допомогою тієї чи іншої наукової або технічної дисципліни.

9.3. Які властивості притаманні алгоритмам?

Роль алгоритмів у житті людини вельми багатогранна і не зводиться тільки до обробки інформації. Проте, в процесі обробки інформації алгоритми відіграють першорядну роль. Треба зпзначити, що наведені вище "визначення" алгоритму досить розпливчаті і, фактично, не є визначеннями. Аби якось уточнити визначення алгоритму, перерахуємо основні властивості, які повинен мати будь-який алгоритм.

Алгоритми мають найважливішу якість – виконання одного і того самого алгоритму в одних і тих же умовах різними людьми (у загальному випадку – виконавцями), як правило, призводить до однакових результатів. Отже, можна стверджувати, що алгоритми мають (точніше, повинні мати) деякі властивості, які забезпечують цей ефект. Окрім вказаної якості, яку прийнято називати **визначеністю (однозначністю) алгоритму**, можна вказати ще **зрозумілість** завдання алгоритму його виконавцеві, **можливість виконання** алгоритму в тих чи інших конкретних умовах, принципову досяжність результату та деякі інші якості. Наявність цих властивостей і робить деякий набір правил або вказівок алгоритмом. Тому серед основних властивостей алгоритму можна виділити:

Детермінованість (визначеність). Припускає отримання однозначного результату обчислювального процесу при заданих початкових даних. Завдяки цій властивості процес виконання алгоритму носить механічний характер;

Результативність. Вказує на наявність таких початкових даних, для яких обчислювальний процес, що реалізовується згідно заданого алгоритму, повинен через кінцеву кількість кроків зупинитися та видати потрібний результат;

Масовість. Ця властивість припускає, що алгоритм повинен бути придатний для розв'язання всіх завдань даного типу;

Дискретність. Означає розчленування визначеного алгоритмом обчислювального процесу на окремі етапи, можливість виконання яких виконавцем (комп'ютером) не викликає сумнівів.

Але є ще одна властивість алгоритму, яка є іноді необхідною (але не достатньою) умовою хорошої програми, – це **ефективність**. Якщо користувач (замовник) формулює завдання у вигляді чіткого алгоритму, то процес проектування істотно полегшиться. Але найчастіше програмістові необхідно самому "поміркувати", щоб отримати більш менш ефективний алгоритм розв'язання поставленої задачі. Як приклад можна навести просте логічне завдання.

Нехай є пательня, на якій одночасно вміщуються чотири біфштекси. Час повного смаження одного біфштекса – 30 хвилин. 15 хвилин з одного боку та 15 хвилин з іншою. Завдання: за мінімальний час підсмажити на пательні шість біфштексів.

При першому розгляді для повного смаження всіх шести біфштексів буде потрібно одну годину часу: спочатку повністю підсмажити чотири біфштекси (це півгодини), потім ще два (це ще півгодини). Недосвідчений програміст може тут же "побігти" кодувати такий простий алгоритм.

Начебто, це рішення правильне. Але, уважно до нього придивившись, можна відмітити, що цілі півгодини сковорода «працює» тільки на половину своєї потужності. Тому правильним рішенням буде таке: покласти на сковороду чотири біфштекси і повністю прожарити з одного боку (15 хвилин), потім два з них перевернути, а два – замінити на сирі. У такому «складі» треба смажити ще 15 хвилин. Отже, два біфштекси підсмажено повністю, а чотири (два на сковороді і два, які були зняті) – тільки з одного боку. Зняті біфштекси треба покласти на пательню і смажити ще 15 хвилин (перевернувши заздалегідь ті, що на ній залишилися). Як бачимо, друге рішення дозволяє підсмажити ті ж біфштекси не за шістдесят, а тільки за сорок п'ять хвилин.

9.4. Способи завдання запису алгоритмів

При завданні алгоритму необхідно поклопотатись про те, щоб алгоритм сприймався всіма можливими виконавцями однозначно і точно, щоб його можна було виконати за будь-яких припустимих початкових умов, і щоб необхідний результат був отриманий за прийнятний час.

Способи завдання (запису) алгоритмів також вельми різноманітні. Зокрема, можна відзначити словесний спосіб завдання алгоритму – на рівні природної мови; запис музичної мелодії у вигляді нот. Графічні способи завдання алгоритму: креслення, які використовуються для виготовлення будь-якої деталі, намальована за спеціальними правилами схема виконання будь-якої послідовності дій (відмітимо, що таку схему прийнято називати блок-схемою алгоритму) тощо.

На практиці найбільш поширені наступні форми подання алгоритмів:

- **словесна** (записи на природній мові);
- **графічна** (зображення з графічних символів – блок-схеми);
- **псевдокоди** (напівформалізовані описи алгоритмів на умовній алгоритмічній мові, що включають як елементи мови програмування, так і фрази природної мови, загальноприйняті математичні позначки і т.і.);
- **програмна** (алгоритмічні мови – тексти на мовах програмування).

Словесним способом запису алгоритмів є опис послідовних етапів обробки даних. Алгоритм задається у довільному викладі на природній мові. Наприклад. Записати алгоритм знаходження найбільшого спільного дільника (НСД) двох натуральних чисел. Алгоритм може бути таким:

1) задати два числа;
2) якщо числа рівні, то взяти будь-яке з них за відповідь та зупинитися, інакше продовжити виконання алгоритму;
3) визначити більше з чисел;
4) замінити більше з чисел різницею більшого і меншого з чисел;
5) повторити дії з кроку 2.

Описаний алгоритм застосовний до будь-яких натуральних чисел і повинен призводити до розв’язання поставленої задачі. Переконаєтеся у цьому самостійно, визначивши за допомогою наведеного алгоритму найбільший спільний дільник чисел 125 та 75.

Словесний спосіб не має широкого розповсюдження по наступним причинам:

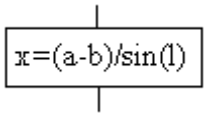
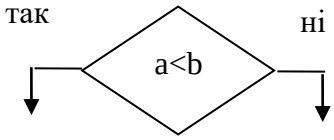
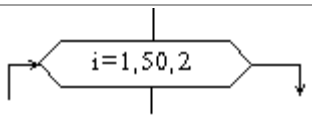
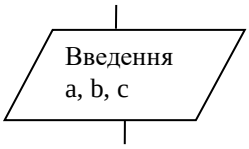
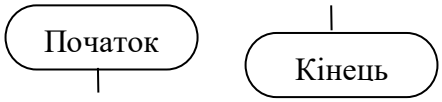
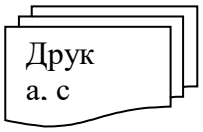
- такі описи строго не формалізуються;
- страждають багатослівністю записів;
- припускають неоднозначність тлумачення окремих розпоряджень.

Графічний спосіб подання алгоритмів є компактнішим та наочнішим у порівнянні зі словесним.

При графічному уявленні алгоритм відображається у вигляді послідовності зв’язаних між собою функціональних блоків, кожен з яких відповідає виконанню одного або декількох дій. Таке графічне уявлення називається схемою алгоритму або блок-схемою.

У блок-схемі кожному типу дій (введенню початкових даних, обчисленню значень виразів, перевірці умов, управлінню повторенням дій, закінченню обробки, тощо) відповідає геометрична фігура, яка представлена у вигляді блокового символу. Блокові символи з’єднуються лініями переходів, що

визначають послідовність виконання дій. У таблиці наведені символи, які найбільш часто вживаються.

Назва символу	Позначка та приклад заповнення	Пояснення
Слідування		Обчислювальна дія або послідовність дій
Розгалуження		Перевірка умов
Модифікація		Початок циклу
Зумовлений процес		Обчислення за підпрограмою, стандартною підпрограмою
Введення-виведення		Введення-виведення у загальному вигляді
Пуск-останов		Початок, кінець алгоритму, вхід у підпрограму та вихід з підпрограми
Документ		Виведення результатів на друк

Блок "слідування" застосовується для позначки дії або послідовності дій, що змінюють значення, форму уявлення або розміщення даних. Для покращення наочності схеми декількох окремих блоків обробки можна об’єднувати в один блок. Подання окремих операцій достатньо вільне.

Блок "розгалуження" використовується для позначки переходів управління за умовою. У кожному блоці "розгалуження" повинні бути вказані питання, умова або порівняння, які він визначає.

Блок "модифікація" використовується для організації циклічних конструкцій. (Слово модифікація означає видозміну, перетворення). Всередині блоку записується параметр циклу, для якого вказуються його початкове значення, гранична умова та крок зміни значення параметра для кожного повтору.

Блок "зумовлений процес" використовується для вказівки звернень до допоміжних алгоритмів, що існують автономно у вигляді деяких самостійних модулів, та для звернень до бібліотечних підпрограм.

Псевдокод є системою позначок та правил, яка призначена для одноманітного запису алгоритмів. Він займає проміжне місце між природною та формальною мовами. З одного боку, він близький до звичайної природної мови, тому алгоритми можуть на ньому записуватися та читатися як звичайний текст. З іншого боку, у псевдокоді використовуються деякі формальні конструкції та математична символіка, яка наближає запис алгоритму до загальноприйнятого математичного запису.

У псевдокоді не прийняті строгі синтаксичні правила для запису команд, як це має місце у формальних мовах, що полегшує запис алгоритму на стадії його проектування та дає можливість використовувати ширший набір команд, який розрахований на абстрактного виконавця. Проте, у псевдокоді зазвичай є деякі конструкції, які властиві формальним мовам, що полегшує перехід від запису на псевдокоді до запису алгоритму на формальній мові програмування. Зокрема, у псевдокоді, так само, як і у формальних мовах, є **службові слова**, сенс яких визначений раз та назавжди. Вони виділяються у друкарському тексті жирним шрифтом, а у рукописному тексті підкреслюються. Єдиного або формального визначення псевдокоду не існує, тому можливі різні псевдокоди, що відрізняються набором службових слів та основних (базових) конструкцій.

На жаль, не існує жодних стандартних псевдокодів, тому багато авторів книг з програмування та алгоритмізації змушені заздалегідь давати опис "свого" псевдокоду. У цьому посібнику не буде застосовуватись вказаний спосіб, а тільки обмежимося невеликим прикладом запису алгоритму на псевдокоді, який є близьким до шкільної алгоритмічної мови. Так, прикладом псевдокоду є шкільна алгоритмічна мова у російській нотації, що описана в підручнику А.Г. Кушніренко та ін. "Основи інформатики і обчислювальної техніки", 1991.

Наприклад, основними базовими управляючими структурами (слідування, розгалуження, цикл), за допомогою яких можна записати практично будь-який алгоритм, можуть бути структури псевдокоду, подані у таблиці російською мовою.

Назва структури	Псевдокод
Присвоєння, введення, вывод	змінна = 0, **Ввод**(змінна), **Вывод**(змінна)
Розгалуження	**Если** умова **то** <операторы> **иначе** <операторы>
Цикл ПОКИ	**Пока** условие **Нцикла** <операторы> **Кцикла**
Цикл ПОВТОРЮЙ ДО	**Повторяй** <операторы> **До** умова

Приклад алгоритму, який виводить максимальне з двох чисел (A і B), поки A ≠ B.

Алгоритм МаксЧисло
Начало
Повторяй
Ввод (A,B)
Если A > B **то**
Вывод ("Максимальное число = ", A)
Иначе
Вывод ("Максимальное число = ", B)
До A = B
Конец

Слідування – це алгоритмічні структури, що створюють лінійну послідовність операцій, які виконуються по черзі у порядку запису. Програмну реалізацію такої алгоритмічної структури називають лінійний блок. Лінійні блоки зазвичай призначені для розв'язку простих завдань, де не передбачені вибір з кількох можливих напрямів ходу програми або циклічне повторення операцій.

Розгалуження надають можливість альтернативного вибору при виконанні програми. У цьому випадку алгоритм може піти по одній з двох можливих гілок залежно від істиності умови, яка перевіряється.

Цикл являє собою послідовність кроків алгоритму, що багато разів повторюється, з використанням логічного блоку для перевірки умови його завершення.

Комп'ютер "розуміє" тільки алгоритми, які задані у вигляді двійкових машинних кодів. Проте цей "природний" для комп'ютерів спосіб завдання алгоритмів дуже складний для використання людиною. Тому в інформатиці застосовується низка спеціальних способів, мов завдання запису алгоритмів, які, по-перше, покликані забезпечити відповідність алгоритму всім необхідним вимогам, а по-друге, пристосовані для їх використання як людиною, так і – після спеціальної обробки – комп'ютером. Такі штучні мови, що використовуються для запису алгоритмів і забезпечують їм наявність усіх необхідних властивостей, називаються алгоритмічними мовами. Зокрема, можна згадати такі популярні мови, як Паскаль, C, C++, Ява (Java).

9.5. Основні види обчислювальних процесів

Якщо є алгоритм обробки інформації або виконання тих чи інших дій, то, в точності виконуючи всі вказівки алгоритму, можна отримати необхідний результат, не маючи ані найменшого уявлення про те, навіщо потрібно виконувати ті або інші дії. Важливо тільки дуже точно виконувати вказані в алгоритмі дії та дотримувати порядок їх виконання. Отже, виконання алгоритмів відносно нескладне. Саме тому процес виконання алгоритмів

вдається формалізувати та доручити його неживим механізмам, автоматичним верстатам, електронно-обчислювальним машинам, тощо. Розробка ж алгоритму, тобто плану виконання дій, є дуже складним творчим процесом, на який іноді витрачаються роки та десятиліття. Розробка алгоритмів розв'язання практичних завдань в різних областях людської діяльності здійснюється фахівцями у сфері обробки даних, яких називають проблемними програмістами.

При всьому різноманіїтті алгоритмів розв'язання завдань в них можна виділити три основні види обчислювальних процесів:

– лінійний;

– розгалужений;

– циклічний.

Лінійним називається такий обчислювальний процес, при якому всі етапи рішення задачі виконуються в природному порядку проходження запису цих етапів.

Розгалуженим називається такий обчислювальний процес, в якому вибір напряму обробки інформації залежить від початкових або проміжних даних (від результатів перевірки виконання будь-якої логічної умови).

Циклом називається багато разів повторювана ділянка обчислень. Обчислювальний процес, що містить два або декілька циклів, називається циклічним. За кількостью виконання цикли діляться на цикли з певною (заздалегідь заданою) кількістю повторів та цикли з невизначеною кількістю повторів. Кількість повторів останніх залежить від дотримання деякої умови, що задає необхідність виконання циклу. При цьому умова може перевірятися на початку циклу – тоді мова йде про цикл з передумовою, або в кінці – тоді це цикл з післяумовою.

Алгоритм, який записаний у спеціальній, "зрозумілій" комп'ютеру формі, прийнято називати ***програмою***, а оброблювану за цією програмою інформацію також записаною у "зрозумілій" для комп'ютера формі, прийнято називати ***даними***.

Контрольні запитання для самоперевірки

1. Що являє собою алгоритм?
2. Яких виконавців алгоритму Ви знаете?
3. Які властивості притаманні алгоритмам?
4. Що являє собою псевдокод?
5. Які основні види обчислювальних процесів Ви знаете?
6. Які існують види циклів?

10. Програмне забезпечення комп’ютера

Що являє собою комп'ютер, більшість людей сьогодні більш-менш уявляють. А ось, що являє собою програма? З цим питанням складніше. Більшість користувачів думають, що програма – це таке віконце з різними кнопочками на екрані монітора. Якщо клацати по цим кнопочкам мишкою, то будуть відбуватися якісь події.

Комп'ютер без програм – проста малокорисна залізяка. Програма без комп'ютера – всього лише набір інструкцій. І якщо ці інструкції нікому буде виконувати, то толку від програми буде мало. Завдання програміста – написати програму (скласти набір команд (інструкцій) для комп'ютера), яка буде змушувати комп'ютер виконувати потрібні користувачу дії. Надалі ми познайомимося з цим більш докладно, а зараз розберемося, що ж таке програмне забезпечення?

Програмне забезпечення – це сукупність програм, які дозволяють здійснити на комп’ютері автоматизовану обробку інформації. Програмне забезпечення ділиться на системне (загальне) та прикладне (спеціальне).

10.1. Системне програмне забезпечення

Системне програмне забезпечення призначене для функціонування та обслуговування комп’ютера, а також для автоматизації процесу створення нових програм. До системного програмного забезпечення належать: операційні системи та призначений для користувача інтерфейс; інструментальні програмні засоби; системи технічного обслуговування.

Операційна система – обов’язкова частина системного програмного забезпечення, яка здійснює ефективне функціонування персонального комп’ютера в різних режимах, організовує виконання програм та взаємодію користувача та зовнішніх пристроїв з ЕОМ. Найпопулярнішими ОС на даний час є родина операційних систем компанії Microsoft: Windows 2000 (або в літературі часто зустрічається позначення W2K, 2K – це, власне, і є 2000: «два кіло»), і особливо – Windows XP (від слова eXPerience – знання, життєвий досвід), Windows-7, Windows-8.

З інших операційних систем для ПК можна відзначити відносно популярну, хоча і складну у налаштуванні безкоштовну ОС Linux з родини Unix. На комп’ютерах Макінтош застосовується операційна система MACOS.

Інтерфейс користувача (сервісні програми) – це програмні надбудови операційної системи (оболонки та середовища), які призначені для спрощення спілкування користувача з операційною системою. Ці програми забезпечують інтерфейс користувача з комп'ютером, зберігають форму спілкування (діалог) користувача з операційною системою, але змінюють мову спілкування (зазвичай мова команд перетворюється у мову меню). Сервісні системи умовно

можна розділити на інтерфейсні системи, оболонки операційних систем та утиліти.

Інтерфейсні системи – це могутні сервісні системи, частіше за все графічного типу, які вдосконалені не тільки для потреб користувача, але і для програмного інтерфейсу операційних систем, зокрема таких, що реалізовують деякі процедури розподілу додаткових ресурсів.

Оболонки операційних систем надають користувачеві якісно новий інтерфейс у порівнянні з тим, що реалізується операційною системою, та роблять необов'язковим знання останньої. Прикладами тут можна назвати системи NC, FAR, TotalCommander.

Утиліти автоматизують виконання окремих типових процедур, що часто використовуються, реалізація яких зажадала б від користувача розробки спеціальних програм. Багато утиліт мають розвинений діалоговий інтерфейс з користувачем і наближається за рівнем спілкування до оболонок.

Інструментальні програмні засоби (системи програмування) – обов'язкова частина програмного забезпечення, за допомогою якої створюються програми. Інструментальні програмні засоби включають до свого складу засоби створення програм (текстові редактори); засоби перетворення програм у вигляд, який є придатним для виконання на комп'ютері (асемблери, компілятори, інтерпретатори, завантажувачі та редактори зв'язків або компоновники), засоби контролю та відлагодження програм (відладчики, турбодебагери).

Текстові редактори дозволяють зручно редагувати, формувати та об'єднувати тексти програм, а деякі – і контролювати синтаксис створюваних програм.

Програма, яка написана на алгоритмічній мові, повинна бути перетворена в об'єктний модуль, який записується на машинній мові (у двійкових кодах). Подібне перетворення виконується трансляторами (асемблером – з мови Assembler та компіляторами – з мов високого рівня). Для деяких алгоритмічних мов використовуються інтерпретатори, які не створюють об'єктний модуль, а при кожному черговому виконанні програми переводять кожен її окремий рядок або оператор на внутрішню машинну мову для наступного виконання. Виконання інтерпретуючих програм вимагає наявності інтерпретатора або середовища інтерпретації. Цей факт використовується в Internet для можливості виконання Internet-програм для різних операційних систем та платформ, для кожної з яких існує середовище інтерпретації, яке однаково працює в різних операційних системах. Об'єктний модуль обробляється завантажувачем – редактором зв'язків, який перетворює його у машинну програму для виконання.

Засоби відлагодження дозволяють виконувати трасування програм або певних їх ділянок (покрокове виконання з видачею інформації про результати виконання), проводити перевірку семантики програми та проміжних результатів у контрольних точках, модифікувати значення змінних у цих точках.

Системами технічного та сервісного обслуговування є програмні засоби контролю, діагностики та відновлення працездатності комп'ютера, дисків тощо.

Прикладне програмне забезпечення використовується для розв'язання потрібних для користувача завдань. Ключовим поняттям тут є пакет прикладних програм.

Пакет прикладних програм – це сукупність програм для розв'язання кола завдань з певної тематики або предмету. Розрізняють такі типи пакетів прикладних програм:

1. Загального призначення – орієнтовані на автоматизацію широкого кола завдань користувача (текстові процесори, табличні редактори, системи управління базами даних, графічні процесори, системи автоматизації проектування і т. і.).
2. Методо-орієнтовані – реалізація різноманітних економіко-математичних методів вирішення завдань (математичного програмування, мережного планування та управління, теорії масового обслуговування, математичної статистики і т. і.).
3. Проблемно-орієнтовані – спрямовані на вирішення певної задачі (проблеми) у конкретній області (банківські пакети, пакети бухгалтерського обліку, фінансового менеджменту і т. і.).

Є так звані ***інтегровані пакети прикладних програм***. Зазвичай це комплекс повністю сумісних між собою програм, який покликаний на всі випадки скласти для користувача єдине у своїй основі комфортне виробниче середовище. Наприклад, до складу офісного пакету Microsoft Office можуть входити: текстовий редактор Word, електронні таблиці Excel, СКБД Access, програма підготовки комп'ютерних презентацій Power Point.

До прикладного програмного забезпечення належать сервісні програмні засоби, які слугують для організації зручного робочого середовища користувача, а також для виконання допоміжних функцій (інформаційні менеджери, перекладачі і т.і.).

Серед прикладного програмного забезпечення нижче особливо хотілося б виділити деякі напрями.

10.2. Що являє собою операційна система?

Очевидно, що людина і комп'ютер говорять «різними мовами». Користувач не може одержати доступу до інформації, яка зберігається в електронному вигляді, безпосередньо, так само як і не може віддавати комп'ютеру будь-які команди без використання спеціальних засобів, які покликані пояснити машині, що і як вона повинна зробити. У багатьох випадках людина, що сидить за клавіатурою, навіть не замислюється над тим, наскільки складні процеси вона ініціює, наприклад, відкриваючи для редагування текстовий документ. Користувач виконує нескладну дію – подвійне клацання мишею, але при цьому комп'ютеру повідомляється безліч різноманітних даних. На який саме об'єкт було здійснено натиснення, який тип

цього об’єкту, де саме він зберігається, що з ним слід зробити, для того щоб на екрані відкрилося те вікно, яке містить необхідну інформацію, тощо. Усі ці відомості обробляються комп'ютером автоматично, користувач лише отримує на екрані готовий результат. Відповідає за обробку таких даних операційна система (ОС).

Таким чином, **операційна система** – це сукупність програмних та апаратних засобів, які виконують дві основні функції: надання користувачеві зручностей, які полегшують діалог із комп’ютером, та підвищення ефективності використання комп’ютера при раціональному управлінні його ресурсами. Тобто операційна система забезпечує зручний, дружній інтерфейс користувача з комп’ютером.

Операційна система відстежує всі операції, які виконує людина на ПК, та «підказує» комп’ютеру, як вона повинна реагувати на ті чи інші її події. На сьогодні найпоширенішими операційними системами є ОС родини Microsoft Windows. Можна сміливо сказати, що базові принципи взаємодії з операційними системами останніх поколінь фактично однакові незалежно від їх версій, оскільки всі ОС даного класу мають схожий призначений для користувача інтерфейс. Інтерфейс операційної системи – це все, що ви бачите на екрані монітора, завантаживши комп’ютер: позначки, кнопки, меню, вікна, панелі, курсори і т. і. Іншими словами, призначений для користувача інтерфейс – це сукупність доступних користувачеві засобів управління операційною системою комп’ютера.

До сучасних операційних систем пред’являються такі вимоги:

– сумісність – ОС повинна включати засоби для виконання додатків (applications), що підготовлені для інших ОС;

– переносимість або мобільність – забезпечення можливості перенесення ОС з однієї апаратної платформи на іншу;

– надійність і відмовостійкість – припускає захист ОС від внутрішніх та зовнішніх помилок, збоїв та відмов;

– безпека – ОС повинна містити засоби захисту ресурсів одних користувачів від інших;

– розширюваність – ОС повинна забезпечувати зручність внесення подальших змін та доповнень;

– продуктивність – система повинна бути достатню швидкісною.

10.3. Програми для роботи з текстами

Текстові редактори (або, як іноді кажуть, **текст-процесори**) – це програми для набору, редагування та підготовки до друку будь-яких документів від маленької нотатки або договору на одну сторіночку до багатотомної енциклопедії. Найвідоміший та поширеніший текст-процесор – це Microsoft Word. Є й інші, але у нас ними мало хто користується. Хіба що та невелика група людей, яка не бажає працювати на крадених піратських програмах, а

купувати Microsoft Office (до складу якого і входить Word) їм не по кишені. Ці педантичні люди цілком можуть скористатися безкоштовним пакетом Open Office, в якому є схожий на Word текстовий редактор.

Новітні версії текстових редакторів за своєю можливістю стоять вже досить близько до видавничих систем (програм верстки), такі як Quark- XPress, PageMaker, FrameMaker або Corel Ventura. Видавничі системи ділять на журнальні та книжні. Перші орієнтовані на видання зі складним дизайном, з великою кількістю окремих текстових вікон, безліччю ілюстрацій (у тому числі і кольорових) та іншої графіки на кожній смузі, з багатоколонковим друком. Другі орієнтовані на лінійне розташування тексту, який складається з багатьох розділів, забезпечені розвиненим апаратом для створення виносок, посилань, змісту, абеткових покажчиків та інших фокусів, які необхідні при книжковій верстці. У гарній видавничій системі можна робити як книги, так і журнали.

Завдання таких програм – підготувати макет книги, газети або журналу, що відповідає найсуворішим поліграфічним вимогам. Макетом (або оригінал-макетом) називають виконаний на папері (а частіше – на спеціальній плівці) примірник видання з усіма ілюстраціями та текстом, який набраний відповідним шрифтом, і який вже іде безпосередньо у друкарню.

Є **програми-перекладачі** – з англійської, німецької, французької, української та інших мов на російську та навпаки (PROMT, Socrat, Плай). Звичайно, самі, без будь-якої нашої участі, машинні перекладачі здатні “створити” такі переклади, в яких розібратися буде важче, ніж в іншомовному оригіналі. Але це не біда: перекладачі можна дуже тонко та точно налаштувати на тематику вашої роботи. А якщо ви придбаєте разом з перекладачем додатковий спеціалізований словник із своєї області діяльності, то перекладач зможе цілком реально допомогти вам – як у ситуації, коли ви не знаєте іноземних мов, так і у ситуації, коли знаєте мову добре, але хочете заощадити час при перекладі текстів великого обсягу.

Електронні словники (Lingvo, Polyglossum, Контекст) не претендують на штучний інтелект програм-перекладачів. Але й не роблять дурних помилок. Мультимедійні версії словників дають не тільки письмовий переклад введених вами слів, але й часто усний – вміють промовляти вголос слова та цілі вирази**.**

Окремо, але у зв’язку з текстовими програмами варто згадати і про програми **розпізнавання образів** або **OCR-програми** (від Optical Character Recognition – «оптичне розпізнавання символів»). Завдяки ним можна використовувати сканер для введення не тільки картинок, але і текстів. Ввести текст як картинку просто, для цього годиться будь-яка програма (вона може приходити в комплекті зі сканером) сканера та більшість графічних програм («малювалок», як їх фамільярно називають). Але картинку не можна редагувати як текст (в який нам дозволяється вставити слово або викреслити два), а крім того, вона займає в десятки разів більше місця пам’яті на диску. OCR-програми вміють на картинці розпізнавати букви та прочитувати із зображення текст.

Існують також програми такого роду, що пристойно працюють з російськими текстами, такі як FineReader або CuneiForm. Клас такої програми визначається відсотком помилок при роботі з оригіналами невисокої (точніше, реальної, життєвої) якості. Якщо помилок, скажімо, 20% від загального числа символів, то працювати з таким текстом практично не можливо: у кожному слові помилка – швидше набрати самому!

Сучасні OCR-программи не просто прочитують текст, але й визначають зображення – нормальне, курсивне, напівжирне або підкреслене, – яким був набраний той чи інший шматок, і відтворюють ці шрифтові виділення у вихідному файлі, розуміють багатоколонкове розташування тексту.

З розпізнаванням написаних від руки документів справа поки йде не дуже успішно, так само як і з **розпізнаванням мови** – комерційних програм для серйозної роботи з російською (українською) мовою поки що не видно. Але розробки у цьому напрямі активно ведуться.

Принаймні, англомовні програмки вже існують, і цілком пристойної якості, як, наприклад, ViaVoice фірми IBM. Правда, вони налаштовуються під вашу вимову не одразу, так що на повну проектну якість виходять через якийсь час. А може, воно і краще, що програм таких мало або навіть зовсім немає? Уявіть собі, як це буде, якщо кожен студент у комп'ютерному класі або співробітник в офісі, замість того щоб тихенько натискати по клавішах, почне розмовляти зі своїм комп'ютером на повний голос!

10.4. Графічні редактори

Графічних редакторів, «рисовалок», існує дуже багато. Є могутні професійні програми, як, наприклад, Adobe Photoshop, Adobe Illustrator, 3D Studio, CORELDRAW! – як правило, дуже великі, з масою допоміжних програм та усіляких додаткових ефектів (за це їх ще називають графічними пакетами). Є простіші і компактніші, наприклад, PaintShop або Paint із стандартного складу Windows.

Графічні редактори діляться на два головні типи – ***растрові*** (або бітмепові, від слова bitmap – «бітова карта») та ***векторні*** за відповідними типами файлів.

Растрові малюють зображення крапками, тобто для кожної крапки картинки окремо задається її колір. Тут можна назвати, наприклад, Paint, Adobe Photoshop, PaintShop і PaintShop Pro, Corel Photo-Paint.

Векторні ж (CORELDRAW!, Adobe Illustrator) малюють одразу цілу лінію – відрізок прямої або якусь стандартну криву, а складні форми представляються як сукупність таких простих кривих та прямолінійних відрізків.

Це абсолютно різні способи подання зображень. Переклад файлів з векторної форми у растрову відбувається елементарно. А ось зворотний – з растрової форми у векторну – є досить складною проблемою.

У кожного з цих типів існують свої переваги та свої недоліки, свої області застосування. Так, векторні "малювалки" дозволяють проробляти дуже складні

трансформації форми малюнка, стиснення та збільшення, будь-які зміни розміру, перетворення контурів. У них можливі повороти будь-якого елементу малюнка на будь-які кути без спотворення форми. І, навпаки, спотворення будь-яких видів: перетворення квадратного малюнка на круглий, циліндричний, криволінійний – в який завгодно. У них дуже легко поєднувати зображення з різного роду написами, які довільним чином розміщені (по дугах або інших кривих, під будь-якими кутами). Але для обробки фотографій вони непридатні. Використовують їх при виготовленні всіх видів емблем та товарних знаків, при книжковій, журнальній та рекламній верстці будь-якої складності.

Растрові "малювалки" використовують не тільки тоді, коли хочеться помалювати, але й тоді, коли треба обробити введені у комп'ютер фотографії, які зроблені цифровою камерою, або скановане зображення (картини, малюнки, звичайні фотографії та слайди). Не дарма у назвах їх часто присутнє слово Photo (фотографія). Основний наголос робиться на ретушування зображень, перетворення кольорів та відтінків, контрасту, яскравості, чіткості, на різного роду розмивання і затуманення, гри зі світлотінню. Але з формою об'єктів вони працюють погано, до того ж файли бітмеп-форматів займають помітно більші розміри у пам'яті, ніж векторні.

Існують і редактори змішаного типу. В сучасних растрових редакторах використовуються елементи векторної графіки, і навпаки, у векторну "малювалку", що вирішується як окремий фрагмент, вставляти растрову графіку.

Деякі графічні пакети включають **програми комп'ютерної анімації**, але існують і незалежні анімаційні програми. Вони використовуються не тільки для створення мультфільмів, але і для оформлення інтернетівських сторіночок усіляким миготливим та підморгуючим ілюстративним матеріалом та рекламою. З серйозних пакетів у першу чергу згадуються AutoDesk Animator, Macromedia Flash і Macromedia Shockware. Простіше у використанні та вивченні такі програми, як Ulead GifAnimator, Adobe ImageReady або Corel R.A.V.E.

10.5. Музичні та звукові редактори

Існує два основних способи комп'ютерного подання музики. Якоюсь мірою вони нагадують два способи подання графіки – векторний (якась математична заготівка) та растровий (переклад справжнього зображення (або звуку) у цифровий вигляд).

У **нотному файлі**, або **MIDI-файлі** (від Musical Instrument Digital Interface – «цифровий інтерфейс музичних інструментів»; це загальноприйнятий стандарт для зв'язку синтезаторів та інших музичних інструментів з комп'ютером або між собою) на декількох незалежних смужках записані одні тільки нотні партії. А вже яким інструментом виконуватиметься та чи інша партія – скрипкою або бубном, – ви задасте потім, виходячи з набору звуків

(**семплов** – від sample, «зразків», або **патчів** – від patch, «шматочок», «латочка»), який надає ваша звукова карта або синтезатор.

Вводиться музика у комп'ютер через MIDI-вхід звукової карти – ви щось граєте на своєму синтезаторі або на спеціальній MIDI-клавіатурі (яка набагато дешевше), а спеціальна програма-редактор записує всі зіграні вами нотки. Втім, можна саджати ноти прямо на нотні лінійки – мишкою, просто це не так зручно. Потім пишете іншу партію, третю.

Для запису та обробки нот використовуються програми, які прийнято називати **секвенсорами** (Cakewalk, Sonar, Cubase, Logic Audio і т.і.). Секвенсор дає і аматору, і професіоналу необмежену владу над нотами: можливості редагування як окремих нот, так і їх груп, для зміни їх висоти, гучності, швидкості відтворення (без зміни тональності), для їх підлаштування під тактову сітку, тощо. MIDI-файлы мають невеликі розміри, у кожного секвенсора – свій власний формат таких файлів, але є й стандартний, хоча досить бідний за своїми можливостями формат (розширення .mid).

Якщо ж ви оцифровуєте (завантажуєте у комп'ютер) звук з мікрофону, лінійного входу або з компакт-диску, то отримуєте файл іншого типу – **звуковий**, або **хвилевий** (wave). Основний стандарт тут – wave-файли (розширення .wav).

Такі файли мають дуже серйозні розміри, для їх обробки потрібно багато оперативної пам'яті та дискового простору, потрібна хороша швидкодія процесора та жорсткого диску. Обробляються такі файли **звуковими редакторами** (Sound Forge, Cool Edit, WaveLab), простий варіант такого редактора – програма Звукозапис (Sound Recorder) у стандартній комплектації Windows.

Звуковий редактор є електронним аналогом магнітофона, посиленого можливостями комп'ютера. Розділити звуки або партії з однієї смужки для їх роздільного редагування тут практично неможливо (спробуйте на магнітофонній стрічці відокремити гітару від барабанів!). Головне в таких редакторах – монтаж та звукові ефекти. Можна відрізати шматок фонограми тут і вставити туди, змінити гучність будь-якої ділянки запису (наприклад, зробити згасання в кінці пісні).

Звукові редактори теж можуть бути **багатосмужечними (мультітрековими)**, що дозволяє створювати в них композиції з багатьох інструментів. Яскравий приклад такого роду – програми Vegas Audio або Nuendo.

Один з основних недоліків комп'ютерної музики – невисока якість звуків (семплів) у звукових картах та недорогих синтезаторах. Убогі «пластикові» звуки – «як би барабани», «як би труби», «як би гітари» одразу відвертають меломана від комп'ютерної музики. Проте існує особливий різновид музичних програм, який дозволяє користуватися високоякісними звуками навіть на дешевому устаткуванні. Це – семплери та програмні синтезатори.

Програма-семплер дозволяє програвати MIDI-файли з використанням високоякісних та навіть професійних семплів. Як достатньо свіжі приклади

назвемо тут програми GigaStudio, Halion, Kontakt. Маючи такий семплер та достатньо обширну бібліотеку звуків, можна отримувати результат дуже хорошої якості.

Програмний синтезатор (або софтвер-синтезатор), такий як Rebirth, Reaktor або DreamStation, за основу бере не живі звуки, а синтезовані безпосередньо у комп'ютері та оброблені цифровими спецефектами, завдяки чому можна отримувати абсолютно фантастичне звучання.

Програми-мікшери дозволяють по ходу справи регулювати гучність і стереобаланс по кожному звуковому каналі (окремо MIDI, Wave, CD, лінійний вхід та вихід, мікрофон). Звукові карти можуть комплектуватися власними програмними мікшерами. А якщо у вашого саундбластера такого немає, ви завжди зможете скористатися стандартним "віндоузівським" мікшером, який управляє як відтворенням, так і записом у комп'ютері.

Програми-конвертери дозволять переводити звук з одного формату в інший. Сьогодні особливий інтерес викликають конвертери MP3, які дозволяють створювати з аудіосмужок звичайних компакт-дисків звукові файли популярного формату MP3, які не сильно поступаються за якістю звуку, а за розміром бувають у десять, та сто разів менше.

Сучасні аудіоредактори, такі як Sound Forge, і навіть деякі програми для запису компакт-дисків забезпечуються вбудованими модулями для перетворення музики у формат MP3. Але одна з самих якісних програм такого роду, – це дуже невелика (і до того ж, абсолютно безкоштовна!) програмка на ім'я Lame.

10.6. Бази даних, електронні таблиці, бухгалтерські програми

З погляду користувача, **база даних** (data base) – це набір як би бланків, точніше, електронних карток, в яких є постійні елементи – заголовки, що підлягають заповненню областей (полів), та змінні (вміст цих полів). Наприклад, набір бланків з обліку кадрів. У кожному з них поля одні і ті ж самі (прізвище, ім'я, по батькові, посада, оклад, стаж роботи, дата та рік народження, номер паспорта і т. і.), а їх вміст змінюється.

В одній базі даних можна мати «картки» (форми) різних типів – наприклад, картки обліку співробітників, заробітної платні за місяцями або по підрозділами, списки завдань для цих співробітників, реквізити контрактів, картки товару з цінами, артикулами, штрих-кодами тощо.

Можна зберігати у базі і графічні файли (фотографії співробітників, діаграми, малюнки) та навіть звукові. Все це організовано в єдину ієрархічну структуру (є спеціальні базові таблиці, а є "дочірні", куди дані приходять з базових). Базу оснащено також різного роду сортуваннями, системами пошуку потрібних даних та їх обробки як по всій базі (підсумувати ціни, скласти список телефонів та адрес за фірмами тощо), так і по будь-якій її частині.

Бази даних можна створювати самому. Процес цей розпадається на дві частини: спершу створюєте заготівку для бази, вказавши програмі, де на екрані має

розташовуватись те чи інше поле, якої довжини буде це поле (з таким запасом, щоб вистачило на найгроміздкіший випадок), які дані сюди вводитимуться (числа, тексти, дата/час, грошові суми і т. і.). А на другому етапі вже заповнюєте відповідні поля іменами, прізвищами та окладами або назвами фірм, номерами контрактів, характеристиками товару тощо.

Створення (програмування) всіх цих чарівних штучок і робота з ними здійснюється за допомогою **систем керуваіння базами даних** – СКБД (у англомовній літературі це називається – DBMS), таких як Access з комплекту Microsoft Office, MySQL, Visual FoxPro, Clarion, Delphi, dBase.

Іноді бази даних формуються програмами автоматично – по мірі того, як ви ними користуєтеся, а ви можете про це навіть не підозрювати. Наприклад, поштова програма зберігає листи у вигляді бази даних, що дозволить вам швидко відшукати потрібний лист за його темою, за адресою відправника або ключовими словами. Програвач Windows Media зберігає у вигляді бази даних усі ваші музичні записи.

До баз даних впритул наближаються **електронні таблиці**. Назвати тут можна, перш за все, Excel з комплекту Microsoft Office, а також Lotus 1-2-3, Quattro Pro. За виглядом це дійсно таблички типу тих, що можна створити у будь-якому пристойному текстовому редакторі, але доповнені можливістю вести досить складні обчислення. Ви самі задаєте спосіб обчислення значень у тому чи іншому стовпчику (наприклад, перемножуванням даних з двох інших стовпців та взяттям від результату відсотка, який заданий у третьому). Міняючи дані у таблиці, міняєте одразу ж і результат. На основі будь-яких рядків та стовпчиків ви зможете побудувати красивий графік або діаграму.

Окремий широкий клас програмних продуктів – фінансові, банківські, бухгалтерські програми, програми для ведення офісної документації, програми планування фінансової, комерційної та виробничої діяльності, на зразок відомої серії ділових програм фірми 1С.

Люди, які планують роботи за серйозним проектом, ведуть облік кадрів та загальну бухгалтерію, товарно-матеріальний облік та облік основних засобів, оформляють прибуткові та витратні ордери, розраховують зарплату, податки та відрахування до різних фондів, без сумніву, повинні освоїти бухгалтерські і ділові програми. Звичайно, доведеться дізнатися багато нового і витратити на це немало сил, але коли операцію, яка раніше займала у вас тиждень напруженої праці, машина виконає за п’ятнадцять хвилин, ви зрозумієте, що сили витрачені не дарма.

10.7. Телекомунікаційні програми

Комунікаційні програми потрібні для того, щоб нам з вами був сенс купувати модем, факс-модем або мережну карту. Найпоширенішими програмами цього типу є **браузери** – програми для прогулянок по Internety. За допомогою браузера ви зможете почитати свіжі новини, подивитися статті в комп’ютерній періодиці, подивитися на персональні сторіночки знаменитостей, завантажити новітні версії

програм, знайти, нарешті, драйвер для своєї відеокарти тощо. Сьогодні найпопулярніший браузер Microsoft Internet Explorer. Є багато шанувальників і у браузера Opera, а також Mozilla та Netscape.

Для ведення електронного листування призначені **поштові програми** (**мейлери**, від англійського «e-mail»). Поштова програма може складати єдиний комплект з браузером (так популярна поштова програма Outlook Express входить у комплект браузера Internet Explorer), а у комплект Опери входить програма M2. Але є і незалежні «листоноші» (The Bat!, Eudora, Pegasus).

Для безпосереднього спілкування через мережу шляхом взаємного листування існують особливі програми – **веб-пейджери**: ICQ, mIRC, Windows Messenger, Miranda, Trillian тощо. Є навіть програми для розмови через Internet «голосом» – за допомогою мікрофону та динаміків (Skype).

Всесвітню Мережу не варто плутати з локальними, у тому числі і домашніми мережами. **Локальна мережа** – це об'єднання всіх комп'ютерів, скажімо, інституту, банку, універмагу або міністерства. Два комп'ютери в одній квартирі, сполучених кабелем, – це теж локальна мережа.

Для цього в комп'ютери ставлять мережні плати (мережні адаптери) і з'єднують їх між собою кабелем. Для розгалужених мереж необхідні також спеціальні пристрої – **концентратори**, **хаби** (hubs) або **світчі** (switchs).

Але одного тільки фізичного під'єднування тут мало. Треба ще якимсь чином організувати спілкування абонентів один з одним. Тобто потрібне якесь програмне середовище. Windows 9x, NT і XP оснащені багатьма необхідними для цього засобами і навіть мережними іграми, в які можуть грати люди, що сидять на різних поверхах або навіть в різних будівлях, які старанно роблять вигляд, що працюють.

До комунікаційних програм належать також **факсні програми** (WinFax, VentaFax), що дозволяють приймати повідомлення з телефаксу безпосередньо у комп'ютер та розсилати на факсні апарати повідомлення, що створені в комп'ютері.

Є багато невеликих, але дуже корисних комунікаційних утиліт. Наприклад, програми для дозвону до Internet за допомогою модему («звонилки»), програми для прискорення перекачування даних («гойдалки»), програми для роботи із спеціальними файловими серверами Internet тощо.

10.8. Ігри

Грати у комп'ютерні ігри – заняття захоплююче. Деякі люди, що іменують себе **геймерами**, здатні сидіти годинами, прагнучи правильно укласти цеглини в «Тетрісе», тижнями бродити по замках, повних привидів, місяцями будувати міста на невідомих планетах, роками палити з усіх видів зброї. Є ігри, в які грають цілими командами, – у локальній мережі або на якомусь сервері в Internet.

Ігри типу «стріляй-біжи» або «бий все, що рухається» – улюблені ігри молодших школярів та деяких старших. Варіацій множина – від найпростіших

та невитіюватих, на зразок Короля-Лева або Аладдіна, до самих наворочених, з об'ємною (**3D**) графікою. Є ігри з простою стріляниною (пістолет, автомат), а є з фантастичною (бластери), є – з бійками («каратешки» – Mortal Combat).

Звичайні ігри цього типу називають **аркадами**, а тривимірні – **3D-Action** («три-де-экшн»). Замість легковажного слова «стрілялка» геймери вживають незрозуміле для необізнаних, а тому крутіше слово **шутер** (shooter). Втім, означає воно в точності те ж саме – "стрілялка".

Ділять "стрілялки" і ще за одним принципом: хто в них головний герой. Якщо герой ви та ігровий світ ви бачите очима свого персонажа, то це називається **FPS** (first person shooter – «стрілялка від першої особи»). Перед вами весь час руки цього персонажа, стискаючі бластер, а ворогів і чудовиськів ви спостерігаєте в основному через проріз прицілу.

Ігри від третьої особи називаються **TPS** (third person shooter). Тут головного гравця, героя показують вам з боку – позаду, збоку, зверху.

Ігри-тренажери (симулятори): різного роду гонки, військові і космічні ігри. Зазвичай в них гравець як би сидить у кабіні літака або автомобіля з екранами, важелями та кнопками. Звичайно, їздити в таких автомобілях і літати в таких літаках простіше, ніж у справжніх.

Більше за все понароблялося ігор-автогонок (The Need For Speed, Driver, Grand Theft Auto) та літакових симуляторів (Microsoft Flight Simulator F19, F29, F117), але є і вертолітні бої (Comanche), і навіть космічні кораблі і роботи (Mechwarrior, зоряні війни X-Wing, Rebel Assault, Wing Commander).

Спортивні симулятори (NBA, FIFA, NHL, Box) – імітації спортивних змагань з футболу, баскетболу, хокею. Правда, управління таким складним об'єктом, як боксуюча або граюча у футбол людина, поки не дуже вдається програмістам. Та й користуватися для цього мишкою, клавіатурою (частіше на мові геймеров – «клавой») або джойстиком не дуже-то зручно. Тому гри цього типу великим попитом не користуються.

У **стратегічних іграх (стратегіях)** ви будуєте міста, країни і навіть цілі планети, управляючи їх розвитком, будуючи будинки та дороги, обкладаючи жителів і підприємства податками, укладаючи союзи і оголошуючи війни. Головна мета – здобуття якихось найважливіших ресурсів – енергії, «спайса», територій, води, грошей – чого завгодно. У таких іграх ви самі не берете участі у діяльності підлеглих вам територій або планет. Працюють інші, а ви їх лідер і мозковий центр – король, президент, генерал, верховний маг.

З погляду правил здійснення ходів, стратегії ділять на покрокові (TBS – turn-based strategy), де ходи здійснюються гравцями строго по черзі, як в шахах, та на стратегії реального часу (RTS – real-time strategy), де кожен гравець робить хід тоді, коли вважає за потрібне. Найвідоміші стратегії: Warcraft, Starcraft, Age of Empires, Command & Conquer, Dune II.

Втім, є і такий різновид стратегій, в яких ви і самі трішки бігаєте і стріляєте. Тобто частково це "стрілялка", частково стратегія. Геймери називають це FPS (First Person Strategy – стратегія від першої особи).

Наприклад, це може бути симулятор бойових роботів, в якому ви не тільки головнокомандувач, але і боєць (Urban Assault, Battlezone, «Паркан»).

Якщо на такому фантастичному світі ви не верховний володар і навіть не генерал, а рядовий учасник – воїн, космічний торговець, то це вже називають ролевою грою або RPG (role playing game). А якщо окрім вас і комп'ютера у цю ж гру на якомусь інтернетівському сервері грає ще тисяча чоловік, то такі забави називають вже багатокористувацькими **ролевими іграми**: MUG (multi-user game) або **MMORPG** (Massively Multiplayer Online RPG).

Ігри-пригоди – звичайно це хитромудрі красиві ігри-казки, ужастики-страшилки (Alone in The Dark), пригоди (Indiana Jones), фантастика. У цих ігор є одна спільна риса: ви часто не знаєте цілі гри і тих засобів, якими її слід добиватися. Ви бродите по світу, повному дивних або цілком звичайних предметів, чиє призначення вам невідоме, і намагаєтеся зрозуміти, що до чого. За це їх і звуть "**броділками**", а також **квестами** (від англійського quest – пошуки). Найвідоміші квести – The Legend Of Kyrandia, The Neverhood, The Mist, популярні також розробки – «Штирліц», «Петька і Василь Іванович», «Операція Пластилін».

«**Настільні**», **логічні ігри** та **головоломки** віддають перевагу тим, для кого гра не основне заняття в житті, а всього лише короткий і необтяжливий відпочинок в офісі – спосіб провести декілька хвилин, поки шеф не повернувся. У офісах дуже люблять пасьянси (Solitaire, Косинка і ін.), люблять Тетріс і деякі його різновиди. Для серйознішого проведення дозвілля існують і серйозніші ігри – шахи (Chess, Chessmaster і ін).

Якщо гра «зламана» нашими комп'ютерними «Кулібіними», то вхід у неї не викликає складнощів. Фірмові ж варіанти при запуску пропонують вам ввести пароль – якесь слово або число, яке ви «можете прочитати у Вашій Інструкції для Користувача на певній сторінці», і якщо ви цього не зробите, то гра на тому і закінчиться. Підібрати потрібне поєднання букв та цифр навмання зазвичай не вдається. Треба дізнаватися у тих, хто дав вам іграшку, як вони в неї входили. На піратських дисках код цей пишуть або на обкладинці, або у спеціальному файлі на цьому ж диску. Файл може називатися sernum, file_id.diz, info.txt або що-небудь з розширенням .nfo.

Контрольні запитання для самоперевірки

1. Які програми призначені для спрощення спілкування користувача з ОС?
2. Які типи графічних редакторів Ви знаете?
3. Основні недолікі комп'ютерної музики?
4. Що являє собою семпл?
5. Яку інформацію можна зберігати у базах даних?
6. Які телекомунікаційні програми набули найбільшого поширення?

11. Програмування та мови програмування

11.1. Що являє собою програмування?

Природною мовою, яку "розуміє" комп'ютер, є **машинна мова**, тобто мова двійкових знаків: нулів та одиниць. Процес опису конкретного алгоритму на мові машинних команд називається **програмуванням у машинних кодах**. Фактично майже ніхто не програмує тепер за допомогою апаратно реалізованих команд. Ця обставина має декілька причин:

1. Для виконання цієї роботи програміст повинен знати коди всіх машинних операцій та їх форматів, призначення та особливості вживання кожної із них, а також пам'ятати адреси конкретних елементів пам'яті, що зберігають ті чи інші операції.

2. Дуже велика відстань між людськими поняттями (рівень вирішуваної проблеми) та примітивним характером команд.

3. Труднощі розуміння довгих програм.

4. Збільшення можливості помилок людини у програмах та трудомісткість процесів їх пошуку та виправлення.

5. Тісний взаємозв'язок між програмою та комп'ютером, на якому вона виконується, що робить неможливим перехід на інший комп'ютер.

Деякі з цих проблем можуть бути розв'язані при застосуванні **системи кодування**, яка дозволяє програмістові записувати команди символічно у змістовних позначеннях. Такі системи кодування називають ***мовою асемблювання*** (замість "мови" команд машини, або машинної мови).

Спеціальна програма, яка називається асемблером (assembler – збирач), яка являє собою символічну мову машинних команд і якою оснащені майже всі ЕОМ, бере на собе перекладання рядків мови асемблера у відповідні команди.

Проте це не вирішує основних згаданих вище проблем. Саме тому дуже рано (з початку 50-х років) були створені мови програмування. Мовами програмування є системи опису програм, достатньо близькі до людської мови, щоб програми можна було легко побудувати, зрозуміти та модифікувати, але у той же час достатньо строго визначені, щоб ті ж програми могли бути врешті виконані комп'ютером.

Окрім цього, програми, які написані на такій мові, були набором вже не окремих машинних команд, а крупніших елементів, названих операторами даної мови. На мові високого рівня вихідна програма складається з низки операторів.

Перші мови програмування були орієнтовані на опис математичних алгоритмів. Тому їх ще називали **алгоритмічними мовами**.

Всі алгоритмічні мови відрізняються, перш за все, наочною формою реалізації алгоритму у вигляді програми, оскільки використовують звичну математичну символіку і обмежений набір зрозумілих ключових слів. Основною перевагою алгоритмічних мов є їх універсальність, тобто незалежність від конкретного типу машин (машинно-незалежні мови).

Щоб виконати програму **P**, яка написана на мові програмування **L**, треба зробити її зрозумілою для машини, оскільки машина "розуміє" лише свою машинну мову. Програма на алгоритмічній мові перед виконанням перекладається на цю мову за допомогою спеціальної програми **транслятора**, назва якої походить від англійського слова translator (перекладач). У програмі-трансляторі "закладені" всі правила алгоритмічної мови і способи перетворення різних її конструкцій на машинну мову. Для цього існують два способи (методи) трансляції.

При трансляції виконується переклад програми, яка зрозуміла людині, на мову, яка зрозуміла комп'ютеру. Якщо мета трансляції – перетворення усього початкового тексту на внутрішню мову комп'ютера (тобто отримання деякого нового коду) і тільки, то така трансляція називається також **компіляцією**. Початковий текст називається також початковою програмою або початковим модулем, а результат компіляції – об'єктним кодом або об'єктним модулем. Якщо ж трансляції піддаються окремі оператори початкових текстів і при цьому отримані коди одразу перетворюються у спеціальне внутрішнє подпння та виконуються, така трансляція називається **інтерпретацією**. Оскільки трансляція виконується спеціальними програмними засобами, то вони мають назву компілятора або інтерпретатора, відповідно.

1. У методі **інтерпретації** відповідність між мовою програмування та машинною мовою встановлюється завдяки спеціальній програмі, так званім **інтерпретатором,** що оперує з програмою **P** та даними **D**, які треба передати програмі **P**. Інтерпретатор декодує **P** та по мірі цього декодування виконує відповідні команди, застосовуючи їх до елементів з **D**. Зауважимо, що інтерпретатор сам по собі є програмою, яка може бути написана на машинній мові.

2. У методі **компіляції** виконання програми включає два етапа (фази): перший – це переклад програми **P** в еквівалентну програму **P'**, яка виражена у машинній мові. Цей переклад виконується спеціальною програмою, яка має назву **компілятор**. Другий етап – це виконання результуючої програми **P'** з даними **D**.

Природно, що інтерпретовані коди виконуються повільніше, ніж скомпільовані, оскільки переклад інструкцій з однієї форми в іншу зазвичай займає у декілька разів більше часу, ніж виконання отриманих готових інструкцій. Але інтерпретація забезпечує велику гнучкість у порівнянні з компіляцією, та у ряді випадків без неї не обійтися.

Інтерпретація найчастіше використовується у простих мовах, де потрібна нескладна трансляція (Бейсик), або там, де компіляція дуже складна або навіть неможлива (мова ЛІСП). Часто використовують обидва ці способи спільно: інтерпретатор – для відлагодження, а компілятор – для трансляції відлагодженої програми.

Насьогодні у світі існує декілька сотень мов програмування. Ця кількість мов знаходиться у приголомшуючій суперечності з тим фактом, що більшість

програмістів застосовують лише декілька мов, а в деяких обчислювальних центрах програмують тільки на одній мові.

З того, що вже нами “пізнано”, повинно бути зрозумілим, що інформатика є дуже могутнім інструментом, яка кидає виклик нашому інтелекту. Велики труднощі цієї дисципліни полягають у тому, що обчислювальна машина дає засіб не боятися завдань таких розмірів, які перевершують все, що раніше розглядалося, тобто такої складності, до засвоєння якої людський розум погано підготовлений. Потреба у такому точному описі крок за кроком послідовності перетворень інформації з такою строгістю, яка не потрібна ні при якому виді людської діяльності (навіть у математиці), викликала у видатних програмістів думки про труднощі програмування, які вимагають не стільки засвоєння якоїсь конкретної мови, скільки нового способу мислення.

11.2. Мови програмування

Програмним забезпеченням (ПЗ) називають сукупність програмних засобів, необхідних при роботі на комп'ютері. Частиною ПЗ є мова програмування.

Мов програмування, як і людських мов, створено багато. Навіщо? Причина – у різноманітності потреб програмістів, на різних рівнях їх кваліфікації та багато у чому іншому. Так, початківцям навряд чи варто пропонувати Асемблер, а професіоналові не потрібний Лого (мова, розрахована на дітей, дозволяє просто та цікаво малювати картинки і програмувати прості ігри). Часто різні мови орієнтовані на різні наукові області. Програмістам поки що не вдалося створити мову, яка задовольняє всіх, та і невідомо, чи можливо та чи треба взагалі її створювати.

Деякі мови програмування створювалися для конкретних цілей. Інші є універсальними і застосовуються для розробки програм, призначених для роботи з різними операційними системами. Однак, незважаючи на відмінності, багато мов програмування дуже схожі, тобто не мають (або майже не мають) принципових відмінностей. А з тих, які принципові відмінності мають, можна виділити три основні класи:

1. Машинні коди.
2. Мови низького рівня.
3. Мови високого рівня.

На зорі комп'ютерної ери всі програми писалися в машинних кодах – це коли в пам'ять обчислювальної машини записуються безпосередньо числа. Кожному числу відповідає певна команда. Процесор читає це число і виконує відповідну команду.

У чому недолік команд машинної мови? У тому, що дії, які викликаються цими командами, дуже малі. Тому програма виконання навіть дуже простого завдання складатиметься з великої кількості команд. Це все одно, що будувати будинок з цегли, а не з крупних блоків.

Оскільки комп'ютери ставали все більш досконалими, а програми – все більш складними, то довелося людям замислитися про способи спілкування програміста і машини. Наступним кроком був перехід на мову низького рівня або мову мнемонічних команд. У мові низького рівня тій чи іншій команді відповідає певний мнемоніческій код, наприклад: ADD AX, BX.

Ця команда з мови асемблера. Команда ADD (скорочення від англійського слова addition – додавання) виконує операцію додавання чисел, що знаходяться в регістрах AX та BX.

Асемблер застосовується досить широко і сьогодні, але попит, як відомо, породжує пропозицію. Тому були створені мови високого рівня (Паскаль, Бейсік, C ++ та інші) Ці мови вже більш зрозумілі для людини. Наприклад, згадану вище операцію додавання, можна представити так: A: = A + B (Паскаль). Тут A і B – це вже не регістри, а змінні (про змінних трохи пізніше).

Оскільки недолік машинної мови був давним-давно зрозумілий усім програмістам, то вони складали з команд машинної мови, а пізніше з операторів мови високого рівня, процедури (або функції, про які йтимиться далі) для виконання найпопулярніших маленьких завдань, таких як:

1. Намалювати коло заданого розміру у заданому місці екрану.
2. Намалювати прямокутник заданого розміру та форми у заданому місці екрану.
3. Намалювати відрізок прямої.
4. Пофарбувати заданим кольором певну область екрану.
5. Відтворити мелодію по заданих нотах.
6. Написати на екрані задане слово, заданий текст.
7. Запам’ятати слово або текст, які введені з клавіатури.
8. Обчислити математичну формулу.

Як бачите, дії, які викликаються цими процедурами, набагато більші, ніж у команд машинної мови. Тому ці процедури зручніші для написання програм, хоча б для таких, як програма, яка обчислює sin(x). Звичайно, хотілося б мати всі подібні процедури всередині комп’ютера. Тому давним-давно існують дискети та компакт-диски, на яких записані цілі "збірки" таких процедур. І кожен охочий може взяти диск, переписати його вміст у комп’ютер та користуватися ним.

Процедури на такому диску записані не роз’єднано, а у комплексі як складові частини особливої великої програми. Якщо ми перепишемо цю велику програму у комп’ютер і запустимо її на виконання, то вона дозволить людині, по-перше, писати власні програми із згаданих процедур, а по-друге, зробить цей процес зручним, тобто виявлятиме багато помилок у ваших програмах, дозволить швидко запускати їх на виконання, виправляти, переписувати на диск і таке інше.

Називають таку комплексну програму складно і по-різному, наприклад, "Середовище і компілятор мови програмування високого рівня". Основне для нас у цій назві – поняття "Мова програмування" або простіше – "мова". Але якщо мова, то яка? У людей є російська, українська, англійська мови. Що являє

собою будь-яка з цих мов спілкування людей? Грубо кажучи, це набір букв, слів, розділових знаків та правил, за якими всі ці елементи потрібно збудувати в ланцюжок, щоб отримати правильне речення. Мова програмування – приблизно те ж саме. Найважливіша частина мови програмування – набір правил, за якими різні об'єкти (у тому числі і звернення до згаданих процедур) потрібно розмістити в ланцюжок, щоб отримати правильну програму. Строго кажучи, процедури не є складовою частиною мови, проте, ви повинні знати, що тримаючи в руках компакт-диск з написом "Turbo Pascal" або «C++» або якою-небудь іншою мовою, ви тримаєте в руках цілий комплекс програм, який містить і велику кількість цих самих процедур, та засобу для зручної розробки ваших програм з їх використанням.

11.2.1. Як «влаштована» мова програмування

Мова програмування використовується для запису алгоритму та подальшого перекладу цього запису на мову машинних команд. Мови програмування схожі на мови природного спілкування. В обох з них можна знайти багато спільного.

Будь-яка природна мова будується з елементарних складових – букв, які створюють абетку мови. Букви використовуються для побудови слів, слова складаються у речення. З речень складається будь-який текст – лист, роман, секретне донесення. Будь-яка мова програмування організована приблизно так само. Є абетка мови, тобто набір символів, які можна використовувати у програмі. З символів абетки складаються ***лексеми*** (лексичні одиниці мови), що відіграють роль слів , – мінімальних смислових елементів мови. До числа лексем входять спеціальні слова, імена різних об'єктів програми (наприклад, імена змінних) тощо. Як це виходить з назви, спеціальні слова мають спеціальне призначення. З них складаються, наприклад, ***оператори***, які описують деяку закінчену дію.

Спеціальні слова бувають двох видів. Існують ***зарезервовані слова***, що мають цілком певний сенс та певне призначення. Призначення зарезервованих слів зазвичай визначається специфікацією або стандартом мови. Зарезервовані слова не можна змінювати: будь-яка неточність в їх написанні є серйозною помилкою. Є також ***ключові слова***, призначення яких залежить від контексту, тобто від місця у програмі, де ці слова використовуються. У Pascal є зарезервовані слова, а ключовими словами, наприклад, є імена типів змінних.

На відміну від природних мов людського спілкування, у мовах програмування можна вводити свої власні слова, наприклад імена змінних, та надавати цим словам свій власний сенс. Невелику програму можна сподобити листу або маленькій розповіді. Великий проект – це роман. Як і звичайний лист, програма може бути написана хорошим або поганим **стилем**, і чим краще стиль, тим зрозуміліше програма, тим менше ймовірність появи у ній помилок.

Головними елементами будь-якої програми є змінні, константи та оператори.

Змінна – це один або декілька фізичних елементів оперативної пам'яті комп'ютера. Такий набір фізичних комірок називають абстрактним елементом пам'яті. Змінною зіставляється цілий набір характеристик (атрибутів). Одним з основних атрибутів є ім'я змінної. У правилах призначення імен змінним слід звернути увагу на те, які символи можна використовувати в іменах, яка максимальна довжина імені, чи розрізняється регістр символів.

Другою характеристикою змінної є її значення (його іноді називають «правим» значенням змінної). Вміст елементу пам'яті, яка пов'язана із змінною, може змінюватися по ходу виконання програми, тому змінна і називається змінною. Вид інформації, що міститься у пам'яті, набір перетворень, які можна виконувати над цією інформацією, та безліч припустимих значень визначаються **типом змінної**. Ще однією найважливішою характеристикою змінної є її **адреса** («ліве» значення). У лінійній моделі пам'яті це порядковий номер першого фізичного елементу пам'яті, що входить до складу абстрактного елементу пам'яті для змінної. У лінійній моделі пам'яті вважається, що елементи пам'яті розташовані в лінію. Послідовно розташовані комірки групуються у сегменти. Ще одним атрибутом змінної є її **область видимості** – ті оператори програми, в яких дану змінну можна використовувати. Зазвичай говорять про **глобальні** та **локальні змінні**. Глобальні змінні «видно» в усій програмі, а локальні – тільки в окремих її частинах.

Константа відрізняється від змінній тим, що її значення фіксоване і не може бути змінено при виконанні програми. Константи бувають двох видів: **буквальні** та **іменовані**. Буквальними константами є значення, які сприймаються у програмі так само, як вони зображені (буквально). Значення можуть бути числовими, символьними або іншими. Приклади буквальних констант:

2003 – ціле число;

1.9274 – дійсне число у форматі з фіксованою крапкою;

1.2e-3 – дійсне число у форматі з плаваючою крапкою; 'A' – символьна константа;

'**Hi, programmer!**' – строкова константа.

Буквальні константи ще називають **літералами**.

На **іменовані константи** посилаються, вказуючи їх ім'я. Ім'я призначається зазвичай за тими ж правилами, що й імена змінних. Приклад вбудованої іменованої константи у Turbo Pascal – Pi (число π).

Оператори задають дії, які повинна виконувати програма. Існують оператори, які дозволяють описати властивості змінних та інших об'єктів, що використовуються у програмі. Вони називаються ***реченнями опису*** або ***декларативними операторами.*** Оператори, що керують ходом виконання програми, називаються ***управляючими операторами.*** Управляючі оператори реалізують основні алгоритмічні конструкції, такі як розгалуження (умовні оператори та оператори вибору) та цикли. Є оператори введення та виведення інформації, за допомогою яких можна організувати спілкування користувача з програмою під час її виконання. Є й інші оператори.

Програмою можуть називати різні речі.

Звідки в комп'ютері вміння робити всі описані вище речі? Потрібно сказати, що колись комп'ютери нічого такого робити не вміли. І їх доводилося вчити. Як навчають комп'ютер? Приблизно так само, як навчають людей, розповідаючи їм, як робити те чи інше. Хай, наприклад, ви живете на 12 поверсі багатоповерхового будинку і до вас у гості приїхала людина, яка ніколи не бувала у місті. Припустимо, що ви хочете навчити її спускатися у двір на прогулянку. Для цього ви даєте їй таку інструкцію, що складається з шести команд:

1. Вийти з квартири
2. Підійти до дверей ліфта
3. Натиснути на кнопку
4. Коли двері відкриються, увійти
5. Натиснути на кнопку з цифрою 1
6. Коли ліфт спуститися і двері відкриються, вийти у двір

Якщо ваш гість вміє ходити та натискати на кнопки, то пам'ятаючи цю інструкцію, він відтепер зможе самостійно спускатися у двір.

А як же навчити зробити що-небудь не людину, а комп'ютер? Наприклад, ви хочете, щоб комп'ютер намалював на екрані монітора жовте сонце з променями. Для цього ви даєте йому на спеціальній, зрозумілій для нього мові інструкцію приблизно такого змісту:

1. Намалювати в такому-то місці екрану коло жовтим кольором.
2. Пофарбувати в жовтий колір внутрішню частину кола
3. Намалювати від кола проміння жовтим кольором.

Якщо комп'ютер вміє малювати різним кольором прямі та коло, а також фарбувати, то він зрозуміє цю інструкцію. Інструкція для комп'ютера на виконання завдання написана на спеціальній, призначеній для нього мові, називається *програмою*. Якщо ж вона написана звичайною українською або іншою людською мовою з розрахунку на те, щоб її зрозумів не комп'ютер, а людина, то вона називається, як ми вже знаємо, алгоритмом. Таким чином, ми тільки що написали алгоритм з трьох команд. Майте на увазі, що це часткове визначення програми і алгоритму. У загальному випадку вони визначаються, як набір правил для отримання потрібного результату.

Для комп'ютера це може бути початковий текст програми – звичайний текстовий файл, що містить запис операторів програми на мові програмування. Такий запис зрозумілий людині, але незрозумілий комп'ютеру. Початковий текст ***компілюється*** (транслюється), тобто перекладається мовою машинних команд, зрозумілою комп'ютеру. У цьому випадку створюється ***файл*** до ***виконання*** (його відмінна риса – ім'я, що закінчується на .exe). Саме файл до

виконання іноді називають програмою. І, нарешті, програмою можуть називати набір усіляких файлів, що виконуються, так і текстових (а можливо, і в інших форматах), таких, що відносяться до одного програмного пакету.

Ми вважатимемо, що програма є послідовністю операторів або інших елементів мови, яка побудована відповідно до певних правил та призначена для розв'язання певного завдання. Правила написання програми, тобто правила, що визначають, які послідовності символів можна використовувати у програмі, називаються ***синтаксисом*** мови програмування. Порушення цих правил призводить до ***синтаксичних помилок***. Ці помилки виявляються на етапі трансляції програми.

Семантика мови програмування – це правила, що визначають які операції і в якій послідовності повинна виконувати ЕОМ, працюючи за програмою.

Отже, для того щоб комп'ютер дещо вмів робити, він повинен мати всередині себе програму цього вміння. І навпаки, якщо комп'ютер дещо вміє, це означає, що хтось колись придумав програму цього вміння та ввів її у комп'ютер. Отже, якщо ваш комп'ютер вміє грати у шахи, це означає, що всередині нього знаходиться програма цієї гри, яку хтось туди ввів. Розучиться ваш комп'ютер грати у шахи тільки тоді, коли ви вилучите програму цієї гри з комп'ютера (або ненавмисно, або щоб звільнити у комп'ютері місце для інших програм).

Якщо повернутися до програми "Намалювати сонечко", то у списку цієї програми є прості (наприклад, привласнити значення координатам центру кола) та складні команди, такі як "Намалювати коло" або "Намалювати лінію". Ці складні команди, які, у свою чергу, складаються з простих команд, вже хтось написав (склав) раніше, щоб ви могли ними користуватися. Чи можете ви доповнити список складних команд потрібними вам командами? Хай ви хочете намалювати паровозик, у якого чотири колеса, та ще зафарбовані різним кольором. Для цього ви створите програму малювання кола в різних місцях екрану, з розфарбовуванням різними кольорами. Потім ви дасте назву новій команді, наприклад, НАМАЛЮВАТИ_КОЛО, і нарешті, повідомляєте цій програмі координати центру кола та колір. Така програма називається ***процедурою***, а нова команда НАМАЛЮВАТИ_КОЛО – зверненням до процедури або викликом процедури. Додаткову інформацію, яку ви їй повідомляєте (передаєте), – називають **параметрами процедури**.

11.2.2. Типи мов програмування

Отже, ми з'ясували, що мови програмування використовуються для запису алгоритмів з метою їх подальшого виконання на комп'ютері, а алгоритм може бути складений і записаний з різним ступенем деталізації. Ступінь деталізації, у свою чергу, залежить від того, наскільки детально враховується архітектура комп'ютера, на якому передбачається виконувати програму. Іноді говорять, що програміст пише програму не для конкретного комп'ютера, а для деякої абстрактної машини. Чим більше деталей пристрою реального комп'ютера

міститься в описі цієї уявної машини, тим нижче рівень абстракції. На одному з найвищих рівнів абстракції міститься уявлення про комп'ютер як фон-нейманівську машину, що складається з процесора, пам'яті та пристроїв введення/виведення.

Слід враховувати, що реальні програми виконуються в середовищі певної операційної системи. Нагадаємо, що **операційна система** є спеціальним програмним комплексом (набором програм), який забезпечує зручний інтерфейс, тобто методи і засоби спілкування користувача з комп'ютером, користувачем та його програмами. Зараз найбільш популярними операційними системами є операційні системи фірми Microsoft (MS-DOS, MS Windows), а також операційні системи родини UNIX. Якщо до абстрактної обчислювальної машини додати операційну систему, отримаємо ***віртуальну машину***. Програми пишуться для абстрактної обчислювальної машини, а для віртуальної машини створюються транслятори і середовища розробки і відлагодження програм.

Основними характеристиками, що дозволяють порівнювати мови програмування, є: потужність та рівень.

Потужність характеризується кількістю і різноманітністю завдань, алгоритми рішення яких можна записати, використовуючи цю мову. Найпотужнішим є машинна мова. Будь-яку задачу, запрограмовану на якійсь мові можна також запрограмувати і на машинній мові.

Рівень мови програмування характеризується складністю вирішення завдань з використанням цієї мови. Чим простіше записується рішення завдань, чим більш безпосередньо реалізуються складні операції і поняття, чим менше обсяг отримуваних програм, тим вище рівень мови.

Для написання програм на низькому рівні абстракції використовуються мови програмування низького рівня. Мови низького рівня використовуються для дуже докладного та детального опису операцій, настільки детального, що при цьому враховується архітектура комп'ютера, пристрій його центрального процесора (перш за все, набір його регістрів). Такі мови називаються ***машинними кодами*** та ***Асемблерами***. Асемблер дає дещо зручніший спосіб запису програм у машинних кодах.

Програма на Асемблері виходить довгою, отже, збільшується ймовірність появи помилок. Для складання такої програми потрібне знання архітектури комп'ютера, це збільшує трудомісткість програмування. Програма виявляється прив'язаною до конкретної архітектури. Все це можна вважати недоліками програмування на мовах низького рівня. Перевагою є можливість «витиснути» з комп'ютера все що можна, і, перш за все, – максимум швидкодії.

Мови програмування високого рівня були створені для того, щоб подолати недоліки низькорівневого програмування. Вони дозволяють використовувати різні операції, не піклуючись про деталі їх реалізації на конкретній архітектурі, тобто немає необхідності враховувати особливості конкретної ЕОМ та її систему команд. Програміст формулює алгоритм за допомогою достатніх узагальнених понять, таких як "арифметичний вираз", "виведення результату" тощо, що дозволяє легко програмувати різноманітні алгоритми

(універсальність), не вдаючись до зайвої деталізації. Програми при цьому виявляються коротшими, надійнішими та універсальними, а процес їх написання скорочується. Програми на мовах високого рівня легко читати, в них простіше розібратися.

Мови високого рівня бувають чотирьох видів:

– *процедурні (імперативні);*

– *функціональні;*

– *логічні;*

– *об'єктно-орієнтовані.*

Основними об'єктами у **процедурних мовах** (їх ще називають імперативними, тобто побудованими на вимогах, наказах) є змінні, оператори привласнення, стандартні алгоритмічні конструкції. Процедурні мови програмування прив'язані до традиційної фон-нейманівської архітектури.

У **функціональних мовах програмування** використовуються функції, значення яких визначаються за заданими параметрами. Традиційні змінні та оператори привласнення при цьому вже не потрібні або, принаймні, не обов'язкові. Прикладом функціональної мови є LISP.

У програмах, які написані на **логічних мовах**, немає певного, фіксованого порядку виконання правил та кроків алгоритму. Вибір відповідної послідовності покладається на систему. Прикладом такої мови є Prolog.

Об'єктно-орієнтовані мови спрощують програмування з використанням технології об'єктно-орієнтованого програмування (Java, C++, C#). Не слід плутати з мовами програмування мови розмітки документів, як, наприклад, HTML.

Pascal є мовою програмування високого рівня. Вона належить до категорії процедурних мов, і спочатку створювався для навчання програмуванню. З цієї причини в ній відсутні деякі важливі властивості, які необхідні для ефективної розробки прикладних програм: немає простих засобів роботи з масивами змінної довжини і т. і. Разом з тим це мова з чіткою логічною структурою, досить надійна. В ролі мови першого знайомства зі світом професійного програмування вона майже ідеальниа! Однією з найпопулярніших реалізацій мови є Turbo Pascal – система розробки програм на Pascal, яка була випущена фірмою Borland ще в середині 80-х років минулого століття. Ця реалізація не єдина, але у даному випадку ми дотримуватимемося саме Turbo Pascal (вер. 7).

11.2.3. Парадигми програмування

Вся півстолітня історія програмування, а може і історія всієї науки – це спроба впоратися зі складністю навколишнього світу. Завдання, що встають перед програмістами, стають все громіздкішими, інформація, яку треба обробити, росте як сніжний ком. Ще нещодавно звичайними одиницями вимірювання інформації були кілобайти та мегабайти, а зараз вже говорять тільки про гігабайти та терабайти. Як тільки програмісти пропонують більш-менш задовільне розв'язання запропонованих завдань, тут же виникають нові,

ще складніші завдання. Програмісти створюють нові методи, розроблюють нові мови. За півстоліття з'явилося декілька сотень мов, запропоновано безліч методів та стилів. Деякі методи та стилі стають загальноприйнятими і утворюють на деякий час так звану ***парадигму програмування***.

Наведемо спочатку визначення з тлумачного словника (наприклад, Велика Енциклопедія Кирила та Мефодія). Парадигма – набір теорій, стандартів та методів, які спільно є способом організації наукового знання, – іншими словами, спосіб бачення світу. За аналогією з цим прийнято вважати, що парадигма у програмуванні – це спосіб концептуалізації, який визначає, як слід проводити обчислення, і як робота, що виконується комп'ютером, має бути структурована та організована.

Відомо декілька основних парадигм програмування, найважливішими з яких на даний час є парадигми *директивного* (процедурне програмування, структурне програмування, модульне програмування), *об'єктно-орієнтованого* та *функціонально-логічного* програмування. Для підтримки програмування відповідно до тієї або іншої парадигми розроблені спеціальні алгоритмічні мови.

Мови C та Turbo Pascal (до версії 5.5) є прикладами мов, які призначені для директивного програмування, коли розробник програми використовує процесно-оріентовану модель, тобто намагається створити код, який належним чином впливає на дані. Активною при цьому підході вважається програма (код), яка повинна виконати всі необхідні дії над пасивними даними для досягнення потрібного результату.

Цей підхід представляється повністю природним для людини, яка лише починає вивчати програмування, та історично виник одним з перших, проте він практично непридатний для створення великих програм. Четверта частина цього посібника присвячена саме директивному програмуванню, оскільки подібний стиль є оптимальним для програмування невеликих програм. А навички, які він дає можливість придбати, необхідні і при використанні інших підходів.

Перші, навіть найпростіші програми, які написані у машинних кодах, складали сотні рядків абсолютно незрозумілого тексту. Для спрощення та прискорення програмування придумали мови високого рівня: FORTRAN, Algol та сотні інших, поклавши рутинні операції по створенню машинних кодів на компілятор. Ті ж програми, що переписані мовами високого рівня, стали набагато зрозумілішими та коротшими. Але життя зажадало розв'язання складніших завдань, і програми знову збільшилися у розмірах, стали неозорими.

Виникла ідея: оформити програму у вигляді декількох, по можливості простих, процедур або функцій, кожна з яких розв'язує своє певне завдання. Написати, відкомпілювати та відлагодити невелику процедуру можна легко та швидко. Потім залишається тільки зібрати всі процедури у потрібному порядку в одну програму. Крім того, один раз написані процедури можна потім використовувати в інших програмах як будівельні цеглини. ***Процедурне***

програмування швидко стало парадигмою. В усі мови високого рівня включили засоби написання процедур та функцій. З'явилося безліч бібліотек процедур та функцій на всі випадки життя.

Виникло питання про те, як виявити структуру програми, розбити програму на процедури, яку частину коду виділити в окрему процедуру, як зробити алгоритм розв'язання задачі простим та наочним, як зручніше зв'язати процедури між собою. Досвідчені програмісти запропонували свої рекомендації, які названи ***структурним програмуванням***. Структурне програмування виявилося зручним та стало парадигмою. З'явилися мови програмування, наприклад Pascal, на яких зручно писати структурні програми. Більш того, на них дуже важко створювати неструктурні програми.

Складність завдань, які стоять перед програмістами, виявилася і тут: програми почали містити сотні процедур, і знову виявилися неозорими. "Цегла" стала дуже маленькою. Був потрібний новий стиль програмування.

У цей же час виявилося, що вдала або невдала структура початкових даних може сильно полегшити або ускладнити їх обробку. Одні початкові дані зручніше об'єднати у масив, для інших більше підходить структура дерева або стека. Ніклаус Вірт навіть назвав свою книгу "**Алгоритми + структури даних = програми**".

Виникла ідея об'єднати початкові дані та всі процедури їх обробки в один модуль. Ця ідея ***модульного програмування*** швидко знайшла багато прихільників і на деякий час стала парадигмою. Програми складалися з окремих модулів, що містять десяток-другий процедур та функцій. Ефективність таких програм тим вище, чим менше модулі залежать один від одного. Автономність модулів дозволяє створювати і бібліотеки модулів, щоб потім використовувати їх як будівельні блоки для програми.

Зараз вельми поширеним став об'єктно-орієнтований підхід, який реалізується, наприклад, мовами C++, Java, Pascal 7.0. При цьому, навпаки, первинними вважаються об'єкти (дані), які можуть активно взаємодіяти один з одним за допомогою механізму передачі повідомлень (який зветься, також, і механізмом виклику методів). Програміст у цьому випадку повинен створити та реалізувати такі об'єкти, взаємодія яких після початку роботи програми призведе до досягнення необхідного кінцевого результату.

Для того щоб забезпечити максимальну незалежність модулів один від одного, треба чітко відокремити процедури, які викликатимуться іншими модулями, – **відкриті** (public) процедури, від допоміжних, які обробляють дані, що вкладені у цей модуль, – **закритих** (private) процедур. Перші перераховуються в окремій частині модуля – **інтерфейсі** (interface), другі беруть участь тільки в **реалізації** (implementation) модуля. Дані, що занесені у модуль, теж діляться на відкриті, вказані в інтерфейсі і доступні для інших модулів, та закриті, доступні тільки для процедур того ж модуля. В різних мовах програмування це ділення проводиться по-різному. У мові Turbo Pascal модуль спеціально ділиться на інтерфейс та реалізацію. В мові C інтерфейс виноситься в окремі "головні" (header) файли. У мові C++, крім того, для опису

інтерфейсу можна скористатися абстрактними класами. У мові Java є спеціальна конструкція для опису інтерфейсів, яка так і називається – interface, але можна створити й абстрактні класи.

Так виникла ідея про приховування, ***інкапсуляцію*** (incapsulation) даних та методів їх обробки. Інкапсуляція, звичайно, проводиться не для того, щоб заховати від іншого модуля щось цікаве. Тут переслідуються дві основні мети. Перша – забезпечити безпеку використання модуля, винести в інтерфейс, зробити загальнодоступними тільки ті методи обробки інформації, які не можуть зіпсувати або вилучити початкові дані. Друга мета – зменшити складність, приховавши від зовнішнього світу непотрібні деталі реалізації.

Знову виникло питання, яким чином розбити програму на модулі? Тут до речі опинилися методи розв'язання старої задачі програмування – моделювання дій штучних та природних об'єктів: роботів, верстатів з програмним управлінням, безпілотних літаків, людей, тварин, рослин, систем забезпечення життєдіяльності, систем управління технологічними процесами.

Насправді, кожен об'єкт – робот, автомобіль, людина – володіє певними характеристиками. Ними можуть бути: вага, зріст, максимальна швидкість, кут повороту, вантажопідйомність, прізвище, вік. Об'єкт може проводити якісь дії: переміщатися у просторі, повертатися, піднімати, копати, зростати або зменшуватися, змінюючи свої первинні характеристики. Зручно змоделювати об'єкт у вигляді модуля. Його характеристики будуть даними, постійними або змінними, а дії – процедурами. Виявилося зручним зробити і зворотне – розбити програму на модулі так, щоб вона перетворилася на сукупність взаємодіючих об'єктів. Так виникло ***об'єктно-орієнтоване програмування*** (object-oriented programming), скорочено ООП (ООР) – сучасна парадигма програмування.

Об'єктно-орієнтоване програмування є *технологією програмування*, в основі якої лежить здатність людини узагальнювати, класифікувати та генерувати абстракції. Концепція ООП базується на таких поняттях як *об'єкти* та *класи*. У повсякденному житті об'єкти зустрічаються на кожному кроці: комп'ютери, телефони, машини, чашки та багато іншого. У програмуванні це можуть бути символи, числа, рядки, а також об'єкти, які створені та реалізовані автором програми. Наприклад, вікно на екрані дисплея – це об'єкт, який має ширину width і висоту height, розташування на екрані, що описується зазвичай координатами (x, y) лівого верхнього кута вікна, а також шрифт, яким у вікно виводиться текст, скажімо, Times New Roman, колір фону color, декілька кнопок, лінійки прокрутки та інші характеристики. Вікно може переміщатися по екрану методом move(), збільшуватися або зменшуватися в розмірах методом size(), згортатися в ярлик методом iconify(), якось реагувати на дії миші та натиснення клавіш. Це повноцінний об'єкт! Кнопки, смуги прокрутки та інші елементи вікна – це також об'єкти зі своїми розмірами, шрифтами, переміщеннями. Зрозуміло, вважати, що вікно само "вміє" виконувати дії, а ми тільки даємо йому доручення: "Згорнися, розгорнися, пересунься", – це

декілька несподіваний погляд на речі, але ж зараз можна подавати команди не тільки мишею та клавішами, але й голосом!

Важливо те, що об'єкти – це не лише деякі значення (дані). В них також стосовно об'єкту можуть бути виконані деякі, заздалегідь визначені дії – ***методи***, а процес виконання цих дій називають ***викликом методу***. В результаті цих дій стан об'єкту може змінитися. Таким чином, об'єкти є об'єднанням структур даних, які містять інформацію, і методів та функцій для управління цими даними.

В багатьох мовах для виклику методу необхідно спочатку вказати об'єкт, до якого застосовується той чи інший метод, а потім після крапки записати ім'я методу. Саме цей синтаксис, в якому головна роль відводиться об'єкту, над яким виконується дія, послужив основою для назви парадигми – "об'єктно-орієнтована".

Довільну об'єктно-орієнтовану мову програмування, окрім інкапсуляції, характеризують ще двома основними властивостями:

– ***успадкування***, тобто створення нових, похідних класів, які успадковують дані і методи від раніше визначених базових класів; при цьому можливе перевизначення або додавання нових даних та методів, в результаті чого створюється так звана ***ієрархія класів***;

– ***поліморфізм***, тобто наявність в різних класах методів з одним і тим самим ім'ям, внаслідок чого кожен з класів має можливість по-своєму виконувати пов'язані з цим методом дії.

Ідея об'єктно-орієнтованого програмування виявилася дуже плідною і почала активно розвиватися. З'ясувалося, що зручно ставити завдання одразу у вигляді сукупності об'єктів, що діють, – виник ***об'єктно-орієнтований аналіз*** (ООА). Вирішили проектувати складні системи у вигляді об'єктів – з'явилося ***об'єктно-орієнтоване проектування***, ООП (OOD – object-oriented design).

Функціональне та логічне програмування використовує мови типу Lisp, Haskell та Prolog. Ця парадигма базується на принципово іншому трактуванні поняття програми. Тут головним є **точне формулювання завдання**, а вибір та вживання необхідного алгоритму для її розв'язання – проблема виконуючої системи, але не програміста. При використанні директивного або об'єктно-орієнтованого підходів та таких мов, як C, C++, Turbo Pascal або Java розмір програми, що розв'язує дане завдання, буде гарантовано набагато більшим. Проте слід відзначити, що інтерпретатори функціональних мов зазвичай працюють досить повільно.

Докладніше принципи об'єктно-орієнтованого, функціонального та логічного програмування вивчатимуться в спеціальних дисциплінах на старших курсах навчання.

11.3. Як навчитися програмувати?

Перш за все, нам хотілося б сказати, що управляти комп'ютером і створювати програми може будь-яка людина. Для створення комп'ютерних програм не потрібно володіти неймовірним інтелектом або вченим ступенем в математичних дисциплінах. Вам знадобиться лише бажання у чомусь розібратися та терпіння, аби не кинути заняття.

Вміння писати програми – це таке саме вміння, як і вміння їздити на велосипеді, плавати або грати у футбол. Деяким людям дійсно вдається робити це набагато краще, ніж іншим, але будь-яка людина зможе досягти певних результатів при належній практиці. Саме з цієї причини діти стають асами програмування у ранньому віці. Діти не обов'язково геніальні, вони просто схильні пізнавати нове і не бояться помилятися.

11.3.1. Програмування – це просто?

Програмування – це ремесло, і кожен програміст повинен досягти потрібного професійного рівня. Подібно до того, як в середні віки утворилися гільдії ремісників – частково для навчання молодих працівників, частково для підвищення професійного рівня, – так і у наш час створені численні навчальні заклади для підготовки програмістів, і все менша і менша їх кількість навчається на власних синцях і шишках. Проте з'ясувалося, що для підготовки майстрів високого класу зусиль одних лише викладачів недостатньо.

У цьому відношенні учнівство в гільдіях володіло незаперечною перевагою. Навчання ремеслу полягало у тому, що учень протягом багатьох років виконував прості допоміжні операції, переймаючи основні прийоми у досвідченіших працівників. Поступово на нього покладалися все серйозніші обов'язки, і після формальної перевірки навичок він отримував офіційне підтвердження своєї професійної компетентності.

У наші дні програмісту-початківцю вже не потрібно сім років (приблизний термін учнівства в ремісничих майстернях), а необхідні технічні знання йому простіше отримати, відвідуючи лекції та вивчаючи літературу. Тепер немає потреби, заглядаючи через плече досвідченого програміста, щоденно спостерігати за його роботою. А ось на те, щоб "набити руку" на виконанні реальних завдань програмістів, засвоїти та закріпити основні методи і принципи роботи, дійсно потрібний час. Зрозуміло, що, прочитавши декілька книг зі столярної справи, не можна одразу взяти та випустити у світ дещо у вигляді шафи. Так чому ж людина, яка прочитала одне-два керівництва з програмування, раптом одразу почне писати стрункі, грамотні програми?

Викладання програмування та його вивчення – непосильна праця. Викладач може всіляко возитися зі студентами, читати лекції, робити критичні зауваження і таке інше Студент може все ретельно записувати, запам'ятовувати, читати, здавати заліки і навіть іспити. Але всі зусилля марні, якщо студент не практикуватиметься у написанні програм, оскільки навички

програмування (як, втім, і інші навички) даються тільки практикою. Більш того, вчитися треба на "справжніх" програмах, а не на спрощених прикладах, якими рясніють більшість керівництв з мов програмування. Просто вивчення мови програмування навряд чи допоможе досягти висот в програмуванні. Тому, **навчитися програмувати можна лише програмуючи.**

11.3.2. Основні прийоми програмування

Здібності, що необхідні програмістові, можна порівняти з тими, які потрібні, наприклад, письменникові. Як і письменник, програміст повинен володіти деякими правилами правопису та граматики, але, всупереч загальноприйнятій думці, для них обох це не головне. Набагато важливіше бути спостережливим та таким, хто шукає, вміти аналізувати та зрозуміло висловлювати свої думки.

Насправді, процес написання програми полягає у записі (кодуванні) деякого алгоритму з використанням тих чи інших засобів, які надаються вам мовою програмування. Тому, для успішного складання програм треба добре знати дві речі: по-перше (і передусім) – алгоритм, і по-друге – набір засобів (прийомів), за допомогою яких цей алгоритм можна реалізувати мовою програмування.

Як складати алгоритми? На жаль, відповісти на це питання однозначно та повно, взагалі, навряд чи можливо! Це все одно, що намагатися відповісти на питання: як писати картини або музику? Вся річ у тому, що процес створення алгоритму і, зрештою, програми – процес творчий. Проте, художник або музикант, окрім таланту, повинні володіти деякою загальною технікою написаня картин або музики. Це також відноситься і до складання алгоритмів.

Розглянемо приклад, який дозволить виявити загальні принципи, згідно з якими, будуються алгоритми для комп'ютерів.

Скласти алгоритм пошуку максимального числа серед заданої послідовності N чисел.

- "А що тут складати – подивився і вибрав"! – можете вигукнути Ви. Так, ми дійсно можемо подивитися і вибрати, але комп'ютер цього зробити не може. А що може? Може складати, віднімати, ділити, множити, порівнювати числа.

Позначимо задані числа як A1, A2, ..., An. Введемо додаткові змінні: Max – для максимального числа і J – порядковий номер елементу.

Опишемо словесно можливу послідовність дій:

1. Ввести числа: N, A1, A2, ., An.
2. Змінній Max присвоїти значення A1.
3. Змінній J присвоїти значення 2.
4. Якщо Ai > Max, тоді змінній Max присвоїти значення Ai.
5. Збільшити значення змінної J на одиницю.
6. Якщо J > N, тоді вивести на екран Max, інакше йти до п.4.

При складанні алгоритму:

– можна використовувати тільки такі інструкції, які здатний виконати виконавець (комп'ютер). Отже, необхідно добре знати можливості виконавця та навчитися зводити розв'язання конкретної задачі до послідовності цих інструкцій;

– необхідно передбачити (продумати) усі можливі варіанти розвитку обчислювального або іншого процесу, включаючи дії машини у разі некоректних початкових даних.

Після того, як алгоритм складений (на папері або "в голові") його необхідно перекласти на алгоритмічну мову.

Зведення складного до простого. Цикл. Отже, щоб змусити комп'ютер щось зробити, потрібно написати програму. І тут мимоволі виникає питання – невже можливо, записуючи одна за одною досить примітивні команди мови програмування, написати програми для всіх тих чудових умінь комп'ютера, деякі з яких ми навели вище? Візьмемо, наприклад, гру у повітряний бій. Адже літачок по екрану повинен рухатися! Але в списках команд більшості мов програмування, що використовуютьтся професійними програмістами для створення таких ігор, немає команди руху. Або візьмемо обчислення траєкторії польоту космічного корабля на Місяць. Для її обчислення потрібно вирішувати складні диференціальні рівняння вищої математики. Але процедури мов Бейсик, Паскаль, які використовуються для цих цілей, не можуть нічого, що виходить за рамки шкільного курсу. І незрозуміло, врешті-решт, як навчити комп'ютер розмовляти, якщо у нашому розпорядженні є тільки команда витягнення з комп'ютера простого звуку заданої висоти.

На всі ці питання відповідь одна: якщо ви добре розбираєтеся у поставленому завданні, то ви обов'язково зможете **розкласти його на багато маленьких завдань**, кожне з яких цілком піддається програмуванню, після чого з багатьох програмок, які вийшли, можна зібрати одну велику програму, яка і розв'язує задачу. Розберемо для ілюстрації декілька основних ідей та прийомів зведення складного до простого.

Візьмемо гру – гонки автомобілів. Тут потрібно задатися питанням – а що таке рух? Розглянемо ілюзію руху, що виникає на екрані кінотеатру. Якщо ви тримали в руках кіноплівку фільму, що зображає, скажімо, рух автомобіля, то повинні були звернути увагу, що вона складається з безлічі нерухомих слайдів (кадрів), на кожному наступному з яких автомобіль знаходиться трохи в іншому місці, ніж на попередньому. Показуючи ці кадри один за одним з великою швидкістю, створюється ілюзія руху автомобіля. Так само поступають зі створенням руху на екрані комп'ютера. Запишемо алгоритм руху по екрану зліва направо звичайного кружечка:

1. Задамо позицію кружечка у лівій частині екрану. Перейдемо до команди 2.
2. Намалюємо кружечок. Перейдемо до команди 3.
3. Зітремо його. Перейдемо до команди 4.
4. Змінимо позицію кружечка на міліметр правіше. Перейдемо до команди 5.
5. Перейдемо до команди 2.

Кожна з наведених команд алгоритму легко програмується на більшості мов. До речі, будь-який комп'ютер, виконавши чергову команду, автоматично переходить до виконання наступної (отже нам не обов'язково було писати, скажімо, у другій команді "Перейдемо до команди 3."). Проте, якщо ми захочемо, то можемо змусити комп'ютер змінити цей порядок, що ми і зробили у команді 5, завдяки чому комп'ютер почав багато разів виконувати послідовність команд 2-3-4-5. Така послідовність, яка багато разів виконується, називається ***циклом***. Цикл – основний засіб змусити комп'ютер зробити багато за допомогою короткої програми.

У короткому проміжку часу після виконання команди 2 кружечок з'являтиметься на екрані і на команді 3 зникати, але цього досить, щоб людське око його помітило. Завдяки циклу кружок меготітиме кожного разу у новому місці, а оскільки зміна "кадрів" буде дуже швидкою, нам здаватиметься, що відбувається плавний рух кружка.

Тепер перейдемо до завдання про траєкторію. Існує наука – обчислювальна математика – яка стверджує, що розв'язання багатьох найскладніших математичних рівнянь можна звести до багаторазового виконання чотирьох дій арифметики, та показує, як це робити. Простіше кажучи, замість розв'язання одного рівняння вона пропонує виконати п'ять мільйонів додавань. Це якраз те, що потрібне комп'ютеру. П'ять мільйонів додавань він виконає за секунду.

Власні процедури. Велику допомогу у спрощенні програмування складних завдань надає можливість, яка доступна програмісту для більшості мов, складати власні процедури (та функції) з команд та операторів мови. Складається процедура так само, як ми складали вище процедуру Намалювати_коло.

Розгалуження (вибір). У програміста-початківця інтерес повинні викликати такі питання: як комп'ютер ухвалює рішення, як він обирає, яку дію з кількох можливих потрібно виконати у даний момент? Візьмемо ту ж гру «гонки автомобілів». Припустимо, що при натисненні на клавішу R автомобіль повертаємо праворуч, при натисненні на клавішу L – ліворуч. Як комп'ютер відчуває натиснення на клавіші, звідки він знає, що потрібно робити при натисненні на кожну з них? Природно, цей вибір робить програма для гри у гонки, сам по собі комп'ютер нічого обрати не може. У програмі заздалегідь пишуться процедури для повороту праворуч і для повороту ліворуч, а вибір між ними робить спеціальна команда вибору, наявна у кожній мові програмування.

Вибір називають розгалуженням по аналогії з деревом, що розгалужується (коли ми залізаємо на дерево, ми час від часу робимо вибір, по якій з декількох гілок лізти далі).

На цьому ми завершимо розгляд основних ідей програмування. Конкретне їх втілення відкладемо до вивчення Паскаля.

Контрольні запитання для самоперевірки

1. Які програми призначені для спрощення спілкування користувача з ОС?
2. Чому перші мови програмування називали алгоритмічними мовами?
3. Які два способи (методи) трансляції Ви знаете?
4. Що являє собою декларативний оператор?
5. Які помилки виявляються на етапі трансляції програми?
6. Що являють собою парадигми програмування?
7. Що являє собою віртуальна машина?
8. Яка основна ідея модульного програмування?

12. Технологія розв’язання завдань за допомогою комп’ютера

12.1. Етапи розв’язання завдань за допомогою комп’ютера

Вирішення завдань за допомогою комп’ютера включає наступні основні етапи, частина з яких здійснюється без участі комп’ютера.

1. **Постановка завдання:**
- збір інформації про завдання;
- формулювання умови завдання;
- визначення кінцевої мети вирішення задачі;
- визначення форми видачі результатів;
- опис даних (їх типів, діапазонів значень, структури і т.і.).

2. **Аналіз та дослідження задачі, моделі:**
- аналіз існуючих аналогів;
- аналіз технічних та програмних засобів;
- розробка математичної моделі;
- розробка структур даних.

3. **Розробка алгоритму:**
- вибір методу проектування алгоритму;
- вибір форми запису алгоритму (блок-схеми, псевдокод та ін.);
- вибір тестів та методу тестування;
- проектування алгоритму.

4. **Програмування:**
- вибір мови програмування;
- уточнення способів організації даних;
- запис алгоритму на обраній мові програмування.

5. **Тестування і відлагодження:**
- синтаксичне відлагодження;
- відлагодження семантики та логічної структури;
- тестові розрахунки та аналіз результатів тестування;
- вдосконалення програми.

6. **Аналіз результатів вирішення задачі та уточнення у разі потреби математичної моделі з повторним виконанням етапів 2–5.**

7. **Супровід програми:**
- доопрацювання програми для вирішення конкретних завдань;
- складання документації до вирішеного завдання, до математичної моделі, до алгоритму, до програми, до набору тестів, до використання.

12.2. Постановка завдання

На цьому етапі розкривається організаційно-економічна сутність задачі:
- формулюється мета її вирішення;
- визначається взаємозв'язок з іншими завданнями;

– вказується періодичність її рішення;
– розкривається склад і форма подання вхідої, проміжної і вихідної інформації;
– характеризуються форми і методи контролю достовірності інформації;
– описуються форми взаємодії користувача з ЕОМ.

Особлива увага приділяється детальному опису вхідної, вихідної і проміжної інформації. При цьому визначається:

– форма подання окремих даних;
– кількість знаків, що виділяються для запису даних, виходячи з їх максимальної значности;
– джерело виникнення даних.

Крім того, для цифрової інформації вказується цілочисельний або дробовий характер даних і припустимий діапазон зміни величин.

Завершується постановка завдання описом контрольного прикладу, який демонструє порядок вирішення завдання традиційним способом.

Основна вимога до контрольного прикладу – це відображення всього різноманіття можливих форм існування вихідних даних.

Користувач добре знає проблемну сторону завдання, але зазвичай слабо уявляє, як вона буде вирішуватися на ЕОМ.

Предметна область користувача часто незнайома програмісту, тому необхідна повна коректна постановка задачі, яка однозначно розуміється користувачем і розробником.

План написання постановки завдання:

1. Найменування завдання.
2. Призначення.
3. Яка досягається мета.
4. Технічні засоби.
5. Періодичність використання.
6. Вхідна інформація.
7. Вихідна інформація (формується за запитами).
8. Метод перевірки правильності (порівнюється з контрольним прикладом).
9. Організація впровадження завдання.
10. Розробка контрольного прикладу (вхідна інформація з конкретними даними, вихідна інформація).
11. Методи захисту.

12.3. Побудова математичної моделі

На цьому етапі проводиться аналіз і дослідження задачі.

Структура етапу:

1. Аналіз існуючих аналогів завдання.
2. Аналіз технічних і програмних засобів.
3. Розробка математичної моделі.

4. Розробка структур даних.

Природна мова, на якій здійснюється постановка задачі, має властивість неоднозначності. Створення математичної моделі дозволяє формалізувати опис завдання.

Математична модель – це система математичних співвідношень – формул, рівнянь, нерівностей і таке інше, що відображають суттєві властивості об'єкту або явища.

Щоб описати явище, необхідно виявити суттєві його властивості, закономірності, внутрішні зв'язки, роль окремих характеристик явища. Виділивши найбільш важливі чинники, можна нехтувати менш суттєвими.

Найефективніше математичну модель можна реалізувати на комп'ютері у вигляді алгоритмічної моделі – так званого "обчислювального експерименту". Звичайно, результати обчислювального експерименту можуть опинитися і не відповідними дійсності, якщо в моделі не будуть враховані якісь важливі сторони дійсності.

Отже, створюючи математичну модель для вирішення завдання, необхідно:

– виділити припущення, на яких має грунтуватися математична модель;

– визначити, що вважати початковими даними та результатами;

– записати математичні співвідношення, які пов'язують результати з початковими даними.

При побудові математичних моделей далеко не завжди вдається знайти формули, які явно відображають шукані величини через дані. У таких випадках використовуються математичні методи, які дозволяють дати відповіді того чи іншого ступеня точності.

Існує не тільки математичне моделювання будь-якого явища, але і візуально-натурне моделювання, яке забезпечується за рахунок відображення цих явищ засобами машинної графіки, тобто перед дослідником демонструється своєрідний "комп'ютерний мультфільм", який знімається у реальному часі. Наочність тут дуже висока.

Математична запис постановки завдання відрізняється високою точністю відображення її сутності, лаконічністю запису, однозначністю розуміння, але вона може бути виконана не для всіх завдань.

При виборі методу вирішення перевага віддається методу, який:

1. Забезпечує необхідну точність і не має властивість виродження (нескінченного зациклення).
2. Дозволяє використовувати вже готові стандартні програми.
3. Орієнтований на мінімальний обсяг інформації.
4. Найбільш швидке отримання результатів.

12.3. Які основні етапи містить процес розробки програм?

Процес розробки програми можна виразити наступною формулою:

РОЗРОБКА ПРОГРАМИ = ВИГОТОВЛЕННЯ + ДОКАЗ ПРАВИЛЬНОСТІ

Наявність помилок у тільки що розробленій програмі це цілком нормальне закономірне явище. Практично неможливо скласти реальну (достатньо складну) програму без помилок. Не можна говорити, що програма правильна, лише на тій підставі, що вона не знехтувана машиною та видала результати. Адже все, що досягнуте у даному випадку, це отримання якихось результатів, не обов'язково правильних. У програмі при цьому може залишатися велика кількість логічних помилок.

Програма є ***правильною***, якщо вона не містить помилок, і працює відповідно до технічного завдання. Така програма не дає невірних результатів, тобто вона надійна.

Надійність програмного забезпечення – властивість програми виконувати задані функції, зберігати свої характеристики у встановлених межах за певних умов експлуатації. Надійність програмного забезпечення визначається його безвідмовністю і відновлюваністю.

Програма є ***точною***, якщо видані нею числові дані мають припустимі відхилення від аналогічних результатів, отриманих за допомогою ідеальних математичних залежностей.

Програма є ***адаптованою***, якщо вона припускає швидку модифікацію з метою пристосування до мінливих умов функціонування.

12.4. Як проконтролювати текст програми до виходу на комп'ютер?

Текст програми можна проконтролювати за столом за допомогою перегляду, перевірки та прокрутки.

Перегляд. Текст програми є видимим на предмет виявлення описок та розбіжностей з алгоритмом. Потрібно проглянути організацію всіх циклів, щоб переконатися у правильності операторів, які задають кількість повторів циклів. Корисно подивитися ще раз умови в умовних операторах, аргументи у зверненнях до підпрограм і т.і.

Перевірка. При перевірці програми програміст по тексту програми у думках прагне відновити той обчислювальний процес, який визначає програма, після чого звіряє його з необхідним процесом. На час перевірки потрібно "забути", що повинна робити програма, та "дізнаватися" про це по ходу її перевірки. Тільки після закінчення перевірки програми можна "пригадати" про те, що вона повинна робити і порівняти реальні дії програми з потрібними.

Прокрутка. Основою прокрутки є імітація програмістом за столом виконання програми на машині. Для виконання прокрутки доводиться задаватися якимись початковими даними та проводити над ними необхідні обчислення. Прокрутка – трудомісткий процес, тому її слід застосовувати лише для контролю логічно складних ділянок програм. Початкові дані повинні обиратися такими, щоб до прокрутки залучалися більшість гілок програми.

Найсучаснмшим методом створення програм, у тому числі для розподілених та паралельних систем, є метод MODEL CHECKING, про який

йшдлся в розділі 9.13. За цим методом на основі опису програми в термінах темпоральної логіки створюється графова автоматна модель програми, перевіряється виконання умов опису програми, тобто виконується її верифікація. Наступним кроком є обход графу автомату по всіх його шляхах віж початкового стану до кінцевого, в процесі якого створюється власно програма.

12.5. Для чого потрібні відлагодження та тестування?

Відлагодження програми – це процес пошуку та усунення помилок у програмі, який здійснюється за результатами її прогону на комп'ютері.

Тестування – це випробування, перевірка правильності роботи програми у цілому, або її складових частин.

Відлагодження та тестування (англ. test – випробування) – це два чітко помітних і несхожих один на одний етапи:

– при відлагодженні відбувається локалізація та усунення синтаксичних помилок та явних помилок кодування;

– у процесі ж тестування перевіряється працездатність програми, яка не містить явних помилок.

Тестування встановлює факт наявності помилок, а відлагодження з'ясовує їх причину.

12.6. У чому полягає відлагодження?

В сучасних програмних системах (Turbo Basic, Turbo Pascal, Turbo C та ін.) відлагодження здійснюється часто з використанням спеціальних програмних засобів, які називають відлагоджувачами. Ці засоби дозволяють досліджувати внутрішню поведінку програми.

Програма-відлагоджувач зазвичай забезпечує наступні можливості:

– **покрокового виконання програми** із зупинкою після виконання кожної команди (оператора);

– **перегляд поточного значення будь-якої змінної** або обчислення значення будь-якого виразу, зокрема, з використанням стандартних функцій; при необхідності можна встановити нове значення змінної;

– **встановлення у програмі "Контрольних точок",** тобто точок, в яких програма тимчасово призупиняє своє виконання, тому можна оцінити проміжні результати, та ін.

При відлагодженні програм важливо пам'ятати наступне:

– на початку процесу відлагодження треба використовувати **прості тестові дані**;

– виникаючі труднощі слід чітко **розділяти та усувати строго по черзі**;

– **не потрібно вважати причиною помилок комп'ютер,** оскільки сучасні комп'ютери та транслятори володіють надзвичайно високою надійністю.

12.7. Що являє собою тест та тестування?

Як би не була ретельно відлагоджена програма, вирішальним етапом, що встановлює її придатність для роботи, є контроль програми за наслідками її виконання на системі тестів.

Програму умовно можна вважати правильною, якщо її запуск для обраної системи тестових початкових даних в усіх випадках дає правильні результати.

Але, як справедливо вказував відомий теоретик програмування Э. Дейкстра, **тестування може показати лише наявність помилок, але не їх відсутність**. Нерідкіми є випадки, коли нові вхідні дані викликають "відмову" або отримання невірних результатів роботи програми, яка вважалася повністю відлагодженою.

Для реалізації методу тестування повинні бути виготовлені або заздалегідь відомі еталонні результати. Обчислювати еталонні результати потрібно обов’язково до, а не після отримання машинних результатів. Інакше є **небезпека мимовільної підгонки обчислюваних значень під бажані, що отримані раніше на комп'ютері.**

12.8. Якими повинні бути тестові дані?

Тестові дані повинні забезпечити перевірку всіх можливих умов виникнення помилок:

– повинна бути випробувана кожна гілка алгоритму;

– черговий тестовий прогін повинен контролювати дещо таке, що ще не було перевірене на попередніх прогонах;

– перший тест повинен бути максимально простий, щоб перевірити, чи працює програма взагалі;

– арифметичні операції в тестах повинні максимально спрощуватися для зменшення об’єму обчислень;

– кількість елементів послідовностей, точність для ітераційних обчислень, кількість проходів циклу у тестових прикладах повинні задаватися з міркувань скорочення об’єму обчислень;

– мінімізація обчислень не повинна знижувати надійності контролю;

– тестування повинно бути цілеспрямованим та систематизованим, оскільки випадковий вибір початкових даних призвів би до труднощів у визначенні ручним способом очікуваних результатів; крім того, при випадковому виборі тестових даних можуть виявитися неперевіреними багато ситуацій;

– ускладнення тестових даних повинно відбуватися поступово.

Приклад. Система тестів для завдання знаходження квадратного кореня у рівнянні $ax^2 + bx + c = 0$:

№ тесту	Перевіряемий випадок	Коефіцієнти			Результати
		a	b	c	
1	d > 0	1	1	-2	x1 = 1, x2 = -2
2	d = 0	1	2	1	Корні: x1 = -1, x2 = -1
3	d < 0	2	1	2	Дійсних коренів немає
4	a = 0, b = 0, c = 0	0	0	0	Усі коефіцієнти дорівнюють нулю. x – будь-яке число
5	a = 0, b = 0, c ≠ 0	0	0	2	Неправильне рівняння
6	a = 0, b ≠0	0	2	1	Один корень: x = -0.5
7	a ≠0, b ≠ 0, c = 0	2	1	0	x1 = 0, x2 = -0.5

12.9. Які характерні помилки програмування?

Помилки можуть бути допущені на всіх етапах рішення задачі – від її постановки до оформлення. Різновиди помилок та відповідні приклади наведені у таблиці:

Вид помилки	Приклад
Неправильна постановка завдання	Правильне рішення невірно сформульованої задачі
Невірний алгоритм	Вибір алгоритму, що приводить до неточного або неефективного вирішення задачі
Помилка аналізу	Неповний облік ситуацій, які можуть виникнути; логічні помилки
Семантичні помилки	Нерозуміння порядку виконання операторів
Синтаксичні помилки	Порушення правил, які визначені мовою програмування
Помилки при виконанні операцій	Дуже велике число, ділення на нуль, здобуття квадратного кореня з від’ємного числа і таке інше
Помилки в даних	Невдале визначення діапазону зміни даних

Друкарські помилки	Переплутані близькі за написанням символи, наприклад, цифра 1 та букви I, l
Помилки введення-виведення	Невірне читання вхідних даних, невірне завдання форматів даних

12.10. Чи є відсутність синтаксичних помилок доказом правильності програми?

Зазвичай синтаксичні помилки виявляються на етапі трансляції. Багато ж інших помилок транслятору виявити неможливо, оскільки **транслятору невідомі задуми програміста.**

Відсутність повідомлень машини про синтаксичні помилки є необхідною, але не достатньою умовою, щоб вважати програму правильною. Приклади синтаксичних помилок:

- пропуск знаку пунктуації;
- неузгодженість дужок;
- неправильне формування оператора;
- невірне утворення імен змінних;
- невірне написання службових слів;
- відсутність умов закінчення циклу;
- відсутність опису масиву і таке інше.

12.11. Які помилки не виявляються транслятором?

Існує безліч помилок, які транслятор виявити не в змозі, якщо оператори, які використовуються у програмі, сформовані вірно. Приклади таких помилок.

Логічні помилки:

- невірна вказівка гілці алгоритму після перевірки деякої умови;
- неповний облік можливих умов;
- пропуск у програмі одного або кількох блоків алгоритму.

Помилки у циклах:

- неправильна вказівка початку циклу;
- неправильна вказівка умов закінчення циклу;
- неправильна вказівка числа повторів циклу;
- нескінченний цикл.

Помилки введення-виведення; помилки при роботі з даними:

- неправильне завдання типу даних;
- організація перегляду меншого або більшого об’єму даних, чим потрібний;
- неправильне редагування даних.

Помилки у використанні змінних:

– використання змінних без вказівки їх початкових значень;
– помилкова вказівка одній змінній замість іншої.

Помилки при роботі з масивами:

– масиви заздалегідь не обнулені;
– масиви неправильно описані;
– індекси слідують у неправильному порядку.

Помилки арифметичних операцій:

– невірна вказівка типу змінної (наприклад, цілочисельного замість дійсного);
– невірне визначення порядку дій;
– ділення на нуль;
– здобуття квадратного кореня з від'ємного числа;
– втрата значущих розрядів числа.

Ці помилки виявляються за допомогою тестування.

12.12. У чому полягає супровід програми?

Супровід програм – це роботи, які пов'язані з обслуговуванням програм у процесі їх експлуатації.

Багаторазове використання розробленої програми для вирішення різних завдань заданого класу вимагає проведення додаткових робіт, що пов'язані з доопрацюваннями програми для вирішення конкретних завдань, проведення додаткових тестових прорахунків і таке інше.

Програма, яка призначена для тривалої експлуатації, повинна мати відповідну документацію та інструкцію по її використанню.

Контрольні запитання для самоперевірки

1. Що називають математичною моделлю?
2. Які вснують етапи розв'язання завдань за допомогою комп'ютера?
3. Для чого потрібні відлагодженн та тестування?
4. Що являє собою тест та тестування?
5. Які характерні помилки програмування?
6. Які помилки не виявляються транслятором?
7. У чому полягає супровід програми?

Список літератури

1. Мациевский С. В., Ишанов С. А., Клевцур С. В. М 367 Информатика: Учебное пособие / Калининград: Изд-во КГУ, 2003. – 140 с.
2. Знакомьтесь: компьютер. // Под ред. Курочкина В.М. – М.: ”Мир, 1989. – 240 с.
3. Филичев С.Ф. Информатика – это просто! – М., ЭКОМ, 1999. – 343 с.
4. Шауцукова Л.З. Информатика 10-11. – М.: Просвещение, 2000 г. – 420 с.
5. Компьютеры приобретают разум. // Под ред. Курочкина В.М. – М.: ”Мир, 1989. – 240 с.
6. Кершан Б., Новембер А., Стоун Дж. Основы компьютерной грамотности. – М.: Мир, 1989.
7. Гвоздева В. А. Введение в специальность программиста: учебник. – 2-е изд., испр. и доп. – М: ИД «ФОРУМ»: ИНФРА-М, 2007. – 208 с.
8. Боглаев Ю.П. Вычислительная математика и программирование. – М.: Высшая школа, 1990.
9. Хасэгава Х. Мир компьютеров в вопросах и ответах. В 2-х томах. // Перевод Чернышов Ю. – М.: Мир, 1990. – 120 с, – 152 с.
10. Фролов Г.Д., Кузнецов Э.И. Элементы информатики. – М.: Высшая школа, 1989. – 303 с.
11. Залогова Л. и др. Информатика. Задачник-практикум в 2 т. /под редакцией И.Г. Семакина и Е.К. Хенера. – М., "Лаборатория базовых знаний", 2000. – 304 с, – 280 с.
12. Потопахин В.В. Искусство алгоритмизации. – М.: ДМК Пресс, 2011. – 320 с.

Навчальне видання

Авраменко Валентин Семенович
Салапатов Володимир Іванович

ВСТУП ДО ПРОГРАМНОЇ ІНЖЕНЕРІЇ

Том 1. Історія розвитку. Основні поняття

Навчальний посібник

Друге видання

Загальний редактор *Г. В. Косенюк*

Комп’ютерне верстання

Підписано до друку 11.06.2012. Формат 60x84/16.
Ум. друк. арк. 31,08. Наклад 300 пр. Зам. № 4351

Видавець і виготовлювач
Черкаський національний університет імені Богдана Хмельницького
Адреса: 18000, м.Черкаси, бул.Шевченка, 81, кімн. 117,
Тел. (0472) 45-32-60, факс (0472) 37-22-33,
e-mail: vydav@cdu.edu.ua, http://www.cdu.edu.ua
Свідоцтво про внесення до державного реєстру
суб’єктів видавничої справи ДК №3427 від 17.03.2009 р.