С. М. Довгалець Р. В. Маслій

АЛГОРИТМІЧНІ МОВИ ТА ПРОГРАМУВАННЯ ЧАСТИНА 1. ОСНОВИ ІНФОРМАТИКИ ТА КОМП'ЮТЕРНОЇ ТЕХНІКИ

Міністерство освіти і науки України
Вінницький національний технічний університет

С. М. Довгалець Р. В. Маслій

АЛГОРИТМІЧНІ МОВИ ТА ПРОГРАМУВАННЯ ЧАСТИНА 1. ОСНОВИ ІНФОРМАТИКИ ТА КОМП'ЮТЕРНОЇ ТЕХНІКИ

Затверджено Вченою радою Вінницького національного технічного університету як навчальний посібник для студентів напряму підготовки «Системна інженерія». Протокол № 13 від 3 липня 2008 р.

Вінниця ВНТУ 2009

УДК [007+681.3] (075)
Д 58

Рецензенти:

Р. Н. Квєтний, доктор технічних наук професор
В. М. Лисогор, доктор технічних наук професор
В. І. Месюра, кандидат технічних наук доцент

Рекомендовано до видання Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України

Довгалець С.М., Маслій Р.В.
Д 58 **Алгоритмічні мови та програмування. Частина 1. Основи інформатики та комп'ютерної техніки.** Навчальний посібник. – Вінниця: ВНТУ, 2009. – 116 с.

В навчальному посібнику розглянуті основи обчислювальної техніки, сучасне апаратне та програмне забезпечення комп'ютерів, математичні та логічні основи ЕОМ, також приділено увагу графічному поданню алгоритмів.

Навчальний посібник призначений для студентів напряму підготовки „Системна інженерія".

УДК [007+681.3] (075)

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

AGP – Accelerated Graphics Port (прискорений графічний порт)

BIOS – Basic Input/Output System (базова система введення/виведення)

CPU – Central processing unit (центральний процесор)

CRT – Cathode Ray Tube (катодно-промінева трубка)

FAT – File Allocation Table (таблиця розміщення файлів)

FDD – Floppy Disk Drive (зовнішній накопичувач інформації на гнучкому диску)

HDD – Hard Disk Drive (зовнішній накопичувач інформації на жорсткому диску)

HTML – HyperText Markup Language (мова розмітки гіпертекстових документів)

IDE – Integrated Development Environment (інтегроване середовище розробки)

IEEE – Institute of Electrical and Electronics Engineers (Інститут Інженерів електротехніки та електроніки)

ISA – Industry Standard Architecture (архітектура індустріального стандарту)

JVM – Java Virtual Machine (віртуальна Java машина)

LCD – Liquid Crystal Display (рідкокристалічний дисплей)

PC – Perconal computer (персональний комп'ютер)

PCI – Peripheral component interconnect (з'єднання периферійних компонентів)

RAD – Rapid Application Development (швидка розробка застосунків)

RAM – Random Access Memory (пам'ять з довільним доступом)

UML – Unified Modeling Language (уніфікована мова моделювання)

USB – Universal Serial Bus (універсальна послідовна шина)

АЛП – арифметично-логічний пристрій

АОМ – аналогова обчислювальна машина

АЦП – аналого-цифровий перетворювач

ЕОМ – електронно-обчислювальна машина

ОЗП – оперативно запам'ятовувальний пристрій

ООП – об'єктно-орієнтоване програмування

ОС – операційна система

ПЗ – програмне забезпечення

ПЗП – постійно запам'ятовувальний пристрій

ПК – персональний комп'ютер

ПУ – пристрій управління

РЧ – реальний час

ЦАП – цифро-аналоговий перетворювач

ВСТУП

З часу виникнення обчислювальної техніки у 1940-х роках застосування і використання комп'ютерів (*computers*) розвивається шаленими темпами. Програмне забезпечення відіграє важливу роль практично у всіх аспектах повсякденного життя: державному управлінні, банківській справі і фінансах, освіті, транспорті, індустрії розваг, медицині, сільському господарстві і юриспруденції. Кількість, якість і області застосування комп'ютерних програм (*programs*) різко збільшились. В результаті сотні мільярдів доларів витрачаються на розробку програмного забезпечення (*software*), і від ефективності цих програм залежать заробітки і навіть життя більшості людей.

В кінці 1990-х років область знань повязана з інформаційними технологіями (information technology) дуже сильно розрослася, в зв'язку з цим було прийнято рішення розділити її на чотири основних дисципліни – інформатика (*computer science*), програмна інженерія (software ingeneering), проектування апаратних платформ (hardware ingeneering) та інформаційні системи (information systems).

Даний навчальний посібник присвячений інформатиці, яка є базовою дисципліною для трьох основних дисциплін. Навчальний посібник є першим з серії посібників призначених для курсу "Алгоритмічні мови та програмування".

В першому розділі посібника розглядається історія обчислювальної техніки, архітектура (*architecture*) та структура (*structure*) комп'ютерів, класифікація комп'ютерів, основні відомості про апаратне (*hardware*) та програмне забезпечення та їх взаємодія.

В другому розділі посібника більш детально розглядається апаратне забезпечення комп'ютера: особливості будови і функціонування материнської плати (*motherboard*), процесора (*processor*), пам'яті (*storage*). Також у розділі розглядається відеосистема комп'ютера: особливості функціонування та різновиди моніторів (*monitor*) та відеокарт (*video card*); аудіосистема комп'ютера. Крім цього у розділі розглянуті особливості будови та різновиди пристроїв введення та виведення.

В третьому розділі посібника більш детально розглядається програмне забезпечення. В підрозділі 3.1 розглядаються операційні системи (*operating system*): базова система введення-виведення BIOS (*Basic Input/Output System)*, дається класифікація операційних систем, розглядаються сімейства операційних систем Windows та UNIX, області застосування операційних систем, операційні системи реального часу (*real-time operating system*), файлові системи (*file system*). В підрозділі 3.2 розглядаються різновиди прикладних програм (*application programs*). В підрозділі 3.3 розглядаються найбільш популярні та використовувані мови високого рівня програмування (*high-level of programming languages*),

спеціалізовані мови програмування. В підрозділі 3.4 розглядаються технології створення програмного забезпечення: модульне програмування (*modular programming*), структурне програмування (*structured programming*), низхідне програмування (*top down programming*), об'єктно-орієнтоване програмування (*object-oriented programming*), технології швидкого створення застосувань (*rapid application development*).

В четвертому розділі посібника розглядаються математичні основи ЕОМ. В підрозділі 4.1 розглядаються форми подання даних *(data)*: способи вираження функцій (*function*), неперервне та дискретне подання даних, аналогово-цифрове (*analog-to-digital*) та цифроаналогове (*digital-to-analog*) перетворення (*transformation*) даних. В підрозділі 4.2 розглядаються основні системи числення (*numeral system)*: двійкова (*binary*), вісімкова (*octal*), десяткова (*decimal*), шістнадцяткова (*hexadecimal*), наводяться правила та приклади переведення як цілих так і дробових чисел з однієї системи в іншу. В підрозділі 4.3 розглядаються формати подання даних: числа з фіксованою точкою і плаваючою точкою; звичайний формат, формат подвійної точності та довгий формат; розглядаються прямий, обернений та додатковий коди подання чисел у комп'ютері.

В п'ятому розділі посібника наводяться основи булевої алгебри (*boolean algebra*), розглянуті базиси (*basis*) логічних функцій (*boolean functions*), наведені закони перетворення логічних функцій. Також у розділі наведений геометричний метод мінімізації логічних функцій (*minimizations of boolean functions*) з використанням карт Карно та розглянуті приклади мінімізації функцій з різною кількістю змінних.

В шостому розділі посібника розглянуті властивості алгоритмів (*algorithms*), подані способи написання алгоритмів. Особлива увага у розділі приділяється графічному способу подання алгоритмів, наведені умовні графічні позначення у схемах алгоритмів, правила графічного запису алгоритмів, розглянуті базові структури алгоритмів. Також у розділі наведені приклади написання схем програм до декількох задач з програмування.

Матеріал, наведений у посібнику, може використовуватися при викладанні курсів: "Алгоритмічні мови та програмування", "Обчислювальні методи та застосування ЕОМ", "Системне програмування", а також використовуватися студентами напряму "Системна інженерія", студентами суміжних спеціальностей.

Розділи 1, 3 та 4 написані С. М. Довгальцем. Розділи 2, 5, 6 написані Р. В. Маслієм.

1 ОСНОВИ ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

1.1 Історія обчислювальної техніки

За всіх часів людям потрібно було рахувати. У далекому минулому вони рахували на пальцях чи роблячи насічки на кістках. Згодом з'явилися перші ручні обчислювальні інструменти. Сьогодні найскладніші обчислювальні задачі, як і безліч інших операцій (*operations*), здавалося б не пов'язаних з числами, виконують за допомогою "електронного мозку" – комп'ютера. Жодна інша машина в історії не привнесла в людську діяльність настільки швидких і глибоких змін. Комп'ютери стали органічною складовою сучасного життя, і обійтися тепер без них майже неможливо.

Незважаючи на бурхливий технічний прогрес, закладення фундаменту комп'ютерної революції відбувалося повільно і далеко не просто. Початком цього процесу можна вважати винахід рахівниці понад 1500 років тому у країнах Середземномор'я. Рахівниці виявилися дуже ефективним інструментом і незабаром поширилися по всьому світу, а в деяких країнах використовуються і донині.

Наступним після рахівниць пристроєм, що полегшує множення і ділення чисел та деякі інші розрахунки, стала логарифмічна лінійка, винайдена наприкінці 1620-х років (уперше логарифми були введені в практику після праці шотландця Джона Непера, опублікованої в 1614 р.).

Перший механічний рахунковий пристрій створив у 40-х роках XVII ст. видатний французький математик, фізик, письменник і філософ Блез Паскаль. Підсумовувальна машина Паскаля являла собою ящик з численними шестернями. Інші операції, крім додавання, виконувалися за досить незручною процедурою повторних додавань.

Першу машину, що дозволяла легко виконувати розрахунки, множення і ділення – механічний калькулятор – винайшов у 1673 р. у Німеччині Готфрід-Вільгельм Лейбніц. І механічний, і електромеханічний калькулятори, так само, як і логарифмічну лінійку, використовували донедавна, поки їх не витіснили електронні калькулятори.

З усіх винахідників минулих часів, які зробили той чи інший внесок у розвиток обчислювальної техніки, найближче до створення комп'ютера в сучасному його розумінні підійшов англієць Чарльз Беббідж. У 1822 р. Беббідж опублікував наукову статтю з описом машини, здатної розраховувати і друкувати великі математичні таблиці. Протягом наступного десятиліття Беббідж, невтомно працюючи над своїм винаходом, намагався її реалізувати на практиці. Однак, продовжуючи працювати в цьому напрямі, він прийшов до ідеї створення ще більш потужної машини, яку назвав аналітичною.

Аналітична машина Беббіджа на відміну від своєї попередниці мала не просто розв'язувати математичні задачі одного типу, але й виконувати різноманітні обчислювальні операції відповідно до інструкцій, заданих оператором. Інструкції чи команди вводилися в аналітичну машину за допомогою перфокарт (аркушів картону з пробитими в них отворами), уперше використаних у 1804 р.

Лише через 19 років після смерті Беббіджа один із принципів, що лежить в основі ідеї аналітичної машини, – використання перфокарт – знайшов втілення в діючому пристрої. Це – статистичний табулятор, побудований американцем Германом Холлерітом з метою прискорити оброблення результатів перепису населення, що проводився в США в 1890 р. Після успішного використання табулятора для перепису Холлеріт організував фірму з виробництва табуляторних машин (Tabulating Machine Company). З роками підприємство Холлеріта зазнало чимало змін – злиттів і перейменувань. Остання така зміна відбулася в 1924 р., за 5 років до смерті Холлеріта, коли він створив Міжнародну корпорацію машин для бізнесу (IBM – International Business Machines Corporation). Ще одним чинником, що сприяв появі сучасного комп'ютера, стали праці з двійкової системи числення. Основну лепту у дослідження двійкової системи числення вніс англійський математик-самоучка Джордж Буль. У праці "Дослідження законів мислення" (1854 р.) він винайшов своєрідну алгебру – систему позначень і правил, застосовну до різних об'єктів, – від чисел і букв до речень.

У 1936 р. випускник одного з американських університетів Клод Шеннон довів, що якщо побудувати електричні кола відповідно до принципів булевої алгебри, то вони могли б виражати логічні висловлення, визначати істинність тверджень, а також виконувати складні обчислення, і впритул наблизився до теоретичних основ побудови комп'ютера.

Першу грубу модель обчислювальної машини на електричних схемах побудував Атанасофф (1939 р.). У 1937 р. Джордж Стібіц склав першу електромеханічну схему, яка виконує операцію двійкового додавання. Ще через два роки Стібіц разом з інженером-електриком фірми Семюелем Уільямсом розробили пристрій, здатний виконувати операції додавання, віднімання, множення і ділення комплексних чисел.

Не маючи ніякого уявлення про праці Чарльза Беббіджа і Буля, Конрад Цузе в Берліні почав розробляти універсальну обчислювальну машину, багато в чому подібну до аналітичної машини Беббіджа. Перший варіант машини, названої Z1, був створений у 1938 р. У другому варіанті машини Z2 дані вводилися за допомогою перфорованої фотоплівки.

За фінансової і технічної підтримки фірми IBM Ейкен приступив до розроблення машини, в основу якої лягли ідеї Беббіджа і технологія XX ст. Опису аналітичної машини самим Беббіджем виявилося більше, ніж досить. Машину, названу Марк-1, уперше успішно випробувано на

початку 1943 р. Машина Марк-1, завдовжки майже 17 м і заввишки понад 2,5 м містила близько 750 тис. деталей, з'єднаних проводами загальною довжиною близько 800 м.

Однак машина Марк-1 застаріла ще до того, як була побудована. У 1941 р., майже за два роки до того, як Марк-1 почала працювати, Конрад Цузе побудував діючий комп'ютер – програмно-керований пристрій, оснований на двійковій системі числення. Машина Z3 була значно меншою від машини Ейкена, а її виробництво набагато дешевшим.

Ще в 1936 р. у віці 24 років Алан Тьюрінг у своїй праці описав абстрактний механічний пристрій – "універсальну машину", яка мала справлятися з будь-якою припустимою, теоретично розв'язною задачею – математичною чи логічною. Деякі ідеї Тьюрінга в кінцевому підсумку знайшли відображення в реальних машинах. Спочатку вдалося створити кілька дешифраторів на основі електромеханічних перемикачів. Однак наприкінці 1943 р. було створено набагато потужнішу машину, яка замість електромеханічних реле містила близько 2 000 електронних вакуумних ламп. Цю машину назвали „Колос". Тисячі перехоплених за день ворожих повідомлень уводилися в пам'ять „Колоса" у вигляді символів *(symbols)*, закодованих на перфострічці.

Водночас у США на потребу воєнного часу з'явилась машина Еніак, що за принципами роботи і застосуванням була теоретично наближена до "універсальної машини" Тюрінга. Як і машина Марк-1 Говарда Ейкена, Еніак призначалася для вирішення завдань балістики.

Однак не встигла Еніак поступити в експлуатацію, як Мочлі й Екерт уже працювали на замовлення військових над новим комп'ютером. Наступна модель – машина Едвак – створена на початку 1951 р. була більш гнучкою. Її внутрішня пам'ять містила не тільки дані, але і програму. Істотно й те, що Едвак кодувала дані вже в двійковій системі, що дозволило значно скоротити кількість електронних ламп.

Моріс Уілкс у 1949 р. завершив створення першого у світі комп'ютера з програмами, збереженими в пам'яті. Комп'ютер одержав назву Едсак.

Епоха масового виробництва комп'ютерів почалася з випуску першого комерційного комп'ютера LEO, що використовувався для розрахунку зарплати працівникам чайних магазинів, які належали фірмі Lyons.

Якісно новий етап у проектуванні комп'ютерів настав, коли фірма IBM запустила свою відому серію машин – IBM/360. Машини цієї серії мали різну продуктивність, різний набір пристроїв і призначалися для розв'язання різних задач, однак їх побудовано за єдиними принципами, що істотно полегшувало модернізацію комп'ютерів і обмін програмами між ними.

У колишньому СРСР розроблення комп'ютерів розпочали наприкінці 40-х років. У 1950 р. в Інституті електротехніки АН УРСР у Києві було випробувано першу вітчизняну електронну обчислювальну машину (ЕОМ) на електронних лампах – малу електронну обчислювальну машину (МЕОМ), яку спроектувала група вчених та інженерів під керівництвом академіка С. А. Лебедєва. У 1952 р. під його керівництвом було створено велику електронну обчислювальну машину (ВЕОМ), що після модернізації в 1954 р. мала високу для того часу швидкодію – 10 000 операцій/с.

1.2 Структура і архітектура комп'ютера

Сукупність основних пристроїв комп'ютера і зв'язків між ними називають *структурою комп'ютера*.

Структура комп'ютера залежить від розрядності процесора, способу організації доступу до пам'яті, способів управління зовнішніми пристроями тощо.

Персональні IBM-сумісні комп'ютери, побудовані за принципом відкритої архітектури, мають структуру, спрощений вигляд якої показаний на рис 1.1.

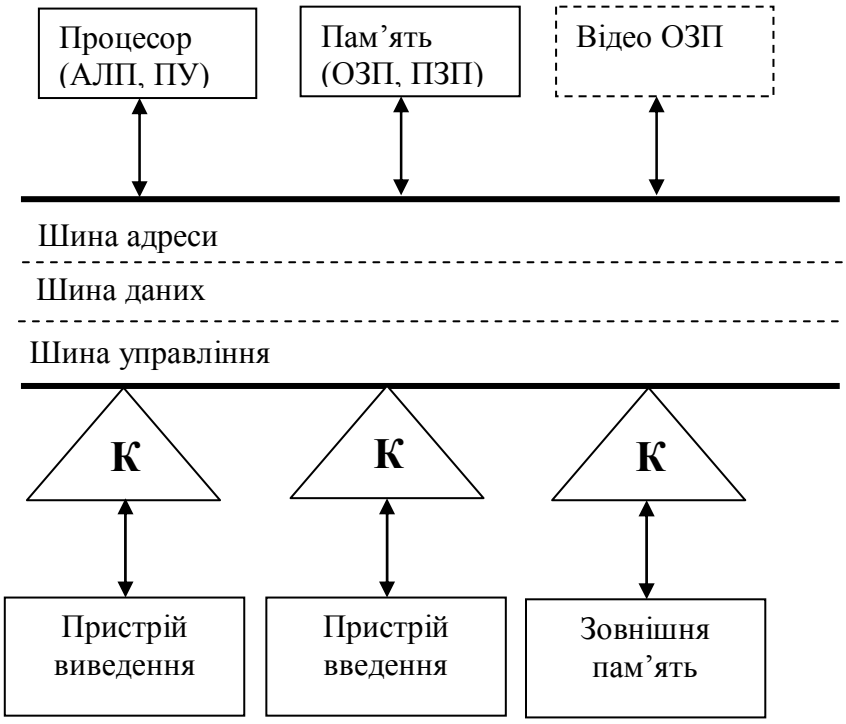


Рисунок 1.1 – Структура комп'ютера

Головною особливістю такої структури є наявність виділеної шини (*bus*) для передачі інформації між функціональними вузлами комп'ютера. Вона складається з трьох частин:

- шина адрес, яка визначає, куди саме направляється інформація по шині;
- шина даних, по якій передається інформація;
- шина управління, яка визначає особливості обміну і синхронізує його.

До шини приєднуються всі пристрої комп'ютера, починаючи від процесора і закінчуючи пристроями введення/виведення (*input/output devices*). Суттєвою особливістю архітектури комп'ютера є наявність спеціалізованих процесорів введення/виведення, які називаються контролерами *(controllers*). На рис. 1.1 контролери зображені літерою **К**. Їх роль полягає в підтримці процесів обміну інформацією для даного пристрою, а також в узгодженні зі стандартною шиною будь-яких зовнішніх пристроїв різних виробників.

Процесор (центральний процесор – *central processing unit* або *CPU*), внутрішня пам'ять і системна шина конструктивно розташовані в окремому блоці, який називають системним. Пристрої зовнішньої пам'яті (це, як правило, накопичувачі на жорстких і гнучких магнітних та оптичних дисках) також розміщують у системному блоці, хоча інколи – і в окремих блоках. Внутрішня пам'ять комп'ютера поділяється на постійну, вміст якої зберігається після вимикання живлення комп'ютера, та оперативну, у якій після вимикання комп'ютера вся інформація втрачається.

Процесор, оперативну та зовнішню пам'ять і всі пристрої введення – виведення під'єднують до системної шини через контролери. Процесор, внутрішню пам'ять, системну шину та контролери розміщують на одному конструктиві, який називають материнською платою.

Пристрої введення та виведення поділяють на дві групи: *стандартні*, до яких належать дисплей та клавіатура (*keyboard*), та *нестандартні* – принтери (*printers*), плотери (*plotters*), сканери (*scanners*), модеми (*modems*), миша (*mouse*) та ін. Нестандартні пристрої отримали ще одну назву – периферійних (*peripheral devices*). Зауважимо, що периферійними пристроями є також накопичувачі на магнітних і оптичних дисках, що конструктивно не входять до складу системного блока.

Архітектурою комп'ютера називають склад і взаємозв'язок основних пристроїв комп'ютера з точки зору програміста. Архітектура комп'ютера визначається операційною системою.

Найбільш поширена архітектура зображена на рис 1.2, відповідає принципам, які були закладені Джоном фон Нейманом в один з перших комп'ютерів „Едвак". За цією концепцією щоб комп'ютер був ефективним і універсальним інструментом він має включати такі компоненти:

- арифметико-логічний пристрій;
- пристрій управління (*control unit*) ;
- запам’ятовувальний пристрій чи пам’ять;
- пристрої введення-виведення інформації.

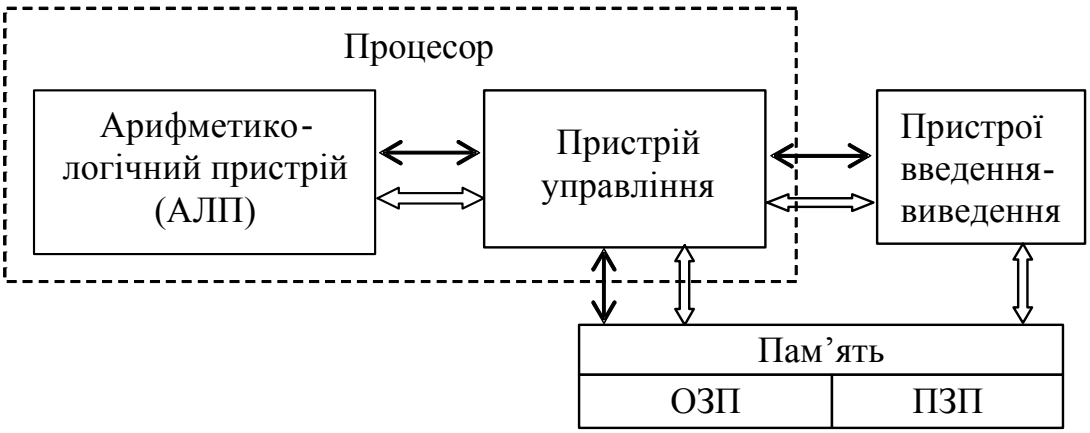


Рисунок 1.2 – Архітектура комп’ютера

Арифметико-логічний пристрій виконує арифметичні та логічні перетворення даних, що надходять до нього.

Пристрій управління автоматично керує процесом оброблення інформації, посилаючи всім іншим пристроям сигнали про виконання тих чи інших дій.

Сукупність арифметико-логічного пристрою та пристрою управління називають *процесором.*

Пам’ять зберігає інформацію, передану з інших пристроїв (зокрема, пристроїв введення), і видає інформацію іншим пристроям комп’ютера, включаючи пристрої виведення.

Пристрої введення/виведення служать для введення даних у машину, виведення результатів і, в разі потреби, для управління процесом оброблення інформації.

Джон фон Нейман також відзначав, що комп’ютер має працювати з двійковими числами, бути електронним, а не механічним пристроєм, і виконувати операції послідовно одну за одною. Принципи, сформульовані фон Нейманом, стали загальноприйнятими тільки тому, що широко застосовувалися увесь час; їх покладено в основу як великих комп’ютерів перших поколінь, так і більш пізніх міні- та мікрокомп’ютерів.

1.3 Класифікація комп’ютерів

Існує досить багато систем класифікації комп’ютерів. Розглянемо деякі з них.

Класифікація за призначенням

Класифікація за призначенням – один з найбільш ранніх методів класифікації. Він пов'язаний з тим, як комп'ютер застосовується. За цим принципом розрізняють великі ЕОМ, міні-ЕОМ, мікро-ЕОМ і персональні комп'ютери (ПК), які в свою чергу поділяють на масові, ділові, портативні, розважальні і робочі станції.

Великі ЕОМ. Це найпотужніші комп'ютери. Їх застосовують для обслуговування дуже великих організацій чи навіть галузей народного господарства. За кордоном комп'ютери цього класу називають мейнфреймами (mainframe). Штат обслуговування великої ЕОМ складає до декількох десятків чоловік. На базі таких суперкомп'ютерів створюють обчислювальні центри, які включають в себе декілька відділів чи груп.

Міні-ЕОМ. Від великих ЕОМ комп'ютери цієї групи відрізняються зменшеними розмірами і, відповідно, меншою продуктивністю і вартістю. Такі комп'ютери використовуються великими підприємствами, науковими установами і деякими навчальними закладами, які поєднують навчальну діяльність з науковою.

Міні-ЕОМ часто застосовують для управління виробничими процесами. Для організації роботи з міні-ЕОМ потрібен спеціальний обчислювальний центр, але з меншим штатом обслуговування ніж для великих ЕОМ.

Мікро-ЕОМ. Комп'ютери даного класу доступні багатьом підприємствам. Організації, які застосовують мікро-ЕОМ, як правило, не створюють обчислювальні центри. Для обслуговування такого класу комп'ютера їм достатньо невеликої обчислювальної лабораторії у складі декількох чоловік.

Персональні комп'ютери (ПК). Ця категорія комп'ютерів інтенсивно розвивалася протягом останніх двадцяти років. ПК призначений для обслуговування одного робочого місця. Не дивлячись на невеликі розміри і відносно невелику вартість, сучасні ПК переважають великі ЕОМ 70-х років, міні-ЕОМ 80-х років і мікро-ЕОМ першої половини 90-х років. Персональний комп'ютер (*personal computer, PC*) спроможний задовольнити більшість потреб малих підприємств і окремих осіб.

Міжнародний сертифікаційний стандарт – специфікація PC99 встановлює такі категорії ПК:

- масовий ПК (Consumer PC);
- діловий ПК (Office PC);
- портативний ПК (Mobile PC);
- робоча станція (Workstation PC);
- розважальний ПК (Entertainment PC).

Більшість ПК, що присутні сьогодні на ринку, попадають під категорію *масових ПК*. Для *ділових ПК* мінімізовані вимоги до засобів відтворення графіки (*graphics*), а до засобів роботи із звуковими даними

вимоги взагалі не висуваються. Для *портативних ПК* обов'язковим є наявність засобів для створення віддаленого доступу, тобто засобів комп'ютерного зв'язку. В категорії *робочих станцій* збільшені вимоги до пристроїв збереження даних, а в категорії *розважальних ПК* – до засобів відтворення графіки.

Класифікація за рівнем спеціалізації

За рівнем спеціалізації комп'ютери поділяють на *універсальні* і *спеціалізовані*. На базі універсальних комп'ютерів можна зібрати обчислювальні системи (*computer system*) довільної конфігурації. Так, наприклад, один і то й же ПК можна використовувати для роботи з текстами, музикою, графікою, фото- і відеоматеріалами.

Спеціалізовані комп'ютери призначені для рішення конкретного пакету задач. До таких комп'ютерів відносяться, наприклад, бортові комп'ютери автомобілів, кораблів, літаків, космічних апаратів тощо. Спеціалізовані міні-ЕОМ, орієнтовані на роботу з графікою, називають *графічними станціями (graphic stations)*. Їх використовують при підготовці кіно- та відеофільмів, а також рекламної продукції. Спеціалізовані комп'ютери, які об'єднують комп'ютери підприємства в одну мережу (*network*), називають *файловими серверами (file server)*. Комп'ютери, які забезпечують передачу інформації між різними учасниками всесвітньої комп'ютерної мережі, називають *мережевими серверами (network server)*.

Класифікація за типорозмірами

Персональні комп'ютери можна класифікувати за типорозмірами. Розрізняють *настільні (desktop)*, *портативні (notebook)* і *кишенькові (palmtop)* моделі.

Настільні моделі розповсюджені найбільш широко. Вони, як правило, знаходяться на робочому місці. Ці моделі відрізняються простотою зміни конфігурації за рахунок нескладного підключення додаткових зовнішніх пристроїв чи встановлення додаткових внутрішніх компонентів.

Портативні моделі зручні для транспортування. Їх використовують бізнесмени, керівники підприємств і організацій, які проводять багато часу в переїздах. З портативним комп'ютером можна працювати при відсутності робочого місця. Особлива привабливість портативних комп'ютерів пов'язана з можливістю їх використання як засобу зв'язку. Підключивши такий комп'ютер до телефонної мережі з будь-якої географічної точки, можна встановлювати зв'язок між цим комп'ютером та центральним комп'ютером своєї організації.

Кишенькові моделі виконують функції „інтелектуальних записних книжок". Вони дозволяють зберігати оперативні дані і отримувати до них швидкий доступ. Деякі кишенькові моделі мають жорстко встановлене

програмне забезпечення, що полегшує роботу, але знижає у виборі гнучкість прикладних програм.

1.4 Програмне забезпечення комп'ютера

Комп'ютер виконує операції під управлінням великого комплексу програм (*programs*), які складають програмне забезпечення процесу обробки інформації. Програмне забезпечення можна розділити на три основні частини: *системне, інструментальне* і *прикладне*.

Системне програмне забезпечення призначене для:

- управління роботою комп'ютера;
- розподілу його ресурсів;
- підтримки діалогу з користувачем – операційні системи;
- автоматизації процесу розробки та відлагодження програм – системне програмування *(system programming)*;
- перекладу мов високого рівня програмування на коди комп'ютера;
- форматування дискет (*diskettes*), архівація файлів тощо – утиліти (*utilities*);
- забезпечення роботи периферійних пристроїв – драйвери (*drivers*).

Інструментальне забезпечення слугує для розробки різних пакетів програм, що застосовуються в різних областях знання. В групу інструментальних програм входять: транслятори (*translator*) з різних алгоритмічних мов (*algorithmic languages*), які переводять текст програми на машинну мову *(machine language)*; відлагоджувачі з допомогою яких знаходять і виправляють помилки, які були допущені при написанні програми; інтегровані середовища розробки, які об'єднують вказані вище компоненти в єдину, зручну для розробки систему.

Прикладне програмне забезпечення – це комплекс програм, заради яких і створювався комп'ютер. Прикладне програмне забезпечення поділяють на прикладне програмне забезпечення *загального призначення,* до якого відносяться ті програми, які широко використовуються різними категоріями користувачів (текстові редактори (*text editors*), електронні таблиці (*spreadsheets*), системи управління базами даних, графічні редактори (*graphics editors*)) та прикладне програмне забезпечення *спеціального призначення*, до якого відносяться програми, які мають специфічне призначення (програми розрахунку руху планет, програми перекладу з староєгипетської мови на сучасні, статистичні обчислення тощо).

Наявність у комп'ютері різноманітних прикладних програм дозволяє виконувати значну частину прикладних задач без безпосереднього програмування користувачем і значно пришвидшує їх розв'язання.

1.5 Апаратне забезпечення комп'ютера

До апаратного забезпечення обчислювальних систем відносяться пристрої і прилади, що утворюють апаратну конфігурацію. Сучасні комп'ютери й обчислювальні комплекси мають блоково-модульну конструкцію – апаратну конфігурацію, необхідну для виконання конкретних видів робіт, можна складати з готових вузлів і блоків.

Усі моделі комп'ютера (від найстарішої до найсучаснішої) мають однаковий *блоковий* принцип компонування.

Основний блок ПК – *системний блок.* Він містить блок електроживлення (*power supply*), кріпильні елементи для материнської (системної) плати, електронних плат і дисководів (*disks drive*).

Стандартним пристроєм виведення ПК є *дисплей (монітор),* а пристроєм введення – *клавіатура.*

За способом розташування пристроїв щодо центрального процесора розрізняють внутрішні й зовнішні пристрої. Зовнішніми, як правило, є більшість пристроїв введення-виведення даних (їх також називають периферійними пристроями) і деякі пристрої, призначені для тривалого зберігання даних.

Деякі пристрої (наприклад, модеми чи жорсткі диски) мають як зовнішні, так і внутрішні варіанти виконання. Крім того, різні класи ПК мають також різне конструктивне виконання. Так, у настільному комп'ютері дисплей, клавіатура і "мишка" – зовнішні пристрої, а портативні комп'ютери являють собою один системний блок, у який вбудовано і дисплей, і клавіатуру, і аналог "мишки".

Узгодження між окремими вузлами й блоками виконують за допомогою перехідних апаратно-логічних пристроїв, названих апаратними інтерфейсами (*interfaces*). Стандарти на апаратні інтерфейси в обчислювальній техніці називають протоколами *(protocols)*. Таким чином, протокол – це сукупність умов, які повинні бути забезпечені розроблювачами пристроїв для успішного узгодження їхньої роботи з іншими пристроями.

Нижче розглядаються основні пристрої комп'ютера та найбільш поширені додаткові пристрої, використовувані в комп'ютері для виконання проектних і науково-інженерних робіт, а також пристрої оброблення і введення-виведення звукових та відеоданих.

1.6 Взаємодія програмного та апаратного забезпечення

Будь-який комп'ютер у процесі роботи оперує не тільки своїми апаратними компонентами, названими *апаратним забезпеченням,* але і збереженими в пам'яті програмами чи програмами, що завантажуються в пам'ять, названими *програмним забезпеченням*.

Засоби програмного забезпечення й апаратні засоби – це два основні компоненти сучасних комп'ютерів. Програмне забезпечення доповнює комп'ютер тими можливостями, які важко чи економічно невигідно реалізовувати тільки апаратними засобами, а також виконує роль посередника між користувачами і комп'ютером, створюючи для користувача зручність взаємодії з комп'ютером.

Використовувані програми висувають певні вимоги до апаратних засобів комп'ютера (наприклад, вимоги до ємності оперативної пам'яті та пам'яті на диску, наявності тих чи тих пристроїв введення-виведення). В свою чергу, більшість апаратних засобів (наприклад, "мишка", сканер чи звукова карта (*audio card*)) потребують для функціонування наявності в пам'яті комп'ютера відповідних програм – *драйверів пристроїв.*

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Що таке структура комп'ютера?
2. Що таке архітектура комп'ютера?
3. Наведіть складові частини виділеної шини у структурі комп'ютера.
4. Призначення контролера у структурі комп'ютера.
5. Які пристрої комп'ютера належать до стандартних пристроїв введення/виведення?
6. Які пристрої комп'ютера належать до нестандартних пристроїв введення/виведення?
7. Які існують класифікації комп'ютерів?
8. Охарактеризуйте класифікацію комп'ютерів за рівнем спеціалізації.
9. Наведіть приклади спеціалізованих комп'ютерів.
10. Охарактеризуйте класифікацію комп'ютерів за типорозмірами.
11. Наведіть категорії ПК згідно із стандартом PC99.
12. Які ви знаєте способи розташування пристроїв щодо центрального процесора?
13. Що таке програмне забезпечення комп'ютера?
14. Що таке апаратне забезпечення комп'ютера?
15. Як ви розумієте термін апаратний інтерфейс?
16. Для чого призначене системне програмне забезпечення комп'ютера?
17. Для чого призначене прикладне програмне забезпечення комп'ютера?
18. Для чого призначене інструментальне програмне забезпечення комп'ютера?
19. Для чого призначені драйвери пристроїв?

2 АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРА

Незважаючи на те, що існують комп'ютери різного класу і різного призначення, їх основні компоненти і принципи компонування несуттєво відрізняються між собою. Тому компоненти комп'ютера надалі будемо розглядати стосовно IBM-сумісних ПК.

Функціонування комп'ютера потребує такого мінімального набору пристроїв:

- материнська плата;
- центральний процесор;
- пам'ять;
- стандартний пристрій виведення – дисплей чи монітор;
- пристрій, що з'єднує дисплей з комп'ютером – відеокарта або відеоконтролер (*video controller*);
- стандартний пристрій введення – клавіатура;
- блок електроживлення.

Оскільки в сучасних операційних системах із графічним інтерфейсом (*graphical interface*) багато операцій виконуються за допомогою "мишки" чи аналогічних їй пристроїв, то один з цих пристроїв також входить у мінімальний склад комп'ютера.

Поряд з основним набором пристроїв комп'ютер, залежно від складу розв'язуваних на ньому задач, містить також додаткові пристрої: принтери, сканери, звукові карти, акустичні системи, мікрофони (*microphones*), модеми тощо.

Розглянемо основні пристрої комп'ютера.

2.1 Материнська плата

Материнська плата – це основний пристрій комп'ютера, що передусім визначає його продуктивність.

Материнська плата містить такі основні компоненти:

- шини;
- базову систему введення-виведення – BIOS;
- кеш-пам'ять;
- системні ресурси (*system resources*) .

Шини – це сукупності ліній (провідників на материнській платі), по яких паралельно й одночасно обмінюються даними компоненти і пристрої комп'ютера. Шину призначено для обміну даними між двома і більше пристроями. Шину, що сполучає тільки два пристрої, називають портом *(port)*.

Зазвичай шина має роз'єм *(connector)* для вмикання зовнішніх пристроїв, які, будучи підключеними, самі стають частиною шини і можуть обмінюватися даними з усіма іншими пристроями.

Шина має власну архітектуру, що містить такі компоненти:

- контролер шини;
- лінії для обміну даними (шини даних);
- лінії для адресації даних (шини адреси);
- лінії для управління даними (шини управління).

Контролер шини керує процесом обміну даними і службовими сигналами.

По *шині даних* відбувається обмін даними між процесором, зовнішніми пристроями й оперативною пам'яттю.

Процес обміну даними можливий лише в тому разі, коли відомі відправник і одержувач цих даних. Кожний компонент комп'ютера, кожний регістр введення-виведення та комірка оперативної пам'яті мають свою адресу і входять у загальний адресний простір комп'ютера. Для адресації до якого-небудь пристрою комп'ютера використовується *шина адреси,* по якій передається унікальний ідентифікаційний код (адреса).

Для того, щоб дані були записані (прочитані) у підключені до шини регістри пристроїв, адреси яких зазначені на шині адреси, потрібні службові сигнали: записування – зчитування, готовності до приймання – передавання даних, підтвердження приймання даних тощо. Усі ці сигнали передаються по *шині управління*.

Стандарти шинного інтерфейсу

Шина визначається алгоритмами, за якими передаються сигнали, правилами інтерпретації сигналів, а також використовуваними мікросхемами (*microcircuits*). Зазначені характеристики визначають стандарт шинного інтерфейсу.

- шину ISA (Industry Standard Architecture);
- шину PCI (Peripheral component interconnect);

Шина ISA. Перший поширений стандарт шинного інтерфейсу – шину ISA розробила фірма IBM під час створення комп'ютера IBM PCAT (1984р.) Ця 16-бітова шина з тактовою частотою (*clock rate*) 8,33 МГц допускає установлення як 8-бітових, так і 16-бітових плат розширення (із пропускною здатністю відповідно 8,33 і 16,6 Мбайт/с).

Шина PCI. Шину PCI розробила фірма Intel з участю ряду інших фірм у 1993 році для нового високопродуктивного процесора Pentium. Останній стандарт PCI – PCI 2.2 визначає як 30-розрядну шину з тактовою частотою 33 МГц і піковою пропускною здатністю 133 Мбайт/с, так і 64-розрядні шини з тактовими частотами 33 і 66 МГц і піковими пропускними здатностями відповідно 266 і 533 Мбайт/с.

Стандарти підключення пристроїв

Для підключення внутрішніх і зовнішніх пристроїв до мостів шини PCI використовують такі основні шини й інтерфейси:

- шину AGP (Accelerated Graphics Port);
- шину USB (Universal Serial Bus);
- шину IEEE 1394 (Fire Wire);
- інтерфейс IDE (ATA);
- інтерфейс SCSI;
- паралельний порт;
- послідовний порт;
- порт PS/2;

Шина AGP. Незважаючи на велику швидкість шини PCI, її можливостей стає недостатньо в умовах зростаючого навантаження на відеосистему (відеокарту і монітор), оскільки реалізація тривимірної графіки і відеоданих потребує передавання великих обсягів даних між монітором, процесором і оперативною пам'яттю.

Стандарт на шину AGP – канал передавання даних між відеокартою і RAM (Random Access Memory) фірма Intel у 1997 розробила на основі стандарту PCI. Цей стандарт призначено для прискорення виведення даних на відеокарту і підвищення продуктивності комп'ютера у процесі оброблення тривимірних зображень (*images*) без установлення спеціалізованих дорогих відеокарт.

Оскільки шина AGP 32-розрядна і її тактова частота дорівнює тактовій частоті системної шини, то в стандартному режимі її пропускна здатність (*bandwidth*) 266 Мбайт/с у два рази перевищує пропускну здатність шини PCI.

Шина USB. Специфікація послідовної універсальної шини USB, розроблена в 1995 р. Для підключення низькошвидкісних зовнішніх пристроїв типу клавіатури і "мишки", забезпечувала швидкість обміну даними 1,5 Мбіт/с з довжиною кабелю до 3 м. Пізніше було прийнято специфікацію USB 1.1, за якою швидкість передавання збільшилась до 12 Мбіт/с, а максимальна довжина кабелю до 5 м. У новій версії специфікації USB – USB 2.0, названій також високошвидкісною USB, швидкості обміну даними по шині становлять 1,5 Мбіт/с, 12 Мбіт/с і 480 Мбіт/с. Специфікація USB 2.0 підтримує зворотню сумісність зі специфікацією USB 1.1. Висока швидкість передавання даних дозволяє підключати до комп'ютера по інтерфейсу USB принтери, сканери й зовнішні пристрої пам'яті, а також обмінюватися даними між комп'ютерами.

Паралельний порт. Порти USB для обміну даними з низько- та середньошвидкісними зовнішніми пристроями з'явилися порівняно недавно. До цього для введення-виведення даних на ці пристрої використовувалися паралельний (*parallel*) і послідовний (*serial*) порти. Хоча зовнішній пристрої поступово переводяться на інтерфейси USB і FireWire, паралельний і послідовний порти входять до складу материнської плати.

У паралельному порту одночасно передаються відразу 8 біт (1 байт) інформації. Тому роз'єм паралельного порту містить 8 ліній для передавання даних, а комп'ютер з двоспрямованим паралельним портом – додатково 8 ліній роз'єму для приймання даних.

Послідовний порт. Паралельний порт забезпечує досить високу швидкість передавання, оскільки дані передаються по байтах. Але, якщо довжина кабелю велика або обмін даними не дуже інтенсивний, зручнішим виявляється послідовний порт.

Послідовний порт передає в одному напрямі одночасно один біт даних. Дані можуть передаватися через цей порт як від комп'ютера до зовнішнього пристрою, так і навпаки.

Характеристики материнської плати

Можливості комп'ютера щодо оброблення даних і модифікації значною мірою визначаються використовуваною в ньому материнською платою. Тому, вибираючи плату, треба враховувати її характеристики, що істотно впливають на вибір інших компонентів комп'ютера.

Основні характеристики материнської плати такі:

- тип гнізда для підключення процесора і кількість гнізд;
- максимальна ємність оперативної пам'яті;
- інтерфейси і кількість слотів розширення;
- склад додаткових компонентів;
- склад і кількість роз'ємів для підключення зовнішніх пристроїв;
- формфактор.

Тип гнізда материнської плати для процесора визначає, які моделі процесорів і з якою тактовою частотою можна вмикати в нього. У документації материнської плати зазвичай зазначають також максимальну тактову частоту підключеного процесора (модель процесора з більшою тактовою частотою не можна підключати до материнської плати). У платах для багатопроцесорних комп'ютерів (*multiprocessor computers*) наявні декілька гнізд для вмикання процесорів.

Інтерфейси і кількість слотів розширення визначають можливості материнської плати щодо вмикання пристроїв. У сучасних материнських платах обов'язковим є слот AGP (у документації зазначається, яку специфікацію підтримує слот – AGP 2.0 чи AGP 3.0). Деякі материнські плати містять вбудовані відеоадаптери. Материнська плата має також декілька слотів PCI.

Компонування материнської плати

Компоненти материнської плати можна поділити на чотири групи:

- набір мікросхем *(chipset)* високого ступеня інтеграції;
- додаткові компоненти;
- роз'єми для вмикання пристроїв;

- допоміжні (сервісні) блоки і вузли.

Набір мікросхем або *чіпсет* визначає функціональні можливості материнської плати: тип і кількість установлюваних процесорів, тип і ємність оперативної пам'яті, тактову частоту системної шини і деякі інші характеристики.

Кількість мікросхем і набір функцій, реалізованих як в окремих мікросхемах набору, так і у всьому наборі мікросхем, істотно залежить від фірми-виробника. Крім того, створюються нові версії наборів мікросхем з новими можливостями і в іншому конструктивному оформленні.

Взаємодія між компонентами і пристроями комп'ютера, підключеними до різних шин, забезпечується за допомогою мостів, реалізованих на одній з мікросхем відповідного набору. Так, головний міст забезпечує інтерфейс із процесором, оперативною пам'яттю і шиною AGP (цей міст часто називають *північним мостом).* Головний міст зв'язаний з мостом ATA/PCI/USB *(південним мостом),* що, у свою чергу, забезпечує зв'язок із пристроями по інтерфейсах PCI, ATA, Serial ATA і USB. Цей міст реалізовує механізм прямого доступу до пам'яті з використанням управління шиною, а також інтерфейс з контролером IEEE 1394.

Контролер IEEE 1394 керує обміном даних між підключеними по інтерфейсу IEEE 1394 пристроями і комп'ютером.

Мікросхема контролера Super I/O містить контролери клавіатури, гнучкого диска, паралельного і послідовного портів, а також інтерфейс із BIOS.

Мікросхема BIOS містить пам'ять тільки для зчитування і пам'ять CMOS.

2.2 Центральний процесор

Процесор, як відомо, це пристрій, що поєднує виконання арифметичних і логічних операцій, пов'язаних з управлінням роботою комп'ютера чи його окремих пристроїв. Процесор, що виконує збережені в оперативній пам'яті команди і керує роботою всього комп'ютера, називають центральним або головним процесором – *CPU.* Слід зазначити, що деякі пристрої сучасних комп'ютерів (наприклад, звукова карта), для збільшення швидкодії можуть мати власні спеціалізовані процесори. Надалі під процесором будемо розуміти центральний процесор комп'ютера.

Процесори IBM-сумісних комп'ютерів

Центральні процесори для IBM-сумісних настільних комп'ютерів виготовляють фірми *Intel, AMD* і *VIA*. Процесори з однаковою чи майже однаковою архітектурою утворюють *сімейства процесорів* (під архітектурою процесора розуміють його структуру і склад компонентів).

Центральний процесор підключається до системної шини і схеми електроживлення материнської плати.

У результаті розвитку Intel 80386 створено процесор Intel 80486DX (1989 р.). Процесор містив 1,2 млн. транзисторів (*transistors*); його тактову частоту в останній моделі сімейства Intel 80486DX4 підвищено до 100 МГц.

Новим етапом у проектуванні і виробництві процесорів став процесор Pentium (1993р.). Тепер виготовляють тільки процесори сімейства Pentium IV (четвертої модифікації Pentium). Тактова частота першої моделі Pentium становила 66 МГц; частоту останніх моделей Pentium IV збільшено до 3,4 ГГц, а кількість транзисторів – з 3,1 до 178 млн. У процесорах сімейства Pentium використовується 64-розрядна шина даних; ємність пам'яті, що адресується, збільшилася до 64 Гбайт.

Відгалуженнями від процесорів сімейства Pentium стали процесори сімейства Xeon, призначені для багатопроцесорних серверів. Перший процесор цього сімейства Pentium II Xeon (1998 р.) містив 7,5 млн. транзисторів і працював з тактовою частотою 100 МГц. Модель Pentium Xeon (2004 р.) містить 169 млн транзисторів і працює з тактовою частотою 3 ГГц.

Іншим відгалуженням процесорів сімейства Pentium стали процесори сімейства Celeron, що являють собою спрощений і, отже, більш дешевий варіант процесорів Pentium. Перша модель процесора цього сімейства (1998 р.) працювала з тактовою частотою 266 МГц і містила 7,5 млн. транзисторів, а останні моделі працюють з тактовою частотою понад 2,8 ГГц.

Крім цього, фірма *Intel* розробила процесор Pentium M (2003 р.) зі зниженим споживанням енергії, що містить 77 млн. транзисторів і працює з тактовою частотою до 1,4 ГГц. Цей процесор спеціально призначено для портативних комп'ютерів.

Перший 64-розрядний процесор фірми *Intel* – Itanium з'явився в 2001 р. Він містив 25 млн. транзисторів і працював з тактовою частотою 733 МГц. Адресна шина цього процесора також стала 64-розрядною, що дозволило адресувати пам'ять ємністю до 1 024 Тбайт. Розрядність шини даних у цьому процесорі збільшилася до 128 байт. Модель Itanium 2 містить 410 млн. транзисторів і працює з тактовою частотою 1,5 ГГц.

Якщо перший процесор Intel 4004 виконано з використанням 10-мікронної технології, то останні моделі процесорів Intel уже виготовляли за 0,13-мікронною технологією. У 2004 р. випущено перший процесор Pentium IV на основі 0,09-мікронної технології.

Процесори AMD – це простіші і дешевші аналоги (клони) процесорів фірми *Intel.* Так, 32-розрядні процесори AMD Athlon XP і AMD Athlon MP є аналогами відповідно Pentium 4 і Xeon, а 64-розрядні процесори AMD Athlon 64 FX і AMD Opteron – аналогами Itanium 2 (перший з них

призначено для настільних і блокнотних ПК, другий – для серверів і робочих станцій).

32-розрядні процесори, які виготовляє корпорація *VIA Technologies* за 0,13-мікронною технологією, відрізняються малими розмірами і низьким енергоспоживанням.

Процесор VIA C3 (спадкоємець процесорів фірми *Cyrix* – колишнього конкурента фірми *Intel),* тактова частота якого до 1,4 ГГц, процесор VIA Antaur, тактова частота якого близько 1,2 ГГц, призначені для використання в невеликих настільних і портативних комп'ютерах з підвищеними вимогами до захисту оброблення даних.

Процесор VIA Eden-N – найменший за розмірами (15x15 мм) клон процесора Intel. Він має тактову частоту 1 ГГц і призначений для материнської плати VIA Nano-iTX з форм-фактором 12x12 см, що передбачається використовувати в портативних комп'ютерах.

Набори команд центрального процесора

Основне функціональне призначення процесора полягає у виконанні інструкцій з оброблення даних – команд машинної мови процесора. Сукупність цих команд утворює *набір команд* цього процесора.

Основний набір команд процесорів Intel називають набором x86. Цей набір уперше був застосований в процесорах Intel 8086 і призначений для виконання таких команд:

- переміщення даних;
- двійкової арифметики (додавання, віднімання, множення і ділення) для чисел з фіксованою точкою (цілих чисел);
- десяткової арифметики для двійково-десяткових чисел;
- логічних операцій (НЕ, І, АБО і виключне АБО);
- зсуву бітів;
- перевірки і модифікації бітів;
- управління (команди умовного і безумовного переходів, циклу, виклику процедур з поверненням з процедур);
- роботи з рядками;
- введення-виведення;
- системних.

У 16-розрядних процесорах Intel 8086, Intel 8088 і Intel 80286 ці команди виконувалися з байтами і словами. У 32-розрядних процесорах, починаючи з Intel 80386, додано команди роботи з подвійними словами та деякі інші команди.

Багатопроцесорні комп'ютери

Останнім часом набули поширення *багатопроцесорні комп'ютери,* тобто комп'ютери, які містять кілька процесорів (зазвичай це потужні сервери).

Функціонування багатопроцесорної системи потребує виконання таких умов:

- материнська плата має підтримувати кілька процесорів, тобто мати додаткові роз'єми для установлення процесорів і відповідний набір мікросхем;
- процесор має працювати в багатопроцесорній системі;
- операційна система має підтримувати декілька процесорів (такими операційними системами є Windows NT/XP і UNIX).

Багатопроцесорна система найбільш ефективна у випадках, коли вона використовується багатозадачними операційними системами і прикладними програмами, створеними за допомогою спеціальних засобів, що дозволяють виконувати паралельне оброблення даних.

У процесі одночасної роботи декількох процесорів операційна система розподіляє різні задачі між процесорами. Існують два режими роботи багатопроцесорних систем – асиметричний і симетричний.

2.3 Пам'ять

Пам'ять сучасного комп'ютера має ієрархічну структуру. Це зумовлено тим, що вартість пам'яті істотно зростає зі збільшенням її швидкодії.

Можна виділити три рівні пам'яті комп'ютера: *регістрову, оперативну* і *зовнішню*.

Регістрова пам'ять – найбільш швидкодіюча, але найменша за ємністю пам'ять (зазвичай декілька сот байтів). Регістрова пам'ять містить деякі системні дані (наприклад, адресу наступної команди, яку буде виконувати процесор), вихідні дані і результати виконання деяких команд процесора, а також часто використовувані в програмі дані.

Оперативна пам'ять має значно більшу ємність (від декількох десятків мегабайтів до декількох гігабайтів) і використовується для тимчасового зберігання програм та даних під час їх оброблення. Так, для запуску програми вона попередньо завантажується з пристрою зовнішньої пам'яті і тільки потім починає виконуватися.

Пристрої регістрової і оперативної пам'яті є енергозалежними, тобто у разі вимикання комп'ютера дані, що знаходяться на цих пристроях, губляться. Пристрої зовнішньої пам'яті є енергонезалежними і призначені для тривалого зберігання даних. Цей рівень пам'яті найменш швидкодіючий і об'єднує велику групу пристроїв, що істотно відрізняються як за обсягом, так і за швидкістю доступу до даних. Пристрої зовнішньої пам'яті можуть бути зі *знімними носіями даних* (наприклад, дисководи гнучких дисків і дисководи *CD-ROM)*. Носії даних інших пристроїв жорстко зафіксовані, наприклад, у дисководах жорстких дисків (хоча існують знімні дисководи жорстких дисків). Ємність і швидкодія

пристроїв зі знімними носіями зазвичай значно нижчі від ємності пристроїв з фіксованими носіями.

Для збільшення швидкості доступу до даних у комп'ютері використовується буферна і кеш-пам'ять.

Буферну пам'ять використовують для збільшення швидкодії комп'ютера у разі обміну із зовнішніми пристроями. Оскільки швидкість доступу до зовнішнього пристрою значно нижча від швидкості оброблення даних у процесорі, процесор простоює, очікуючи закінчення введення-виведення. Щоб цього не відбувалося, фрагмент даних для читання попередньо зчитується з пристрою в буферну пам'ять. Для записування даних фрагмент даних спочатку передається в буферну пам'ять, а потім з буферної пам'яті на пристрій. Збільшення швидкодії відбувається за рахунок того, що обмін між буферною пам'яттю і зовнішнім пристроєм виконується без участі процесора (він у цей час може виконувати інші процеси). Буферна пам'ять реалізується або апаратно в електронній схемі пристрою введення-виведення, або програмно в оперативній пам'яті комп'ютера.

Кеш-пам'ять крім функції буферизації, зберігає найчастіше використовувані дані або ті дані, до яких недавно відбувалося звернення. Ці дані поміщають у більш швидкодіючу пам'ять, ніж пам'ять, використовувана для зберігання цих даних. Так, доцільно таблиці розміщення файлів на дисках помістити в оперативну пам'ять, оскільки під час оброблення даних звернення до цієї таблиці відбувається досить часто. Такі самі дані оперативної пам'яті поміщають у кеш-пам'ять, виконану на більш швидкодіючих елементах.

Внутрішня пам'ять

До внутрішньої пам'яті комп'ютера належать:

- оперативна, чи системна пам'ять (RAM);
- пам'ять BIOS (ROM BIOS);
- пам'ять пристроїв комп'ютера.

За способом зберігання даних пам'ять поділяють на тимчасову енергозалежну і постійну енергонезалежну пам'ять.

В оперативну пам'ять завантажуються як системні програми, зокрема модулі операційної системи, так і прикладні програми користувачів. Будь-яку програму можна виконувати тільки з оперативної пам'яті. Історично оперативну пам'ять називають пам'яттю з довільним доступом RAM на відміну від зовнішньої пам'яті з послідовним доступом SAM (Serial Access Memory), реалізованої на магнітній стрічці. Однак така назва тепер уже не може служити означенням тільки оперативної пам'яті, оскільки всі пристрої пам'яті (за винятком нагромаджувачів на магнітній стрічці) – це пристрої з довільним доступом. Проте таку назву оперативної пам'яті можна досить часто знайти в літературі.

Основні характеристики елементів пам’яті такі:

- тип пам’яті;
- технологія виготовлення;
- глибина адресного простору, розрядність і ємність;
- максимальна пропускна здатність;
- тимчасова діаграма доступу;
- тимчасова діаграма робочого циклу;
- тип корпусу;
- тип модуля оперативної пам’яті;
- наявність контролю правильності даних;
- наявність у модулі буферної пам’яті;
- напруга електроживлення;

Тип пам’яті (DDR SDRAM, QDR-II SRAM чи FLASH пам’ять) однозначно визначає як вид пам’яті (динамічна, статична чи постійна), так і її основні можливості.

Зовнішня пам’ять

Зовнішня пам’ять чи *зовнішні запам’ятовувальні пристрої* використовуються для довгострокового зберігання інформації. Обсяги даних, збережені в таких пристроях, в 100 чи 1000 разів перевищують ємність оперативної (внутрішньої) пам’яті комп’ютера.

Зовнішні запам’ятовувальні пристрої можна класифікувати за такими ознаками:

- способом зберігання і доступу до даних (магнітоелектричні чи магнітні, магнітооптичні, оптичні, електричні);
- видом носія даних (дискета, жорсткий диск, магнітна стрічка, магнітооптичний диск, компакт-диск, DVD-диск і Flash-пам’ять);
- режимом доступу до пам’яті (пам’ять з довільним доступом RAM (дискета, усі види дисків і Flash-пам’ять, пам’ять з послідовним доступом – SAM (магнітна стрічка));
- типом носіїв даних – знімними носіями (дискета, компакт-диск, DVD-диск, знімні жорсткі диски, магнітна стрічка), незнімними носіями (жорсткий диск, магнітооптичний диск);
- можливістю перезаписування даних на перезаписувані носії (дискета, жорсткий і магнітооптичний диски, диски CD-RW, DVD-RW, DVD-RAM і DVD+RW, магнітна стрічка і Flash-пам’ять), носії з однократним записом («тільки для зчитування») (диски CD-ROM, CD-R, DVD-ROM, DVD-R і DVD+R);
- типом пристрою зберігання даних: внутрішні (вбудовуються в системний блок) і зовнішні (підключаються до системного блока через один з портів чи за допомогою плати розширення).

Пристрої зовнішньої пам’яті поділяють на такі основні групи:

- дисководи гнучких магнітних дисків (FDD – Floppy Disk Drives);

- дисководи жорстких дисків (HDD – Hard Disk Drives);
- пристрої зчитування-записування на магнітну стрічку;
- магнітооптичні дисководи (magneto-optical drives);
- оптичні дисководи (optical drives);
- пристрої зовнішньої Flash-пам’яті.

Дисководи FDD використовують дискети розміром 3,5 дюймів. Гнучкий магнітний диск ємністю 1,44 Мбайт поміщено у жорсткий пластмасовий корпус. Виріз у корпусі для доступу до дискети у звичайному стані закритий пересувною металевою чи пластмасовою шторкою. Для заборони записування на дискету передбачено вікно захисту. Отвір у правому верхньому куті служить ознакою того, що дискета має ємність 1,44 Мбайт. Перші дискети 3,5" ємністю 720 кбайт не мають отвору. Дисководи для дискет надвисокої щільності ED ємністю 2,88 Мбайт не одержали поширення.

Перші дисководи ZIP появилися в 1995 р. і були розраховані на дискети ємністю 100 Мбайт. Нові дисководи зчитують і записують дискети ємністю 250 і 750 Мбайт, але підтримують також і старий формат – 100 Мбайт. Швидкість обертання диска в дисководі ZIP становить 3000 обертів за хвилину.

Структура дисковода ZIP майже така сама, як і дисковода FDD. Великі ємності дискети досягаються за рахунок збільшення щільності записування і використання більш точної системи позиціювання головок зчитування – записування.

Сьогодні випускаються декілька десятків типів *дисководів жорстких дисків – HDD*, які називають також *вінчестерами*. Однак принципи функціонування більшості дисководів однакові. Основні елементи конструкції типового дисковода жорстких дисків:

- магнітні диски;
- головки зчитування – записування;
- механізм приводу головок;
- двигун приводу дисків;
- гермоблок ;
- друкована плата з електронною схемою управління;
- роз’єми.

Для зберігання двійкових даних оптичні носії використовують зміну відображення світла від матеріалу носія. Так, повне чи майже повне відображення світла можна зіставити з одиницею, а відсутність відбитого від носія світла – з нулем. Як *оптичну зовнішню пам’ять* комп’ютера використовують два види носіїв – компакт-диски та DVD диски.

Компакт-диски, названі також дисками CD, використовують для записування аудіоданих у цифровому вигляді та програють на спеціальних пристроях – плеєрах CD.

Аналогічно диски DVD використовують для записування відеоданих, вони зчитуються на плеєрах DVD, підключених до телевізора.

Останнім часом *Flash-пам'ять* стали використовувати не тільки як внутрішню пам'ять, але і як зовнішню, підключену до комп'ютера по інтерфейсу USB. Як уже зазначалось, перевага цього інтерфейсу полягає в тому, що пристрій можна вмикати і вимикати без перезавантаження комп'ютера.

Пристрій Flash-пам'яті містить такі компоненти:

- одну чи кілька мікросхем Flash-пам'ять в корпусі TQPF;
- контролер, що керує доступом до мікросхем Flash-пам'яті;
- адаптер USB.

Пристрої Flash-пам'яті мають невеликі розміри і найрізноманітніші форми корпусів. Звичайний пристрій має індикатор, що загоряється під час доступу до даних, та перемикач, що дозволяє чи забороняє записування у пам'ять. Крім того, деякі пристрої мають перемикач, що дозволяє чи забороняє зчитування даних без використання пароля. Пристрій Flash-пам'яті можна використовувати як завантажувальний пристрій.

2.4 Відеосистема комп'ютера

Монітори

Дисплей або монітор – обов'язковий пристрій ПК.

Монітор, так само як і телевізор, призначений для виведення чорно-білих або кольорових рухомих зображень.

Виведені на екран монітора зображення є растровими, тобто являють собою прямокутну матрицю точок, названих пікселами. За такого подання враховують обмежену здатність ока, яка, починаючи з деякої відстані, сприймає дві близько розміщені точки як одну. Тобто, матриця на екрані монітора сприймається як цілісне зображення. У зв'язку з цим до монітора комп'ютера ставляться суворіші вимоги щодо кількості пікселів і їх відстані між ними, ніж до екрана телевізора, оскільки зображення на екрані розглядають зазвичай з відстані 50...70 см, а рекомендована відстань до екрана телевізора – не менше 3 м.

У разі виведення чорно-білих зображень пікселі зображення можуть бути або чорного, або білого кольору.

Для формування кольорових зображень використовують адитивну кольорову модель RGB, що містить три колірні компоненти: червону, зелену і синю. Тоді кожна точка зображення формується з трьох точок, кожна з яких задає інтенсивності червоного, зеленого чи синього кольорів. При цьому враховують ще одну особливість зору: якщо в зображенні наявні близько розміщені кольорові деталі, то, починаючи з деякої відстані, око не розрізняє кольору окремих деталей, а сприймає їх як

деякий сумарний колір (так, дві близько розміщені точки – червона і зелена – на великій відстані буде сприйматися як одна жовта точка).

Під час формування рухомого зображення і в моніторах комп'ютерів, і на екранах телевізорів та кіно враховується інерційність світлового подразника сітківки ока (після закінчення дії світлового подразника час зберігання світлового порушення становить 0,4...1,0 с). Тому рухомі зображення передаються як послідовність нерухомих зображень (кадрів), що швидко змінюються і відображають окремі фази руху. За досить високої частоти змінювання кадрів окремі кадри перестають сприйматися, і рух на екрані стає плавним.

Тепер для ПК використовують три основні типи моніторів*:*

- CRT-монітори;
- LCD-монітори;
- плазмові монітори.

CRT-монітори. Монітори на основі електронно-променевої трубки – CRT-монітори (Cathode Ray Tube – катодна променева трубка) містять такі основні компоненти:

- електронно-променеву трубку;
- електронну схему управління;
- кнопки управління;
- роз'єми або з'єднувальні кабелі;
- корпус.

Електронно-променева трубка – це скляна трубка, що нагадує велику колбу з пробкою без повітря усередині.

Рідкокристалічний монітор, чи *LCD-монітор* (Liquid Crystal Display) містить такі самі компоненти, що й CRT-монітор, однак формують пікселі зображення не пучки електронів, а *рідкі кристали.* Ці речовини названі так тому, що вони зазвичай знаходяться в рідкому стані, але при цьому мають властивості кристалічних тіл. Фактично це рідини, що мають *анізотропію* (неоднорідність в різних напрямах) властивостей (зокрема, оптичних), пов'язаних з упорядкованістю орієнтації молекул. Під впливом електрики молекули рідких кристалів, що мають довгасту форму, можуть змінювати свою орієнтацію і внаслідок цього змінювати властивості світлового променя, що проходить крізь них. Уперше застосували рідкі кристали в чорно-білих (точніше, у чорно-сірих) дисплеях для калькуляторів та годинників, а потім – у моніторах для портативних комп'ютерів. Останнім часом LCD-монітори дедалі більше використовують і в настільних комп'ютерах.

Плазмові монітори. Плазмові монітори містять такі самі компоненти, що й CRT-монітор, але зображення формується за допомогою плазмових екранів, основаних на використанні електричного розряду в плазмі.

Основні характеристики моніторів:

- розмір екрана по діагоналі;
- форма екрана;
- тип маски (тільки для CRT-моніторів);
- крок точки; роздільна здатність;
- кут огляду; яскравість;
- коефіцієнт контрастності;
- інерційність;
- інтерфейс і роз'єми;
- частота регенерації;
- режим розгортки;
- підтримання Plug&Play;
- споживана потужність;
- відповідність стандартам TCO.

Діагоналлю екрана монітора, як і телевізора, називають відстань між лівим нижнім і правим верхнім кутами екрана. Цю відстань вимірюють у дюймах (позначають символом "). Однак для CRT-моніторів під діагоналлю екрана виробники зазвичай вказують розмір діагоналі електронно-променевої трубки. Оскільки трубку укладено в корпус, розмір екрана стає трохи меншим від розміру трубки (деякі виробники вказують обидва розміри). Так, для монітора 17" (43,18 см) дійсний розмір екрана становить 40,55 см.

Відеокарти

Відеокарта чи *відеоадаптер* здійснює безпосереднє управління монітором. Вона передає сигнали управління електронній схемі монітора, тобто контролює процес формування зображення на екрані.

Конструктивно відеокарта являє собою плату, що вставляється в слот розширення PCI чи AGP.

Основні компоненти сучасної відеокарти такі:

- Video BIOS;
- набір мікросхем;
- відеопам'ять;
- RAMDAC;
- роз'єми підключення до шини і зовнішніх пристроїв.

Операції із зображеннями на екрані монітора виконуються за допомогою спеціальних команд. Набір таких команд міститься в постійному запам'ятовувальному пристрої (Flash-пам'яті – Video BIOS). Сучасні відеокарти підтримують також стандарт Plug&Play, тому в Video BIOS містяться відомості про модель, виробника і параметри відеокарти.

Набір мікросхем (chipset) визначає можливості відеокарти. Зазвичай в його склад входять 64-розрядний чи 128-розрядний спеціалізований суперконвеєрний процесор та набір апаратних засобів як для оброблення відеозображень різних форматів, так і для оброблення векторної графіки.

Найпотужніші процесори фактично є багатопроцесорними, оскільки містять кілька незалежних процесорів (по одному на кожний конвеєр).

Для оброблення відеоданих на комп'ютері вихідні дані зазвичай вводять з відеопристроїв: телевізора, відеокамери, відеомагнітофона чи відеоплеєра, або безпосередньо з телеантени (наземної чи супутникової), або з виходу системи кабельного телебачення. Крім того, відеодані можуть бути створені (за допомогою спеціалізованих програмних засобів) безпосередньо в комп'ютері і збережені в зовнішній пам'яті. Результат оброблення відеоданих часто виводиться на екран телевізора чи відеомагнітофона (наприклад, під час створення відеокліпів чи рекламних роликів).

Пристрої оброблення відеоданих виконують такі основні функції:

- приймання аналогових чи цифрових відеоданих від зовнішніх відеопристроїв або безпосередньо з телеантени чи телевізійного кабелю;
- відображення прийнятих відеоданих у реальному часі на екрані монітора і виведення звукового супроводу на звукову карту;
- "захоплення" окремого кадру чи фрагментів (відеокліпів) прийнятих відеоданих.

Відеопам'ять являє собою пам'ять довільного доступу (RAM). Для відеокарт без графічного прискорювача використовували такі самі моделі динамічної пам'яті, що й для оперативної пам'яті, або моделі пам'яті, спеціально розроблені для зберігання відеоданих, наприклад, VRAM (Video RAM), WRAM (Window RAM), SGRAM (Synchronous Graphics RAM) і MDRAM (Multi-bank RAM). Ці моделі відрізняються від моделей оперативної пам'яті підвищеною пропускною здатністю, потрібною для швидкого оброблення зображень. Вимоги до ємності та швидкодії пам'яті нових моделей відеокарт (із прискорювачем) істотно підвищились, тому у відеокарті зазвичай використовують пам'ять DDR SDRAM і DDR 2 SDRAM або спеціально розроблену для графічних прикладних задач графічну пам'ять SGDDR3 DRAM. Особливістю використання пам'яті у відеокартах є 128-бітова чи 256-бітова шина даних між пам'яттю і графічним процесором, що здатна швидко передавати великі обсяги даних.

Роз'єми відеокарти можуть, крім роз'ємів для підключення монітора, містити також роз'єми для введення-виведення відеозображень у різних телевізійних форматах, підключення зовнішнього DVD-плеєра й інших пристроїв.

Основні характеристики відеокарт: підтримувані роздільні здатності (по горизонталі та вертикалі); частота регенерації; підтримувані режими роботи (текстовий чи графічний); кількість кольорів; тип шини (PCI чи AGP); ємність відеопам'яті; підтримуваний інтерфейс (інтерфейси); додаткові можливості.

У текстовому режимі відеокарта виводить тільки символи кодування *(coding)* ANSI. Кожний символ подається у вигляді прямокутної

матриці (розміром, наприклад, 9x14 пікселів), у якій за допомогою пікселів різних кольорів (наприклад, чорного і білого) зображується заданий піксел. Раніше подання (образи) символів зберігалися в постійній пам'яті відеокарти, тепер їх формує комп'ютер. Текстовий режим тепер використовують рідко, здебільшого для розв'язання старих прикладних задач операційної системи MS-DOS. У графічному режимі зображення на екрані монітора формується процесором і відеокартою по пікселях.

2.5 Аудіосистема комп'ютера

Характеристики звуку змінюються невпинно, тобто є аналоговими величинами. Усередині комп'ютера звукові дані подаються в дискретному (цифровому) вигляді. Перетворення звуку з аналогової форми в цифрову (на вході) і обернене перетворення (на виході) виконує основний компонент звукової системи – *звукова карта.* Для введення аналогового звуку використовують *мікрофон,* а для виведення – *акустичну систему.* Ці три компоненти й утворюють базову звукову систему комп'ютера. Крім них, за професійної роботи зі звуком, можна використовувати магнітофони, CD-плеєри, DVD-плеєри, музичні клавіатури, синтезатори й інші пристрої введення-виведення.

2.6 Пристрої введення

Клавіатуру (keyboard) призначено для введення в комп'ютер інформації від користувача.

Основні характеристики клавіатури: технологія фіксації клавіш; інтерфейс підключення; дотримання вимог до ергономіки, випромінювання, безпеки й екології; наявність додаткових клавіш і пристроїв.

Сучасні клавіатури підключають до материнської плати через *інтерфейс PS/2,* однак цей інтерфейс дедалі більше витісняється інтерфейсом USB. Існують також і безпровідні клавіатури, підключені до комп'ютера з використанням інфрачервоного випромінювання чи технології Bluetooth.

З популярністю графічних інтерфейсів великого поширення набули *маніпулятори (pointing devices)* – пристрої введення, що керують положенням курсора на екрані та дозволяють виконувати одну чи декілька фіксованих команд у точці перебування курсора. Існують такі основні типи маніпуляторів:

- "мишка";
- трекбол;
- джойстик;
- сенсорна панель;

- сенсорний екран;
- графічні планшети.

Маніпулятор “мишка” одержала свою назву завдяки формі та принципу роботи: вона «бігає» під управлінням користувача по поверхні робочого столу.

Сканер – це пристрій введення, що перетворює зображення в його цифрову форму (по точках) і передає цей образ у комп’ютер. Зображення, що вводиться, може бути текстом, малюнком, фотографією і навіть тривимірним об’єктом невеликої висоти.

Основні компоненти сканера:

- джерело світла;
- світлочутливі елементи (сенсори);
- алфавітно-цифровий перетворювач.

Джерело світла спрямовує світловий потік на сканований об’єкт (у деяких типах сканерів як джерело світла використовується природне освітлення).

Світловий потік, відбиваючись від об’єкта (непрозорого) чи проходячи через об’єкт, потрапляє на *сенсори*, що перетворюють значення інтенсивності падаючого на них світла в аналогові значення електричної напруги.

Сенсор може просто фіксувати наявність і відсутність світла в певній точці об’єкта (чорно-біле сканування). Якщо сенсор фіксує загальну інтенсивність падаючого на нього світла, то таке сканування називають «сірим». Для відображення кольору об’єкта використовують модель RGB. У цьому разі для вимірювання інтенсивності кожного кольорового компонента (червоного, зеленого і синього) застосовують окремий сенсор.

Сканери мають такі основні характеристики:

- тип сканера;
- використовувана технологія сканування;
- типи оброблюваних зображень;
- глибина кольору;
- можливість роботи з прозорими і непрозорими оригіналами;
- розмір сканованої ділянки;
- роздільна здатність;
- динамічний діапазон;
- глибина різкості;
- тип інтерфейсу з комп’ютером;
- швидкість сканування;
- додаткові можливості.

Тепер випускають *типи сканерів,* що розрізняються за способом переміщення зчитувального пристрою сканера і паперу один відносно одного. Це такі сканери:

- ручні;

- барабанні;
- планшетні;
- рулонні;
- роликові;
- плівкові (сканери слайдів);
- проекційні.

2.7 Пристрої виведення

Плотери, як і принтери, призначені для одержання твердих копій даних. Назва пристрою походить від слова plot (креслити), і дійсно перший тип плотерів – перові плотери призначалися для виведення креслень, зображень чи текстів на папір за допомогою друкарського вузла (на відміну від принтера, що друкує по точках). У друкарських вузлах використовуються спеціальні фломастери, чорнильні і кулькові ручки, а також інші пристрої, що забезпечують різну ширину ліній, насиченість, колірну палітру й інші параметри.

Крім способу виведення, плотери відрізняються від принтерів ще й тим, що дані виводяться на носій великого формату: ISO A2 (42x59,4 см), A1 (59,4x84 см), A0 (84x111,8 см). У США використовують набір форматів ANSI, у якому формат C (43,2x55,9 см) відповідає формату A2, формат D (55,9x86,3 см) – формату A1, а формат B (86,3x111,8 см) – формату A0. Деякі моделі плотерів можуть виводити дані на носії ще більших форматів, ніж A0.

Використовують такі типи плотерів:
- олівцево-перові;
- електростатичні;
- прямого виведення зображення;
- струминні;
- світлодіодні;
- різальні;
- перові.

Друкувальні пристрої комп'ютерів чи *принтери (printers)* використовують для одержання "твердої" копії (hard copy) файлів у пам'яті комп'ютера. Основний вид носія для одержання "твердої" копії – паперові аркуші різних форматів. Однак деякі типи сучасних принтерів дозволяють виводити текст і зображення на інші носії (наприклад, прозорі плівки).

Перші принтери для комп'ютерів були створені на основі друкарської машинки. Для друкування літери важельці з положенням букв, цифр та інших знаків ударяли по паперу через фарбувальну стрічку і залишали на ній відбиток (таку технологію друкування називають *ударною* чи *контактною).* Ці принтери мали невисоку швидкість виведення і могли

друкувати тільки одним шрифтом, хоча пізніше з'явилися принтери типу "селектрик" зі змінним шрифтом і вищою швидкістю друкування.

Тепер використовують такі основні типи принтерів:

- матричні принтери *(matrix printer)* – звичайні і лінійні (рядкові);
- струминні і світлодіодні принтери – звичайні і портативні (мобільні);
- лазерні принтери – чорно-білі і кольорові;
- твердочорнильні принтери;
- сублімаційні принтери;
- воскові принтери;
- автохромні принтери.

Матричний принтер діє так само, як друкарська машинка, але виводить текст чи зображення на папір по точках.

Матричний принтер містить такі основні компоненти:

- картридж із фарбувальною стрічкою;
- каретку;
- блок переміщення каретки;
- валик;
- блок управління.

Розміри CRT і LCD-моніторів не дозволяють показувати зображення на екрані великій кількості людей, наприклад, під час проведення лекцій чи презентацій.

Вирішити цю проблему можна застосуванням *проекторів (projector)*, на які безпосередньо (замість монітора чи поряд з монітором) подається сигнал з відеокарти комп'ютера.

Існує два типи проекторів зображень: дзеркальні та діапроектори.

У *дзеркальних проекторах (overhead projectors)* світло, що проходить крізь прозорий оригінал чи відбите від непрозорого оригіналу, проектується дзеркалом на великий екран. Для виведення зображень з комп'ютера замість прозорого оригіналу використовують панель LCD, на яку подаються сигнали з виходу відеокарти. Панель LCD являє собою виконаний окремо LCD-монітор із кнопками управління, який за допомогою кабелю підключений до роз'єму відеокарти (через спеціальний перехідник можна підключати до роз'єму відеокарти і монітор, і панель).

Панелі LCD дедалі більше витісняються мультимедійними проекторами, що дуже подібні до звичайних діапроекторів для перегляду слайдів, тому їх часто називають діапроекторами. Існує кілька типів таких проекторів, але найпоширеніші такі:

- проектори LCD;
- полісиліконові проектори;
- проектори DLP.

Основний елемент *проектора LCD* – мініатюрний рідкокристаліч-ний екран, виконаний на основі технології TFT. Світло від проекційної

лампи великої потужності потрапляє на лінзу-конденсор, що перетворює світловий потік з розбіжного в паралельний, і після проходження крізь об'єктив зображення проектується на екран.

Більш високу яскравість і якість зображення забезпечують проектори, що базуються не на проходженні світлового потоку через об'єкт, а на відображенні світлового потоку від об'єкта (при цьому нагрівання об'єкта істотно нижче). Існують кілька технологій використання відбитого світлового потоку в проекторах, але найбільшого поширення набула технологія цифрового оброблення світла *DLP* (Digital Light Processing), якою передбачено цифрові мікродзеркальні пристрої *DMD* (Digital Micromirror Devices). Проектори, виконані за цією технологією, називають проекторами DLP.

У проекторах DLP панель TFT замінено панеллю, що складається з безлічі елементів DMD. Кожний елемент DMD являє собою мініатюрне дзеркало (розміром близько одного мікрона), що або відбиває світло, яке перпендикулярно падає на нього, або у разі попадання на нього сигналу управління повертається на невеликий кут і відхиляє світло.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Що називають мінімальним набором пристроїв комп'ютера?
2. Наведіть основні компоненти материнської плати.
3. Що називають шиною?
4. Наведіть стандарти шинного інтерфейсу.
5. Що таке базова система введення-виведення комп'ютера?
6. Наведіть основні характеристики материнської плати.
7. Яке призначення головного процесора комп'ютера?
8. Де застосовуються багатопроцесорні комп'ютери?
9. Які існують види пам'яті комп'ютера?
10. Що таке регістрова пам'ять комп'ютера?
11. Що таке буферна пам'ять комп'ютера?
12. На які основні групи поділяються пристрої зовнішньої пам'яті? Які їх особливості?
13. Дайте класифікацію зовнішних запам'ятовувальних пристроїв. Наведіть приклади.
14. Дайте характеристику основних типів моніторів.
15. Яке призначення відеоадаптера?
16. Які компоненти утворюють аудіосистему комп'ютера?
17. Які існують пристрої введення?
18. Наведіть основні характеристики сканерів.
19. Які існують пристрої виведення?
20. Наведіть основні характеристики проекторів.

3 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРА

3.1 Операційні системи

Операційна система являє собою комплекс системних і службових програмних засобів. З одної сторони вона опирається на базове програмне забезпечення (ПЗ) комп'ютера, яке входить в його систему BIOS, з іншої сторони вона сама є опорою для ПЗ більш високих рівнів – прикладних і більшості службових застосувань.

Основна функція всіх ОС – посередницька. Вона полягає в забезпеченні декількох видів інтерфейсів:

- інтерфейсу між користувачем і програмно-апаратними засобами комп'ютера (інтерфейс користувача);
- інтерфейсу між програмним і апаратним забезпеченням (апаратно-програмний інтерфейс);
- інтерфейсу між різними видами програмного забезпечення (програмний інтерфейс).

Всі ОС спроможні забезпечити як пакетний, так і діалоговий режим роботи з користувачем. Пакетний режим ОС автоматично виконує задану послідовність команд. Суть діалогового режиму полягає в тому, що ОС знаходиться в очікуванні команди користувача, і отримавши її приступає до виконання, а виконавши чекає чергову команду. Діалоговий режим роботи оснований на використанні переривань процесора і переривань BIOS. Опираючись на ці апаратні переривання ОС створює свій комплекс системних переривань.

За реалізацією інтерфейсу користувача відрізняють *неграфічні* і *графічні ОС*. Неграфічні ОС реалізують інтерфейс командного рядка. Основним пристроєм управління в даному випадку є клавіатура. Управляючі команди вводяться в поле командного рядка. Виконання команди починаються після її підтвердження, наприклад натиснення клавіші ENTER. Прикладом неграфічної ОС може служити MS-DOS. Графічні ОС реалізують більш складний тип інтерфейсу, в якому як орган управління крім клавіатури може використовуватися маніпулятор "мишка" чи адекватний пристрій позиціювання. Робота з графічною ОС основана на взаємодії активних і пасивних елементів управління. Прикладом графічної ОС може служити ОС з сімейства Windows, наприклад Windows XP.

Всі ОС забезпечують свій автоматичний запуск. Для дискових ОС в спеціальній(системній) області диска створюється запис системного коду. Звертання до цього коду здійснюють програми, які знаходяться в BIOS. Завершуючи свою роботу вони дають команду на завантаження і виконання вмісту системної області диска.

Базова система введення-виведення BIOS

Базова система введення-виведення BIOS комп'ютера – це програма, що служить інтерфейсом між апаратним забезпеченням комп'ютера і операційною системою. Пристрої сучасних комп'ютерів також містять свої BIOS, що, у свою чергу, служать інтерфейсом між BIOS комп'ютера і відповідним пристроєм.

Програма BIOS комп'ютера містить такі компоненти:

- програму початкового завантаження;
- тестову програму перевірки системи Post;
- програму конфігурації комп'ютера CMOS Setup;
- драйвери пристроїв.

Програма BIOS розміщується в модулі пам'яті ROM на материнській платі. Ця пам'ять енергонезалежна, тобто її вміст зберігається після вимикання комп'ютера. Дані про конфігурацію комп'ютера можна змінити, тому вони зберігаються в модулі пам'яті RAM. У цій пам'яті зберігаються також поточні дата та час. Ця пам'ять реалізується на напівпровідниках, виконаних з використанням технології КМОН(CMOS). Оскільки ця пам'ять енергозалежна, тобто її вміст губиться під час вимикання електроживлення, до складу материнської плати входить акумуляторна батарея, що забезпечує автономним електроживленням пам'ять CMOS.

3.1.1 Класифікація операційних систем

Операційні системи можуть розрізнятися особливостями реалізації внутрішніх алгоритмів управління основними ресурсами комп'ютера (процесорами, пам'яттю, пристроями), особливостями використаних методів проектування, типами апаратних платформ, областями використання і багатьма іншими властивостями.

Від ефективності алгоритмів управління локальними ресурсами комп'ютера залежить ефективність всієї мережної ОС в цілому. Тому, характеризуючи мережеву ОС, часто приводять важливі особливості реалізації функцій ОС із управління процесорами, пам'яттю, зовнішніми пристроями автономного комп'ютера. Так, наприклад, в залежності від особливостей використаного алгоритму управління процесором, операційні системи поділяють на *багатозадачні* і *однозадачні*, *багатокористувацькі* і *однокористувацькі*, на системи, які підтримують багатониткову обробку, і системи, які не підтримують її, на багатопроцесорні і однопроцесорні системи.

За числом задач, які одночасно виконуються, ОС можуть бути поділені на два класи:

- однозадачні (MS-DOS, MSX).
- багатозадачні (ОС EC,OS/2, UNIX, Windows 95).

Однозадачні ОС в основному виконують функцію подання користувачу віртуальної машини, роблячи більш простим і зручним процес взаємодії користувача з комп'ютером. Однозадачні ОС включають методи управління периферійними пристроями, методи управління файлами, методи спілкування з користувачем.

Багатозадачні ОС крім вище згаданих функцій, управляють розділенням використаних ресурсів, таких як процесор, оперативна пам'ять, файли і зовнішні пристрої.

За числом користувачів ОС, які працюють одночасно, поділяють на:

- однокористувацькі (MS-DOS, Windows 3.x, ранні версії OS/2);
- багатокористувацькі (UNIX, Windows NT).

Головною відмінністю багатокористувацьких систем від однокористувацьких є наявність методів захисту інформації кожного користувача від несанкціонованого доступу інших користувачів. Слід зауважити, що не кожна багатозадачна система є багатокористувацькою, і не кожна однокористувацька ОС є однозадачною.

Найважливішим розділяючим ресурсом є процесорний час. Спосіб розподілення процесорного часу між кількома одночасно існуючими в системі процесами значною мірою визначає специфіку ОС. Серед багатьох існуючих варіантів реалізації багатозадачності можна виділити дві групи алгоритмів:

- невитискальна багатозадачність (NetWare, Windows 3.x).
- витискальна багатозадачність (Windows NT, OS/2, UNIX).

Основною різницею між витискуючим і невитискуючим варіантами багатозадачності є ступінь централізації механізму планування процесів. В першому випадку механізм планування процесів повністю зосереджений в операційній системі, а в другому – розподілений між системою та прикладними програмами. При невитискальній багатозадачності активний процес виконується до тих пір, поки він сам, за власною ініціативою, не віддасть управління операційній системі для того, щоб та вибрала з черги інший готовий до виконання процес. При витискальній багатозадачності рішення про переключення процесора з одного процесора на інший приймається операційною системою, а не самим активним процесом.

Важливою особливістю операційних систем є можливість розпаралелювання обчислень в рамках однієї задачі. Багатониткова ОС розділяє процесорний час не між задачами, а між їх окремими гілками (нитками).

Іншою важливою особливістю ОС є відсутність або наявність в ній методів підтримки багатопроцесорної обробки – мультипроцесування. Мультипроцесування призводить до ускладнень всіх алгоритмів управління ресурсами.

В наш час стає загальноприйнятим введення в ОС функції підтримки багатопроцесорної обробки даних. Такі функції є в операційних системах

Solaris 2.x фірми Sun, Open Server 3.x компанії Santa Crus Operations, OS/2 фірми IBM, Windows NT фірми Microsoft і NetWare4.1 фірми Novell.

Багатопроцесорні ОС класифікуються за способом організації обчислювального процесу в системі з багатопроцесорною архітектурою: асиметричні ОС і симетричні ОС. Асиметрична ОС виконується тільки на одному з процесорів системи, розділяючи прикладні задачі між іншими процесорами. Симетрична ОС повністю децентралізована і використовує всі процесори, поділяючи їх між системними і прикладними задачами.

Раніше були розглянуті характеристики ОС пов'язані з управлінням одним типом ресурсів – процесором.

Важливий вплив на вигляд операційної системи в цілому, на можливості її використання в тій чи іншій області, показують особливості і інших підсистем управління локальними ресурсами – підсистем управління пам'яттю, файлами, пристроями введення-виведення.

Специфіка ОС проявляється і в тому, яким чином вона реалізовує мережеві функції: розпізнання і перенаправлення в мережу запитів до видалених ресурсів, передача повідомлень по мережі, виконання видалених запитів. При реалізації функцій виникає комплекс задач, пов'язаних з розподільним характером зберігання і обробки даних в мережі: введення довідкової інформації про всі доступні в мережі ресурси і сервери, адресація взаємодіючих процесів, забезпечення прозорості доступу, тиражування даних, підтримка безпеки даних.

3.1.2 Сімейство операційних систем Windows

На цей день сімейство ОС Windows фірми Microsoft у всіх реалізаціях, безперечно, вважається найпоширенішою ОС ПК: у світі понад 150 млн. IBM PC-сумісних комп'ютерів, і систему Windows установлено більше, ніж на 100 млн. з них. Проаналізуємо деякі версії цієї ОС.

Операційна система Windows NT 4.0

Windows NT 4.0 (була створена в 1996 р.) – це не просто чергова версія популярної ОС. Вона являє собою основу для нового покоління програмних продуктів, орієнтованих на роботу в мережі Internet.

Зовні Windows NT 4.0 аналогічна системі Windows 95. Єдина ознака, що дозволяє з першого погляду розрізнити ці дві системи, це стартове меню, де зазначено, в якому середовищі відбувається робота. До пакета входить низка прикладних програм: Internet Information Server 2.0, Index Server, FrontPage, Internet Explorer, Domain Name System (DNS) Server, Proxy Server і Internet Resource Center, усі пакети Service Pack, Plus та низка додаткових утиліт, серед яких є як нові, наприклад, Administrative Wizards чи Imager, так і вдосконалені версії старих програм, наприклад Task Manager.

Що стосується безпеки, то Windows NT 4.0 спроектована з урахуванням вимог стандартів безпеки точно так само, як і її попередники. Архітектурні зміни не торкнулися підсистеми захисту інформації, що, як і інші підсистеми, виконується у вигляді окремого процесу у режимі користувача.

Операційна система Windows 2000

Windows 2000 – ОС, основана на технології Windows NT, була створена групою програмістів під керівництвом Дейва Катлера, який раніше працював у фірмі DEC над проектом VMS.

Windows 2000 – 32-розрядна ОС із пріоритетною багатозадачністю та поліпшеною реалізацією роботи з пам'яттю. В основі системи лежать ті самі принципи, що колись забезпечили успіх NT:

- *сумісність* (compatibility). Система має звичний інтерфейс ОС сімейства Windows, підтримку файлових систем NTFS5, NTFS4, FAT16 та FAT32. Більшість застосувань, написаних для MS-DOS, Win9x, NT4, а також деякі програми під OS/2 та POSIX запускаються і функціонують без проблем. При проектуванні Windows 2000 враховувалася можливість роботи системи в різних мережевих середовищах, тому в постачання входять засоби для роботи в UNIX та Novell-мережах;
- *переносимість* (portability). Система працює на різних процесорах сімейства x86 виробництва Intel та AMD. Реалізація підтримки МП інших архітектур можлива, але вимагає деяких зусиль;
- *масштабування* (scalability). У системі реалізовано підтримку технологій для роботи симетричної мультипроцесорної системи (SMP) та COW (Cluster Of Workstations). Кількість МП при використанні SMP може досягати 32 (можливе збільшення до 64);
- *система безпеки* (security) цілком задовольняє специфікації у цій галузі;
- *розподілена обробка* (distributed processing). Windows 2000 має вбудовані в систему мережеві можливості, що забезпечує можливість зв'язку з різними типами комп'ютерів завдяки наявності різноманітних транспортних протоколів та технології клієнт-сервер;
- *надійність і стійкість до збоїв* (reliability and robustness). Архітектура ОС захищає застосування від ушкодження одного іншим і самою ОС. При цьому використовується структурована обробка критичних ситуацій на всіх архітектурних рівнях, що включає відновлювану файлову систему NTFS і забезпечує захист за допомогою вбудованої системи безпеки та удосконалених методів управління пам'яттю;
- *локалізація* (localization). Система надає можливості для роботи в багатьох країнах світу національними мовами, що досягається застосуванням стандарту ISO Unicode;

- *розширюваність* (extensibility). Завдяки модульній побудові системи стає можливим додавання нових модулів на різні архітектурні рівні ОС. Існує чотири модифікації постачання: Windows 2000 Professional, Windows 2000 Server, Windows 2000 Advanced Server і Windows 2000 Data Center. Відрізняються вони одна від одної, по-перше, кількістю служб і програм, які входять до постачання, по-друге, ступенем підтримки апаратного забезпечення. Наприклад, перша модифікація не підтримує більше 2 МП, друга – вже 4 МП, третя – 8 МП, а четверта – 64.

Операційна система Windows XP

Windows XP – це потужна операційна система, в основі якої лежить Windows 2000. У Windows XP зроблена спроба, об'єднати дві, що раніше існували незалежно, вітки Windows 9x і NT. Назва XP походить від англійської *experience* (досвід, професійний). На відміну від попередньої системи Windows 2000, яка поставлялася як в серверному, так і в клієнтському варіантах, Windows XP є виключно клієнтською системою. Її серверним варіантом є випущена пізніше система Windows Server 2003. Windows XP і Windows Server 2003 побудовані на основі одного і того ж ядра операційної системи, в результаті їх розвиток і оновлення йде більш менш паралельно. У Windows XP з'явилася можливість використовувати «Visual Styles», що дозволяють змінити графічний інтерфейс користувача. Система надає можливості для роботи в багатьох країнах світу національними мовами, що досягається застосуванням стандарту ISO Unicode.

3.1.3 Сімейство операційних систем UNIX

UNIX – це операційна система, яка спочатку розроблялася протягом 1969-1970-х років групою співробітників підрозділу Bell Labs корпорації AT&T, яка включала Кена Томпсона, Денніса Рітчі та Дугласа Макілроя. ОС UNIX є прикладом винятково вдалої реалізації простої мультипрограмної ОС з багатьма користувачами. Сьогодні існує безліч різних UNIX-систем, які, в свою чергу, об'єднуються в сімейства. В їх розробці в різний час брали участь AT&T, деякі комерційні фірми, а також некомерційні організації. UNIX стала першою операційною системою, майже повністю написаною мовою програмування високого рівня, а саме Сі, що суттєво спростило встановлення системи на інші архітектури. Перші версії UNIX були розраховані на машини без диспетчера пам'яті. Процеси завантажувалися в єдиний адресний простір. Ядро системи розміщувалося в нижніх адресах ОЗП, починаючи з адреси 0. ОС UNIX створювалася перш за все для професіоналів, і тому ніколи не містила зручного графічного інтерфейсу. З цієї причини UNIX, насамперед, має просту, але потужну командну мову і незалежну від пристроїв файлову систему.

3.1.4 Області застосування операційних систем

Багатозадачні ОС поділяються на три типи відповідно до критеріїв ефективності :

- системи пакетної обробки (ОС ЕС);
- системи розподілення часу (UNIX, VMS);
- системи реального часу (QNX, RT/11).

Системи пакетної обробки призначені для розв'язування задач обчислювального характеру які не потребують швидкого результату. Головною метою і критерієм ефективності системи пакетної обробки є розв'язування максимального числа задач за одиницю часу. Для досягнення цієї мети використовується така схема функціонування: спочатку формується пакет задач, кожне завдання має вимоги до системних ресурсів; з цього пакета задач формується множина задач, які одночасно виконуються. Вибір нового завдання з пакета задач залежить від внутрішньої ситуації, яка склалася в системі, тобто обрані "вигідні" задачі. Тому в таких ОС не можна гарантувати виконання того чи іншого завдання протягом певного часу. Взаємодія користувача з обчислювальною машиною, на якій встановлена система пакетної обробки, зводиться до того, що він приносить завдання до оператора і в кінці дня після виконання всього пакета завдань отримує результат.

Системи розподілу часу призначені для виправлення основних недоліків систем пакетної обробки – ізоляція користувача-програміста від процесу виконання цієї задачі. Кожному користувачу системи розподілу часу дається термінал, з якого він може обмінюватися інформацією зі своїми програмами. В системах розподілу часу кожній задачі виділяється квант процесорного часу. Якщо квант досить невеликий, то у всіх користувачів, які працюють на одній машині, складається враження, що кожен з них одноосібно використовує машину. Критерієм ефективності системи поділу часу є не максимальна пропускна властивість, а зручність і ефективність роботи користувача.

Вимоги *систем реального часу (РЧ)* не задовольняють ОС загального призначення (MS-DOS. Windows, UNIX та ін.). Для виконання цих вимог і створюються ОС РЧ.

Як відомо, система РЧ має давати відгук на будь-які непередбачені зовнішні впливи протягом передбачуваного інтервалу часу. Для цього мають бути забезпечені такі властивості:

- обмеження часу відгуку, тобто після настання події реакція на неї гарантовано відбудеться до наперед встановленого крайнього терміну. Відсутність такого обмеження розглядається як серйозний недолік ПЗ;
- одночасність обробки; навіть якщо настає більше однієї події одночасно, усі тимчасові обмеження для всіх подій мають бути витримані. Це означає, що у системі РЧ має бути забезпечений паралелізм. Паралелізм

досягається використанням кількох процесорів у системі і/або багатозадачного підходу.

Прикладами систем РЧ є: системи управління атомними електростанціями або технологічними процесами, медичний моніторинг, управління озброєнням космічної навігації, розвідки, управління лабораторними експериментами, управління автомобільними двигунами, робототехніка, телеметричні системи управління, системи антиблокування гальм, системи сигналізації тощо.

ОС загального призначення відрізняються від ОС РЧ тим, що перед ними ставляться різні завдання, які докорінно відрізняються. Наприклад, ОС загального призначення, особливо багатокористувацькі, такі як UNIX, орієнтовані на оптимальний розподіл ресурсів комп'ютера між користувачами і задачами (системи поділу часу). У ОС РЧ подібна задача відходить на другий план – усе відступає перед головним завданням – встигнути зреагувати на події, що відбуваються на об'єкті.

Основні вимоги до ОС РЧ це *мультипрограмність* та *багатозадачність*. Крім цього ОС РЧ має активно використовувати переривання для диспетчеризації.

Задачі РЧ висувають свої вимоги до обчислювально-керуючих систем, у тому числі до ОС, в яких реалізовано ПЗ РЧ. Система РЧ має встигнути відреагувати на подію, що відбулася на зовнішньому об'єкті протягом часу, критичного для цієї події. Величина критичного часу для кожної події визначається об'єктом і самою подією і, природно, може бути різною, але час реакції системи має бути передбачуваним (обчисленим) при створенні системи. Відсутність реакції в передбачений час вважається помилкою для систем РЧ. Залежно від типу зовнішніх подій час відгуку однієї й тієї ж системи РЧ може змінюватися в широких межах. Наприклад, від 500 мс до 5 хв.

Мережева ОС QNX є потужною ОС, що дозволяє проектувати складні програмні системи, які працюють у РЧ як на одному комп'ютері, так і в локальній обчислювальній мережі. Вбудовані засоби ОС QNX забезпечують підтримку багатозадачного режиму на одному комп'ютері і взаємодію задач, що виконуються паралельно на різних комп'ютерах, які працюють у середовищі локальної обчислювальної мережі. Основною мовою програмування у системі є мова Сі. Основне операційне середовище відповідає стандартам POSix-інтерфейсу. Це дозволяє з невеликими доробками перенести необхідне накопичене ПЗ до операційного середовища ОС QNX для організації його роботи у середовищі розподіленої обробки.

Деякі операційні системи можуть поєднувати в собі властивості систем різного типу, наприклад, частина задачі може виконуватись в режимі пакетної обробки, а частина – в режимі реального часу і в режимі

розподілення часу. В такому випадку режим пакетної обробки часто називають фоновим режимом.

3.1.5 Файлові системи

Файлова система – це частина операційної системи, призначення якої полягає в тому, щоб забезпечити користувача зручним інтерфейсом при роботі з даними, які зберігаються на диску, і забезпечити спільне користування файлами декількома користувачами і процесами. Принцип організації файлової системи – табличний. Поверхня жорсткого диску розглядається як тривимірна таблиця, вимірами якої є номери *поверхні*, *циліндра* і *сектора*. Під циліндром розуміють сукупність всіх доріжок, що належать різним поверхням, і які знаходяться у спеціальних таблицях розміщення файлів (FAT-таблицях).

В широкому розумінні поняття "файлова система" включає:

- сукупність всіх файлів на диску;
- набори структур даних, які використовуються для управління файлами, такі, наприклад, як каталоги файлів, дескриптори файлів, таблиці розподілення вільного і зайнятого місця на диску;,
- комплекс системних програмних засобів, які реалізують управління файлами, зокрема: створення, знищення, читання, запис, назва, пошук і інші операції з файлами.

До функцій обслуговування файлової структури відносять такі операції, які здійснюються під управлінням ОС:

- створення файлів і призначення їм імен;
- створення каталогів (*directories*) і призначення їм імен;
- перейменування фалів і каталогів (папок);
- копіювання і переміщення файлів між дисками комп'ютера і між каталогами (папками) одного диска;
- видалення файлів і каталогів;
- навігація по файловій структурі з метою доступу до вказаного файла, каталога (папки);
- управління атрибутами файлів.

Назви файлів

Файл – це іменована послідовність байтів довільної довжини. Оскільки з цього означення витікає, що файл може мати і нульову довжину, то фактично створення файла полягає в присвоєні йому імені і його реєстрація в файловій системі.

Файли ідентифікуються назвами. Користувачі дають файлам символьні назви, при цьому враховуються обмеження ОС як на використовувані символи так, і на довжину назви. До недавнього часу ці обмеження були досить вузькими. Так, в популярній файловій системі FAT

(File Allocation Table) довжина назви обмежувалась відомою схемою 8.3 (8 символів – назва, 3 символи – розширення назви), а в ОС UNIX System V назва не може містити більше 14 символів. Однак користувачу набагато зручніше працювати з довгими назвами, оскільки вони дозволяють дати файлу дійсно мнемонічну назву, за якою навіть через достатньо великий проміжок часу можна буде згадати, що містить цей файл. Тому сучасні файлові системи, як правило, піддержують довгі символьні назви файлів. Наприклад, Windows NT в своїй новій файловій системі NTFS (New Technology File System) встановлює, що назва файла може містити до 255 символів, не враховуючи кінцевого нульового символу.

При переході до довгих назв виникає проблема сумісності з раніше створеними додатками, які використовували короткі назви. Щоб додатки могли звертатися до файлів відповідно до раніше узгоджених, файлова система повинна вміти давати еквівалентні короткі назви (псевдоніми) файлам, які мають довгі назви. Таким чином, однією з важливих задач стає проблема генерації відповідних коротких імен.

Довгі назви підтримуються не тільки новими файловими системами, але й новими версіями відомих файлових систем. Наприклад, в ОС Windows 95 використовується файлова система VFAT, яка являє собою досить змінений варіант FAT. Серед багатьох інших удосконалень одним із головних переваг VFAT є підтримка довгих назв. Крім проблеми генерації еквівалентних коротких назв, при реалізації нового варіанта FAT, важливою задачею була задача зберігання довгих назв при умові, щоб принципово метод зберігання і структура даних на диску не повинні бути змінені.

Зазвичай різні файли можуть мати однакові символьні назви. В цьому випадку файл однозначно ідентифікується так званою складовою назвою, яка являє собою послідовність символьних назв каталогів. В деяких системах одному й тому ж файлу не може бути надано декілька різних назв, а в інших таке обмеження відсутнє. В останньому випадку операційна система ОС присвоює файлу додаткову унікальну назву так, щоб можна було встановити взаємно однозначну відповідність між файлом і його унікальною назвою. Унікальна назва являє собою числовий ідентифікатор і використовується програма ОС. Прикладом такої унікальної назви файла є номер індексного дескриптора в системі UNIX.

Типи файлів

Файли бувають різних типів: звичайні файли, спеціальні файли, файли-каталоги.

Звичайні файли в свою чергу поділяються на текстові (*text files*) й двійкові (*binary files*). Текстові файли складаються з рядків символів поданих в ASCII-коді (American Standard Code for Information Interchange). Це можуть бути документи, вихідні тексти програм тощо. Текстові файли

можна прочитати на екрані і роздрукувати на принтері. Двійкові файли не використовують ASCII-коди, вони часто мають складну внутрішню структуру, наприклад, об'єктний код програми або архівний файл. Всі ОС повинні вміти розпізнавати хоча б один тип файлів – їх власні виконувані файли.

Спеціальні файли – це файли, які асоціюються з пристроями введення-виведення, які дозволяють користувачу виконувати операції введення-виведення, використовуючи звичайні команди запису в файл або читання з файла. Ці команди обробляються спочатку програмами файлової системи, а потім на деякому етапі виконання запиту перетворюються в ОС в команди управління відповідними пристроями. Спеціальні файли, так само як і пристрої введення-виведення, поділяються на блок-орієнтовані і байт-орієнтовані.

Каталог – це, з одного боку, група файлів, які об'єднані користувачем, виходячи з деяких міркувань (наприклад, файли, які містять програми ігор, або файли, які містять один програмний пакет), а з іншого боку – це файл, який містить системну інформацію про групу файлів, які містяться в ньому. В каталозі міститься список фалів, які до нього входять і встановлюється відповідність між файлами і їх характеристиками (атрибутами).

В різних файлових системах можуть використовуватись як атрибути різні характеристики, наприклад:

- інформація про розширений доступ;
- пароль для доступу до файлів;
- власник файла;
- ознака "тільки для читання";
- ознака "прихований файл";
- ознака "системний файл";
- ознака "архівний файл";
- ознака "двійковий /символьний";
- ознака "тимчасовий" (знищити після завершення процесу);
- ознака блокування;
- довжина запису;
- вказівник на ключове поле запису;
- довжина ключа;
- дати створення, останнього доступу і останньої зміни;
- поточний розмір файла;
- максимальний розмір файла.

Каталоги можуть безпосередньо мати значення характеристик файлів, як це зроблено в файловій системі MS-DOS, або посилатись на таблиці, які містять ці характеристики, як це реалізовано в ОС UNIX. Каталоги можуть утворювати ієрархічну структуру за рахунок того, що

каталог більш низького рівня входить в каталог більш високого рівня (рис. 3.1).

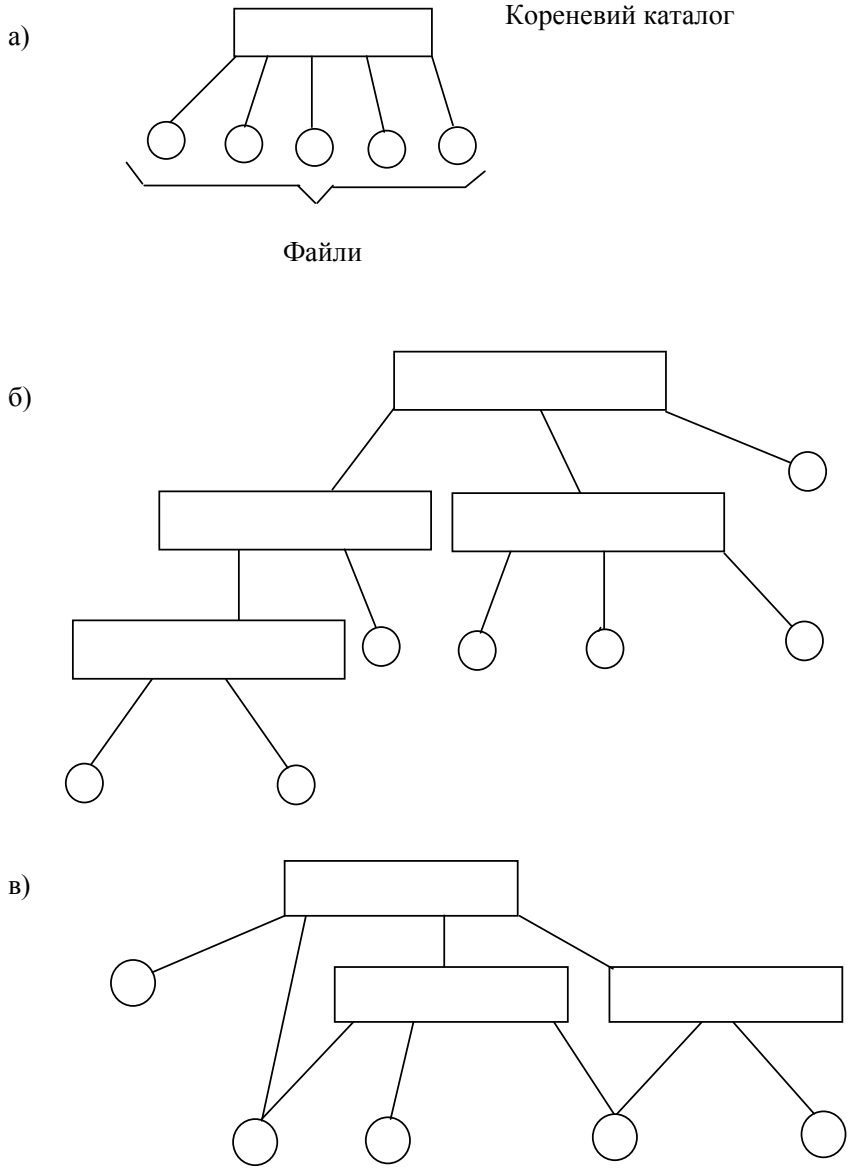


Рисунок 3.1 – Логічна організація файлової системи: а) однорівнева, б) ієрархічна (дерево), в) ієрархічна (сітка)

Ієрархія каталогів може бути деревом або сіткою. Каталоги утворюють дерево, якщо файлу дозволено входити в один каталог, і сітку – якщо файл може входити одразу в декілька каталогів. В MS-DOS каталоги утворюють деревовидну структуру, а в UNIX – сіткову. Як і будь-який інший файл, каталог має символьну назву і однозначно ідентифікується складовою назвою, яка містить ланцюжок символьних назв всіх каталогів, через які проходить шлях від кореня до даного каталогу.

Загальна модель файлової системи

Функціонування будь-якої файлової системи можна подати багаторівневою моделлю (рис. 3.2), в якій кожний рівень дає певний інтерфейс (набір функцій) рівню, що знаходиться над ним, а сам, в свою чергу, для виконання своєї роботи використовує інтерфейс (працює з набором запитів), що лежить на нижньому рівні.

Задачею символьного рівня є визначення за символьною назвою файла його унікальної назви. В файлових системах, в яких кожний файл може мати тільки одну символьну назву (наприклад, MS-DOS), цей рівень відсутній, оскільки символьна назва є одночасно унікальною і може бути використана ОС. В других файлових системах, в яких один і той же файл може мати декілька символьних назв, на даному рівні розглядається ланцюг каталогів для визначення унікальної назви файла.

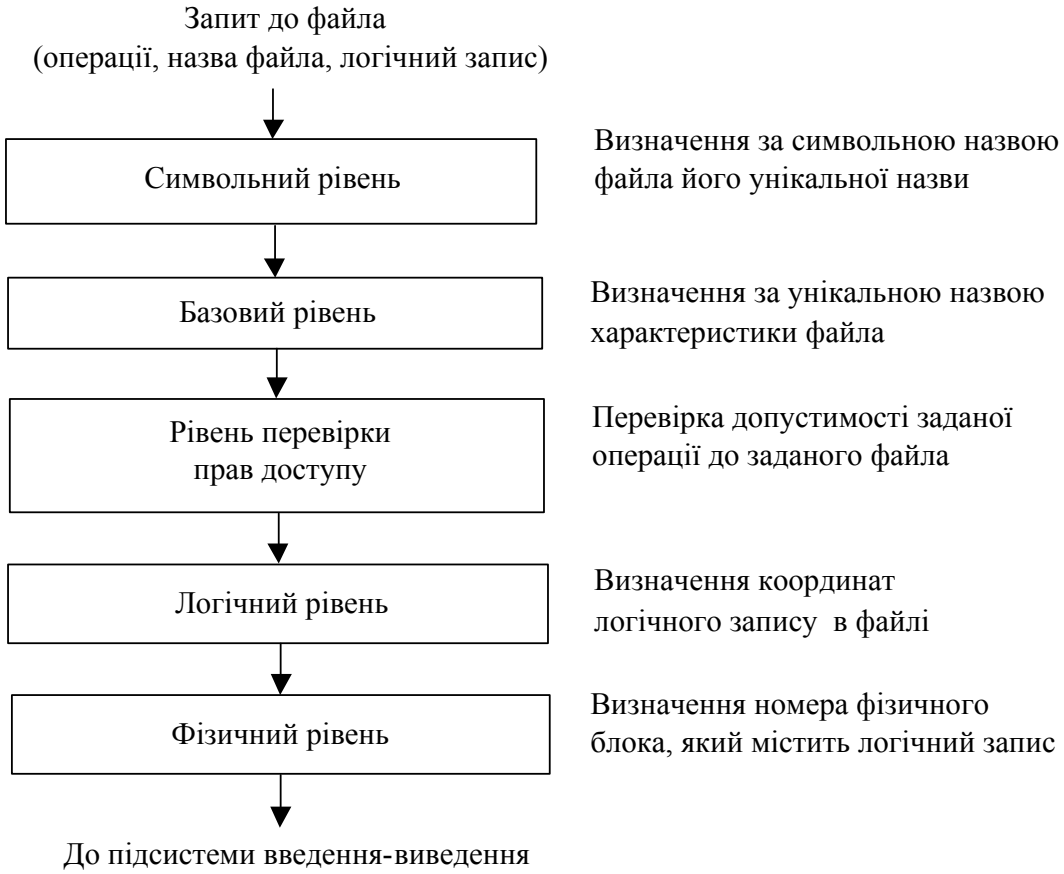


Рисунок 3.2 – Загальна модель файлової системи

На наступному, базовому рівні за унікальною назвою файла визначаються його характеристики: право доступу, адреса, розмір і інші. При відкритті файла його характеристики переміщуються з диска в оперативну пам’ять, щоб зменшити середній час доступу до файла. В деяких файлових системах (наприклад, HPFS) при відкритті файла разом з його характеристиками в оперативну пам’ять переміщуються декілька перших блоків файла, які містять дані.

Наступним етапом реалізації запиту до файла є перевірка прав доступу до нього. Для цього порівнюються права користувача або процеси зі списком дозволених видів доступу до даного файла.

На логічному рівні визначаються координати потрібного логічного запису в файлі, тобто потрібно визначити на якій відстані (в байтах) від початку файла знаходиться потрібний логічний запис. Алгоритм роботи даного рівня залежить від логічної організації файла. Для визначення координат логічного запису в файл з індексно-послідовною організацією зчитування таблиці індексів (ключів), в якій безпосередньо вказується адреса логічного запису.

На фізичному рівні файлова система визначає номер фізичного блока, який містить потрібний логічний запис, і зміщення логічного запису в фізичному блоці.

Після визначення номера фізичного блока, файлова система звертається до системи введення-виведення для виконання операцій обміну із зовнішнім пристроєм. У відповідь на цей запит в буфері файлової системи буде переданий потрібний блок, в якому на основі отриманого при роботі фізичного рівня зміщення вибирається потрібний логічний запис.

3.2 Прикладне програмне забезпечення

Прикладне програмне забезпечення забезпечує рішення задач в різних областях застосування комп'ютерних систем обробки даних. Нижче перераховані деякі різновиди прикладних програм.

Текстові редактори. Основні функції цього класу прикладних програм полягають у введенні й редагуванні текстових даних.

Текстові процесори. Основна відмінність текстових процесорів від текстових редакторів у тому, що вони дозволяють не тільки вводити й редагувати текст, але й форматувати його, тобто оформляти.

Графічні редактори. Це великий клас програм, призначених для створення й (або) обробки графічних зображень. У даному класі розрізняють такі категорії: растрові редактори, векторні редактори й програмні засоби для створення й обробки тривимірної графіки (3D-редактори).

Системи управління базами даних. Базами даних називають величезні масиви даних, організованих у табличні структури.

Основними функціями систем управління базами даних є:

- створення порожньої (незаповненої) структури бази даних;
- надання засобів її заповнення або імпорту даних таблиць з іншої бази;
- забезпечення можливості доступу до даних, а також надання засобів пошуку й фільтрації.

Багато систем управління базами даних додатково надають можливості проведення найпростішого аналізу даних і обробки.

Електронні таблиці. Електронні таблиці являють собою комплексні засоби для зберігання різних типів даних і обробки.

Системи автоматизованого проектування (CAD-системи). Призначені для автоматизації проектно-конструкторських робіт. Застосовуються в машинобудуванні, архітектурі.

Експертні системи. Призначені для аналізу даних, що втримуються в базах знань, і видачі рекомендацій із запиту користувача. Такі системи застосовують у тих випадках, коли вихідні дані добре формалізуються, але для ухвалення рішення потрібні великі спеціальні знання.

Редактори HTML(Web-редактори). Це особливий клас редакторів, що поєднують у собі властивості текстових і графічних редакторів. Вони призначені для створення й редагування так званих Web-документів (Web-сторінок Інтернету).

Браузери (засоби перегляду Web). До цієї категорії ставляться програмні засоби, призначені для перегляду електронних документів, виконаних у форматі HTML (документи цього формату використаються як Web-документи).

Інтегровані системи діловодства. Являють собою програмні засоби автоматизації робочого місця керівника.

Бухгалтерські системи. Це спеціалізовані системи, що сполучать у собі функції текстових і табличних редакторів, електронних таблиць і систем управління базами даних.

Фінансові аналітичні системи. Програми цього класу використовуються в банківських і біржових структурах. Вони дозволяють контролювати й прогнозувати ситуацію на фінансових, товарних і сировинних ринках, робити аналіз поточних подій, готовити зведення й звіти.

Геоінформаційні системи (ГІС). Призначені для автоматизації картографічних і геодезичних робіт на основі інформації, отриманої топографічними методами.

Системи відеомонтажу. Призначені для цифрової обробки відеоматеріалів, їхнього монтажу, створення відеоефектів, усунення дефектів, накладення звуку, титрів і субтитрів.

3.3 Мови програмування

Процесор комп'ютера – це велика інтегральна мікросхема. Всі команди й дані він одержує у вигляді електричних сигналів. Фактично процесор можна розглядати як величезну сукупність досить простих електронних елементів – транзисторів.

Команди, що надходять у процесор по його шинах, насправді є електричними сигналами, і їх можна подати як сукупності нулів і одиниць,

тобто числами. Різним командам відповідають різні числа. Тому, реально, програма, з якою працює процесор, являє собою послідовність чисел, яку називають машинним кодом *(machine code)*.

Управляти комп'ютером потрібно за певним алгоритму. *Алгоритм* – це точно визначений опис способу рішення завдання у вигляді кінцевої (за часом) послідовності дій. Такий опис ще називається формальним. Для подання алгоритму у вигляді, зрозумілому комп'ютеру, служать мови програмування. Спочатку завжди розробляється алгоритм дій, а потім він записується однією з таких мов. У підсумку виходить текст програми – повний, закінчений і детальний опис алгоритму мовою програмування. Потім цей текст програми спеціальними службовими додатками, які називаються трансляторами, або переводиться в машинний код, або виконується.

Мови програмування – штучні мови. Від природних вони відрізняються обмеженою кількістю „слів", значення яких зрозуміло транслятору, і дуже строгими правилами запису команд-*операторів (operators)*. Сукупність подібних вимог формує *синтаксис (syntax)* мови програмування, а смисл кожної команди та інших конструкцій мови – його *семантику (semantics)* .

Процес пошуку помилок в програмі називається *тестуванням (testing)*, процес усунення помилок – *налагодженням (debugging)*.

Компілятори й інтерпретатори

З допомогою мови програмування створюється не готова програма а тільки її текст, що описує раніше розроблений алгоритм. Щоб одержати робочу програму, треба цей текст або автоматично перевести в машинний код (для цього служать програми-компілятори (*compiler*)) і потім використати окремо від вихідного тексту, або відразу виконувати команди мови, зазначені в тексті програми (цим займаються програми-інтерпретатори (*language processors*)).

Інтерпретатор бере черговий оператор мови з тексту програми, аналізує його структуру й потім відразу виконує (звичайно після аналізу оператор транслюється в деяке проміжне подання або навіть машинний код для більш ефективного подальшого виконання).

Компілятори повністю обробляють весь текст програми(що інколи називають вихідними текстами). Вони переглядають його в пошуках синтаксичних помилок (іноді кілька разів), виконують певний значеннєвий аналіз і потім автоматично переводять (транслюють) на машинну мову – генерують машинний код.

У результаті закінчена програма виходить компактною і ефективною, працює в сотні разів швидше програми, виконуваної за допомогою інтерпретатора, і може бути перенесена на інші комп'ютери із процесором, що підтримує відповідний машинний код.

Основний недолік компіляторів – трудомісткість трансляції мов програмування, орієнтованих на обробку даних складної структури, часто заздалегідь невідомої або динамічно змінюваної під час роботи програми.

Рівні мов програмування

Мовою найнижчого рівня є мова асемблера, що просто подає кожну команду машинного коду, але не у вигляді чисел, а за допомогою символьних умовних позначок, названих мнемоніками. Однозначне перетворення однієї машинної інструкції в одну команду асемблера називається транслітерацією. Через те, що набори інструкцій для кожної моделі процесора відрізняються, конкретній комп'ютерній архітектурі відповідає своя мова асемблера, і написана нею програма може бути використана тільки в цьому середовищі.

Мови програмування високого рівня значно ближчі й зрозуміліші людині, ніж комп'ютеру. Особливості конкретних комп'ютерних архітектур у них не враховуються, тому створювані програми на рівні вихідних текстів легко перенести на інші платформи, для яких створений транслятор цієї мови.

Мови асемблера

Мови асемблера ізоморфні до машинної мови, тобто кожній команді (оператору) мови асемблера відповідає, як правило, одна машинна команда. Це дозволяє програмістові, що створює програму мовою асемблера, скористатися всіма можливостями системи команд мікропроцесора *(microprocessor)*. Крім того, програми, записані мовами асемблера, найбільш "економічні", тобто потребують менших ресурсів пам'яті та часу виконання. Це приводить до широкого застосування мов асемблера при програмуванні низки задач на основі мікропроцесорів.

Мови асемблера є основними при створенні ПЗ із вбудованими мікропроцесорами та мікропроцесорними контролерами. Це пояснюється економічністю цих мов при програмуванні з погляду використання ресурсів пам'яті. В інших випадках варто відзначити загальну тенденцію зменшення використання мов асемблера, що зумовлено недоліками цього класу мов – трудомісткістю і значними термінами такого програмування. У зв'язку з цим створено низку інструментальних засобів проектування ПЗ, які ґрунтуються на мовах програмування високого рівня та нових сучасних принципах та технологіях створення програм.

Мови програмування високого рівня

Серед найбільш використовуваних на сьогоднішній день мов програмування високого рівня можна виділити *Фортран, Бейсік, Паскаль, Сі, Сі++, Java.*

Basic (Бейсік). Для цієї мови є і компілятори, і інтерпретатори, а за популярністю вона займає перше місце в світі. Була створена в 60-х роках як навчальна мова.

Pascal (Паскаль). В мові Паскаль, створеній в кінці 70-х Ніклаусом Віртом, з'явилася низка вимог до структури програми і є можливості, які дозволяють успішно застосовувати її при створенні великих проектів.

Найбільш важливі характеристики мови:

- наявність у багатьох реалізаціях версій компілятора як вбудованого в інтегроване середовище, так і автономного (для компіляції програм великих розмірів);
- висока швидкість компіляції та генерація оптимізованого коду, що забезпечує швидке виконання програм;
- можливість створення за рахунок механізму модулів, що компілюються окремо. Це суттєво підвищило технологічність мови;
- наявність розширеного набору числових цілих типів та типів даних з плаваючою точкою стандарту IEEE;
- можливість організації ефективного інтерфейсу з *мовами асемблера* та *Сі* на рівні об'єктного коду;
- наявність в поширених реалізаціях (*Turbo Pascal, Borland Pascal, Delphi*) вбудованого інтегрованого налагоджувача, який забезпечує повну перевірку змінних, структур даних та виразів за покрокового налагоджування або в певних точках програми. Налагоджувач має можливість модифікації значень змінних та структур даних у процесі налагоджування програми.

С (Сі). Ця мова створена в лабораторії Bell і спочатку не розглядалась як масова. Вона планувалася для заміни асемблера, щоб мати можливість створювати як ефективні, так і компактні програми, і в той же час не залежати від конкретного типу процесора. *Сі* багато в чому схожа на *Паскаль* і має додаткові засоби для прямої роботи з пам'яттю (*покажчики*). Цією мовою в 70-ті роки написано багато прикладних та системних програм і ряд відомих операційних систем (UNIX).

До переваг мови *С* належать:

- мобільність. Це означає, що програма мовою *Сі*, яка створена для однієї програмної або апаратної платформи, легко переноситься на іншу на рівні вихідного коду практично без доробок. Теоретично це справедливо і для мови програмування *Паскаль*, але практичне втілення процесу перенесення складної програми може привести, практично, до її повної модифікації;
- підтримка сучасних технологій створення програмного забезпечення. У першу чергу це стосується наявності гнучких керуючих конструкцій, які асоціюються з мовами асемблеру і рекомендуються для застосування теоретичними та практичними методами програмування. Структура мови спонукає програміста використовувати у роботі низхідне

або структурне програмування та покрокову розробку модулів. Результатом такого підходу є надійна програма, що добре читається;

- успадковування мови. Багато нових мов програмування ґрунтуються на концепціях та технологіях застосування мови *Ci*. Їх прикладом можуть бути *Ci++, Ci#, Java* тощо.

C++ (Ci++). Це обєктно-орієнтоване розширення мови *Ci*, створене Бьярном Страуструпом в 1980 році. Велика кількість нових потужних можливостей, що дозволили різко збільшити продуктивність програмістів.

Java (Джава, Ява). Ця мова була створена компанією Sun на початку 90-х на основі *Ci++*. Вона була призвана спростити розробку програм на *Ci++* шляхом виключення з неї низькорівневих можливостей. Та головна особливість цієї мови – компіляція не в машинний код, а в платформо-незалежний байт-код (кожна команда займає один байт). Цей байт-код може виконуватися з допомогою *інтерпретатора* – віртуальної Java-машини *JVM* (Java Virtual Machine), версії якої створені сьогодні для будь-яких платформ.

Особливу увага в розвитку цієї мови приділяється двом напрямам: підтримці мобільних пристроїв та мікрокомп'ютерів, що влаштовані в побутовій техніці (технологія *Jini*) і створенню платформо-незалежних програмних модулів, які здатні працювати на серверах в глобальних і локальних мережах з різними операційними системами (технологія *Java Beans*).

Крім розглянутих мов існують спеціалізовані мови програмування високого рівня, наприклад:

Fortran (Фортран). Мова створена в 50-і роки і орієнтована на математичні розрахунки. Хоча в Фортрані вперше була реалізована низка важливих понять програмування, зручність створення програм було принесено в жертву можливості отримання ефективного машинного коду. Однак для цієї мови була створена велика кількість бібліотек, тому Фортран продовжує активно використовуватися в багатьох організаціях.

Cobol (Кобол). Ця мова розроблена на початку 60-х для застосування в економічній області і рішення бізнес задач. Цією мовою створено дуже багато програм, які активно експлуатуються і сьогодні.

Крім розглянутих вище існують такі мови програмування високого рівня, як *Algol (Алгол), PL/I (ПЛ/1), Smalltalk (Смолток), Lisp (Ліспі), Prolog (Пролог), Ada (Ада), Forth (Форт)* тощо.

Мови програмування для Інтернету

З активним розвитком всесвітньої мережі було створено багато реалізацій популярних мов програмування, адаптованих спеціально для Інтернету. Всі вони відрізняються характерними особливостями: мови є інтерпретованими, інтерпретатори для них розповсюджуються безкоштов-

но, а самі програми знаходяться в текстах. Такі мови називають скрипт-мовами.

Найбільш використовуваними скрипт-мовами є *HTML, Perl, Tcl/Tk, VRML, XML*.

Мови програмування баз даних

Ця група мов відрізняється від алгоритмічних мов насамперед розв'язуваними завданнями. База даних – це файл (або група файлів), що представляє собою впорядкований набір записів, що мають однакову структуру й організованих по єдиному шаблону (наприклад, у табличному вигляді). База даних може складатися з декількох таблиць.

Мови моделювання

При створенні програм і формуванні структур баз даних нерідко застосовуються формальні способи їхнього подання – формальні нотації, за допомогою яких можна візуально представити (зобразити за допомогою "мишки") таблиці баз даних, об'єкти програми й взаємозв'язку між ними в системі, що має спеціалізований редактор і генератор вихідних текстів програм на основі створеної моделі. Такі системи називаються CASE-системами. У них активно застосовуються нотації IDEF, а останнім часом все більшу популярність завойовує мова графічного моделювання UML (Unified Modeling Language).

3.4 Технології створення програмного забезпечення

Під *технологією програмування (software ingeneering)* розуміють сукупність узагальнених та систематизованих знань або науку про оптимальні способи проведення процесу програмування, що забезпечує в заданих умовах одержання програмної продукції із заданими властивостями. Технологія програмування в загальному випадку залежить від подальшого використання створених застосувань. Можна виділити такі етапи створення ПЗ:

- розробка, узгодження та затвердження технічного завдання на створюваний програмний комплекс;
- створення детальних специфікацій програмного комплексу. Треба визначити структури даних, склад модулів, їх інтерфейсні частини, алгоритми реалізації. На цьому етапі мають бути визначені мови реалізації проекту, версії;
- обґрунтування необхідної технології розробки;
- написання програмного комплексу;
- визначення кількості та складу комплекту тестів для створюваного комплексу і його написання. У загальному випадку для

складної системи обсяг тестів може перевищувати обсяг програмного комплексу як такого;

- налагоджування та тестування ПЗ;
- створення необхідної програмної документації;
- верифікація ПЗ. Здійснюється перевірка ПЗ на кожному етапі проектування системи на предмет виконання всіх вимог, які сформульовано на попередньому етапі;
- атестація та валідація ПЗ. Здійснюється тестування та оцінка інтегрованої системи (апаратного та програмного забезпечення) на предмет відповідності функціональним вимогам, а також вимогам до інтерфейсу та продуктивності;
- проведення попередніх випробувань створеного програмного комплексу;
- проведення міжвідомчої комісії та приймальних випробувань створеного програмного комплексу;
- створення та затвердження технічних умов на програмний проект;
- сертифікація створеного ПЗ.

Природно, що у деяких випадках та у різних технологіях певні етапи можуть бути відсутні і навпаки, у разі необхідності, можливе введення додаткових етапів.

Після виникнення комп'ютерних технологій було створено різні технології програмування. Розглянемо найбільш відомі та поширені з них. Варто зазначити, що всі технології потребують відповідної програмної підтримки (інструментальних засобів), які створюються різними фірмами та колективами програмістів. На ринку ПЗ зараз існує багато відповідних програмних продуктів, вартість яких є суттєвою.

Модульне програмування

Модульне програмування є подальшим розвитком та вдосконаленням процедурного програмування. Основна властивість модульної технології – це стандартизація та паспортизація інтерфейсу між окремими програмними одиницями. Модуль є окремою функціонально завершеною програмною одиницею, яка структурно ідентифікується (або оформлюється) у стандартний спосіб відносно компілятора і відносно об'єднання її з іншими аналогічними одиницями і завантаження. Як правило, модуль містить паспорт, в якому вказано всі основні його характеристики: мову програмування, обсяг, вхідні та вихідні змінні, їх типи, точки входження в модуль. Параметри настроювання тощо.

Модульне програмування – це мистецтво функціональної декомпозиційної задачі на певну сукупність різних модулів, вміння широко використовувати стандартні модулі шляхом їх параметричного настроювання, автоматизації збирання готових модулів з бібліотек тощо.

Структурне програмування

Структурне програмування ґрунтується на фіксації для програміста допустимих керуючих структур. Певні структури програмістам вживати заборонено. Будь-який алгоритм на будь-якому рівні проектування має бути записаний тільки за допомогою допустимих структур. Зазвичай кількість допустимих керуючих структур дуже обмежена – дві, три. Найбільше розповсюдження одержали такі три структури: лінійна (композиція операторів), вибору (if) та циклічна. Усі структури мають один вхід та один вихід. Дозволяється необмежене та рекурсивне вкладення структур одна в одну. Технологія структурного програмування визначає роботу програміста як суперпозицію допустимих структур.

Для суперпозиції допустимих структур не потребується використання операторів goto, тому структурне програмування називають ще програмуванням без goto. Структурні програми на відміну від звичайних мають просту деревоподібну архітектуру, легко читаються та модифікуються. У структурній програмі відсутній клубок переходів уперед-назад. Це досягається частково і тим, що на етапі проектування створені деякі штучні ускладнення – заборонено використання оператора goto, конструювання програм дозволяється проводити тільки з використанням допустимих структур певної мови. Ці ускладнення подовжують процес проектування програми, що призводить до її додаткового пророблення. З'являться можливість провести формальними методами доведення правильності процесу проектування програми. Результатом цього є значне скорочення процесу налагодження, отже скорочується весь процес створення програмного проекту. Варто зазначити, що мови асемблеру незручні для структурованого програмування в зв'язку з відсутністю в них блокових структур.

Низхідне програмування

Низхідне проектування – це певна багаторівнева дисципліна створення програм. На верхньому рівні вихідний алгоритм записується у вигляді деякої ієрархічної схеми, елементи якої описуються природною для цієї проблеми мовою. Кожний такий опис можна розглядати як послідовність коментарів, заготовок або команд деякого віртуального проблемно-орієнтованого комп'ютера. Кожна команда такого комп'ютера моделюється або інтерпретується командами іншої віртуальної машини більш низького рівня і так далі до команд реального комп'ютера або операторів відповідної мови програмування. Проектування програмного комплексу виконується у такий спосіб, що опис системи верхнього рівня не залежить від опису функціонування блоків більш низьких рівнів. Вся система проектується та налагоджується за рівнями зверху вниз. Кожний нижній рівень налагоджується на системі тестів, які створено та перевірено

на попередньому верхньому рівні. У цілому технологія низхідного проектування дозволяє:

- почати програмування фактично одночасно та паралельно з розробкою відповідного алгоритму;
- формально, у вигляді програми певної віртуальної машини, фіксувати кожний етап створення відповідного алгоритму; легко модифікувати програму за рівнями шляхом заміни однієї віртуальної машини придатною іншою;
- спрощувати налагодження програм шляхом розосередження її за рівнями та проведення незалежно від нижче розташованих рівнів деталізації.

Об'єктно-орієнтоване програмування

Об'єктно-орієнтоване програмування (ООП) було започатковано у таких мовах програмування, як Ада, Smalltalk, Сі++, Паскаль (реалізація Object Pascal). До появи ООП панувало процедурне програмування. Тоді основою програм були функції та процедури, тобто дії. Програміст визначав, які процедури та функції потрібні йому для розв'язання поставленої задачі, реалізовував ці функції та поєднував їх у програму. Програма звичайно мала досить чіткий алгоритм роботи – послідовність операцій, що починається в одній точці та закінчується в інших.

В ООП і проектуванні головною відправною точкою є не процедура, не дія, а *об'єкт (class object)*. Взагалі кажучи, такий підхід є природним, оскільки у реальному світі ми маємо справу саме з об'єктами (людьми, предметами, технічними пристроями), які взаємодіють один з одним. Взаємодія користувача з комп'ютерною програмою – це також взаємодія двох об'єктів – програми і людини, що обмінюються один з одним певними *повідомленнями.*

Прикладна програма, побудована за принципами ООП, – це не послідовність якихось операторів, не деякий жорсткий алгоритм. Об'єктно-орієнтована програма – це сукупність об'єктів і способів їхньої взаємодії. Обмін між об'єктами відбувається за допомогою повідомлень.

ООП ґрунтується на трьох основоположних принципах:

- інкапсуляції *(encapsulation)* та приховуванні даних;
- успадковуванні *(inheritance)*;
- поліморфізмі *(polymorphism)*.

Об'єкт можна визначити як деяку сукупність даних і способів роботи з ними. Дані можна розглядати як поля запису. Це характеристики об'єкта. Користувач і об'єкти програми повинні мати можливість читати дані об'єкта, певним чином їх обробляти і записувати в об'єкт нові значення. Тут найважливіше значення має принцип інкапсуляції та приховування даних. Цей принцип полягає в тому, що зовнішнім об'єктам і

користувачеві прямий доступ до даних, як правило, заборонений. Робиться це з двох міркувань:

- для надійного функціонування об’єкта треба підтримувати цілісність і несуперечність його даних. Якщо не подбати про це, то зовнішній об’єкт чи користувач можуть занести до об’єкта неправильні дані та він почне функціонувати з помилками;
- необхідно ізолювати зовнішні об’єкти від особливостей внутрішньої реалізації даних. Для зовнішніх споживачів даних має бути доступний тільки інтерфейс користувача – опис того, які присутні дані та функції і як їх використовувати.
- внутрішня реалізація – це справа розробника об’єкта. При такому підході розробник може у будь-який момент модернізувати об’єкт, модифікувати структуру збереження і форму подання даних, але, якщо при цьому не торкатися інтерфейсу, зовнішній споживач цього навіть не помітить. І у такий спосіб у зовнішній програмі нічого не доведеться модифікувати.

Щоб реалізувати принцип приховування даних, в об’єкті зазвичай визначають процедури і функції, які забезпечують усі необхідні операції з даними: їх читання, обробку, запис. Ці функції та процедури називаються *методами* і через них відбувається спілкування з даними об’єкта.

Сукупність даних та методів їх читання і запису називаються *властивостями об’єкта*. Властивості можна встановлювати в процесі проектування, їх можна змінювати програмно під час виконання прикладної програми. Зовні все це виглядає так, начебто об’єкт має якісь дані, наприклад, цілі числа, які можна прочитати, використати в необхідних обчисленнях та записати в об’єкт нові їх значення.

Крім методів, які працюють з окремими даними, в об’єкті є методи, які працюють з усією їх сукупністю, що змінює їхню структуру. Таким чином, об’єкт є сукупністю властивостей і методів. Але це не можна вважати точним означенням об’єкта. Для того, щоб дати повне означення, треба розглянути взаємодію об’єктів один з одним.

Середовищем взаємодії об’єктів є *повідомлення*, які генеруються у результаті виникнення різних *подій*. Події настають, зазвичай, унаслідок дій користувача – переміщення курсора “мишкою”, натискання кнопок “мишки” або клавіш клавіатури. Але події можуть наставати і в результаті функціонування об’єктів як таких. У кожному об’єкті визначено низку подій, на які він може реагувати. У конкретних примірниках об’єкта можуть бути визначені *оброблювачі* певних подій, які й визначають реакцію екземпляра об’єкта. В оброблювачах аналізується згенерована подія і виконуються необхідні дії: зміна властивостей об’єктів, виконання певних методів, генерація нових подій. До написання цих оброблювачів зводиться, як буде видно далі, основне програмування при розробці *графічного інтерфейсу користувача* у візуальному середовищі проектування.

Тепер можна остаточно визначити *об'єкт* як сукупність *властивостей* і *методів*, а також *подій*, на які він може реагувати. Умовно це зображено на рис. 3.3 Зовнішнє управління об'єктом здійснюється через оброблювачі подій. Ці оброблювачі звертаються до методів і властивостей об'єкта. Початкові значення даних об'єкта можуть задаватися також за рахунок вихідних даних проектування. У результаті виконання методів об'єкта можуть генеруватися нові події, які сприймаються іншими об'єктами чи програмами користувача.

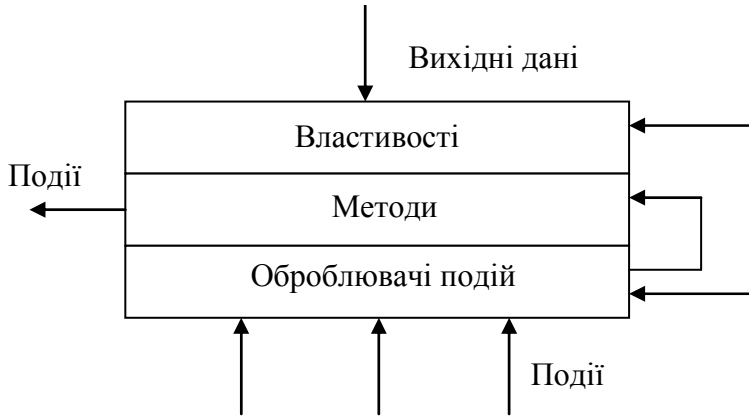


Рисунок 3.3 – Об'єкт як сукупність властивостей, подій та методів

Як правило, складна програма – це не просто певна сукупність об'єктів. У процесі роботи об'єкти можуть створюватися і знищуватися. Таким чином, структура програми є динамічним утворенням, що модифікується під час виконання. Основна мета створення і знищення об'єктів – економія ресурсів комп'ютера і, насамперед, пам'яті.

З метою організації динамічного розподілу пам'яті до всіх об'єктів закладено методи їх створення – *конструктори (constructors)* та знищення – *деструктори (destructors)*. Конструктори об'єктів, що постійно мають бути присутні у прикладній програмі, спрацьовують під час запуску програми. Деструктори всіх об'єктів, що є в даний момент у застосуванні, спрацьовують при завершенні його роботи. Але нерідко і під час виконання різні нові об'єкти (наприклад, нові вікна документів) динамічно створюються і знищуються за допомогою їх конструкторів та деструкторів.

Створювати об'єкти у своїй програмі можна двома способами: вручну включати до неї відповідні оператори або шляхом *візуального програмування*, використовуючи заготівки – *компоненти (components)*.

Програмування вручну різних потрібних користувачеві вікон, кнопок, меню, обробка подій "мишки" та клавіатури, включення до програми зображень і звуку вимагало все більше і більше часу програміста. У низці випадків весь цей сервіс починав займати до 80-90 % обсягу програмних кодів.

Вихід з цієї ситуації позначився завдяки двом підходам. Перший з них – стандартизація багатьох функцій інтерфейсу, завдяки чому з'явилася можливість використовувати бібліотеки, наявні, наприклад, у системі Windows. У підсумку при зміні стилю графічного інтерфейсу (наприклад, при переході від Windows 98 до Windows 2000) застосування могли автоматично пристосовуватися до нової системи без будь-якого перепрограмування. На цьому шляху створилися чудові умови для розв'язання однієї з найважливіших задач удосконалювання техніки програмування – повторного використання коду. Один раз створені форми, компоненти, функції могли бути згодом неодноразово використані для вирішення їхніх задач. Кожен програміст одержав доступ до напрацювання інших програмістів і до низки бібліотек, створених різними фірмами. Причому була забезпечена сумісність ПЗ, розробленого різними алгоритмічними мовами.

Другим революційним кроком була поява візуального програмування, що виникло спочатку в системі розробки застосувань Visual Basic і знайшло блискуче втілення в системах Delphi, Visual C++ та Borland C++ Builder фірми Borland.

Візуальне програмування дозволило звести проектування інтерфейсу користувача до простих і наочних процедур, що надають можливість за лічені години зробити те, на що раніше йшли місяці роботи. У сучасному вигляді в названих системах це виглядає так.

Програміст працює в інтегрованому середовищі розробки (Integrated development environment – *IDE*). Середовище надає форми з депозитарію форм (у застосуванні їх може бути кілька), на яких розміщуються компоненти. Зазвичай це віконна форма, хоча можуть бути і невидимі форми. На форму за допомогою "мишки" переносяться і розміщаються піктограми компонентів, які є в бібліотеках. За допомогою нескладних маніпуляцій можна змінювати розміри і розташування цих компонентів. При цьому увесь час у процесі проектування видно результат – зображення форми і розташованих на ній компонентів.

Але переваги візуального програмування не зводяться до цього. Найголовніше полягає в тому, що під час проектування форми і розміщення на ній компонентів система розробки застосувань автоматично формує коди програми, включаючи до неї відповідні фрагменти, які описують компоненти. Потім у відповідних діалогових вікнах користувач може модифікувати задані за замовчуванням значення певних властивостей цих компонентів і, за необхідності, написати оброблювачі подій. У такий спосіб проектування зводиться, фактично, до:

- розміщення компонентів на формі;
- задання деяких їхніх властивостей;
- написання, за необхідності, оброблювачів подій.

Компоненти можуть бути *візуальними*, видимими при роботі застосування, та *не візуальними*, що виконують ті чи інші службові функції. Візуальні компоненти відразу видимі на екрані монітора в процесі проектування в такому ж вигляді, в якому їх побачить користувач під час виконання застосування. Це дозволяє легко вибрати місце їхнього розташування та їхній дизайн – форму, розмір, оформлення, текст, колір тощо. Не візуальні компоненти видимі на формі в процесі проектування у вигляді піктограм, але користувачеві під час виконання вони не видимі, хоча і виконують для нього "за кадром" дуже корисну роботу.

До бібліотек візуальних компонентів системи розробки застосувань включено велику кількість типів компонентів та їх номенклатура швидко розширюється від версії до версії.

Типи об’єктів і, зокрема, компонентів бібліотек системи розробки, застосувань формуються у вигляді *класів (class)*. Класи – це типи, обумовлені користувачем. У класах описуються властивості об’єкта, його методи і події, на які він може реагувати. Мови ООП передбачають тільки інструментальні засоби створення класів. Самі класи створюються розробниками ПЗ. Утім, творці системи розробки застосувань уже розробили багато корисних класів і включили їх до бібліотек.

Якби при створенні нового класу все доводилося починати з нуля, то ефективність цього заняття була б зведена нанівець. Якби при розробці нового компонента, наприклад, будь-якої нової кнопки, довелося б створювати все спочатку (рисувати зображення, описувати усі властивості, що визначають її місце розташування, розміри, написи і картинки на її поверхні, колір, шрифти, описувати методи, що реалізують її поведінку – зміна розмірів, видимість, реакції на повідомлення, що надходять від клавіатури та миші), то ймовірно, програміст би відмовився б від розробки нової кнопки.

На щастя, у дійсності все відбувається по-іншому завдяки важливому принципу ООП – *успадковуванню*. Новий *похідний клас (derived class)* може успадковувати властивості, методи, події свого *батьківського класу (base class)*, тобто того класу, на основі якого він створюється. Наприклад, при створенні нової кнопки можна взяти за основу один із уже розроблених класів кнопок і тільки додати до нього якісь нові властивості або скасувати властивості та методи батьківського класу.

Третій важливий принцип ООП – *поліморфізм* – означає можливість визначення методу з одним і тим же ім’ям, який можна застосовувати одночасно до всіх примірників об’єктів ієрархії успадковування. У цьому випадку кожен примірник об’єкта ієрархії може “замовляти” особливості реалізації цього методу “над собою”. Принцип поліморфізму підвищує гнучкість використання об’єктів при створенні складних програмних комплексів.

Технологія швидкого створення застосувань

Технологія швидкого створення застосувань (Rapid Application Development – *RAD*) є новітньою технологією, що створена завдяки візуальному ООП для середовищ Windows. Першою інструментальною системою для цієї технології було середовище Visual Basic. Ця система сформувала новий стиль взаємодії програміста або користувача програми з комп'ютером, дозволяючи наочно конструювати користувацький інтерфейс за допомогою миші, а не звичайного для колишніх часів шляху: написанням кодів, їхньою компіляцією і виконанням програми, після чого тільки і можна було подивитися, як же це виглядає на екрані.

Хоча середовище Visual Basic знайшло широкий попит і допомогло відкрити світ програмування для людей, не занадто ним спокушених, воно має багато проблем. Головні з них – низька продуктивність створюваних застосувань при їхньому виконанні, недостатня строгість та рудиментарна об'єктна орієнтованість мови, що сприяє скоріше швидкій розробці виробів, а не створенню потужних ефективних застосувань, а також низка інших недоліків.

Системи Delphi, Borland C++Builder та Visual C++ – це наступний крок у розвитку середовищ швидкої розробки застосувань. Вони виправляють багато дефектів, виявлених у Visual Basic. Автори цих систем створили інструменти, що на перший погляд виглядають схожими на середовище Visual Basic, хоча в дійсності вони суттєво кращі.

Інтегровані середовища розробки в Delphi та C++Builder виглядають однаково. Весь користувацький інтерфейс, усі бібліотеки, усі прийоми роботи з цими системами практично однакові. Основне розходження Delphi і C++Builder полягає в мовах програмування, що лежать у їхній основі. Delphi базується на мові Паскаль (версія Object Pascal), а C++Builder – на мові Сі++.

Під терміном швидке створення застосувань розуміють процес створення ПЗ, що місить три елементи:

- невелику команду програмістів (2-10 осіб);
- короткий, але ретельно пророблений виробничий графік (від двох до шести місяців);
- ітеративний цикл, завдяки якому програмісти, за мірою того, як застосування починає набувати форму, запитують та реалізують у програмному продукті вимоги, які отримують у процесі взаємодії із замовником.

Варто відзначити, що технологія швидкого створення застосувань, як і будь-яка інша, не може претендувати на універсальність, вона є продуктивною, у першу чергу для відносно невеликих проектів, які створюються під конкретного замовника. Якщо ж створюється типова система, яка не є завершеним програмним продуктом, а являє собою комплекс типових компонент, що централізовано супроводжуються,

адаптуються до програмно-технічних платформ, систем управління базами даних, засобів телекомунікації, організаційно-економічних особливостей об'єктів впровадження та таких, що інтегруються із існуючими програмними системами, на передній план виступають такі покажчики проекту, як керованість та якість, які можуть увійти в протиріччя з простотою та швидкістю розробки. Для таких проектів необхідні високий рівень планування та жорстка дисципліна проектування, строга відповідність створеним заздалегідь протоколам та інтерфейсам, що знижує швидкість розробки.

Технологія RAD непридатна для розробки складних розрахункових програм, ОС, програм управління складними об'єктами (супутники, космічні кораблі, атомні реактори тощо), тобто програм, які потребують написання великого обсягу (сотні тисяч рядків) унікального вихідного коду.

Не підпадають під технологію швидкого створення застосувань програми, в яких відсутня яскраво окреслена інтерфейсна частина, що наочно визначає логіку роботи системи (наприклад, застосування, що функціонують у РЧ) та застосування, від роботи яких залежить безпека людей (управління літаком, атомною електростанцією), у зв'язку з тим, що ітеративний підхід передбачає, погану працездатність перших кількох версій, а це в даному випадку абсолютно виключено.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Що таке операційна система?
2. Наведіть класифікацію операційних систем.
3. В чому різниця між однозадачними та багатозадачними ОС?
4. Для чого призначені ОС пакетної обробки.
5. Які вимоги висовуються до ОС реального часу?
6. Яка відмінність між ОС загального призначення і ОС РЧ?
7. Що таке файлова система?
8. Наведіть приклади типів файлів.
9. Опишіть загальну модель файлової системи.
10. Порівняйте ієрархію каталогів MS-DOS та UNIX.
11. Що таке мова програмування?
12. В чому різниця між компіляторами і інтерпретаторами? Які з них вам відомі?
13. Які мови програмування активно використовуються сьогодні? Їх особливості і де застосовуються.
14. Поясніть поняття ООП.
15. Яка відмінність між об'єктом та класом?
16. Охарактеризуйте технологію RAD.

4 МАТЕМАТИЧНІ ОСНОВИ ЕОМ

4.1 Форми подання даних

Будь-яка форма людської діяльності, будь-який процес функціонування технічного об'єкта пов'язані з передаванням і перетворенням інформації.

Інформацію, виражену і зафіксовану в деякій матеріальній формі (наприклад, на дискеті) називають *даними.* Дані можуть мати постійне значення (бути константами) чи змінюватися (бути змінними).

Прикладом константи може служити число $\pi = 3{,}14159$, ім'я людини (наприклад, "Петро") тощо.

Як приклад змінних величин можна навести швидкість автомобіля (під час зупинок автомобіля вона дорівнює нулю, а під час руху змінюється від нуля до деякого максимального значення, що залежить від марки автомобіля). Змінною величиною є також результат кидання монети. Цей результат може набувати одного з двох значень – "орел" (бік монети, на якому зображено герб) чи "решка" (бік монети, на якому зазначено її номінал).

Отже, величини можуть набувати як числових значень, так і бути нечисловими.

Будь-яку нечислову інформацію можна перетворити в числову форму. Зазвичай для цього кожному з можливих значень величини зіставляється своє унікальне число. Цей процес часто називають *кодуванням інформації.* Так, кожному можливому імені людини можна присвоїти свій унікальний порядковий номер (кількість таких імен дуже велика, але скінченна). Результату кидання монети "орел" можна присвоїти числове значення 0, а результату кидання "решка" – значення 1.

Спосіб кодування інформації залежить від розв'язуваної задачі. Так, для імен людей кодування краще виконувати не за самими іменами, а за буквами (наприклад, за алфавітом). У першому випадку, якщо для кожної букви використати їх порядковий номер в алфавіті, то числовий еквівалент імені "Петро" дорівнюватиме 1706201816. У другому випадку істотним є не назви результатів кидання монети, а те, що можливих результатів усього два, і тому їх можна закодувати будь-якими двома числами.

Надалі будемо вважати, що всі постійні та змінні величини мають числові значення.

Усі цілі та дробові числа (додатні, від'ємні і нуль) називаються *раціональними числами (rational numbers).* Раціональні числа утворюють нескінченну множину, що має такі властивості:

- упорядкована множина – для кожних двох різних раціональних чисел *a і b* можна вказати, яке з них менше. Множина всюди щільна – між кожними двома різними раціональними числами *a* і *b (a < b)* існує ще принаймні одне раціональне число *c (a < c < b),* а, отже, і нескінченна множина раціональних чисел;

- арифметичні дії (додавання, віднімання, множення і ділення) – над будь-якими двома різними раціональними числами завжди можливі і дають у результаті певне раціональне число. Виняток – ділення на нуль;

- сукупність раціональних чисел не вичерпує всієї множини допустимих чисел. Так, існують числа, що виражають довжини відрізків, несумірних з довжиною масштабу (тобто відрізків, які не можна виразити цілим чи дробовим числом). Це, наприклад, число π, що, як відомо, є відношенням довжини кола до його діаметра. Числа, подібні до числа π, називають *ірраціональними (irrational number).*

Усі раціональні й ірраціональні числа називають дійсними або натуральними. Крім властивостей раціональних чисел 1–3, натуральні числа мають також властивість *неперервності.*

Будь-яке раціональне число можна подати у вигляді *m/n,* де *m* і *n* – цілі числа. Ірраціональні числа в такому вигляді точно подати не можна, однак будь-яке ірраціональне число можна з будь-яким ступенем точності замінити раціональним числом так само, як число π.

Змінні величини можуть бути пов'язані між собою функціональною залежністю, якщо кожному заданому значенню однієї з декількох величин, названих *аргументами (argument of a function),* відповідає одне чи кілька значень змінної величини *функції.* Сукупність допустимих значень аргументів називають областю визначення функції *(domain of function)*, якій відповідає множина значень функції *(ranging check a function)*.

Існують три основні способи вираження функцій:

- аналітичне уявлення;
- табличне подання;
- графічне зображення.

Аналітично функцію можна описати за допомогою однієї чи декількох формул. Залежність функції (швидкості руху автомобіля) від часу можна спрощено подати за допомогою таких формул:

$v = (v_{\max} / t_1)t$, якщо $0 < t < t_1$ – розгін автомобіля;

$v = v_{\max}$, якщо $t_1 < t \le t_2$ – рух автомобіля;

$v = v_{\max} - \{v_{\max} / t\}(t - t_2)$, якщо $t_2 < t \le t_2 + t_1$ – гальмування автомобіля,

де v – швидкість автомобіля;

t – час;

v_{max} – максимальна швидкість автомобіля;

t_1 – час досягнення автомобілем максимальної швидкості;

t_2 – час початку гальмування автомобіля.

Часто функціональну залежність не вдається подати у вигляді формули. У цьому разі значення аргументу і відповідні значення функції можна задати у вигляді *таблиці.*

Для залежності швидкості руху від часу, якщо $v_{max} = 50$ км/год; $t_1 = 0{,}05$ год і $t_2 = 0{,}5$ год, значення функції v наведено в табл. 4.1.

Таблиця 4.1 – Табличні значення функції

Час *t,* год	0	0,01	0,02	0,03	0,04	0,05	0,06-0,5	0,51	0,52	0,53	0,54	0,55
Швидкість *v,* км/год	0	10	20	30	40	50	50	40	30	20	10	0

Результат кидання монети (0 чи 1) – випадкова величина, тому його не можна виразити формулою, але йому можна надати табличного вигляду (табл. 4.2) як залежності від номера випробування (у теорії імовірностей експеримент, у результаті проведення якого отримують випадкову величину, називають *випробуванням).*

Таблиця 4.2 – Приклад результатів випробування – кидання монети

Номер випробування *n*	1	2	3	4	5	6	7	8	9	10
Результат *y*	0	0	1	0	1	1	1	0	0	1

Щоб *графічно* зобразити задану функціональну залежність, на горизонтальній осі (осі абсцис) позначають ряд значень однієї зі змінних величин (зазвичай аргументу), а на вертикальній осі (осі ординат) – відповідні значення функції. Тоді графік залежності швидкості автомобіля від часу матиме вигляд, як показано на рис. 4.1, а графік залежності результату кидання монети y від номера випробування n – як на рис. 4.2.

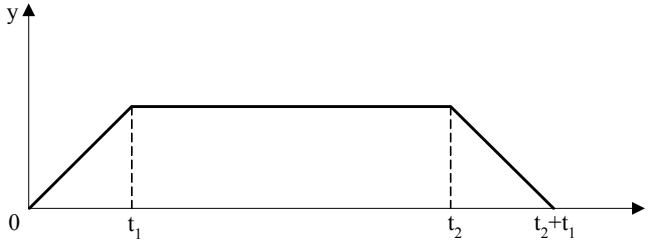


Рисунок 4.1 – Графік залежності швидкості автомобіля від часу

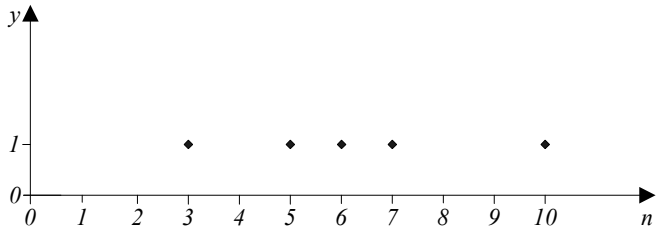


Рисунок 4.2 – Графік залежності результату кидання монети від номера випробовування

Функції, графіки яких зображено на рис. 4.1 і 4.2, істотно відрізняються за характером зміни значень аргументу і функції.

Значення аргументу і функції у випадку руху автомобіля мають натуральні значення і змінюються неперервно. Такі змінні називають *неперервними (digital data),* чи *аналоговими (analog data)* даними. У природі і техніці багато даних змінюються неперервно (наприклад, зміна освітленості протягом дня, звукові коливання, зміна електричної напруги залежно від сили струму тощо).

У випадку кидання монети аргументи і функції набувають тільки фіксованих значень. Такі змінні називають *дискретними.* Дискретними є, зокрема, усі числові дані.

Відповідно до характеру зміни даних усі пристрої оброблення і передавання даних (комп'ютери теж) поділяють на два класи: неперервної дії – *аналогові* і дискретної дії – *цифрові.*

В *аналогових обчислювальних машинах* (*АОМ*) оброблювана інформація подається відповідними значеннями аналогових величин: струму, напруги, кута повороту будь-якого механізму і т. ін. Аналогові обчислювальні машини з'явилися навіть раніше, ніж комп'ютери. Так, механічний обчислювальний пристрій – *«диференціальний аналізатор»,* здатний розв'язувати складні диференціальні рівняння, був створений в США ще в 1930 р. Недолік АОМ – невисока точність обчислень, тому на сьогоднішній день сфера їх застосування дуже обмежена (здебільшого у складі різних моделювальних пристроїв для розроблення складних зразків техніки).

Тепер під словом «комп'ютер» зазвичай розуміють пристрій, у якому дані подаються й оброблюються в дискретній (числовій) формі.

Хоча комп'ютер не може обробляти безпосередньо аналогові дані, але їх можна вводити в комп'ютер після перетворення у дискретну форму. Цю операцію виконують спеціальні пристрої уведення – *аналого-цифрові перетворювачі* (*АЦП*). Щоб перетворити неперервний аналоговий сигнал у числову форму, АЦП через задані проміжки (кванти) часу вимірює величину аналогового сигналу і вводить її в комп'ютер для наступного оброблення. Так, для неперервної функції (див. рис. 4.1) значення швидкості, зведені в таблицю залежності v від t (див. табл. 4.1), – це

дискретні подання цієї функції з інтервалом (кроком) квантування 0,01 год. (36 с.). Аналогічно аналоговий сигнал, поданий на рис. 4.3, перетворюється в цифровий. При цьому для величини сигналу обрано діапазон значень 0...9, і на інтервалі зміни сигналу зроблено 10 вимірювань із кроком квантування одна секунда. У результаті сигнал виражається десятьма числами: 5, 7, 8, 8, 5, 4, 2, 1, 6, 5.

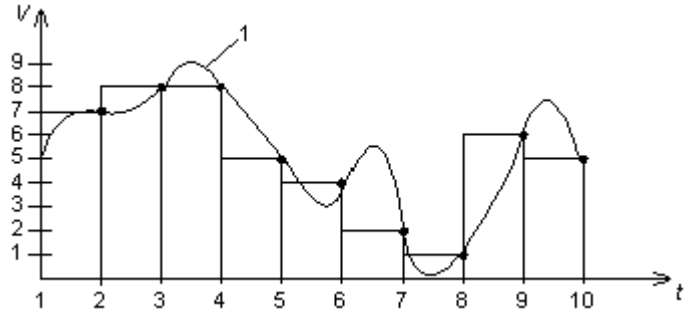


Рисунок 4.3 – Перетворення аналогового сигналу в дискретний в діапазоні значень 0...9 із кроком квантування одна секунда: 1 – вихідний аналоговий сигнал

Точність подання звукового сигналу (див. рис. 4.3) як за часом, так і за значенням явно недостатня. Якщо збільшити діапазон значень сигналу в 5 разів (від 0 до 45) і зменшити крок квантування в 4 рази (0,25 с), то результат буде таким, як показано на рис. 4.4. У цьому разі буде отримано вже 37 значень дискретної величини.

Отже, чим більший діапазон значень і чим менший крок квантування, тим точнішою буде аналогова змінна, але тим більшими будуть і обсяг отриманих дискретних даних, і час їх оброблення.

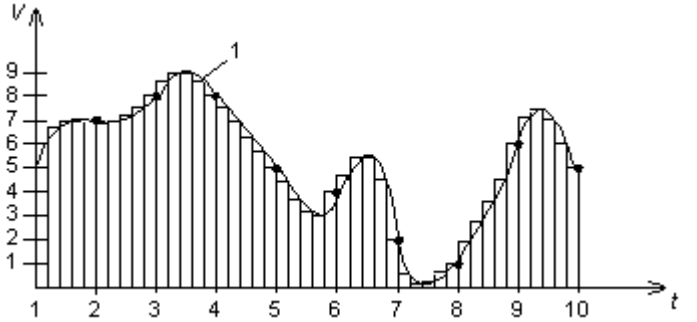


Рисунок 4.4 – Перетворення аналогового сигналу в дискретний в діапазоні значень 0...45 із кроком квантування 0,25 секунд: 1 – вихідний аналоговий сигнал

Сучасні комп’ютери дозволяють перетворювати дискретний сигнал в аналоговий. Пристрій, що виконує цю операцію, називають *цифро-аналоговим перетворювачем (ЦАП)*. Спочатку перетворювач ставить у відповідність кожному дискретному числу відповідний рівень аналогової величини (наприклад, для електричних сигналів значення напруги чи

струму). Одержувана пилкоподібна крива пропускається через спеціальний електронний фільтр, що згладжує її, перетворюючи в неперервний сигнал, який потім можна подати на вхід аналогового пристрою (наприклад, гучномовця комп'ютера).

Отже, щодо можливості подання будь-якої інформації в числовій формі комп'ютери – це найбільш універсальні пристрої оброблення даних для розв'язання багатьох задач у різних галузях науки, техніки, бізнесу й управління.

4.2 Системи числення

Під *системою числення* розуміють спосіб подання будь-якого числа за допомогою деякого алфавіту символів, названих *цифрами (digits)*.

Систему числення називають *позиційною,* якщо одна і та ж сама цифра має різне значення, обумовлене позицією цифри в послідовності цифр, що зображує число (прикладом непозиційної системи є римська система числення).

Кількість різних цифр в алфавіті позиційної системи називають *основою S* цієї системи. Система числення, що використовується в повсякденному житті, має десять різних цифр (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) і тому її називають *десятковою системою числення.*

Будь-яке число *N* у позиційній системі числення можна виразити за формулою:

$$N_s=a_mS^m+a_{m-1}S^{m-1}+\ldots+a_1S^1+a_0S^0+a_{-1}S^{-1}+a_{-2}S^{-2}+\ldots \qquad (4.1)$$

де N_s – число подане у системі з основою *S;*
m – номер розряду *(digits)*;
a_m – цілий коефіцієнт взятий з алфавіту системи числення;
S – основа системи числення (наприклад, 2,8,10,16).

Скорочений запис числа *N* має вигляд:

$$N_s=a_ma_{m-1}\ldots a_1a_0,a_{-1}a_{-2}\ldots$$

У цій послідовності кома відокремлює цілу частину числа від дробової частини. Кома опускається, якщо немає від'ємних степенів. Позиції *m* коефіцієнтів *a* називають *розрядами.* У позиційній системі числення значення кожного розряду більше від значення сусіднього правого розряду в *S* раз.

У комп'ютерах застосовуються такі позиційні системи числення: десяткова, двійкова, вісімкова і шістнадцяткова.

Алфавіт *десяткової системи числення* складається з десяти різних цифр: 0,1, 2,..., 9. У цій системі "вага" кожного розряду в 10 разів більша від "ваги" попереднього. Наприклад, у записі 1987 цифра 1 означає кількість тисяч, цифра 9 – кількість сотень, цифра 8 – кількість десятків і цифра 7 – кількість одиниць.

Будь-яке число в десятковій системі числення можна виразити відповідно до формули (4.1) сумою різних цілих степенів десяти *(S* = 10) з відповідними коефіцієнтами *a* (0,1,2, ...9):

$$N_{10} = a_m 10^m + a_{m-1} 10^{m-1} + ... + a_1 10^1 + a_0 10^0 + a_{-1} 10^{-1} ...,$$

де N_{10} – число в десятковій системі числення;

m – номер розряду;

$a_0, a_1, ..., a_m$ – кількість одиниць, десятків, сотень і т. д.;

$a_{-1}, a_{-2}, ...$ – кількість десятих, сотих, тисячних і т. д. часток одиниці.

Ірраціональні числа, наприклад число π, а також деякий дріб, наприклад 1/3, не можна точно виразити за допомогою кінцевої послідовності цифр. У цьому разі беруть їх наближення із заданою точністю.

Вибір тієї чи іншої системи числення для подання чисел довільний. Так, вибір десяткової системи пояснюється тим, що людина має на руках 10 пальців. Однак різні народи в різні періоди часу користувалися й іншими системами числення. Так, у стародавньому Вавилоні поряд з десятковою системою числення широко використовували і шістдесяткову систему числення. Сліди шістдесяткових дробів зберігаються й донині в діленні кола на 360°, години на 60 хвилин і хвилини на 60 с.

Зрозуміло, що не існує максимальної основи системи числення, тобто основа системи числення може бути як завгодно велика. Водночас існує мінімальна основа системи числення, що дорівнює 2. Цю систему числення називають *двійковою системою числення,* в алфавіті якої є тільки дві цифри: 0 і 1.

Будь-яке дійсне число в двійковій системі числення можна виразити у вигляді суми цілих степенів основи S = 2, помножених на відповідні коефіцієнти (0 чи 1).

Наприклад, двійкове число 11011.011 можна подати так:

$$11011.01_2 \approx 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} =$$

$$= 16 + 8 + 2 + 1 + 0.25 = 27.25_{10}$$

Для фізичного зображення чисел потрібні елементи, здатні знаходитися в одному з декількох стійких станів. Кількість цих станів мають дорівнювати основі прийнятої системи числення. Тоді кожний стан буде мати відповідну цифру з алфавіту цієї системи числення. Найпростіші

з погляду технічної реалізації двопозиційні елементи здатні знаходитися в одному з двох стійких станів.

Прикладами таких двопозиційних елементів можуть бути:

- електромагнітне реле (стан: замкнутий чи розімкнутий);
- феромагнітна поверхня (стан: намагнічена чи розмагнічена);
- магнітний сердечник (стан: намагнічений в одному напрямі чи в іншому);
- транзистор (стан: проводить струм чи не проводить струму).

Одному із цих стійких станів ставим у відповідність 0, а другому – цифру 1.

Саме простота і забезпечила найбільше поширення в комп'ютерах двійкової системи числення.

Двійкове подання числа порівняно з десятковим потребує більшої кількості розрядів (для багаторозрядного числа приблизно в 3,3 раза). Завдяки простоті, швидкодії і дешевизні технічної реалізації двопозиційних елементів двійкова система числення на тепер є основною системою, застосовуваною в комп'ютерах для подання інформації та виконання арифметичних (*arithmetic operations*) і логічних операцій *(logical operations)*.

За допомогою відповідних програм десяткові числа з введенням у комп'ютер перетворюються в двійкові числа, а в разі виведення виконується обернене перетворення.

Для того щоб перевести десяткове число в двійкове, досить знати структуру коду і значення розрядів. Вони зростають справа наліво в порядку зростання номерів розрядів. Їх легко запам'ятати: 1, 2, 4, 8, 16 і т.д. Користуючись табл. 4.3, неважко знайти десятковий еквівалент будь-якого двійкового числа розрядністю не більше 8.

Таблиця 4.3

Вагові значення розрядів і коди чисел											Приклади десяткових чисел
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	
128	64	32	16	8	4	2	1	0,5	0,25	0,125	
					1	1	1				7
				1	1	0	1				13
		1	1	1	0	1	0				58
		1	1	0	1	0	1,	0	1	1	53,375
1	1	1	1	1	1	1	1				255
							0,	0	1	0	0,250
1	0	0	0	0	0	0	1				129

Так, десяткове число 7 можна зобразити сумою трьох вагових значень молодших розрядів: $7_{10} = 4 + 2 + 1$

Відповідно до цього позначаємо одиниці в правих трьох розрядах: $7_2 = 111_2$. У всіх інших розрядах (в цьому прикладі попереду трьох одиниць) позначаємо нулі, тобто можна записати: $7_{10} = 00000111_2$. Це було б еквівалентним записом. Неважко переконатися, що число 13 в двійковій системі буде мати вигляд 00001101.

Відмітимо, що найбільше десяткове число, яке можна записати восьмирозрядним двійковим числом – 256, а шістнадцятирозрядним – 65535 (відповідно $2^8 = 256$, а $2^{16} = 65536$).

Так само кодуються і дробові числа. Для цього важливо враховувати, що вагові значення розрядів дробової частини числа зменшуються після коми в такому порядку: 0,5; 0,25; 0,125 і т.д. Таким чином дробовому числу 7,125 відповідає двійковий код 00000111, 0010000, а числу 58,5625 – 111010,1001.

Як же вчинити якщо десятковий дріб не кратний 5?

Існує правило:

Необхідно дробову частину числа помножити на 2, якщо результат за кількістю знаків виходить більше дробової частини, то отриману одиницю (а виникає тільки 1) записують в розряд цілої частини (це і є двійковий код), а якщо результат за кількістю знаків збігається з числом дробової частини, то на місце цілої частини ставлять нуль (це теж двійковий код).

Приклад 4.1. Перевести десяткове дробове число $0{,}28125_{10}$ у двійкову систему числення.

Обчислення проводять за такою схемою:

0	28125
	×
	2
0	56250
	×
	2
1	12500
	×
	2
0	25000
	×
	2
0	50000
	×
	2
1	00000

Результат: $0{,}28125_{10}=0{,}01001_2$

Приклад 4.2. Перевести десяткове дробове число $0{,}9_{10}$ у двійкову систему числення.

0,	9 × 2
1	8 × 2
1	6 × 2
1	2 × 2
0	4 × 2
0	8
	……

Очевидно, що цей процес може продовжуватись без кінця, даючи все нові і нові знаки в відображенні двійкового еквівалента числа $0{,}9_{10}$. Так, за чотири кроки ми отримаємо число $0{,}1110_2$, а за сім кроків – число $0{,}1110011_2$, яке є найбільш точним поданням числа $0{,}9_{10}$ в двійковій системі числення і т.д. Такий нескінченний процес переривають на певному кроці, коли вважають, що отримана потрібна точність подання числа.

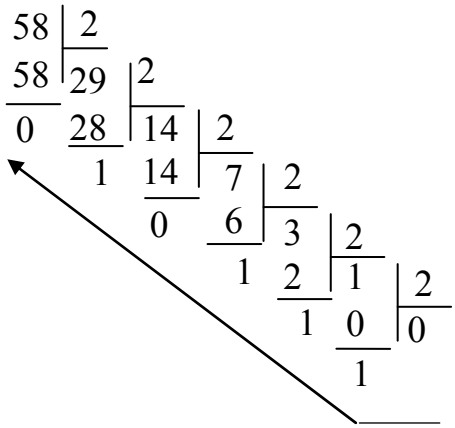
Оскільки люди в науковій і виробничій практиці користуються десятковою системою числення, то виникає задача пошуку простих правил (правильніше, алгоритмів) для переводу чисел із десяткової системи в двійкову:

1) поділити число на 2. Зафіксувати залишок (0 чи 1) і різницю;
2) якщо частка не дорівнює 0, то поділити її на 2 і так далі.

Якщо частка дорівнює 0, то записати всі отримані залишки, починаючи з першого, зліва направо.

Приклад 4.3. Записати десяткове число 58_{10} у двійковій системі числення.

Ділення у стовпчик виконується так, як показано нижче. Цифрами шуканого числа, рівного даному, є всі залишки. Поява у частці нуля вказує на кінець процесу ділення:



Залишки, починаючи з останнього, виписані знизу вверх, утворюють шукане двійкове число: $58_{10} = 111010_2$.

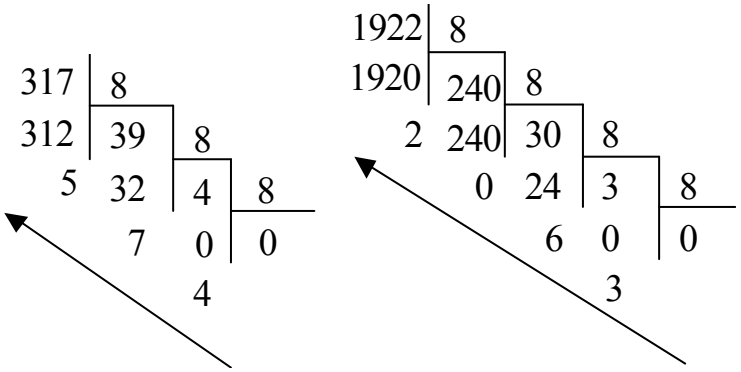
У процесі програмування і налагодження програм часто доводиться використовувати двійкові коди команд програми, адрес і даних. Двійкові числа довгі і, крім того, важкі для сприйняття. Тому для скороченого і зручного запису двійкових чисел часто використовують вісімкову і шістнадцяткову системи числення.

У *вісімковій системі числення* використовують вісім цифр – від 0 до 7, а будь-яке число подають сумою цілих степенів основи 8, помножених на відповідні коефіцієнти *a* (0, 1, ..., 7).

Наприклад, число 215_{10} записується у вісімковій системі числення в такий спосіб: $215_{10} = 3 \cdot 8^2 + 2 \cdot 8^1 + 7 \cdot 8^0 = 327_8$.

Для того, щоб замінити десяткове ціле число на рівне йому вісімкове використовується алгоритм послідовного ділення цього числа на 8.

Приклад 4.4. Записати десяткові числа 317_{10} і 1922_{10} у вісімковій системі числення:



Отже, маємо: $317_{10}=475_8$; $1922_{10}=3602_8$.

Достатньо простий і зворотний перехід від двійкового подання будь-якого числа у вісімкове.

Для цього в двійковому записі числа потрібно виділити тріаду (вліво і вправо від коми) і замінити кожну тріаду відповідною вісімковою цифрою. У випадку необхідності неповні тріади доповняються нулями.

Приклад 4.5. Перевести двійкові числа 1010110_2 та $11110100{,}010\ 11101111000011_2$ у вісімкову систему числення.

1) $1010110_2 = \underbrace{001}_{1}\ \underbrace{010}_{2}\ \underbrace{110}_{6}{}_2 = 126_8$

2) $11\ 110\ 100,\ 010\ 111\ 011\ 110\ 000\ 11_2 =$

$\underbrace{011}_{3}\ \underbrace{110}_{6}\ \underbrace{100}_{4},\ \underbrace{010}_{2}\ \underbrace{111}_{7}\ \underbrace{011}_{3}\ \underbrace{110}_{6}\ \underbrace{000}_{0}\ \underbrace{110}_{6}{}_2 = 364{,}273606_8$

Приклад 4.6. Перевести десяткове дробове число $0{,}65625_{10}$ у вісімкову систему числення.

0,	65625	
	× 8	
5	25000	
	× 8	
2	00000	

Відповідь: $0{,}65625_{10} = 0{,}52_8$.

При переводі змішаних десяткових чисел (наприклад, $124{,}25_{10}$) у вісімкову систему, окремо переводять цілі і дробові частини.

Приклад 4.7. Перевести дробове число $124{,}25_{10}$ у вісімкову систему.

124	8	
4	15	8
7	1	

0,	× 25	
	8	
2	00	

Відповідь: $124{,}25_{10} = 174{,}2_8$.

У *шістнадцятковій системі числення* алфавіт цифрових знаків складається із 16 символів, причому як перші десять символів

використовують арабські цифри від 0 до 9, а додатково до них – буквені символи: 10 – A(a), 11 – B(b), 12 – C(c), 13 – D(d), 14 – E(e), 15–F(f).

Число 215_{10} у шістнадцятковій системі числення записують так:

$$215_{10} = D \cdot 16^1 + 7 \cdot 16^0 = D7_{16}$$

Приклади 4.8

$15_{10}=F_{16}$;

$31_{10}=1*16^1 + F*16^0=1F_{16}$;

$167_{10}=10*16^1+7*16^0=A7_{16}$;

$6C_{16}=\underbrace{0110}_{6}\ \underbrace{1100}_{C}{}_{2}$;

$11111011111_2=\underbrace{0111}_{7}\ \underbrace{1101}_{D}\ \underbrace{1111}_{F}{}_{2}=7DF_{16}$;

$11101001000,11010010_2=\underbrace{0111}_{7}\ \underbrace{0100}_{4}\ \underbrace{1000}_{8},\ \underbrace{1101}_{D}\ \underbrace{0010}_{2}=748,D2_{16}$.

Приклад 4.9. Перевести десяткове число $0,65625_{10}$ в шістнадцяткову систему числення.

0,	65625 ×
	16
10	50000 ×
(A)	16
8	00000

Відповідь: $0,65625_{10}=0,A8_{16}$.

Правила переводу *двійково-десяткового коду* в десяткову систему і навпаки аналогічні переводу по тетрадах двійкових чисел в шістнадцяткову систему числення з урахуванням заборонених комбінацій.

Приклад 4.10. Перевести двійково-десятковий код $010101110011,0100_{2\text{-}10}$ у десяткову систему.

$$\underbrace{0101}_{5}\ \underbrace{0111}_{7}\ \underbrace{0011}_{3}\ \underbrace{0100}_{4}{}_{2\text{-}10}=573,4_{10}$$

Приклад 4.11. Перевести десяткове число 362,8910 у двійково-десятковий код.

$$362{,}89_{10} = \underbrace{11\ 0110\ 0010}_{3\ \ 6\ \ 2},\ \underbrace{1000\ 1001}_{8\ \ 9}{}_{2\text{-}10}.$$

Для переводу правильного дробу, записаного в системі числення з основою p, в дріб, записаний в системі числення з основою q існує таке правило:

1) основу нової системи числення виразити цифрами вхідної системи числення і всі наступні дії проводити у вхідній системі числення;

2) послідовно множити дане число і отримані дробові частини ділення на основу нової системи до тих пір, поки дробова частина ділення не стане дорівнювати нулю чи не буде отримана потрібна точність подання числа;

3) отримані цілі частини ділення, які є цифрами числа в новій системі числення, привести відповідно до алфавіту нової системи числення;

4) додати дробові частини числа в новій системі числення, починаючи з цілої частини першого ділення.

Існують різні способи переведення чисел з однієї системи числення в іншу. Розглянемо загальні правила переведення чисел з однієї позиційної системи числення в іншу.

Переведення *цілого числа з десяткової системи числення в систему з основою S* здійснюється послідовним діленням його на основу *S* нової системи числення доти, доки частка буде меншою від *S.* Число в новій системі запишеться у вигляді остачі ділення, починаючи з останнього.

Переведення *правильного дробу* (меншого за 1) з *десяткової системи числення в систему з основою S* здійснюється послідовним множенням її на основу *S,* при цьому перемножуються тільки дробові частини. Дріб у новій системі числення записують у вигляді цілих частин отриманих добутків, починаючи з першого.

Для переведення *неправильного дробу* (більшого за 1) потрібно виконати окремо переведення цілої і дробової частин.

Операції ділення і множення виконуються в десятковій системі числення. Для переведення *чисел із системи числення S у десяткову систему числення* зручніше скористатися формулою (4.1). Оскільки основи вісімкової і шістнадцяткової систем числення відповідають цілим степеням числа 2 ($8 = 2^3$, $16 = 2^4$), для них застосовують прості правила переведення в двійкову систему числення і навпаки. Кожні три цифри двійкового числа перетворяться в одну цифру вісім нового числа (якщо довжина двійкового числа не кратна трьом, спочатку додається відповідна кількість нулів). У

разі оберненого перетворення кожна цифра вісімкового числа перетвориться в три двійкові цифри.

Аналогічно виконуються взаємні перетворення шістнадцяткових і двійкових чисел, за винятком того, що число двійкових цифр дорівнює чотирьом.

Приклад 4.12

1. Переведення числа 377_{10} у шістнадцяткову систему числення:
377:16 = 23 (остача 9); 23:16 = 1 (остача 7); 1:16 = 0 (остача 1).
Результат: 179_{16}.

2. Переведення дробого десяткового числа $0{,}6875_{10}$ у вісімкову систему числення:

$$0.6875 \cdot 8 = 5.5000(5); \quad 0.5000 \cdot 8 = 4.0000(4)$$

Результат: $0{,}54_8$.

3. Переведення числа $FCA1_{16}$ у десяткову систему числення:

$$FCA1_{16} = 15 \cdot 16^3 + 12 \cdot 16^2 + 10 \cdot 16^1 + 1 \cdot 16^0 = 61440 + 3072 + 160 + 1 = 64673_{10}$$

Результат: 64673_{10}

4.Переведення числа 1111111_2 у шістнадцяткову систему числення:
$01111111_2 =$ | 0111 | 1111 | $=7F_{16}$

5. Переведення числа $6C8_{16}$ у двійкову систему числення:
$6C8_{16} =$ | 0110 | 1100 | 1000 | $= 011011\ \ 001000_2$

Арифметичні операції в системі числення S виконуються так само, як і в десятковій системі, але треба враховувати, що у разі додавання і множення одиниця переводиться в старший розряд, коли сума чи добуток чисел більші від основи *S*. У разі віднімання в старшому розряді позичається кількість одиниць, що також дорівнює основі *S*.

Системи числення, які використовуються в ЕОМ, наведені табл. 4.4.

Таблиця 4.4 – Системи числення

Система числення	Основа	Алфавіт	Приклади відповідності десятковим числам
Двійкова	2	1,0	A=11011; A=27
Вісімкова	8	0,1,2,3,4,5,6,7	A=23; A=19
Шістнадцяткова	16	0,1,2,3,4,5,6,7 A,B,C,D,E,F	A=2B; A=43
Двійково-десяткова	2	1,0	A=01000101; A=45

4.3 Формати подання даних

Будь-яка інформація (числа, команди, алфавітно-цифрові записи тощо) подається в комп'ютері у вигляді двійкових кодів. Окремі елементи двійкового коду, що набувають значення 0 чи 1, називають *розрядами* чи *бітами.*

У старих комп'ютерах, призначених для обчислювальних задач, мінімальною одиницею інформації, доступної для оброблення, була *комірка.* Кількість розрядів у комірці орієнтовано на подання чисел і вона різна у різних комп'ютерах (24 біт, 48 біт і т. д.). Однак такий великий розмір комірки був незручний для подання символів, оскільки для подання символьної інформації достатньо 5...8 біт. Це дає можливість подати від 32 до 256 символів. Мінімальною одиницею інформації, оброблюваної в сучасному комп'ютері, є *байт,* що складається з восьми двійкових розрядів (бітів). Кожен байт, розміщений у пам'яті комп'ютера, має свою адресу, що визначає його місцезнаходження і задається відповідним кодом. Адреси пам'яті починаються з нуля для першого байта і послідовно збільшуються на одиницю для кожного наступного біта.

Похідні одиниці від байта – кілобайт (2^{10} байт) – Кбайт; мегабайт ($2^{20)}$ байт) – Мбайт; гігабайт (2^{30} байт) – Гбайт; терабайт (2^{40}байт) – Тбайт і петабайт (2^{50} 6айт) – Пбайт.

Для подання чисел використовують один чи декілька послідовно розміщених байтів. Групи байтів утворюють двійкові слова, що, у свою чергу, можуть бути як фіксованої, так і змінної довжини.

Формати даних *фіксованої довжини* (півслово, слово і подвійне слово) складаються відповідно з одного, двох і чотирьох послідовно розміщених байтів. Звернення до цих даних виконується за адресою крайнього лівого байта числа, що для слова має бути кратним числу 2, а для подвійного слова – числу 4.

Формат даних *змінної довжини* складається з групи послідовно розміщених байтів від 1 до 256. Адресація таких даних виконується, як і у форматах фіксованої довжини, за адресою найлівішого байта.

Залежно від характеру інформації використовують формати подання даних як фіксованої, так і змінної довжини. Так, у форматах даних фіксованої довжини зазвичай подаються двійкові числа, команди і деякі логічні дані, а у форматах даних змінної довжини – десяткові числа, алфавітно-цифрова і деяка логічна інформація.

У сучасних комп'ютерах застосовують дві форми подання чисел: з фіксованою точкою (комою) і з плаваючою точкою (комою). Ці форми, крім того, називають відповідно *природною* і *напівлогарифмічною*.

У разі подання *чисел з фіксованою точкою* (в першій формі) положення точки фіксується у певному місці відносно розрядів числа. У перших комп'ютерах точка фіксувалася перед старшим розрядом числа,

тому подані числа за абсолютною величиною були менші від одиниці. У сучасних комп'ютерах точка фіксується праворуч від наймолодшого розряду і тому можуть подаватися тільки цілі числа. При цьому використовують два варіанти подання цілих чисел: зі знаком і без знака.

Для *числа зі знаком* крайній розряд ліворуч потрапляє під знак числа. У цьому розряді записується нуль для додатних чисел і одиниця – для від'ємних. *Числа без знака* займають усі розряди числа, тобто числа можуть бути тільки додатними. Нумерація розрядів числа зазвичай ведеться справа наліво.

У комп'ютерах числа з фіксованою точкою мають три основні формати – один байт (півслово), 16-розрядне слово (короткий формат) і 32-розрядне подвійне слово (довгий формат).

На рис. 4.6 показано формати подання чисел з фіксованою точкою зі знаком (рис. 4.6, *а*) і без знака (рис. 4.6, *б)* довжиною в півслово (числа в короткому і довгому форматах мають аналогічні подання).

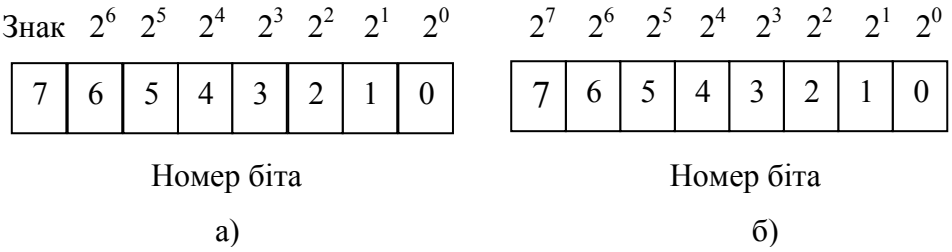


Рисунок 4.6 – Подання числа з фіксованою точкою довжиною в півслово: *а)* зі знаком; *б)* без знака

Слід зазначити, що інтерпретацію числа з фіксованою точкою, як і числа зі знаком чи без знака, має виконувати програма оброблення цих чисел. Так, число $01110011_2 = 73_{16}$ буде мати десяткове значення 115_{10} і як число зі знаком, і як число без знака. Однак число $11101101_2 = ED_{16}$ буде мати десяткове значення мінус 109_{10} як число зі знаком і мінус 237_{10} – як число без знака.

Діапазон зміни чисел з фіксованою точкою зі знаком X становить:

$-2^{n-1} \le X \le 2^{n-1}-1$, а чисел без знака: $0 \le X \le 2^n-1$, де n – розрядність числа.

Так, для $n = 8$ діапазон зміни чисел зі знаком – від -128_{10} до $+127_{10}$, а чисел без знака - від 0_{10} до 255_{10}.

Для подання чисел, що не вкладаються в діапазон подвійного слова, у сучасних комп'ютерах можна використовувати два варіанти:

- уведення й оброблення чисел довільної довжини (наприклад, 8 чи 10 байт);
- використання масштабних коефіцієнтів.

За обома варіантами час виконання операцій над числами істотно збільшується (за першим варіантом через велику довжину числа, за

другим – через потребу вручну відслідковувати положення десяткової точки в числі).

Для подання чисел з фіксованою точкою відносна точність виконуваних розрахунків залежить від величини чисел і є максимальною у разі виконання операцій з максимально можливими числами. Тому поряд з поданням чисел з фіксованою точкою у сучасних комп'ютерах використовують також другу форму – *подання чисел з плаваючою точкою.*

Будь-яке число *N,* подане як число з плаваючою точкою, є добутком двох співмножників:

$$N = m \cdot S^p,$$

де *m* – мантиса числа N(|m| < 1);

S – основа системи числення;

p – цілочисловий порядок.

Зі зміною порядку в той чи інший бік точка нібито «плаває» у зображенні числа.

Прикладом записування десяткового числа як числа з плаваючою точкою є *експонентна форма* запису, наприклад, $0.35 \cdot 10^{12}$ чи $-0.1563 \cdot 10^{-12}$.

Отже, для подання чисел із плаваючою точкою потрібно записати в комп'ютер зі своїми знаками мантису *m* і порядок *p.* І мантиса, і порядок записуються в двійковому вигляді, тобто зі значенням *S=2.* Знак числа при цьому збігається зі знаком мантиси.

Щоб спростити операції з порядками, їх зводять до дій над цілими додатними числами використанням *зміщеного порядку,* що завжди додатний. Зміщений порядок *p* утвориться додаванням до порядку *p* числа 2^n (де *n* – кількість бітів, що відводиться для значення порядку числа). Наприклад, для *n-1* $p_m = p + 64$ порядок набуватиме значення від 0 (якщо *p* = - 64) до 127 (якщо *p* = 63). У цьому разі, якщо *p* = -15, зміщений порядок = -15 + 64 = 49.

Кількість розрядів, виділених для зображення порядків, визначає діапазон чисел із плаваючою точкою у комп'ютері. Крім того, велике значення має точність подання чисел, що підвищується зі збільшенням кількості розрядів мантиси.

Тому з урахуванням різних вимог, пропонованих до точності розв'язання задач, у комп'ютерах зазвичай використовують кілька форматів. У комп'ютерах для подання значень із плаваючою точкою використовується формат *IEEE* (Institute of Electrical and Electronics Engineers – Інститут інженерів з електротехніки й електроніки), що визначає три формати подання чисел: звичайний, подвійної точності і довгий.

Звичайний формат займає подвійне слово (32 біт), що складається: з біта знака, 7-бітового двійкового порядку і 24-бітової мантиси, що подає число в діапазоні 1,0 ... 2,0. Оскільки старший біт мантиси завжди дорівнює одиниці, він не зберігається в пам'яті. Це подання дає сім значущих цифр і діапазон значень приблизно $3{,}4 \cdot 10^{-38}$... $3{,}4 \cdot 10^{+38}$.

Формат подвійної точності займає подвійне слово (64 біт). Цей формат аналогічний короткому формату, за винятком того, що порядок займає 11 біт, а мантиса – 52 біт (плюс неявний старший одиничний біт). Це подання містить 15 значущих цифр і діапазон значень чисел – близько $1{,}7 \cdot 10^{-308}$... $1{,}7 \cdot 10^{+308}$.

Довгий формат займає 10 байт (80 біт). Його подання аналогічне поданню чисел з подвійною точністю, за винятком того, що мантиса займає 68 біт. Кількість значущих цифр для цього формату – 19, а діапазон значень – близько $3.4 \cdot 10^{-4932}$... $1.1 \cdot 10^{+4932}$.

Третя форма подання чисел у комп'ютерах – це *двійково-десяткова форма.* У цій формі кожна цифра десяткового числа зберігається в чотирьох бітах, тобто дві цифри на один байт. Цифри від 0 до 9 виражені двійковими кодами від 0000 до 1001. Двійкові значення 1100 і 1101 використовують відповідно для знаків *«+»* і «-». Положення десяткової точки в цьому випадку фіксується і відслідковується програмними засобами. Наприклад: -142_{10}=1101 0001 0100 $0010_{2\text{-}10}$

Для цієї форми подання чисел, так само, як і для чисел з фіксованою і плаваючою точками, визначено арифметичні операції. Однак таку форму подання чисел тепер використовують украй рідко, здебільшого для оброблення великих масивів десяткових чисел.

Для спрощення арифметичних операцій числа в комп'ютері подаються спеціальними кодами – прямим, оберненим і додатковим.

Прямий код двійкового числа містить цифрові розряди, ліворуч від яких записується знаковий розряд. Додавання в прямому коді чисел, що мають однакові знаки, виконується досить просто. Цифрові розряди чисел додаються за правилами арифметики, і сумі присвоюється код знака доданків. Значно складніше реалізовується в прямому коді операція алгебричного додавання, тобто додавання чисел, що мають різні знаки. У цьому разі доводиться визначати більше за модулем число, вираховувати числа і присвоїти різниці знак більшого за модулем числа.

За допомогою оберненого і додаткового кодів операція віднімання (чи алгебричного додавання) зводиться до арифметичного додавання, спрощується визначення знака результату операції, а також полегшується вироблення ознак переповнення результату (коли в результаті арифметичних операцій число стає більшим від максимально допустимого для цієї форми значення). *Обернений код* від'ємного числа одержується за таким правилом: у знаковий розряд числа записується одиниця, у цифрових розрядах нулі замінюються одиницями, а одиниці – нулями.

Додатковий код від'ємного числа отримують з оберненого коду додаванням одиниці до молодшого розряду.

Під час виконання операції алгебричного додавання з використанням оберненого чи додаткового коду додатні числа подаються прямим кодом, а від'ємні – оберненим або додатковим кодом. Потім виконується арифметикне підсумовування цих кодів, включаючи знакові розряди, що при цьому розглядаються як старші. У разі використання оберненого коду виникла одиниця перенесення зі знакового розряду циклічно додається до молодшого розряду суми кодів, а у разі використання додаткового коду ця одиниця вилучається.

Як відомо, комп'ютери можуть обробляти тільки інформацію, подану в числовій формі. Під час зчитування документів, текстів програм та інших матеріалів введені букви кодуються відповідними числами, а у разі виведення їх для читання людиною (на монітор, принтер і т. ін.) за кожним числом будується зображення символу. Відповідність між набором символів і їх кодів називають *кодуванням символів (symbolic coding).*

Зазвичай код символу зберігається в одному байті. Код символу розглядається як число без знака і, отже, може набувати значень від 0 до 255. Такі кодування називають однобайтовими; вони дозволяють використовувати до 256 різних символів. Тепер дедалі більшого поширення набуло двобайтове кодування *Unicode,* за якого коди символів можуть набувати значень від 1 до 65535. У цьому кодуванні є номери для майже всіх застосовуваних символів (букв та ієрогліфів різних мов, математичних, декоративних символів тощо).

Кодування символів зазвичай визначається використовуваною операційною системою чи програмною оболонкою.

Загалом наявність у сучасних комп'ютерах різних форм і форматів подання чисел дозволяє вибирати ті з них, що найбільшою мірою відповідають вимогам розв'язуваних задач.

КОНТРОЛЬНІ ЗАПИТАННЯ ТА ЗАВДАННЯ

1. *Наведіть властивості нескінченної множини, яку утворюють раціональні числа.*
2. *Які існують способи вираження функцій?*
3. *У чому відмінність аналогових сигналів від дискретних?*
4. *Яким чином аналогові сигнали перетворюють у дискретні?*
5. *Що таке система числення?*
6. *Наведіть приклади позиційних та непозиційних систем числення.*
7. *Перетворіть десяткове число 144_{10} у двійкове.*

8. *Перетворіть десяткове число 221_{10} у двійкове.*
9. *Перетворіть десяткове число 512_{10} у двійкове.*
10. *Перетворіть дробове десяткове число $123,43_{10}$ у двійкове.*
11. *Перетворіть дробове десяткове число 101.25_{10} у двійкове.*
12. *Перетворіть вісімкове число 44_8 у десяткове.*
13. *Перетворіть вісімкове число 99_8 у двійкове.*
14. *Перетворіть вісімкове число 145_8 у двійкове.*
15. *Перетворіть вісімкове число 267_8 у шістнадцяткове.*
16. *Перетворіть вісімкове число 341_8 ушістнадцяткове.*
17. *Перетворіть десяткове число 299_{10} у шістнадцяткове.*
18. *Перетворіть десяткове число 172_{10} у шістнадцяткове.*
19. *Перетворіть десяткове число 236_{10} у вісімкове.*
20. *Перетворіть десяткове число 196_{10} у вісімкове.*
21. *Перетворіть шістнадцяткове число $2F_{16}$ у десяткове.*
22. *Перетврість шістнадцяткове число $6B3_{16}$ у десяткове.*
23. *Перетворіть шістнадцяткове число AC_{16} у вісімкове.*
24. *Перетворіть шістнадцяткове число $D9_{16}$ у двійкове.*
25. *Перетворіть шістнадцяткове число $F9_{16}$ у двійкове.*
26. *Перетворіть двійкове число 10111001_2 у десяткове.*
27. *Перетворіть двійкове число 111100001_2 у десяткове.*
28. *Перетворіть двійкове число 111001_2 у шістнадцяткове.*
29. *Перетворіть двійкове число 101011_2 у шістнадцяткове.*
30. *Перетворіть двійкове число 100100111_2 у вісімкове.*
31. *Перетворіть двійкове число 101000101_2 у вісімкове*
32. *Перетворіть двійково-десяткове число $111011110,1001_{2\text{-}10}$ у десяткове.*
33. *Перетворіть двійково-десяткове число $110010100,1101_{2\text{-}10}$ у десяткове.*
34. *Перетворіть десяткове число 234_{10} у двійково-десяткове.*
35. *Перетворіть десяткове число 168_{10} у двійково-десяткове.*
36. *Перетворіть десяткове число 317_{10} у двійково-десяткове.*
37. *Наведіть правила переведення правильного дробу з однієї системи в іншу.*
38. *Яким чином здійснюється переведення цілого числа з десяткової системи числення в систему з основою S?*
39. *Яким чином в сучасних комп'ютерах здійснюється подавання чисел з плаваючою точкою?*
40. *Які існують формати подання чисел у компютері?*
41. *Для чого здійснюється кодування символів у комп'ютері?*
42. *Яким чином здійснюються арифметичні операції в системах числення з основою S?*

5 ЛОГІЧНІ ОСНОВИ ЕОМ

5.1 Основи булевої алгебри

Булева алгебра оперує логічними змінними, які можуть приймати тільки два значення: істина або хибність, які позначають відповідно 1 і 0. Як відмічалось раніше основною системою числення ЕОМ є двійкова система, в якій також використовуються цифри 1 і 0. Таким чином одні і ті ж пристрої ЕОМ можуть використовуватись для обробки та зберігання як числової інформації, поданої в двійковій системі числення, так і логічних змінних. Це зумовлює порівняно просту схемну реалізацію процесу обробки інформації в ЕОМ.

Сукупність значень логічних змінних $x_1, x_2 \ldots x_n$ називається набором змінних. Набір логічних змінних зручно відображати у вигляді *n*-розрядного двійкового числа, кожний розряд якого дорівнює значенню однієї із змінних. Неважко помітити, що кількість можливих наборів в n двійкових розрядах рівна 2^n.

Логічною функцією від набору логічних змінних *(logical values)*(аргументів) $f(x_1, x_2 \ldots x_n)$ називається функція, яка може приймати також тільки два значення: істина *(true)* або хибність *(false)*. Область визначення логічної функції скінченна і залежить від числа можливих наборів аргументів. Будь-яка логічна функція може бути задана за допомогою таблиці істинності, в лівій частині якої записуються можливі набори аргументів, а в правій – відповідні їм значення функції.

У випадку великої кількості аргументів табличний спосіб задання логічної функції стає громіздким і втрачає наглядність. Так, уже для шести логічних аргументів потрібна таблиця в 64 рядки. Тому логічні функції зручно виражати через інші, більш прості логічні функції одної чи двох змінних, які описуються за допомогою простих таблиць. Сукупність таких елементарних логічних функцій (або логічних операцій), за допомогою яких можна виразити логічну функцію будь-якої складності, називаються *функціонально повними системами логічних функцій*.

Загальне число логічних функцій *n* змінних визначається за формулою 2^{2^n}. Таким чином існує 4 логічних функції однієї логічної змінної (таблиця 5.1). Дві з цих функцій є константами: $f_0(x)=0$, $f_3(x)=1$.

Функція $f_2(x)$ називається запереченням змінної або інверсією і позначається як $f_2(x) = \bar{x}$. Функція $f_1(x)$ повторює значення змінної, тобто $f_1(x) = x$.

Таблиця 5.1

x	$f_0(x)$	$f_1(x)$	$f_2(x)$	$f_3(x)$
0	0	0	1	1
1	0	1	0	1

Число логічних функцій двох змінних рівно 16 (табл. 5.2). Із них 6 функцій є виродженими:

$f_0(x_1, x_2) = 0$; $f_{10}(x_1, x_2) = \overline{x_2}$;

$f_3(x_1, x_2) = x_1$; $f_{12}(x_1, x_2) = \overline{x_1}$;

$f_5(x_1, x_2) = x_2$; $f_{15}(x_1, x_2) = 1$.

Таблиця 5.2 – Логічні функції двух змінних

Змінні та їх стани					Назва функції	Вираження функції через елементарні логічні операції	Позначення операції
x_1	0	0	1	1			
x_2	0	1	0	1			
$f_0(x_1,x_2)$	0	0	0	0	-	0	-
$f_1(x_1,x_2)$	0	0	0	1	-	$x_1 \cdot x_2$	-
$f_2(x_1,x_2)$	0	0	1	0	Заперечення за x_2	$x_1 \cdot \overline{x_2}$	-
$f_3(x_1,x_2)$	0	0	1	1	-	x_1	-
$f_4(x_1,x_2)$	0	1	0	0	Заперечення за x_1	$\overline{x_1} \cdot x_2$	-
$f_5(x_1,x_2)$	0	1	0	1	-	x_2	-
$f_6(x_1,x_2)$	0	1	1	0	Додавання за модулем 2 (нерівнозначність)	$x_1 \cdot \overline{x_2} \vee \overline{x_1} \cdot x_2$	$x_1 \oplus x_2$
$f_7(x_1,x_2)$	0	1	1	1	-	$x_1 \vee x_2$	-
$f_8(x_1,x_2)$	1	0	0	0	Стрілка Пірса (заперечення диз'юнкції)	$\overline{x_1 \vee x_2} = \overline{x_1} \cdot \overline{x_2}$	$x_1 \downarrow x_2$
$f_9(x_1,x_2)$	1	0	0	1	Еквівалентність (рівнозначність)	$\overline{x_1} \cdot \overline{x_2} \vee x_1 \cdot x_2$	$x_1 \infty x_2$
$f_{10}(x_1,x_2)$	1	0	1	0	-	$\overline{x_2}$	-
$f_{11}(x_1,x_2)$	1	0	1	1	Імплікація	$x_1 \vee \overline{x_2}$	$x_2 \rightarrow x_1$
$f_{12}(x_1,x_2)$	1	1	0	0	-	$\overline{x_1}$	-
$f_{13}(x_1,x_2)$	1	1	0	1	Імплікація	$\overline{x_1} \vee x_2$	$x_1 \rightarrow x_2$
$f_{14}(x_1,x_2)$	1	1	1	0	Штрих Шефера (заперечення кон'юнкції)	$\overline{x_1 \cdot x_2} = \overline{x_1} \vee \overline{x_2}$	x_1 / x_2
$f_{15}(x_1,x_2)$	1	1	1	1	-	1	-

Функції $f_1(x_1, x_2)$ і $f_7(x_1, x_2)$ разом з функцією інверсії складають функціонально–повну систему логічних функцій, які найбільш часто використовуються на практиці. В цій системі визначені 3 елементарні логічні операції: інверсія (заперечення), кон'юнкція (логічне множення), диз'юнкція (логічне додавання). Операція кон'юнкції (функція f_1)

позначається знаком $\wedge$, точкою або зовсім без знаку. Операція диз'юнкції (f_7) позначається знаком $\vee$ або знаком +. На рисунку 5.1 наведені таблиці істинності цих логічних операцій.

Інверсія	
x	$\overline{x}$
0	1
1	0

Кон'юнкція		
x_1	x_2	$x_1 \cdot x_2$
0	0	0
0	1	0
1	0	0
1	1	1

Диз'юнкція		
x_1	x_2	$x_1 + x_2$
0	0	0
0	1	1
1	0	1
1	1	1

Рисунок 5.1 – Таблиці істинності логічних операцій: інверсії, кон'юнкції та диз'юнкції

Інші вісім логічних функцій двох змінних з таблиці 5.2. виражаються через операції інверсії, кон'юнкції та диз'юнкції і разом з ними визначають основні операції алгебри логіки. Деякі операції, що виконуються цими функціями мають свої позначення.

Логічні зміни об'єднані знаками логічних операцій, складають логічні вирази. При обчисленні значення логічного виразу визначено такий порядок виконання логічних операцій: спочатку виконується інверсія, потім кон'юнкція, і в останню чергу диз'юнкція. Для зміни вказаного порядку використовують дужки. Наприклад значення логічної функції трьох змінних $f(x_1, x_2, x_3) = (x_1 \cdot x_2 + \overline{x_1 \cdot x_2}) \cdot x_1 + \overline{x_3}$ при наборі змінних (0, 1, 1) буде хибність, а при наборі (1, 0, 1) – істина.

5.2 Мінімізація логічних функцій

Використовуючи аналітичні форми подання, зазвичай прагнуть до спрощення логічних функцій, виражаючи складні логічні функції через інші функції. При цьому систему логічних функцій називають *функціонально повною*, якщо будь-яку логічну функцію можна подати в аналітичній формі через деякий набір функцій – *базис*.

Найпоширеніший базис – це набір трьох функцій: кон'юнкції (& – І), диз'юнкції (V – АБО) та інверсії (НЕ). Функціонально повними є також системи, що складаються з двох логічних функцій: диз'юнкції (АБО) та інверсії (НЕ), або кон'юнкції (І) та інверсії (НЕ). Функціональну повноту мають системи, що складаються тільки з однієї логічної функції: інверсії кон'юнкції (І-НЕ) чи інверсії диз'юнкції (АБО-НЕ). Існують також інші функціонально повні системи логічних функцій. В алгебрі логіки визначено чимало законів, за допомогою яких можна перетворювати логічні функції:

Аксіоми:

$\overline{0} = 1;\ \overline{1} = 0;$

$x + 0 = x;\ x \cdot 1 = x;$

$x + 1 = 1;\ x \cdot 0 = 0;$

$x + x = x;\ x \cdot x = x;$

$x \cdot \overline{x} = 0;\ x + \overline{x} = 1;$

$\overline{\overline{x}} = x;$

Комутативний (переміщуваний) закон:

$x_1 \cdot x_2 = x_2 \cdot x_1;$

$x_1 + x_2 = x_2 + x_1;$

Асоціативний (сполучний) закон:

$(x_1 \cdot x_2) \cdot x_3 = (x_1 \cdot x_3) \cdot x_2 = x_1 \cdot (x_2 \cdot x_3);$

$(x_1 + x_2) + x_3 = x_1 + (x_2 + x_3)$

Ці закони ідентичні аналогічним законам звичайної алгебри.

Дистрибутивний (розподільний) закон:

$x_1 \cdot (x_2 + x_3) = (x_1 \cdot x_2) + (x_1 \cdot x_3);$

$x_1 + (x_2 \cdot x_3) = (x_1 + x_2) \cdot (x_1 + x_3);$

Закон поглинання. У диз'юнктивній формі логічної функції кон'юнкція меншого рангу, тобто з меншим числом змінних, поглинає всі кон'юнкції більшого рангу, якщо в них містяться їх зображення. Це справедливо і для кон'юнктивних форм:

$x_1 + (x_1 \cdot x_2) = x_1;$

$x_1 \cdot (x_1 + x_2) = x_1;$

Закон склеювання:

$x_1 \cdot x_2 + x_1 \cdot x_2 = x_1 \cdot x_2;$

$(x_1 + x_2) \cdot (x_1 + x_2) = x_1 \cdot x_2;$

Правило де Моргана:

$\overline{x_1 + x_2} = \overline{x_1} \cdot \overline{x_2};$

$\overline{x}_1 \cdot x_2 = \overline{x}_1 + x_2.$

Для перевірки наведених залежностей можна використовувати або аналітичні перетворення виразів, або побудувати таблиці істинності для логічних функцій, що знаходяться в лівій і правій частинах.

Проектування пристроїв комп'ютера передбачає виконання таких етапів:

- визначення таблиці істинності функції для розв'язуваної цим пристроєм задачі;
- за таблицями істинності формування функцій, виражених в аналітичному вигляді в диз'юнктивній чи кон'юнктивній досконалій нормальній формі (д.д.н.ф чи к.д.н.ф.);
- мінімізація логічної залежності за допомогою наведених законів алгебри логіки;
- подання отриманих виразів в обраному базисі;
- побудова функціональної схеми пристрою – схеми, що показує зв'язок між різними логічними елементами, позначеними умовними позначками.

У такий спосіб можна побудувати технічний пристрій, що має мінімальні апаратні витрати.

5.3 Геометричні методи мінімізації булевих функцій

Найбільш поширеним є метод, оснований на використанні карт Карно. Карти Карно являють собою графічне подання таблиць істинності булевих функцій. Структури карт Карно для функцій двох, трьох та чотирьох змінних показано на рис. 5.2 – 5.4. Вони являють собою таблиці, які містять по 2^n прямокутні комірки, де n – кількість логічних змінних. Карта розмічається системою координат, яка відповідає значенням вхідних змінних. Комбінація цифр, якими позначається кожний стовпець, показує, для яких значень змінних обчислюється функція, що розміщується у клітинках цього стовпця. Якщо на наборі $(\alpha_1, ..., \alpha_n)$ значення функції дорівнює „одиниці", то її д.д.н.ф. містить елементарну кон'юнкцію $x_1^{\sigma 1} .. x_n^{\sigma n}$, яка набуває на цьому наборі значення 1. Тому комірки карти Карно, що подають функцію, містять стільки 1, скільки елементарних кон'юнкцій міститься в її д.д.н.ф., причому, кожній 1 відповідає одна з елементарних кон'юнкцій.

Координати рядків та стовпців у карті Карно слідують не у природному порядку зростання двійкових кодів, а у порядку 00,01,11,10. Модифікацію порядку слідування наборів зроблено для того, щоб сусідні набори (тобто набори, що відрізняються один від одного тільки цифрою будь-якого одного розряду), були сусідніми у геометричному розумінні.

Процес мінімізації полягає у формуванні прямокутників, які містять по 2^k комірки, що заповнені одиницями (k – ціле число). У прямокутники поєднуються сусідні комірки, що відповідають "сусіднім" елементарним кон'юнкціям. Сукупність прямокутників, що покривають усі „одиниці", називається покриттям. Одна і та сама комірка може бути покритою двома чи кількома прямокутниками. Можна зробити такі висновки.

1. Формула, яку отримано в результаті мінімізації булевої функції за допомогою карт Карно, містить стільки елементарних кон'юнкцій, скільки прямокутників є у покритті.

2. Чим більше комірок у прямокутнику, тем менше змінних у відповідній їй елементарній кон'юнкції.

На рис. 5.5, а прямокутнику, який містить чотири комірки, відповідає кон'юнкція $\bar{x}_3 x_4$ двох змінних, а квадрату, що складається з однієї комірки, – кон'юнкція $\bar{x}_1 x_2 x_3 \bar{x}_4$. Функція, що відповідає покриттю, має вигляд: $f(x_1, x_2, x_3, x_4) = \bar{x}_3 x_4 + \bar{x}_1 x_2 x_3 \bar{x}_4$

x_1 \ x_2	0	1
0	f(0,0)	f(0,1)
1	f(1,0)	f(1,1)

а)

x_1	x_2	$f(x_1,x_2)$
0	0	f(0,0)
0	1	f(0,1)
1	0	f(1,0)
1	1	f(1,1)

б)

Рисунок 5.2 – а) карта Карно для функції двох змінних; б) таблиця істинності для функції двох змінних

x_1 \ x_3 x_2	00	01	11	10
0	f (0,0,0)	f (0,0,1)	f(0,1,1)	f (0,1,0)
1	f (1,0,0)	f(1,0,1)	f(1,1,1)	f (1,1,0)

а)

x_1	x_2	x_3	$f(x_1,x_2,x_3)$
0	0	0	f (0,0,0)
0	0	1	f(0,0,1)
0	1	0	f (0,1,0)
0	1	1	f (0,1,1)
1	0	0	f (1,0,0)
1	0	1	f (1,0,1)
1	1	0	f (1,1,0)
1	1	1	f (1,1,1)

б)

Рисунок 5.3 – а) Карта Карно для функції трьох змінних; б) таблиця істинності для функції трьох змінних

		$\bar{x}_3$		x_3	
		$\bar{x}_4$	x_4		$\bar{x}_4$
		00	01	11	10
$\bar{x}_1$	$\bar{x}_2$	00	f (0,0,0,0)	f (0,0,0,1)	f (0,0,1,1)	f (0,0,1,0)
	x_2	01	f (0,1,0,0)	f (0,1,0,1)	f (0,1,1,1)	f (0,1,1,0)
x_1		11	f (1,1,0,0)	f (1,1,0,1)	f(1,1,1,1)	f(1,1,1,0)
	$\bar{x}_2$	10	f (1,0,0,0)	f (1,0,0,1)	f (1,0,1,1)	f (1,0,1,0)

Рисунок 5.4 – Карта Карно для функції чотирьох змінних

Незважаючи на те, що карти Карно зображуються на площині, насправді комірки розміщені на поверхні тора. Верхня та нижня межі карти Карно склеюються таким чином, що утворюється поверхня циліндра, а склеювання бічних меж утворює тороїдальну поверхню. Тому комірки на рис. 5.5, б та рис. 5.5, в утворюють прямокутники.

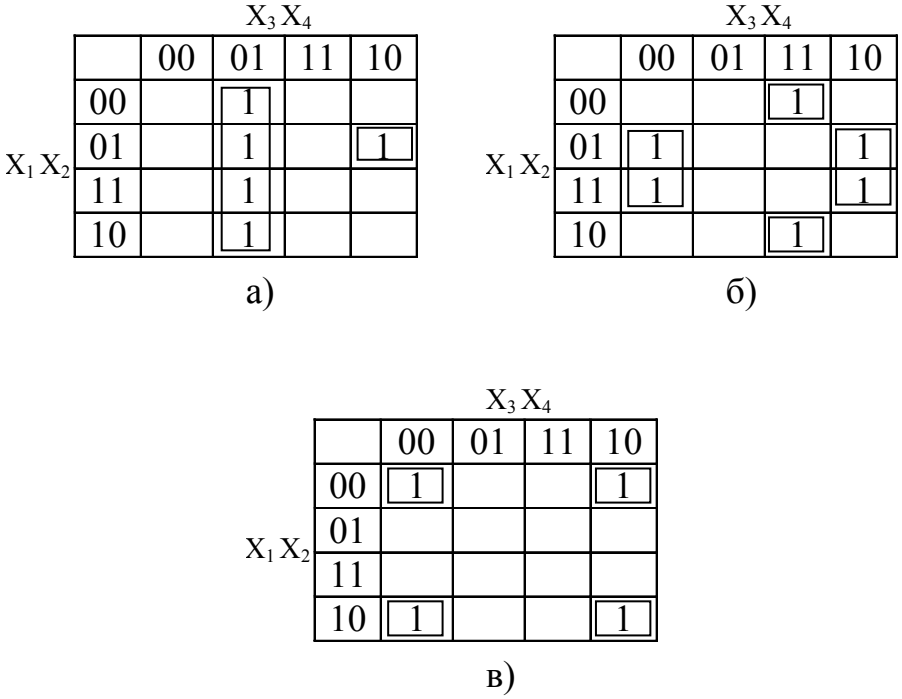


Рисунок 5.5 – Покриття карт Карно

Послідовність дій при мінімізації булевих функцій за методом карт Карно.

1. Зображується таблиця для *n* змінних та виконується розмітка її бічних сторін.

2. Комірки таблиці, що відповідають наборам змінних, на яких значення функції дорівнює 1, заповняються одиницями, інші комірки не заповнюються (або заповнюються нулями).

3. Вибирається найкраще покриття таблиці правильними прямокутниками. Найкращим вважається покриття, яке утворене мінімальною кількістю прямокутників, а якщо таких варіантів кілька, то з них вибирається той, що дає максимальну сумарну площу прямокутників. Якість мінімізації оцінюється функціоналом, який називається коефіцієнтом покриття:

$$k = \frac{m}{s}, \qquad (5.1)$$

де m – загальна кількість прямокутників;

s – сумарна площа прямокутників.

Для рис. 5.5, а k = 2/5, рис. 5.5, б k = 2/6, а для рис. 5.5, в k = 1/4. Кращим вважається те покриття, для якого значення функціоналу k буде меншим.

Приклад 5.1. Мінімізувати булеву функцію:

$f(x_1, x_2, x_3) = x_1x_2x_3 + x_1\bar{x}_2x_3 + \bar{x}_1x_2\bar{x}_3 + \bar{x}_1\bar{x}_2\bar{x}_3$.

Мінімізуємо дану булеву функцію з допомогою карт Карно. Використаємо зображену на рис. 5.3, а карту Карно для трьох змінних. Виконавши перші два пункти зі згаданої вище послідовності дій при мінімізації методом карт Карно, отримаємо таку карту Карно:

		X_2X_3			
		00	01	11	10
X_1	0	1			1
	1		1	1	

Згідно з третім пунктом здійснюємо покриття прямокутниками розглянуту вище карту Карно, отримаємо таку карту:

		X_2X_3			
		00	01	11	10
X_1	0	1			1
	1		1	1	

Оскільки варіант об’єднання в прямокутники на розглянутій карті тільки один, отримуємо таку мінімізовану булеву функцію: $f(x_1, x_2, x_3) = x_1 x_3 + \bar{x}_1 \bar{x}_3$.

Приклад 5.2. Мінімізувати булеву функцію:

$f(x_1, x_2, x_3, x_4) = \bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 + \bar{x}_1 x_2 \bar{x}_3 \bar{x}_4 + x_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 + \bar{x}_1 \bar{x}_2 x_3 x_4 + \bar{x}_1 x_2 x_3 x_4 +$

$+ x_1 x_2 x_3 x_4 + x_1 \bar{x}_2 x_3 x_4 + \bar{x}_1 x_2 x_3 \bar{x}_4$

Для мінімізації використаємо зображену на рис. 5.4 карту Карно для чотирьох змінних. Заповнимо „одиницями” карту Карно згідно з вхідною булевою функцією:

$X_1 X_2$ \ $X_3 X_4$	00	01	11	10
00	1		1	
01	1		1	1
11			1	
10	1		1	

Виконаємо таке покриття карти Карно:

$X_1 X_2$ \ $X_3 X_4$	00	01	11	10
00	1		1	
01	1		1	1
11			1	
10	1		1	

Для такого покриття обчислимо коефіцієнт покриття *k* з формули 5.1 В цьому випадку кількість прямокутників *m=5,* площа прямокутників *s*=8. Таким чином *k*=5/8.

Згідно з цим покриттям отримуємо таку мінімізована булеву функцію:

$f(x_1, x_2, x_3, x_4) = \bar{x}_1 \bar{x}_3 \bar{x}_4 + x_1 x_2 x_3 + \bar{x}_1 x_2 x_3 + x_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 + \bar{x}_1 \bar{x}_2 x_3 x_4$.

Виконаємо ще один варіант покриття карти Карно:

$X_1 X_2$ \ $X_3 X_4$	00	01	11	10
00	1		1	
01	1		1	1
11			1	
10	1		1	

Для такого покриття коефіцієнт покриття k=3/8, оскільки кількість прямокутників m=3, площа прямокутників s=8.

Згідно з цим покриттям отримуємо таку мінімізована булеву функцію: $f(x_1, x_2, x_3, x_4) = \bar{x}_1 x_2 \bar{x}_4 + \bar{x}_2 \bar{x}_3 \bar{x}_4 + x_3 x_4$.

Другий варіант покриття кращий ніж перший, оскільки коефіцієнт покриття у другому випадку менший за перший. Крім того видно, що у другому випадку булева функція більш мінімізована.

КОНТРОЛЬНІ ЗАПИТАННЯ ТА ЗАВДАННЯ

1. *Дайте означення логічної функції.*
2. *Які існують способи задання логічних функцій?*
3. *Що називають функціонально повними системами логічних функцій?*
4. *Що таке інверсія, кон'юнкція та диз'юнкція?*
5. *Наведіть етапи проектування пристроїв комп'ютера.*
6. *Що таке базиси?*
7. *Наведіть закони перетворення логічних функцій?*
8. *Наведіть послідовність дій при мінімізації методом карт Карно.*
9. *Мінімізуйте за допомогою карт Карно булеву функцію*
   $f(x_1, x_2, x_3, x_4) = \bar{x}_1 x_2 \bar{x}_3 \bar{x}_4 + \bar{x}_1 x_2 \bar{x}_3 x_4 + x_1 \bar{x}_2 x_3 \bar{x}_4 + \bar{x}_1 \bar{x}_2 x_3 x_4 + \bar{x}_1 x_2 x_3 x_4 +$
   $+ x_1 x_2 x_3 x_4 + \bar{x}_1 \bar{x}_2 x_3 \bar{x}_4 + x_1 x_2 x_3 \bar{x}_4$
10. *Мінімізуйте за допомогою карт Карно булеву функцію*
    $f(x_1, x_2, x_3) = \bar{x}_1 x_2 x_3 + \bar{x}_1 \bar{x}_2 x_3 + \bar{x}_1 x_2 \bar{x}_3 + x_1 \bar{x}_2 \bar{x}_3$
11. *Мінімізуйте за допомогою карт Карно булеву функцію*
    $f(x_1, x_2, x_3) = x_1 x_2 x_3 + \bar{x}_1 x_2 x_3 + x_1 x_2 \bar{x}_3 + \bar{x}_1 \bar{x}_2 \bar{x}_3$
12. *Мінімізуйте за допомогою карт Карно булеву функцію*
    $f(x_1, x_2, x_3, x_4) = x_1 x_2 \bar{x}_3 \bar{x}_4 + \bar{x}_1 x_2 x_3 x_4 + x_1 \bar{x}_2 x_3 \bar{x}_4 + \bar{x}_1 \bar{x}_2 x_3 x_4 + \bar{x}_1 x_2 x_3 x_4 +$
    $+ x_1 x_2 x_3 x_4 + x_1 \bar{x}_2 x_3 \bar{x}_4 + x_1 x_2 x_3 \bar{x}_4$
13. *Мінімізуйте за допомогою карт Карно булеву функцію*
    $f(x_1, x_2, x_3) = \bar{x}_1 x_2 \bar{x}_3 + \bar{x}_1 x_2 x_3 + \bar{x}_1 x_2 \bar{x}_3 + x_1 \bar{x}_2 \bar{x}_3$
14. *Мінімізуйте за допомогою карт Карно булеву функцію*
    $f(x_1, x_2, x_3) = x_1 \bar{x}_2 x_3 + \bar{x}_1 x_2 x_3 + x_1 x_2 \bar{x}_3 + \bar{x}_1 x_2 \bar{x}_3$
15. *Мінімізуйте за допомогою карт Карно булеву функцію*
    $f(x_1, x_2, x_3) = \bar{x}_1 \bar{x}_2 x_3 + \bar{x}_1 x_2 x_3 + x_1 x_2 \bar{x}_3 + x_1 \bar{x}_2 \bar{x}_3$
16. *Мінімізуйте за допомогою карт Карно булеву функцію*
    $f(x_1, x_2, x_3) = x_1 \bar{x}_2 x_3 + \bar{x}_1 x_2 x_3 + \bar{x}_1 x_2 \bar{x}_3 + \bar{x}_1 \bar{x}_2 \bar{x}_3$

6 ОСНОВИ АЛГОРИТМІЗАЦІЇ

Алгоритмом називається чітко і однозначно визначена послідовність дій, необхідна для досягнення поставленої мети.

Використання обчислювальних машин як виконувачів алгоритмів висуває ряд вимог до алгоритмів. Алгоритми мають такі властивості:

- зрозумілість;
- однозначність;
- дискретність;
- масовість;
- скінченність;
- результативність;
- правильність.

Під *зрозумілістю* алгоритмів мають на увазі вказівки, які зрозумілі виконувачам.

Під *однозначністю* алгоритмів розуміють єдиний спосіб трактування правил виконання дій і порядку їх виконання.

Під *дискретністю* алгоритмів розуміють можливість розбиття всієї послідовності дій на окремі елементарні дії, виконання яких людиною, чи машиною не викликає сумнівів.

Під *масовістю* алгоритмів розуміють можливість їх виконання для рішення цілого класу конкретних задач, що відповідають загальній постановці задачі.

Результативність полягає у тому, що при будь-якому наборі вихідних даних алгоритм забезпечує або отримання шуканого результату, або повідомлення про те, що результат не можна отримати.

Існує чотири способи написання алгоритмів:

- вербальний (словесний);
- алгебраїчний (за допомогою літерно-цифрових позначень виконуваних дій);
- графічний;
- з допомогою алгоритмічних мов програмування.

Словесна форма запису алгоритмів використовується в різних інструкціях, призначених для виконання їх людиною.

Алгебраїчна форма найчастіше використовується у теоретичних дослідженнях фундаментальних властивостей алгоритмів.

Графічна форма відповідно до державних стандартів (ГОСТ) на оформлення документації прийнята як основна для опису алгоритмів.

Алгоритм записаний з допомогою алгоритмічної мови програмування називається програмою. Алгоритм у такій формі може бути введений у ЕОМ і після відповідної обробки виконаний з метою отримання шуканого результату.

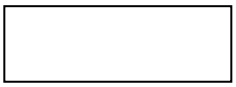
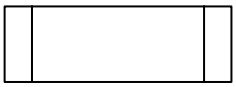
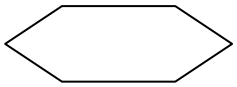
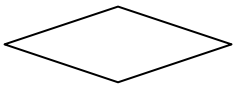
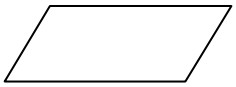
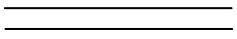
Графічний спосіб подання алгоритмів

Схема – наочне графічне зображення алгоритму, коли окремі його дії (етапи) зображаються за допомогою різних геометричних фігур (блоків), а зв'язки між етапами указуються за допомогою стрілок, що сполучають ці фігури.

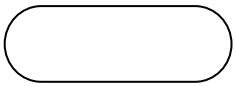
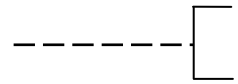
Схеми відображають кроки, які повинні виконуватися комп'ютером, і послідовність їх виконання.

Умовні графічні позначення у схемах алгоритмів наведені у табл. 6.1.

Таблиця 6.1 – Умовні графічні позначення у схемах алгоритмів

Назва символу та зміст позначення	Позначення
1 Процес Виконання операцій в результаті яких змінюються значення	
2 Наперед визначений процес Виконання операцій у відповідності до окремо описаного алгоритму	
3 Модифікація Зміна значення змінної, яка керує виконанням циклічного алгоритму	
4 Розв'язання Перевірка умови та обрання напрямку виконання алгоритму	
5 Введення виведення Введення або виведення за допомогою стандартного пристрою введення-виведення	
6 Паралельні дії Виражає синхронізацію двох чи більше паралельних операцій	
7 Границя циклу Відображення початку та закінчення циклу	 

Продовження таблиці 6.1

Назва символу та зміст позначення	Позначення
8 Лінія Зображує потік даних або управління	
9 Пунктирна лінія Альтернативний зв'язок між символами. Використовують для обведення анотованої частини	
10 З'єднувач Позначення зв'язків між перерваними лініями потоку	
11 Пуск-зупин Початок та кінець алгоритму	
12 Коментар	
13 Пропуск Зображення пропуску символу або групи символів	

Інструкції з виконання будь-яких дій поміщаються всередину прямокутників (табл. 6.1, підпункт 1).

Раніше створені і окремо описані алгоритми і програми (а точніше підпрограми) зображаються у вигляді прямокутника з бічними лініями (табл. 6.1, підпункт 2). Усередині такого "подвійного" прямокутника указуються ім'я алгоритму (підпрограми), параметри, при яких він повинен бути виконаний.

Шестикутник (табл. 6.1, підпункт 3) використовують для зміни параметра змінної, яка керує виконанням циклічного алгоритму, також зазначаються умови завершення циклу.

Блок вибору, що визначає шлях, по якому підуть ці дії (наприклад, обчислення) далі, залежно від результату аналізу даних, зображається у вигляді ромбу (табл. 6.1, підпункт 4). Сама умова записується усередині ромба. Якщо умова, що перевіряється, виконується, тобто має значення "істина", то наступним виконується етап по стрілці "так". Якщо умова не виконується ("хибність"), то здійснюється перехід по стрілці "ні".

Введення початкових даних і виведення результатів зображаються паралелограмом (табл. 6.1, підпункт 5). Усередині нього пишеться слово «введення» або «друк» і перераховуються змінні, що підлягають введенню або виведення.

Виконання декількох операцій одночасно, тобто паралельно, зображується за допомогою двох горизонтальних паралельних відрізків (табл. 6.1, підпункт 6).

Умови для ініціалізації циклу, його приросту, завершення і т.д. розміщуються всередині символу (табл. 6.1, підпункт 7), на початку чи в кінці в залежності від розміщення операції, умови, яка перевіряється.

Для зображення напрямку потоку даних чи управління використовують лінію (табл. 6.1, підпункт 8). За необхідності чи для підвищення зрозумілості алгоритму можуть використовуються стрілки, які вказують напрямок функціонування алгоритму.

Пунктирну лінію (табл. 6.1, підпункт 9) використовують для виділення анотованої частини програми та для зображення альтернативного зв'язку між символами алгоритму.

Стрілками зображаються можливі шляхи алгоритму, а малими кружечками (табл. 6.1, підпункт 10) – розриви цих шляхів (ліній) потоку на сторінці.

Початок і кінець алгоритму зображаються за допомогою овалів (табл. 6.1, підпункт 11). Усередині овалу записується "початок" або Є "кінець".

Коментарі використовуються в тих випадках, коли пояснення не поміщається усередині блока (табл. 6.1, підпункт 12).

Символ (три крапки) (табл. 6.1, підпункт 13) використовують в схемах для відображення пропуску символу або групи символів, для яких не визначені ні тип, ні кількість символів. Його використовують лише в символах лінії чи між ними. Найчастіше він використовується в схемах, що відображують загальний розв'язок з невідомим числом повторень.

Існують такі правила графічного запису алгоритмів:

- блоки алгоритмів з'єднуються лініями потоків інформації;
- лінії потоків не повинні перетинатися;
- будь-який алгоритм може мати лише один блок початку і один блок кінця;
- для полегшення аналізу і опису алгоритмів блоки можуть нумеруватися. Номери проставляються у розриві контуру блока у верхній лівій частині.

Будь-який, навіть найскладніший, алгоритм може бути поданий у вигляді комбінацій кількох елементарних фрагментів, які називають базовими структурами. До них належать:

- послідовне проходження (рис. 6.1);
- розгалуження „якщо-то" (рис. 6.2);
- розгалуження „якщо-то-інакше" (рис. 6.3);
- обирання варіанта за ключем (рис. 6.4);
- цикл з параметром (рис. 6.5);
- цикл з передумовою (рис. 6.6);

- цикл з післяумовою (рис. 6.7).
- використання коментарю (рис. 6.8);
- цикл з передумовою або післяумовою (рис. 6.9).

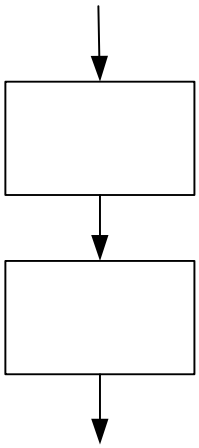


Рисунок 6.1 – Послідовне проходження

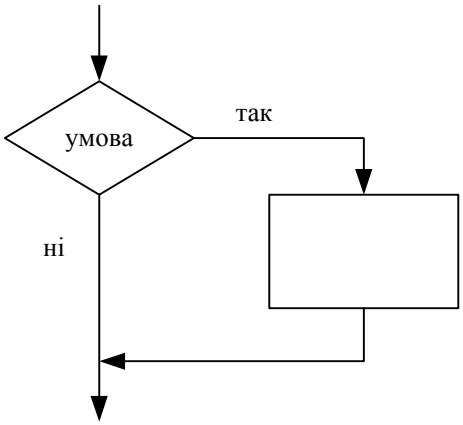


Рисунок 6.2 – Розгалуження „якщо-то”

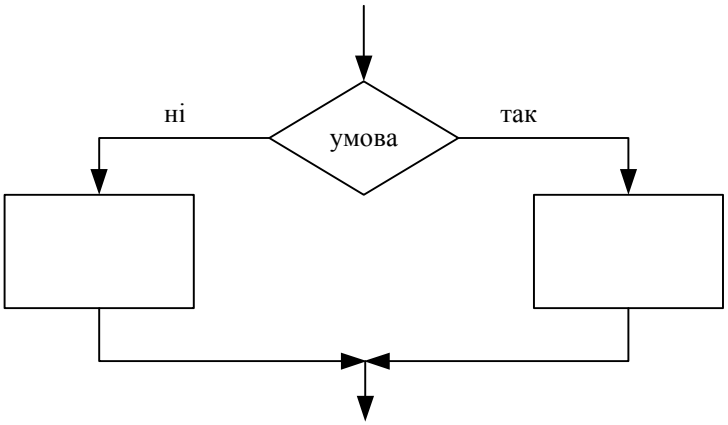


Рисунок 6.3 – Розгалуження „якщо-то-інакше”

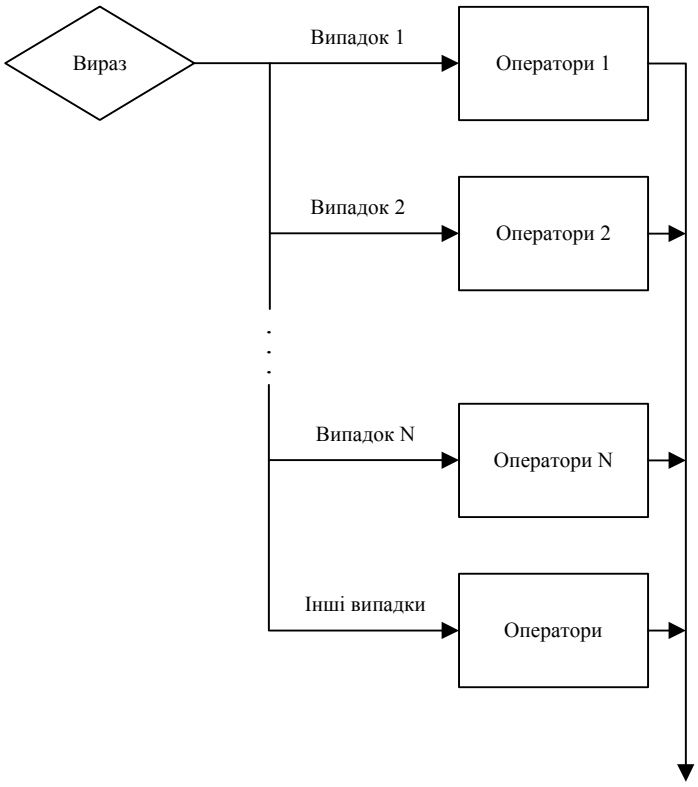


Рисунок 6.4 – Обирання варіанта за ключем

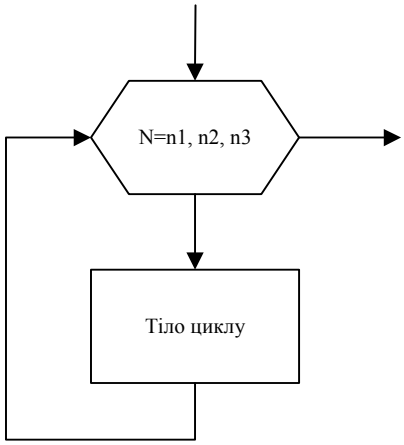


Рисунок 6.5 – Цикл з параметром

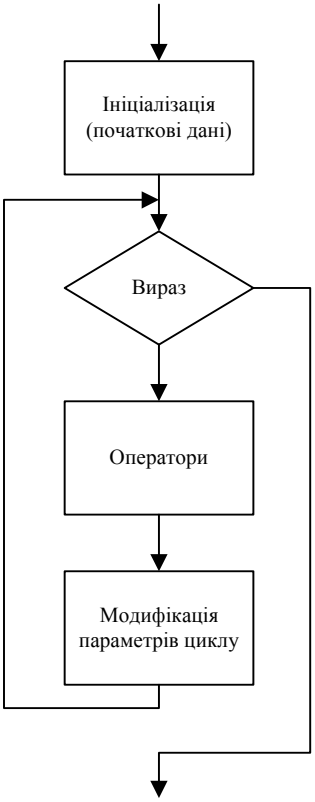


Рисунок 6.6 – Цикл з передумовою

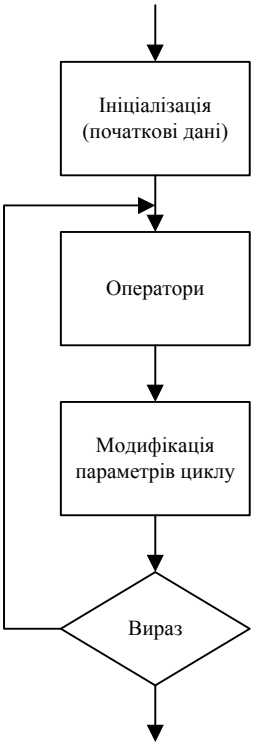


Рисунок 6.7 – Цикл з післяумовою

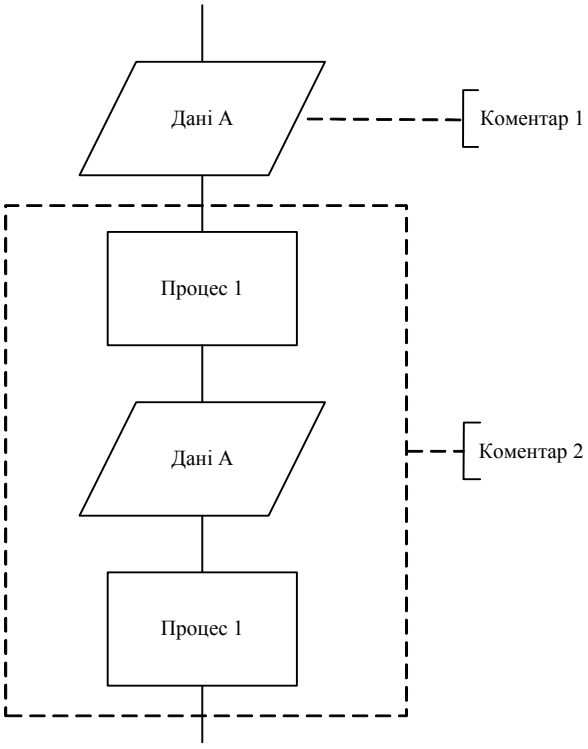


Рисунок 6.8 – Використання коментарю

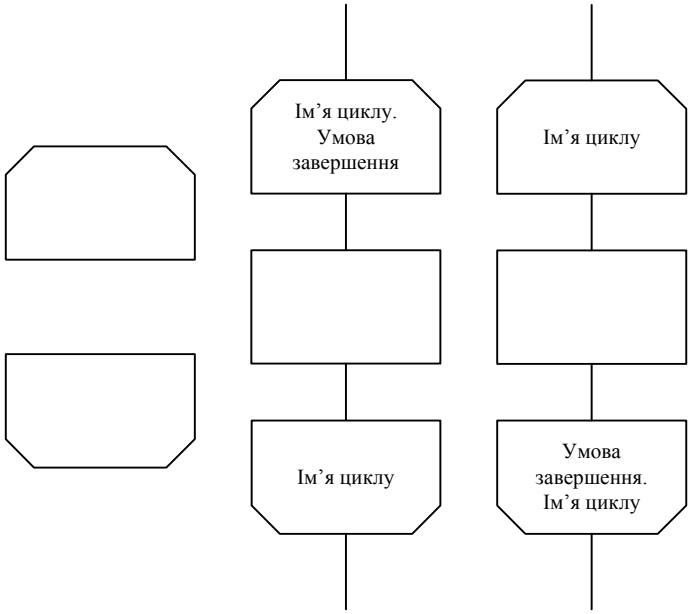


Рисунок 6.9 – Цикл з передумовою або післяумовою

Приклад 6.1. Як приклад опишемо алгоритм природною мовою. Розглянемо алгоритм для знаходження коренів квадратного рівняння.

1. Почати.
2. Введення a, b, c.
3. D присвоїти b^2 - 4ac.
4. Якщо D<0, то йти до 6.
5. Якщо D>0, то йти до 8.
6. Дійсного коріння немає.
7. Перейти до 10.
8. Обчислити: $x_1 = (-b - \sqrt{D})/2 \cdot a$, $x_2 = (-b + \sqrt{D})/2 \cdot a$.
9. Вивести значення x_1 і x_2.
10. Закінчити.

Кожному операторові відповідає стандартний блок з табл. 6.1 (ці умовні оператори замінюються конкретними командами мови програмування, на яку перекладається алгоритм).

1. Оператор 1 – початок роботи;
2. Оператор 2 – введення початкових даних;
3. Оператор 3 – початкове присвоювання;
4. Оператори 4, 5 – перевірка умови;
5. Оператори 6, 8 – дії;
6. Оператор 7 – лінія шляху виконання;
7. Оператор 9 – виведення результатів;
8. Оператор 10 – закінчення роботи.

У остаточному вигляді схема програми зображена на рис. 6.8:

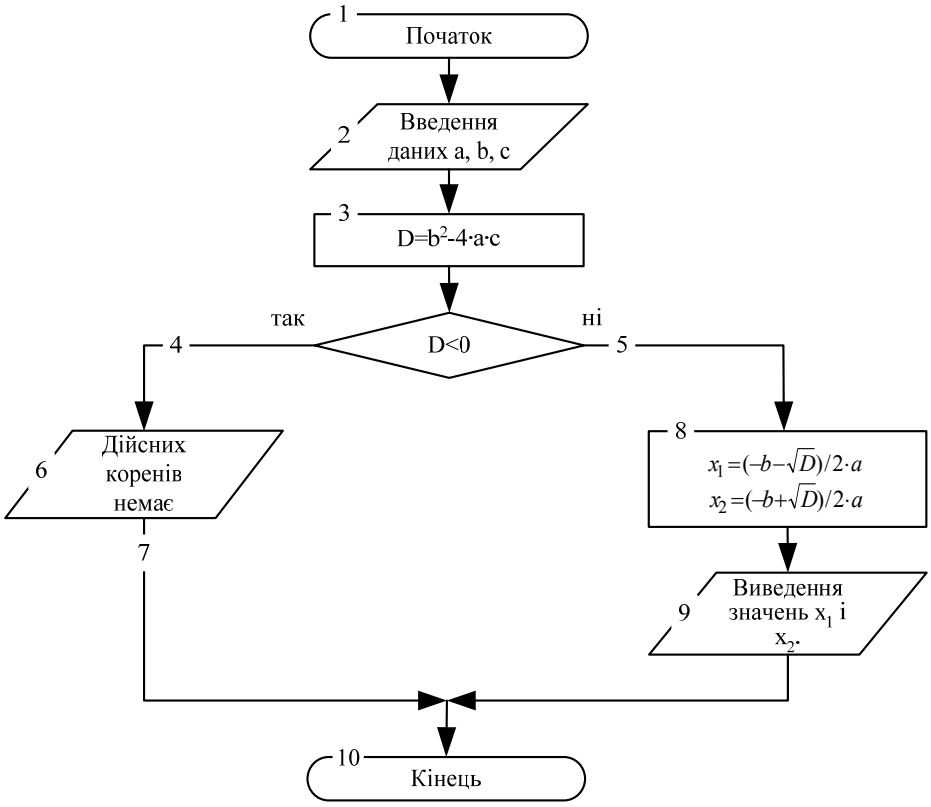


Рисунок 6.10 – Схема програми знаходження коренів квадратного рівняння

Приклад 6.2. Скласти схему програми, яка виводить на екран таблицю значень функції $f(x) = 2x^2 - 5x - 8$ в діапазоні від -4 до 4. Крок зміни аргументу 0.5.

Введемо змінні, які будуть використані у програмі.

Вхідні дані:

a – ліва границя відрізка, змінна дійсного типу;

b – права границя відрізка, змінна дійсного типу;

n – кількість вузлів, в яких обчислюється значення заданої змінної, змінна цілого типу.

Вихідні дані:

x – аргумент функції *f(x)*, змінна дійсного типу;

y – значення функції *f(x)*, змінна дійсного типу;

Проміжні дані:

dx – крок зміни аргументу, змінна дійсного типу;

i – параметр циклу, змінна цілого типу.

Опишемо алгоритм природною мовою.

1. Почнемо.
2. Об’явимо змінні, які будуть використані у програмі.
3. Вводимо з клавіатури значення змінних *a*, *b* та *dx*.
4. Обчислюємо кількість вузлів *n*, в яких будемо обчислювати значення аргументу.

5. Початковому значення аргументу – змінній *x* присвоюємо значення лівої границі *a.*

6. Задаємо початкове значення лічильника циклу *i=1*.

7. Задаємо цикл з умовою *i<=n.*

8. Якщо вираз *i<=n* приймає значення *true* (так) переходимо до 10.

9. Якщо вираз *i<=n* приймає значення *false* (ні) переходимо до 14.

10. Обчислюємо значення функції $f(x) = 2x^2 - 5x - 8$.

11. Виводимо на екран значення змінних *x* та *y.*

12. Збільшуємо на крок *dx* поточне значення аргументу *x.*

13. Збільшуємо значення лічильника циклу *i* на 1.

14. Закінчимо.

Для графічного подання алгоритму використаємо стандартні блоки з табл. 6.1 та базові структури розглянуті на рис. 6.1-6.9. Згідно з розглянутим вище вербальним поданням алгоритму формуємо графічне подання у вигляді схеми програми, яка зображена на рис. 6.11.

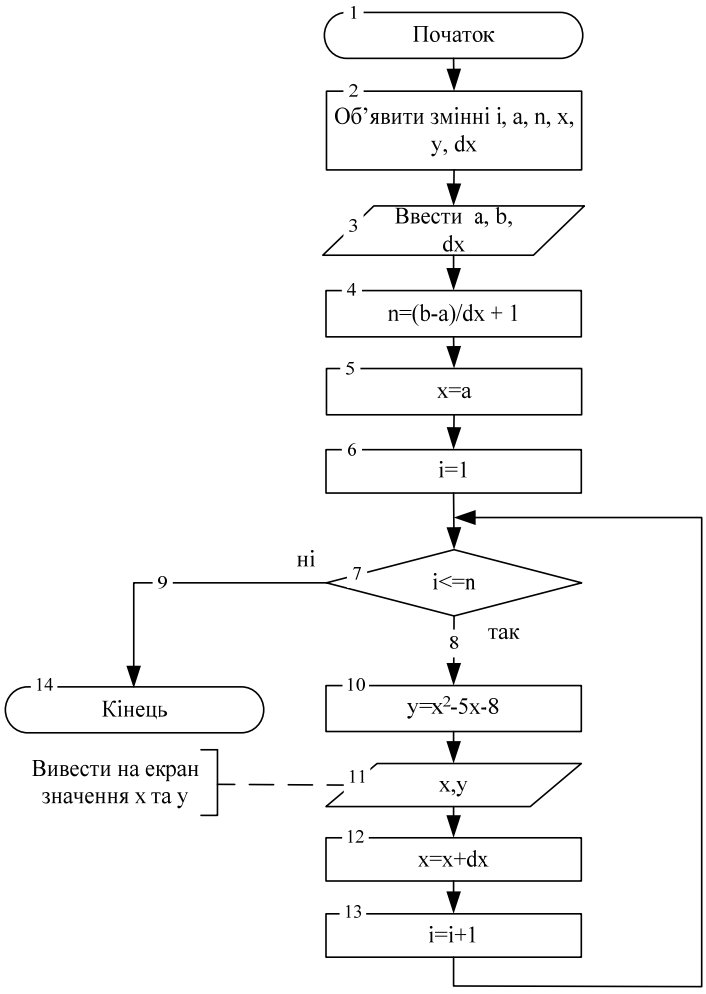


Рис. 6.11 – Схема програми табулювання функції

В схемі програми, зображеної на рис. 6.11, використано декілька базових структур розглянутих на рис. 6.1-6.9. Крім послідовного

проходження (див. рис. 6.1) використаний цикл з передумовою (див. рис 6.6). Крім цього, для блока виведення на екран значень *x* та *y*, показаний приклад застосування коментарю (див. рис. 6.8).

Приклад 6.3. Скласти схему програми, яка обчислює середнє арифметичне послідовності дробових чисел. Після введення останнього числа програма повинна вивести на екран мінімальне та максимальне число послідовності. Кількість чисел послідовності задається під час роботи програми.

Введемо змінні, які будуть використані у програмі.

Вхідні дані:

a – поточне дійсне число послідовності, змінна дійсного типу;

n – кількість дійсних чисел, для яких обчислюється середнє арифметичне, змінна цілого типу.

Вихідні дані:

sred – середнє значення послідовності, змінна дійсного типу;

max – максимальне значення послідовності, змінна дійсного типу;

min – мінімальне значення послідовності, змінна дійсного типу;

Проміжні дані:

sum – сума послідовності чисел;

i – параметр циклу, змінна цілого типу.

Опишемо алгоритм природною мовою.

1. Почнемо.
2. Об’явимо змінні, які будуть використані у програмі.
3. Присвоюємо нуль змінній *sum*.
4. Вводимо з клавіатури кількість чисел послідовності у змінну *n,* а також перше число послідовності у змінну *a*.
5. Ініціалізуємо змінні *min*, *max* та *sum* значенням змінної *a*.
6. Задаємо цикл з умовою *i<n* , початковим значенням *i=1,* та збільшенням лічильника циклу на 1.
7. Якщо вираз *i<n* приймає значення *true* (так) переходимо до 9.
8. Якщо вираз *i<n* приймає значення *false* (ні) переходимо до 19.
9. Вводимо з клавіатури поточне число послідовності у змінну *a.*
10. Додаємо до значення змінної *sum* значення змінної *a.*
11. Вводимо умову *a<min.*
12. Якщо вираз *a<min* приймає значення *true* переходимо до 14.
13. Якщо вираз *a<min* приймає значення *false* переходимо до 15.
14. Присвоюємо змінній *min* значення змінної *a*.
15. Вводимо умову *a>max.*
16. Якщо вираз *a>max* приймає значення *true* переходимо до 18.
17. Якщо вираз *a>max* приймає значення *false* переходимо до 6.
18. Присвоюємо змінній *max* значення змінної *a*.
19. Обчислюємо значення середнього арифметичного *sred*.

20. Виводимо на екран значення *sred.*

21. Закінчимо.

Для графічного подання алгоритму використаємо стандартні блоки з табл. 6.1 та базові структури, розглянуті на рис. 6.1-6.9 . Згідно з розглянутим вище поданням вербального алгоритму формуємо графічне подання у вигляді схеми програми, яка зображена на рис. 6.12.

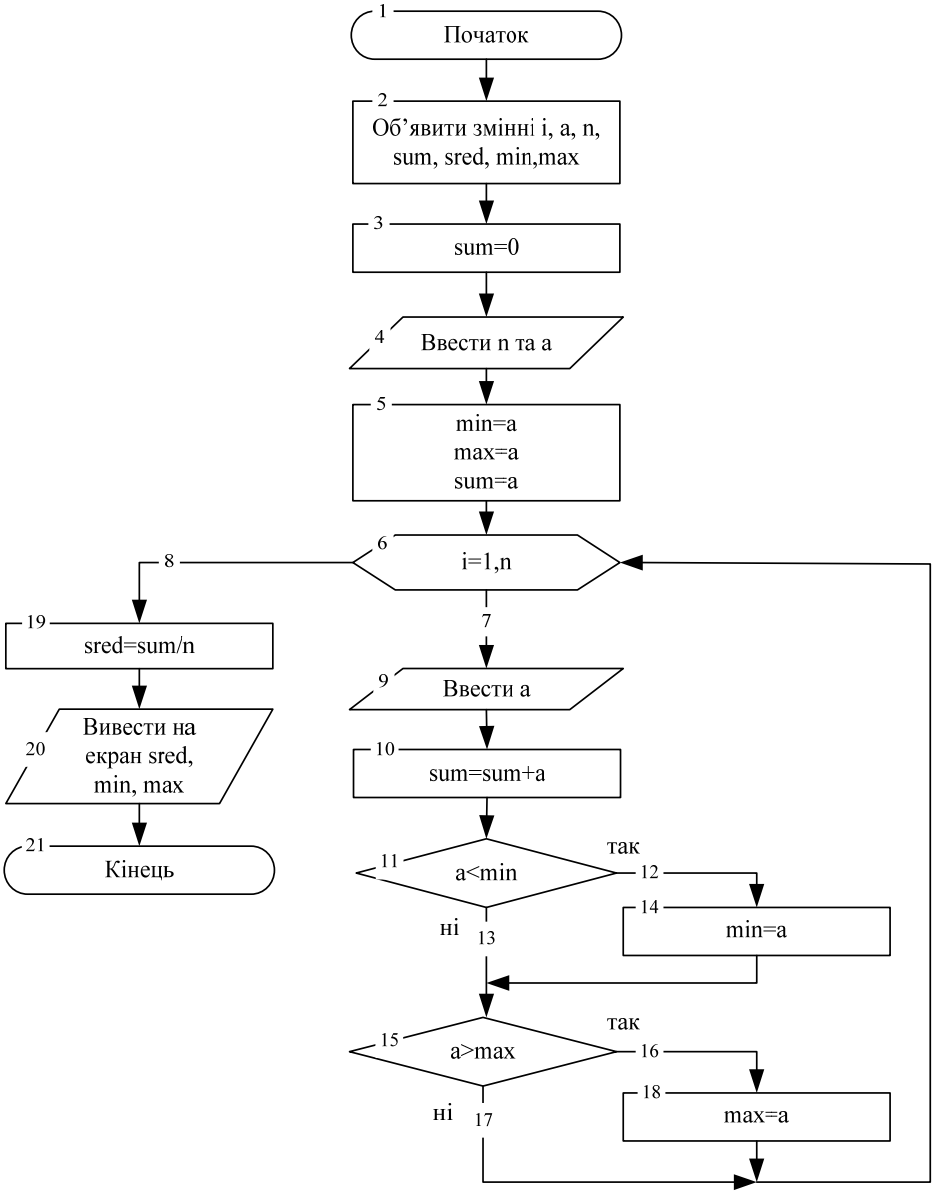


Рисунок 6.12. Схема програми, яка обчислює середнє арифметичне

В схемі програми зображеної на рис. 6.12 використано декілька базових структур розглянутих на рис. 6.1-6.9. Крім послідовного проходження (див. рис. 6.1) використаний цикл з параметром (див. рис 6.5). Крім цього, для знаходження *min* та *max* використано розгалуження „якщо-то" (див. рис. 6.2).

КОНТРОЛЬНІ ЗАПИТАННЯ ТА ЗАВДАННЯ

1. *Що таке алгоритм?*
2. *Наведіть властивості алгоритмів.*
3. *Що розуміють під дискретністю алгоритмів?*
4. *Які існують способи подання алгоритмів?*
5. *Який спосіб подання алгоритмів прийнятий за основний?*
6. *Що таке схема програми?*
7. *Які графічні символи використовуються у схемах алгоритмів?*
8. *Як позначають розриви в схемах алгоритмів?*
9. *Яким чином у схемах алгоритмів зображується символ „модифікація”?*
10. *Яким чином у схемах алгоритмів зображується символ „введення-виведення”?*
11. *Яким чином у схемах алгоритмів зображується символ „процес”?*
12. *Яким чином у схемах алгоритмів зображується символ „розв’язання”?*
13. *Яким чином у схемах алгоритмів зображується символ „коментар”?*
14. *Яким чином у схемах алгоритмів зображується символ „наперед визначенний процес”?*
15. *Яким чином у схемах алгоритмів зображується символ „паралельні дії”?*
16. *Наведіть правила графічного запису алгоритмів.*
17. *Що таке базова структура?*
18. *Як зображується розгалуження „якщо-то” у схемах алгоритмів?*
19. *Як зображуєтсья розгалуження „якщо-то-інакше” у схемах алгоритмів?*
20. *Як зображується вибір за ключем у схемах алгоритмів?*
21. *Як зображується цикл з параметром у схемах алгоритмів?*
22. *Як зображується цикл з передумовою у схемах алгоритмів?*
23. *Як зображується цикл з післяумовою у схемах алгоритмів?*
24. *Наведіть приклад використання коментарю у схемах алгоритмів.*
25. *Зобразіть схему програми знаходження коренів квадратного рівняння.*
26. *Зобразіть схему програми обчислення середнього арифметичного масиву чисел.*
27. *Зобразіть схему програми табулювання функції.*

ЛІТЕРАТУРА

1. Верлань А. Ф., Апатова Н. В. Інформатика: Підручник. – К.: Форум, 2000. – 223с.

2. Власюк В. Х. Основи програмування алгоритмічними мовами: Навчальний посібник / МОН України. – Вінниця: ВНТУ, 2004. – 157с.

3. Глушаков С. В., Коваль А. В., Смирнов С. В. Язык программирования С++. – Харьков: Фолио, 2003. – 500с.

4. Гунжій А. М., Коряк С. Ф., Самсонов В. В., Скляров О. Я. Архітектура, принцип функціонування та керувальні ресурси IBM PC. – Харків: "Компанія СМІТ", 2003. – 512 с.

5. Дибкова Л. М. Інформатика та комп'ютерна техніка: Посібник. для студентів вищих навчальних закладів. – К.: Академія, 2002. – 320с.

6. Інформатика: Комп'ютерна техніка; Комп'ютерні технології: Посібник / За ред. О. І. Пушкаря. – К.: "Академія", 2001. – 696с.

7. Інформатика та комп'ютерна техніка: Навчальний посібник / МОН України; Уклад.: С. І. Перевозніков, Г. О. Лосєв, Г. О. Ваховська, І. Р. Арсенюк. – Вінниця: ВДТУ, 2002. – 102с.

8. Інформатика. Комп'ютерна техніка. Комп'ютерні технології: Підручник. – К.: Каравела, 2004. – 464с.

9. Інформатика і комп'ютерна техніка: Навчальний посібник / За ред. М. Є. Рогози. – К.: ВЦ "Академія", 2006. – 368с.

10. Заєць В. М. Функціональне програмування: Навчальний посібник / МОН України. – Львів: "Бескид Біт", 2003. – 160с.

11. Клименко О. Ф., Головко Н. Р., Шарапов О. Д. Інформатика та комп'ютерна техніка: Навчально-методичний посібник / МОН України. – К.: КНЕУ, 2002. – 534с.

12. Ковалюк Т. В. Основи програмування: Підручник для студентів ВНЗ / За заг. ред. М. З. Згуровського. – К.: ВГ ВНМ, 2005. – 384с.

13. Кравчук С. О., Шонін В. О. Основи комп'ютерної техніки. Компоненти, системи, мережі: Навчальний посібник для студентів ВНЗ. – К.: ІВЦ "Видавництво Політехніка": вид-во "Каравела", – 2005. – 344с.

14. Круподьорова Л. М. Алгоритмізація і основи програмування: Навчальний посібник / МОН України. – Вінниця: ВНТУ, 2005. – 168с.

15. Круподьорова Л. М. Програмування мовою Delphi: Навчальний посібник / МОН України. – Вінниця: ВНТУ, 2005. – 152с.

16. Литвин І. І., Конончук О. М., Дещинський Ю. Л. Інформатика: теоретичні основи і практикум: Підручник. – Львів: "Новий світ-2000", 2004. – 300с.

17. Лужецький В. А., Северин Л. І., Каплун В. А. Основи інформатики та інформаційних технологій: Навчальний посібник / МОН України. – Вінниця: ВНТУ, 2004. – 70с.

18. Макарова М. В., Карнаухова Г. В., Запара С. В. Інформатика та комп'ютерна техніка: Навчальний посібник для студ. ВНЗ / За заг. ред. М. В. Макарової. – 2-ге вид., стер. – Суми: "Університетська книга", 2005. – 642с.

19. Перевозніков С. І., Сілагін О. В., Лосєв Г. О. Системне програмування і операційні системи: Навчальний посібник / МОН України. – Вінниця: ВДТУ, 2002. – 77с.

20. Погорілий С. Д. Основоположні математичні відомості. Програмне забезпечення: Навчальний посібник. – К.: "Київський університет", 2002. – 288с.

21. Поплавський А. В., Коренний А. А., Коренна Т. М. Інформатика та комп'ютерна техніка: Практикум / МОН України. ч.2. – Вінниця: ВНТУ, 2005. – 110с.

22. Інформатика: Комп'ютерна техніка; Комп'ютерні технології: Посібник / За ред. О. І. Пушкаря. – К.: Академія, 2001. – 696с.

23. Рогоза М. Є., Клименко В. І. ХР: Windows, Word, Exel для самостійного вивчення: Навчальний посібник / МОН України. – К: Центр навчальні літератури, 2003. – 294с.

24. Руденко В. Д., Макарчук О. М., Патланжоглу М. О. Практичний курс інформатики: Навчальний посібник / За ред. В. М. Мадзігона. – К.: Фенікс, 1997. – 304с.

25. Роберт У. Себеста Основные концепции языков программирования, 5-е изд.: Пер. с англ. – М.: Издательский дом «Вильямс», 2001. – 672с.

26. Семеренко В. П. Програмування мовами С та С++ в середовищі Windows: Навчальний посібник / МОН України. – Вінниця: ВДТУ, 2002. – 128 с.

27. Симонович С. В. Информатика: Базовый курс: Учебное пособие. – СПб: Питер, 2003. – 640с.

28. Системне програмування мовою Асемблер: Лабораторний практикум / МОН України. ч.1, Уклад.: В. П. Семеренко. – Вінниця: ВНТУ, 2003. – 71с.

29. Уейк М., Прата С., Мартин Д. Язык Си. Руководство для начинающих: Пер. с англ. – М.: Мир, 1988. – 512с.

30. Шишкова О. М. Основи програмування мовою Паскаль у прикладах і завданнях: Навчальний посібник / МАУП. – К.: МАУП, 2004. – 112с.

31. Щедріна О. І. Алгоритмізація та програмування процедур обробки інформації: Навчальний посібник / МОН України. – К.: КНЕУ, 2001. – 240с.

32. Ярмуш О. В., Редько М. М. Інформатика і комп'ютерна техніка: Навчальний посібник. – К.: Вища освіта, 2006. – 359с.

СЛОВНИК НАЙБІЛЬШ ВЖИВАНИХ ТЕРМІНІВ

Алгоритм – algorithm
Алгоритмічна мова – algorithmic language
Аналогові дані – analog data
Аналогово-цифрове перетворення – analog-to-digital transformation
Апаратне забезпечення – hardware
Аргумент функції – argument of a function
Арифметична операція – arithmetic operations
Багатопроцесорний комп'ютер – multiprocessor computer
Базис – basis
Базова система введення-виведення – Basic Input/Output System
Батьківський клас – base class
Блок електроживлення – power supply
Булева алгебра – boolean algebra
Відеокарта – videocard
Відеоконтролер – videocontroller
Вісімкова система числення – octal numeral system
Графіка – graphics
Графічна станція – graphic station
Графічний інтерфейс – graphical interface
Графічні редактори – graphics editors
Дані – data
Двійкова система числення – binary numeral system
Двійковий файл – binary file
Деструктор – destructor
Десяткова система числення – decimal numeral system
Дзеркальний проектор – overhead projector
Диз'юнкція – disjunction, logical addition
Дискета – diskette
Дисковод – disk drive
Дискретні дані – digital data
Драйвер – driver
Електронна таблиця – spreadsheet
Звукова карта – audio card
Зображення – image
Інкапсуляція – encapsulation
Інтерпретатор – language processor
Інтерфейс – interface
Інформатика – computer science
Ірраціональне число – irrational number
Істина – true
Каталог – directory

Кишенькові ПК – palmtop
Клавіатура – keyboard
Клас – class
Кодування – coding
Кодування символів – symbolic coding
Комп'ютер – computer
Компілятор – compiler
Компоненти – components
Кон'юкція – conjunction, logical multiplication
Конструктор – constructor
Контролер – controller
Логічна змінна – logical value
Логічна операція – logical operation
Логічна функція – boolean functions
Маніпулятор – pointing device
Материнська плата – motherboard
Машинна мова – machine language
Машинний код – machine code
Мережа – network
Мережевий сервер – network server
Миша – mouse
Мікропроцесор – microprocessor
Мікросхема – microcircuit
Мікрофон – microphone
Мінімізація логічних функцій – minimization of boolean functions
Множина значень функції – ranging check a function
Модем – modem
Монітор – monitor
Набір мікросхем – chipset
Налагоджування – debugging
Настільні ПК – desktop
Низхідне програмування – top down programming
Об'єкт класу – class object
Об'єктно-орієнтоване програмування – object-oriented programming
Область визначення функції – domain of function
Обчислювальна система – computer system
Оператор – operator
Операційна система – operating system
Операція – operation
Пам'ять – storage
Паралельний порт – parallel
Периферійні пристрої – peripheral devices
Персональний комп'ютер – personal computer

Плотер – plotter
Поліморфізм – polymorphism
Порт – port
Портативні ПК – notebook
Послідовний порт – serial
Похідний клас – derived class
Прикладне програмне забезпечення – application software
Принтер – printer
Пристрій управління – control unit
Пристрої введення/виведення – input/output devices
Програма – program
Програмне забезпечення – software
Проектор – projector
Пропускна здатність – bandwidth
Протокол – protocol
Процесор – processor
Раціональне число – rational number
Робоча станція – Workstation PC
Роз'єм – connector
Розряд – digit
Семантику – semantics
Символ – symbol
Синтаксис – syntax
Система числення – numeral system
Системне програмування – system programming
Системні ресурси – system resources
Сканер – scanner
Структурне програмування – structured programming
Тактова частота – clock rate
Текстовий редактор – text editor
Текстовий файл – text file
Тестування – testing
Технологія програмування – software ingeneering
Транзистор – transistor
Транслятор – translator
Успадкування – inheritance
Утиліти – utilities
Файловий сервер – file server
Функція – function
Хибність – false
Цифра – digit
Цифро-аналогове перетворення – digital-to-analog transformation
Шістнадцяткова система числення – hexadecimal numeral system

Навчальне видання

Сергій Михайлович Довгалець
Роман Васильович Маслій

Алгоритмічні мови та програмування. Частина 1. Основи інформатики та комп'ютерної техніки

Навчальний посібник

Оригінал-макет підготовлено Маслієм Р. В.

Редактор В. О. Дружиніна

Коректор З. В. Поліщук

Науково-методичний відділ ВНТУ
Свідоцтво Держкомінформу України
серія ДК № 746 від 25.12.2001
21021, м. Вінниця, Хмельницьке шосе, 95, ВНТУ

Підписано до друку
Формат 29,7×42 1/4
Друк різографічний
Тираж прим.
Зам. №

Гарнітура Times New Roman
Папір офсетний
Ум. друк. арк.

Віддруковано в комп'ютерному інформаційно-видавничому центрі
Вінницького національного технічного університету
Свідоцтво Держкомінформу України
серія ДК № 746 від 25.12.2001
21021, м. Вінниця, Хмельницьке шосе, 95, ВНТУ