МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ПОЛІТЕХНІКА»
Кафедра теплових електричних станцій та енергозберігаючих технологій

НАВЧАЛЬНИЙ ПОСІБНИК

З ДИСЦИПЛІНИ

«ОСНОВИ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ПРОГРАМУВАННЯ»

Для здобувачів інституту енергетики та комп´ютерно - інтегрованих систем управління

Перший (освітньо-науковий) рівень вищої освіти
Спеціальність – 144 ТЕПЛОЕНЕРГЕТИКА
Освітня програма – *Теплоенергетика*
Рік впровадження ОПП -2020

Затверджено
на засіданні кафедри ТЕСЕТ
Протокол № 10 від 22.03.2022р

ОДЕСА 2022

Навчальний посібник з дисципліни «Основи інформаційних технологій та программування» Державного університету «Одеська політехніка» за освітньою програмою 144- Теплоенергетика. Укладачі: Губар Л.Б., доц. Лужанська Г.В. - Одеса , ДУОП, 2022. – с.278.

Рецензент: Г.А.Баласанян, д.т.н., професор, рекомендовано Витягом з протоколу №10 від 22.03.22р. засідання кафедри теплових електричних станцій та енергозберігаючих технологій, як навчальний посібник для здобувачів вищих навчальних закладів, які навчаються за освітньо-професійною програмою бакалавра із спеціальності 144- «Теплоенергетика».

Навчальний посібник містить теоретичні та практичні аспекти для опанування дисциплін «Основи інформаційних технологій та програмування». На прикладі середовища розробки Microsoft QBasic представлені прийоми візуального створення програмних продуктів, розглянуті базові алгоритми та структури даних. Особлива увага приділена алгоритмам, що використовуються у математичному моделюванні – сортування, пошуку, обробки динамічних структур даних.

Посібник призначений для здобувачів, що навчаються за спеціальністю 144 - «Теплоенергетика» та вивчають сучасні інформаційні технології.

ПЕРЕДМОВА

Навчальний посібник призначений для вивчення дисципліни «Основи інформаційних технологій та програмування», яка є основною для формування практичних знань і навичок здобувачів зі спеціальності 144-«Теплоенергетика ». Даний курс спирається на отримані теоретичні знання здобувачами з дисципліни «Основи інформаційних технологій та програмування» і призначений для формування у здобувачів бакалавру спеціальності 144 - «Теплоенергетика» науково обґрунтованого підходу до алгоритмізації математичних задач та розробки програмних продуктів з використанням сучасних технологій.

Коло питань, що розглядаються в навчальному посібнику, включає в себе основні теми, які полягають в розкритті понять про організацію і принципи функціонування комп'ютерних систем, алгоритмічні мови та системи програмування, основні принципи розробки алгоритмів і програм, технології створення програмних продуктів з використанням спеціалізованих середовищ візуального програмування. У навчальному посібнику розглянуто мову програмування QBasic, наведено приклади реалізації базових алгоритмів і структур даних. Виділене коло питань визначає структуру посібника, який складається з восьми розділів.

У першому розділі розглядаються особливості побудови і функціонування сучасних комп'ютерних систем, які визначають основні апаратні і програмні компоненти, а також наведені особливості операційних систем сучасних персональни комп'ютерів.

Другий розділ містить відомості про розробку користувацького інтерфейсу. Розглянуто переваги і недоліки усіх

типів користувацьких інтерфейсів, їх основні елементи.

Третій розділ присвячений алгоритмічним мовам та системам програмування. Розглянуто способи перекладу програм з мови програмування на мову мікропроцесору, засоби створення програм, розглянуто особливості використання інтегрованих середовищ розробки програмних засобів, середовищ бистрого проектування, надано відомості про основні системи програмуваннярозгладаються етапи рішення задач на ЕОМ, форми запису алгоритмів, мова блок-схем та особливості структурного підходу до розробки програм.

Четвертий розділ присвячений вивченню арифметичних основам компютерамвластивостей алгоритмів, системам числення використовують фахівці спілкування з комп'ютером.

П'ятий розділ присвячений основним принципам розробки алгоритмів програм, єтапам рішення завданнь на ЕОМ, вибіру методу рішення, розробки алгоритму, складання програми та її налагодження., вивченню властивостям алгоритмів та її ріновидам.

Шостий розділ присвячений вивченню програмуванню мовою QBasic , запуск програми, головного меню, формальні відомості, алфавіту та інформації у мови QBasic.

У сьомому розділі наведено основні команди програмі QBasic та правила наіменування,оператори. які використовуются у програмі QBasic та принцип їх роботи.

Восьмий розділ присвячений графічній обробки даних у програмі QBasic за допомогою операторів, виведення тексту у графічному режимі, збереження графічної інформації та вивод її на екран за допомогою графічних операторів.

Дев' ятий розділ присвячений вивченню операційної системи Windows.

Десятий розділ присвячений освоенню лекційного матеріалу та проведенню здобувачами бакалавру практичних робіт з

дисципліни «Основи інформаційних технологій і програмування» за освітньою програмою 144-Теплоенергетика.

Розділ І

ОРГАНІЗАЦІЯ ФУНКЦІЮВАННЯ КОМП'ЮТЕРНИХ СИСТЕМ

Компоненти комп'ютерної система

Комп'ютерна система складається з наступних компонентів:

1. **Апаратура (hardware)** комп'ютера, основні частини якої – **центральний процесор (Central Processor Unit - CPU**), що виконує **команди (інструкції)** комп'ютера; **пам'ять (memory), що** зберігають дані й програми, і **пристрої введення- виведення, або зовнішні пристрої (input-output devices, I/O devices**), що забезпечують введення інформації в комп'ютер і виведення результатів роботи програм у формі, яка сприймається користувачем-людиною або іншими програмами.
2. **Операційна система (operating system)** – системне програмне забезпечення, що управляє використанням апаратури комп'ютера різними програмами й користувачами.
3. **Прикладне програмне забезпечення (applications software)** – програми, призначені для рішення різних класів завдань. До них відносяться, зокрема, **компілятори**, що забезпечують трансляцію програм з мов програмування, наприклад, C++, у машинний **код** (команди); **системи управління базами даних (СУБД**); **графічні бібліотеки**, **ігрові програми**, **офісні програми**. Прикладне програмне забезпечення утворює наступний, більш

високий рівень, у порівнянні з операційною системою, і дозволяє вирішувати на комп'ютері різні прикладні й повсякденні завдання.

4. **Користувачі (users)** – люди й інші комп'ютери.
5. Користувач фактично стає частиною обчислювальної системи в процесі своєї роботи на комп'ютері, тому що повинен підкорятися певним строгим правилам, порушення яких приведе до помилок або неможливості використання комп'ютера, і виконувати великий обсяг типових рутинних дій – майже як сам комп'ютер. Одна з важливих функцій ОС саме й полягає в тому, щоб позбавити користувача від більшої частини такої рутинної роботи (наприклад, резервного копіювання файлів) і дозволити йому зосередитися на роботі творчій. Інші комп'ютери в мережі також можуть відігравати роль користувачів (**клієнтів**) стосовно комп'ютера, який виступає в ролі **сервера**, що, наприклад, використовується для зберігання файлів або виконання великих програм.

Функціонування комп'ютерної системи

Користувачам комп'ютера доступні системні й прикладні програми (наприклад, компілятори, текстові редактори, системи керування базами даних). Ці програми взаємодіють із операційною системою, що, у свою чергу, управляє роботою комп'ютера.

Класифікація комп'ютерних систем.

ОС – від **суперкомп'ютерів до мобільних пристроїв, - і підсумуємо вимоги до ОС для цих класів комп'ютерів.**

Суперкомп'ютери (super-computers) – потужні

багатопроцесорні комп'ютери, найбільш сучасні, які мають продуктивність до декількох **petaflops** (1015 дійсних операцій у секунду; абревіатура **flops** розшифровується як **floating-point operations per second**). Суперкомп'ютери використовуються для обчислень, що вимагають великих обчислювальних потужностей, надвисокої продуктивності й великого обсягу пам'яті. Особливістю суперкомп'ютерів є їхня паралельна архітектура - як правило, всі вони є багатопроцесорні.

Багатоцільові комп'ютери, або **комп'ютери загального призначення (mainframes)** – традиційна історична назва для комп'ютерів, розповсюджених в 1950-х – 1970-х рр., ще до епохи загального поширення персональних комп'ютерів. Саме для mainframe-комп'ютерів створювалися перші ОС. Типові приклади таких комп'ютерів: IBM 360/370; з вітчизняних – М-220, БЭСМ-6. На таких комп'ютерах вирішувалися всі необхідні завдання – від розрахунку зарплати співробітників в організації до розрахунку траєкторій космічних ракет. Подібний комп'ютер виглядав досить незграбно й громіздко й міг займати цілий великий зал. Параметри ранніх mainframe-комп'ютерів були досить скромними: швидкодія - кілька тисяч операцій у секунду, оперативна пам'ять – кілька тисяч комірок (слів). Недостатньо зручним був користувальницький інтерфейс (інтерактивна взаємодія з комп'ютерами була реалізована набагато пізніше, в 1960-х рр.). Проте, на таких комп'ютерах вирішувалися досить серйозні завдання оборонного й космічного призначення. З появою персональних і портативних комп'ютерів класичні mainframe-комп'ютери відійшли в минуле. Однак варто підкреслити, що в саме в операційних системах для mainframe-комп'ютерів були реалізовані всі основні методи й алгоритми, що розглянуті в даному посібнику, і які згодом були використані в ОС для персональних, кишенькових комп'ютерів і **мобільних**

пристроїв.

Кластери комп'ютерів (computer clusters) – групи комп'ютерів, фізично розташовані поруч і з'єднані один з одним високошвидкісними шинами й лініями зв'язку. Кластери комп'ютерів використовуються для високопродуктивних паралельних обчислень. Найбільш відомі у світі комп'ютерні кластери, розташовані в дослідницькому центрі CERN (Швейцарія) - тому самому, де перебуває великий адронний коллайдер. Як правило, комп'ютерні кластери розташовуються в дослідницьких інститутах і в університетах. Операційна система для кластерів повинна, крім загальних можливостей, надавати засоби для конфігурування кластера, керування комп'ютерами (процесорами), що його складають, розпаралелювання рішення завдань між комп'ютерами кластера й моніторингу кластерної комп'ютерної системи. Прикладами таких ОС є ОС фірми Microsoft - Windows 2003 for clusters; Windows 2008 High-Performance Computing (HPC).

Настільні комп'ютери (desktops) – це найпоширеніші в наш час комп'ютери, якими користуються вдома або на роботі всі люди, від школярів і студентів до хатніх господарок. Такий комп'ютер розміщається на робочому столі й складається з монітора, системного блоку, клавіатури й миші.

Параметри (2010 р.) **настільного комп'ютера**, найбільш прийнятні для використання сучасних ОС: швидкодія процесора 1 – 3 ГГц, оперативна пам'ять – 1 – 8 гігабайт і більше, обсяг жорсткого диска (hard disk drive – HDD) – 200 Гб – 1 Тб і більше (1 терабайт, Тб = 1024 Гб). Вся розмаїтість сучасних операційних систем (Windows, Linux і ін.) – до послуг користувачів **настільних комп'ютерів**.

При необхідності на **настільному комп'ютері** можна встановити дві або більше операційних системи, розділивши його

дискову пам'ять на кілька розділів (partitions) і встановивши на кожний з них свою операційну систему, так що при включенні комп'ютера користувачеві надається стартове меню, з якого він вибирає потрібну операційну систему для завантаження.

Портативні комп'ютери (laptops, notebooks – дослівно "комп'ютери, що містяться на колінах"; " комп'ютери-зошити") – це мініатюрні комп'ютери, які по своїм параметрам не уступають настільним, але по своїм розмірам вільно містяться в невелику сумку або рюкзак, або, наприклад, на колінах користувача, що летить у літаку у відрядження й не бажає втрачати час даром. Ноутбуки коштують звичайно в кілька разів дорожче, ніж настільні комп'ютери з аналогічними характеристиками. На **ноутбуках** використовуються ті ж операційні системи, що й для настільних комп'ютерів (наприклад, Windows або MacOS). Характерними рисами **портативних комп'ютерів** є всілякі убудовані порти й адаптери для бездротового зв'язку:

Wi-Fi (офіційно IEEE 802.11) – вид радіозв'язку, що дозволяє працювати в бездротовій мережі із продуктивністю 10-100 мегабіт у секунду (використовується звичайно на конференціях, у готелях, на вокзалах, аеропортах – тобто в зоні радіусом у кілька сотень метрів від джерела прийому-передачі);

Bluetooth – також радіозв'язок на більш коротких відстанях (10 – 100 м для Bluetooth 3.0), використовувана для взаємодії комп'ютера з мобільним телефоном, навушниками, плеєром і ін.

Зовнішні пристрої (додаткові жорсткі диски, принтери, іноді навіть DVD-ROM) підключаються до ноутбука через порти USB. Ще років 10 назад на ноутбуках активно використовувалися **інфрачервоні порти** (**IrDA**), які, однак, незручні, тому що вимагають присутності "відповідного" IrDA – порту іншого пристрою на відстані 20-30 см від порту ноутбука, при відсутності

між ними перешкод.

Інша характерна риса ноутбуків – це наявність кард-ридеров – портів для читання всіляких карт пам'яті, використовуваних у мобільних телефонах або цифрових фотокамерах; забезпечується також інтерфейс FireWire (офіційно – IEEE 1394) для підключення цифрової відеокамери; таким чином, ноутбуки добре пристосовані для введення, обробки й відтворення обробки мультимедійної інформації. Нині портативний комп'ютер є майже в кожного студента, він використовується для підготовки до відповіді на іспиті, або для рішення завдань практикуму, іноді прямо в університетському буфеті. Один із критичних параметрів ноутбука – час роботи його батарей без підзарядки; дуже добре, якщо цей час становить порядку 10 годин, що поки порівняно рідко зазвичай цей час становить не більше 5 годин. Популярний різновид ноутбука нині – це **нетбук** - ноутбук, призначений для роботи в мережі, звичайно менш потужний і тому більш дешевий, а також більш мініатюрний.

Кишенькові портативні комп'ютери й органайзери (КПК, handhelds, personal digital assistants – PDA) – це "іграшки для дорослих" у вигляді мініатюрного комп'ютера, що міститься на долоні або в кишені, але по своїй швидкодії іноді не уступає ноутбуку. При всій привабливості, серйозні недоліки КПК, з погляду автора, - це незручність введення інформації (доводиться користуватися паличкою-стайлусом, - адже не носити ж із собою ще й громіздку клавіатуру), а також незручність читання інформації на маленькому екрані.

Сучасні КПК мають фактично ті ж порти й адаптери, що й ноутбуки - Wi-Fi, Bluetooth, IrDA, USB. Операційні системи для КПК аналогічні ОС для ноутбуків, але все-таки враховують більше тверді обмеження КПК по обсязі оперативної пам'яті. У цей час для

КПК широко використовується ОС Windows Mobile - аналог Windows для мобільних пристроїв. Донедавна була також широко поширена PalmOS для організайзерів типу PalmPilot фірми 3COM. Зрозуміло, для КПК є апаратура й програмне забезпечення для підключення до ноутбука або настільного комп'ютера з метою синхронізації даних, що забезпечує додаткову надійність.

Мобільні пристрої (mobile intelligent devices – мобільні телефони, комунікатори) – це пристрої, якими кожний з нас користується постійно для голосового зв'язку, рідше – для запису або обробки якої-небудь інформації або для виходу в Інтернет. Найбільш важливі параметри мобільного пристрою – це якість голосового зв'язку й час автономної роботи батареї. Однак все більшого значення набувають убудовані в них цифрові фото і відеокамери.

Операційні системи для мобільних пристроїв відрізняються більшою компактністю, через більш тверді обмеження по пам'яті. Епоха домінування на ринку мобільних телефонів операційних систем типу Symbian, закінчується вони поступаються місцем більше сучасним Google Android і Microsoft Windows Mobile. Для мобільних пристроїв, як і для КПК, досить важлива характеристика ОС – це її надійність, зокрема, схоронність даних після переповнення пам'яті, що виникає, наприклад, у результаті прийому великого числа SMS- Повідомлень, інтенсивної фото- або відеозйомки. У мобільних телефонах використовується платформа ("видання") JME – **Java Micro Edition**, і будь-який мобільний телефон, що випускається от уже більше 10 років, підтримує Java. Програми на Java для мобільних телефонів називаються **мидлетами** (від абревіатури **MID – Mobile Intelligent Device**).

Комп'ютери, що носяться, (wearable computers) - для повсякденного життя досить екзотичні пристрої, однак для

спеціальних застосувань (наприклад, убудовані в скафандр космонавта або в кардіостимулятор) вони життєво важливі. Зрозуміло, їхня пам'ять і швидкодія значно менше, ніж у настільних комп'ютерів, але критичним фактором є їхня надвисока надійність, а для їхніх операційних систем і іншого програмного забезпечення – мінімальний можливий **час відповіді (response time)** – інтервал, протягом якого система обробляє інформацію від датчиків, від користувача або з мережі, перевищення якого загрожує катастрофічними наслідками. Із цього погляду, ОС для **комп'ютерів, що** носяться, можна віднести **до систем реального** часу.

Розподілені системи (distributed systems) – це системи, що складаються з декількох комп'ютерів, об'єднаних у провідну або бездротову мережу. Фактично, такі нині всі комп'ютерні системи (згадаєте девіз **"Мережа – це комп'ютер**"). Всі операційні системи повинні, таким чином, підтримувати розподілений режим роботи, засоби мережевої взаємодії, високошвидкісну надійну передачу інформації через мережу.

Системи реального часу (real-time systems) – обчислювальні системи, призначені для керування різними технічними, військовими й іншими об'єктами в режимі реального часу. Характеризуються основною вимогою до апаратури й програмного забезпечення, у тому числі до операційної системи: **неприпустимість перевищення часу відповіді** системи, тобто очікуваного часу виконання типової операції системи. Для ОС вимоги реального часу накладають досить тверді обмеження - наприклад, в основному циклі роботи системи неприпустимі переривання (тому що вони приводять до неприпустимих тимчасових витрат на їхню обробку). Системи реального часу - особлива досить серйозна й специфічна область, вивчення якої

виходить за рамки даного курсу.

Наведений огляд дає деяке подання про розмаїтість комп'ютерних систем у наш час. Для кожної з них повинна бути розроблена адекватна операційна система.

Основні компоненти операційної системи

Розглянемо тепер основні частини ОС.

Ядро (kernel) – низькорівнева основа будь-якої операційної системи, яка виконується апаратурою в особливому **привілейованому режимі**. Ядро завантажується у пам’ять один раз іперебуває в пам'яті **резидентно** – постійно, по одних і тих же адресах. **Підсистема управління ресурсами (resource allocator)** – частина операційної системи, що управляє обчислювальними ресурсами комп'ютера - оперативною й зовнішньою пам'яттю, процесором і ін.

Управляюча програма (control program, supervisor) – підсистема ОС, що управляє виконанням інших програм і функціонуванням пристроїв вводу-виводу.

Архітектура комп'ютерної системи

Комп'ютерна система має модульну структуру. Для кожного пристрою (пам'ять, зовнішні пристрої) у системі є спеціальний пристрій управління (інакше кажучи, спеціальний процесор), який називається **контролером пристрою**. Всі модулі (центральний процесор, пам'ять і контролер пам'яті, зовнішні пристрої та їхні контролери) з'єднані між собою **системною шиною (system bus),** через яку вони обмінюються сигналами. Як ми вже знаємо роботою кожного контролера керує **драйвер -** спеціалізована низькорівнева програма, що є частиною ОС.

Ось типова структура сучасної настільної або портативної комп'ютерної системи, із вказівкою найпоширеніших типів пристроїв.

Центральний процесор – пристрій, що виконує **команди (instructions)** комп'ютерної системи. У сучасних комп'ютерах, як правило, він є **багатоядерним**, тобто має у своєму складі від 2 до 32 ядер (копій) процесора, що паралельно працюють у загальній пам'яті, або **гібридним**, що складається із центрального й графічного процесорів. Продуктивність кожного ядра – 3 – 3.2 GHz. Помітимо, що під продуктивністю розуміється в цьому випадку **тактова частота процесора (ядра)** – час виконання ним однієї найпростішої машинної команди. Однак є й інші важливі фактори, що визначають загальну продуктивність системи, - тактова частота пам'яті й системної шини. Фактично підсумкову продуктивність системи можна оцінити по самій повільній із цих частин системи (звичайно це системна шина). Ці характеристики необхідно брати до уваги при виборі й покупці комп'ютера.

Оперативна (основна) пам'ять, або просто **пам'ять** – пристрій, що зберігає оброблювані дані. Обсяг пам'яті - 1 - 16 гігабайт і більше; менший обсяг пам'яті використовувати не рекомендується, тому що це може привести до значного вповільнення системи. Тактова частота пам'яті - 667 MHz - 1.5 GHz.

Системна шина – пристрій, до якого приєднані всі модулі комп'ютера й через який вони обмінюються сигналами, наприклад, про переривання. Тактова частота шини – 1 – 1.5 GHz (це і є фактично якась сумарна продуктивність системи). Звичайно використовується шина типу **PCI (Personal Computer Interface).** До неї можуть бути

приєднані процесор, пам'ять, диски, принтер, модем і інші зовнішні пристрої.

Порти – пристрої з роз'ємами для підключення до комп'ютера зовнішніх пристроїв. Кожний порт має свій контролер (і, відповідно, свій драйвер).

Найчастіше використовується **порт USB (Universal Serial Bus),** з характерним плоским роз'ємом, розміром порядку 1 см, із зображенням тризубця. До портів USB можуть підключатися більшість видів пристроїв, причому для цього не потрібно попередньо відключати комп'ютер і підключати пристрій, що дуже зручно. Є кілька стандартів USB з різною швидкодією. Найпоширеніший нині стандарт USB 2.0, що забезпечує швидкодію порту 240 – 260 мегабіт у секунду. Для порівняння, попередній стандарт – USB 1.0 – забезпечував лише 10 – 12 мегабіт у секунду. Розпізнати тип USB- Порту на Вашім комп'ютері можна, якщо вивести інформацію про пристрій; в Windows: **Мій комп'ютер / (права кнопка миші) Властивості / Устаткування / Диспетчер пристроїв / Пристрою USB**. При цьому контролер порту USB 2.0 буде позначений як **розширений (enhanced).** Якщо це не так, Вам необхідно модернізувати порти USB або сам комп'ютер, інакше при запису на флешку Вам доведеться чекати в 20 разів довше (!). Існують також "перехідники" USB 1.0 -> USB 2.0. Новітній стандарт USB 3.0, реалізація якого тільки почалася, забезпечить швидкодію не менш 1 гигабіта в секунду. До порту USB можна підключати клавіатуру, мишу, принтери, сканери, зовнішні жорсткі диски, флешки й навіть TV-**Тюнери** - пристрої для прийому телевізійного сигналу з антени й показу телевізійного зображення на комп'ютері. Рекомендується кожний пристрій підключати завжди до того самого порту USB, інакше для деяких пристроїв (наприклад, того ж TV-Тюнера) можуть виникнути проблеми.

Порти COM (communication ports) – порти для підключення різних комунікаційних пристроїв, наприклад, **модемів** – пристроїв

для виходу в Інтернет і передачі інформації по аналоговій або цифровій телефонній лінії. Більш стара назва стандарту COM-Порту – **RS-232**. У комп'ютерах 10-15 – літньої давнини до COM-Порту часто підключалася мишка (зараз вона, зрозуміло, підключається через USB). Роз'єми COM-Портів мають два формати – "великий" (з 25 контактами - **pins**) і "малий" (з 9 контактами). У сучасних комп'ютерах часто роз'єми COM-Порти відсутні, але операційна система, за традицією, імітує наявність у системі **віртуальних COM- Портів** – уявлюваних COM-Портів, які ОС як би інсталює в систему при установці, наприклад, драйверів для взаємодії через Bluetooth або через кабель комп'ютера з мобільним пристроєм. При цьому фізично мобільний телефон або органайзер може бути підключений до порту USB (або з'єднаний з комп'ютером без дротяним зв'язком), але однаково для взаємодії з ним ОС використовує віртуальний COM- Порт, звичайно з більшим номером (наприклад, 10 або 15).COM-Порт інакше називають **послідовним портом (serial port),** тому що, з погляду ОС і драйверів, COM-Порт – це символьний пристрій послідовної дії.

Порт LPT (від line printer), або **паралельний порт** – це нині вже застарілий вид порту для підключення принтера або сканера, з товстим у перетині кабелем і великим роз'ємом. Всі нові моделі принтерів і сканерів працюють через USB-Порти. Однак іноді доводиться вирішувати завдання підключення до нового комп'ютера старого принтера. Якщо на комп'ютері немає LPT-Порту, доводиться купувати спеціальний перехідник, що підключається до USB або інших портів. Однак і тут можливий сюрприз – роз'єм LPT-Порту має трохи не сумісні один з одним модифікації. Незручність LPT-Порту в тім, що він вимагає попередньо вивантажити ОС і виключити принтер, і тільки після цього виконувати приєднання до комп'ютера, інакше можливий вихід з ладу принтера або

комп'ютера. LPT-Порт може, як правило, працювати й для введення інформації, наприклад, зі сканером, але для цього потрібна низькорівнева утиліта Setup, запустивши її при завантаженні ОС, встановіть для LPT-Порту спеціальний режим роботи: **EPP – Extended Parallel Port**.

Порти SCSI і SCSI-Пристрою. **SCSI (Small Computer System Interface**; вимовляється "**скАзи",** з наголосом на першому складі) – інтерфейс, адаптери й порти для підключення широкого спектра зовнішніх пристроїв – жорстких дисків, CD-ROM / DVD- ROM, сканерів і ін. Стандарт SCSI був запропонований на початку 1980-х рр. і одержав широке поширення, завдяки фірмі Sun, що широко використовувала його у своїх робочих станціях. Характерною зручною можливістю SCSI є можливість підключення до одному SCSI-Порту **гірлянди (ланцюжка) SCSI-Пристроїв** (до 10), кожний з яких має унікальний для даного з'єднання **SCSI ID** – число від 0 до 9, установлюване звичайно на задній панелі SCSI-Пристрою. Наприклад, за традицією, SCSI ID сканера дорівнює 4. На одному з кінців ланцюжка – SCSI-Порт із контролером, на іншому – **термінатор –** перемикач на задній панелі пристрою, установлюваний у певне положення як ознака кінця SCSI-Ланцюжка. Кожний пристрій, крім останнього, з'єднано з наступним SCSI-Пристроєм спеціальним кабелем. SCSI-Роз’єм нагадує роз’єм порту LPT, однак має з боків спеціальні металеві захвати ("лапки") для більшої надійності підключення. Перевага SCSI, крім можливості використання гірлянд пристроїв, у його швидкодії та надійності. Ранні моделі SCSI мали швидкість обміну інформацією до 10-12 мегабіт у секунду, зараз - 240-250 мегабіт у секунду. Є кілька стандартів SCSI (у тому числі - Wide SCSI, Ultra Wide SCSI), на жаль, не сумісних по роз’ємах.

Порт **VGA (Video Graphic Adapter)** використовується для

підключення **монітора (дисплея),** керованого **графічним контролером (процесором)**.

IEEE 1394 (FireWire) – порти для підключення цифрових відеокамер або фотоапаратів. Характерна риса – невеликий блискучий плоский роз'єм шириною 3-5 мм (є два його стандарти). Порт працює в дуплексному режимі, тобто дозволяє управляти не тільки введенням інформації з камери в комп'ютер, але й установками самої камери (наприклад, перемотуванням стрічки) за допомогою комп'ютерної програми (наприклад, Windows Movie Maker). За допомогою такого ж порту може бути підключений також телевізор, що має інтерфейс FireWire. Характерною рисою сучасних комп'ютерів є те, що FireWire-Порти монтуються прямо на **материнській платі (motherboard) –** основній друкованій платі комп'ютера, на якій змонтовані процесор і пам'ять. У таких випадках у технічних характеристиках комп'ютера звичайно вказується: **"FireWire on board (на борті)".** Читачам рекомендується не плутати **FireWire** з Wi-**Fi** – стандартом швидкого бездротового зв'язку.

HDMI (High Definition Multimedia Interface) – інтерфейс і порт, що дозволяє підключити до комп'ютера телевізор або інше відеоустаткування, що забезпечує найкращу якість відтворення (HD – High Definition). Роз'єм HDMI нагадує роз'єм USB. HDMI-Порт входить у комплектацію всіх сучасних портативних комп'ютерів.

Bluetooth – пристрій для бездротового підключення (за допомогою радіозв'язку) до комп'ютера мобільних телефонів, органайзерів, а також навушників, плейєрів і т.ін. Зручність Bluetooth у тім, що комп'ютер і телефон залишаються з'єднаними, навіть якщо відійти від комп'ютера з телефоном на деяку відстань, не більше 10-
15 метрів (Bluetooth 2.0). Новий стандарт Bluetooth 3.0 забезпечує

взаємодію на відстані 200-250 м. Звичайно портативні комп'ютери комплектуються убудованими адаптерами Bluetooth, або можна придбати адаптер Bluetooth, що підключається через USB. Недолік Bluetooth - відносно маленька сумарна швидкість передачі інформації. Наприклад, при пересиланні на комп'ютер через Bluetooth з мобільного телефону Nokia 3230 цифрової фотографії обсягом 500 кілобайт потрібно чекати порядку 10 - 15 секунд. **Інфрачервоний порт (IrDA)** – порт для підключення ноутбука до мобільного телефону (або двох ноутбуків один до одного) через інфрачервоний зв'язок. Незручність портів IrDA – необхідність установки двох пристроїв, що з'єднуються, поруч, на відстані 20-30 см один від одного, без фізичних перешкод між ними. Швидкість передачі інформації – 10-12 мегабіт у секунду. Сучасні ноутбуки вже не комплектуються портами IrDA.

Є також **мережні пристрої – порти й адаптери –** для підключення комп'ютера до локальної мережі.

Функціонування комп'ютерної системи

Перевага описаного модульного підходу до апаратури в тім, що центральний процесор, пам'ять і зовнішні пристрої можуть функціонувати паралельно. Роботою кожного пристрою управляє спеціальний контролер. При необхідності виконання введення-виведення центральний процесор генерує переривання, у результаті якого викликається операційна система, у свою чергу, як реакція на переривання запускається драйвер пристрою, який активізує його контролер. Кожний контролер пристрою має локальний буфер – спеціалізовану пам'ять для обміну інформацією між комп'ютером і пристроєм. Для того, щоб контролер міг почати вивід на пристрій, попередньо центральний процесор (точніше, драйвер пристрою, запущений на ньому) повинен переслати інформацію із заданої

ділянки оперативної пам'яті в буфер пристрою. Далі контролер пристрою вже виконує вивід інформації з буфера на сам пристрій (наприклад, записує її в задану ділянку жорсткого диска). По закінченні обміну інформацією, контролер генерує сигнал про **переривання (interrupt)** по системній шині, цим інформуючи процесор про закінчення операції. Для того, щоб уникнути повторних пересилань великих обсягів інформації, у сучасних комп'ютерах застосовують **DMA (Direct Memory Access) – контролери** – контролери із прямим доступом до оперативної пам'яті. Такі контролери використовують при обміні із пристроєм не свою спеціалізовану пам'ять, а прямо ділянку оперативної пам'яті, де і розміщається буфер обміну.

Обробка переривань

Операційну систему можна розглядати як **програму,керовану перериваннями (interrupt-driven program).** Переривання центрального процесора передає керування підпрограмі обробки даного виду переривань, яка є частиною ОС. У більшості комп'ютерів цей механізм реалізований через **вектор переривань (interrupt vector)** – резидентний масив в оперативній пам'яті, у якому зберігаються доступні по номерах переривань адреси підпрограм- оброблювачів переривань (модулів ОС). При обробці переривання апаратура й ОС зберігають **адресу перерваної команди.** При поновленні обчислень виконання перерваної команди буде знову повторене.

Очевидно, що при обробці переривання, у свою чергу, може виникнути інше переривання. У цьому випадку нове вхідне переривання **затримується (disabled),** і інформація про нього запам'ятовується в **черзі переривань** – системній структурі ОС, що забезпечує почергову обробку всіх виниклих переривань.

Крім переривань, генеруємих апаратурою неявно при обчисленнях, можливо також **програмоване переривання (trap**; дослівно – **пастка**) за допомогою спеціальної команди процесора. У випадку такого переривання працює загальний механізм запуску оброблювача переривання - частини ОС. Таким чином, зі спрощеної точки зору, ОС можна розглядати як набір оброблювачів переривань.

При перериванні ОС зберігає **стан процесора –** значення регістрів і значення **лічильника команд (program counter – PC)** – адреси перерваної команди**.** Оброблювач переривання в ОС визначає по вмісту сегмента об'єктного коду, якого виду переривання виникли і які дії по його обробці варто почати. Серед можливих видів переривань, крім фіксації різних помилок, є також **переривання по таймеру** – періодичні переривання через певний квант часу, призначені для **опитування пристроїв (polling)** – дій операційної системи по періодичній перевірці стану всіх портів і зовнішніх пристроїв, що можуть мінятися із часом: наприклад, до USB-Порту була підключена флешка; принтер закінчив печатку й звільнився, і т.д. ОС виконує реконфігурацію системи й коректує системні таблиці, що зберігають інформацію про пристрої.

Архітектура введення-виведення

Є два різновиди режиму введення-виведення – **синхронний** і **асинхронний** .

Синхронне введення-виведення – це введення-виведення, виконання якого приводить до переходу програми в стан очікування, доти, поки операція введення-виведення не буде повністю закінчена. На апаратному рівні – команда введення-виведення переводить процесор у стан очікування (idle) до наступного переривання. При даному режимі в кожний момент виконується не більше одного

запиту на введення-виведення; одночасне введення-виведення відсутньо. Синхронне виведення виконують відомі всім програмістам оператори виду **println(x).** При їхньому використанні в програмах ми не замислюємося над тим, що використовуємо досить неефективний варіант синхронного введення-виведення. Однак мислення більшості програмістів - послідовне, у тому розумінні, що про свою програму вони мислять як про послідовно виконувану, і взагалі не думають про можливість якого-небудь распаралелювання. При налагодженні програми, або якщо розмір виведеної інформації невеликий, це цілком припустимо.

Асинхронне введення-виведення – введення-виведення, виконуване паралельно з виконанням основної програми. Після того, як починається асинхронне введення-виведення, керування вертається користувальницькій програмі, без очікування завершення введення-виведення (останнє може бути виконано спеціальною явною операцією). Таким чином, операція асинхронного обміну як би розбивається на дві: **почати введення-виведення** і **закінчити введення-виведення**. Остання виконується для того, щоб у цьому місці програма все-таки очікувала завершення обміну, якщо його результат необхідний для подальших обчислень. Такий підхід до реалізації обміну більш важкий для розуміння програмістами й може привести до помилок (наприклад, використана тільки операція початку введення-виведення, а виклик операції її закінчення забута).

Структура пам'яті

Основна (оперативна) пам'ять – єдина велика частина пам'яті, до якої процесор має безпосередній доступ. Як відомо, уміст основної пам'яті не зберігається після перезавантаження системи або після вимикання комп'ютера. **Зовнішня (вторинна) пам'ять** – розширення основної пам'яті, що забезпечує функціональність

стійкої(що зберігається) пам'яті великого обсягу.

Як вторинна пам'ять найчастіше використовуються **жорсткі диски (hard disks).** Фізично вони складаються із твердих пластин з металу або скла, які покриті магнітним шаром для запису. Поверхня диска логічно ділиться на **доріжки (tracks),** які, у свою чергу, діляться на **сектори.** Контролер диска визначає логіку взаємодії між пристроєм і комп'ютером.

Пристрій жорсткого диска показано на рис. 1.

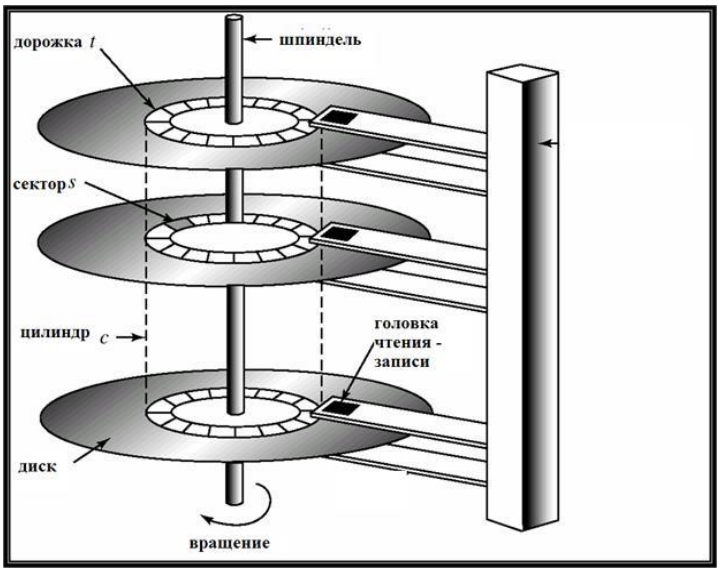


Рис. 1. Пристрій жорсткого диска

Як видно з рисунка, **циліндр** - це група вертикально розташованих один під одним секторів різних магнітних дисків з одним і тим же номером доріжки.

Системи пам'яті організовані в **ієрархію**, виходячи з їхньої швидкодії, вартості й можливості збереження інформації (стійкості). Для оптимізації роботи пам'яті будь-якого виду використовується **асоціативна пам'ять (кэш – cache), яка** розташована в більш швидкодіючих системах пам'яті, вона зберігає елементи більш повільної пам'яті, які частіше використовуються. Із цього погляду, оперативну пам'ять можна розглядати як кеш для зовнішньої пам'яті. Кеш-пам'ять – це, по суті справи, асоціативний список пар **(Адреса, Значення),** причому апаратний пошук у ньому відбувається за

адресою як по ключу. Таким чином, перед звертанням до повільної зовнішньої пам'яті спочатку відбувається пошук по заданій адресі в кеш-пам'яті, і якщо він не привів до успіху, виконується стандартне звертання до зовнішньої пам'яті. Принцип кешування дуже важливий і дозволяє істотно прискорити роботу із зовнішньою пам'яттю. Однак він вимагає реалізації спеціальної політики керування кеш-пам'яттю, тому що кешування вводить додатковий рівень в ієрархії пам'яті й вимагає погодженості даних, збережених одночасно на різних рівнях пам'яті. Апаратура й ОС підтримують **кеш команд, кеш даних, кеш жорсткого диска** й т.д. – для всіх видів пам'яті.

Більше швидкі види пам'яті на схемі розташовані вище, більше повільні - нижче. Схема особливих коментарів не вимагає. Деякі часто використовувані види зовнішньої пам'яті:

- **флеш-пам'ять (флешка)** – зовнішня пам'ять компактного розміру, модуль якої підключаються через USB-Порт. Параметри: обсяг - до 128 гігабайт і більше; швидкість обміну через порт USB 2.0: 240 – 260 мегабіт у секунду;
- **зовнішній жорсткий диск (ZIV drive** і інші) – обсяг до 1 терабайта; працює також через порт USB;

BluRay – диски – новий різновид компакт-дисків великого об’єму (однобічні – 25 гігабайт, двосторонні – 50 гігабайт). Для порівняння, стандартний об’єм диска DVD становить 4.7 гігабайт.

Апаратний захист пам'яті й процесора

З метою спільного використання системних ресурсів (пам'яті, процесора, зовнішніх пристроїв) декількома програмами, потрібно, щоб апаратура й операційна система забезпечили неможливість впливу програми, що виконується некоректно, на інші користувальницькі програми. Для цього необхідна апаратна

підтримка, як мінімум, двох режимів виконання програм – **користувальницького (непривілейованого) режиму (user mode) –** для виконання програм користувачів і **системного (привілейованого, режиму ядра - system mode, monitor mode)** - для модулів операційної системи. Ідея двох режимів у тім, щоб виконувані в привілейованому режимі модулі ОС могли виконувати розподіл і виділення системних ресурсів, зокрема, формувати нові адреси, а користувальницькі програми, у результаті помилок або навмисних атак, виконуючись у звичайному режимі, не могли б звернутися до ділянки пам'яті операційної системи або іншого завдання, змінити їх і цим порушити їхню цілісність. Для визначення поточного режиму виконання команд в апаратурі вводиться **біт режиму**, рівний 0 для системного й 1 – для користувальницького режиму. При перериванні або збої апаратура автоматично перемикається в системний режим. Деякі привілейовані команди, що змінюють системні ресурси й стан системи (наприклад, регістр стану процесора), повинні виконуватися тільки в системному режимі, це захистить системні ресурси від випадкового або навмисного псування при виконанні команд звичайної користувальницької програми.

Для **захисту введення-виведення** всі команди введення-виведення вважаються привілейованими. Необхідно гарантувати, щоб користувальницька програма ніколи не одержала керування в системному режимі й, зокрема, не могла б записати нову адресу у вектор переривань, що, як ми вже відзначали, містить адреси підпрограм обробки переривань, зокрема, зв'язаних з введенням-виведенням.

Особливості ОС для персональних комп'ютерів

Персональні комп'ютери призначені, як правило, для одного

користувача. Проте, ОС для персональних комп'ютерів повинна передбачати режим мультипрограмування (багатозадачності), тому що користувачам часом зручніше виконувати кілька завдань паралельно - наприклад, набирати деякий текст у редакторі, приймати електронну пошту й одночасно друкувати на принтері які-небудь документи. Крім того, при роботі в локальній мережі можливий віддалений вхід на комп'ютер інших користувачів. Тобто, ОС для персональних комп'ютерів повинна підтримувати також режим розподілу часу.

Персональні комп'ютери мають різноманітний набір пристроїв введення-виведення, роботу з якими повинна підтримувати операційна система за допомогою **драйверів –** низькорівневих системних програм для керування цими пристроями. Для користувача зручніше за все, якщо всі необхідні драйвери убудовані в операційну систему. Однак ситуація ускладнюється тим що драйвери пристроїв розробляє зазвичай фірма-розроблювач відповідного пристрою - в англомовній термінології, **Original Equipment Manufacturer (OEM),** а не фірма-розроблювач ОС. Тому при випуску й установці на комп'ютер нової ОС можуть виникнути проблеми із драйверами – який-небудь пристрій нова ОС "не розуміє". На практиці, повинно пройти не менш двох-трьох років експлуатації нової ОС, перш ніж для неї з'являться драйвери для всіх використовуваних зовнішніх пристроїв, хоча останнім часом щодо цього ситуація значно покращилася - нові ОС стають усе більше"тямущими" і мають у своєму складі величезні набори драйверів.

Персональний комп'ютер має традиційні клавіатуру й мишу, що підключаються через USB-Порт, або бездротові клавіатуру й мишу, блок керування який також підключається через USB-Порт. Портативний комп'ютер може мати також убудований маніпулятор

типу trackball (кулька для переміщення курсору миші) або touchpad (плоска пластинка для цієї ж мети). До комп'ютера підключений монітор: для настільного комп'ютера – до порту VGA, для портативного – монітор убудований у комп'ютерну систему, але додатково може підключатися через порт VGA зовнішній монітор або мультимедійний проектор. До традиційних додаткових зовнішніх пристроїв ставиться також принтер (підключається через порт USB, більше старі моделі – через так званий **паралельний порт,** або **LPT – абревіатура від Line PrinTer)**. Рідше використовується **сканер** – пристрій для оцифровки паперових зображень, наприклад, підписаних або рукописних документів. Сканер може також підключатися через порт USB, однак деякі моделі сканерів підключаються через інший інтерфейс – **SCSI**, використовуваний і для жорстких дисків. Є внутрішній жорсткий диск (hard drive) ємністю 250 GB – 1 TB і більше, що підключається через інтерфейс IDE або SATA. Можуть підключатися через порт USB також зовнішні накопичувачі - flash-**пам'ять**, **ZIV drives** і інші різновиди **зовнішніх жорстких дисків**, що мають ємність до 1 терабайта. Операційна система повинна забезпечувати їхнє використання як частин комп'ютерної системи (наприклад, на зовнішній ZIV-Диск може бути навіть встановлене програмне забезпечення, у тому числі - інша операційна система). Для настільного комп'ютера в комплект входить пристрій читання й запису компакт-дисків у різних форматах - CD- ROM, CD-RW (з можливістю запису на CD); DVD-ROM/DVD-RW; DVD-RAM (останнє означає пристрій з режимом безпосереднього запису на компакт-диск, як у пам’ять); BluRay - більш сучасний формат компакт-дисків ємністю до 25 або 50 GB і ін. Досить важливим зовнішнім пристроєм, особливо для портативного комп'ютера, є порт для підключення цифрової відеокамери (IEEE 1394, або

FireWire), більш мініатюрний, ніж USB. Він має дуплексний режим роботи, так що, наприклад, перемотування відеострічки на відеокамері може запускатися програмним шляхом з комп'ютера.

Найбільш важливими властивостями ОС для персонального комп'ютера повинні бути, звичайно, простота й зручність у використанні, дружність до користувача. Це досягається насамперед, зручним і сучасним апаратним і програмним користувальницьким інтерфейсом, наприклад, інтерфейсом типу multi-touch (з доступом безпосередньо до екрана), ноутбуками типу Tablet PC (з можливістю повороту екрана й введення інформації дотиком до екрана).

При розробці ОС для ПК використовуються ті ж технології, які застосовуються й в "великих" ОС (для mainframe-комп'ютерів). Однак, оскільки користувач має персональний доступ до комп'ютера, він часто не має потреби в яких-небудь системних програмах для оптимізації роботи процесора або в поліпшених засобах захисту (останньою, однак, не слід зневажати й відключати її, тому що на комп'ютер можливі мережні атаки).

На тому самому персональному комп'ютері можуть бути встановлені, при необхідності, дві або більше операційних системи - такий комп'ютер зветься **double bootable system**, і при його включенні користувачеві видається початкове меню для уточнення, яку саме ОС потрібно запустити – **boot loader** (завантажник ОС). Таке використання комп'ютера рекомендується, наприклад, для студентів, що вивчають ОС і бажають спробувати нову операційну систему, або вивчити іншу вже відому, на яку дотепер бракувало часу, - наприклад, установити на одному комп'ютері Windows і Linux. Для установки другої ОС необхідно скористатися спеціальною утилітою (наприклад, **Partition Magic**) для виділення на диску для інсталяції нової ОС окремого **розділу (partition)** –

суміжної ділянки дискової пам'яті, що має певне позначення, найчастіше – у вигляді латинської букви.

Персональні комп'ютери мають **мережні адаптери (мережні карти) –** пристрої для підключення до локальної мережі. Відповідно, ОС для персональних комп'ютерів мають у своєму складі драйвери мережних адаптерів і користувальницький інтерфейс для настроювання підключення комп'ютера до локальної мережі.

Паралельні комп'ютерні системи й особливості їх ОС

Паралельні комп'ютерні системи – це мультипроцесорні системи з декількома безпосередньо взаємодіючими процесорами. Класичні приклади: із закордонних комп'ютерів - CRAY, з вітчизняних - "Ельбрус"; з більше сучасних - комп'ютери серії СКІФ. У цей час випускаються мультипроцесорні робочі станції - наприклад, купивши або одержавши в подарунок настільний комп'ютер, Ви можете виявити в його складі два або навіть чотири процесори. Відповідно, ОС повинна забезпечувати реконфігурацію такої системи, підключення нових процесорів або видалення процесорів із системи, распаралелювання рішення завдання на декількох процесорах і синхронізацію вирішальних її паралельних процесів.

Серед паралельних комп'ютерів виділяються **тісно зв'язані (tightly coupled) системи**, у яких процесори розділяють загальну пам'ять і таймер (такти); взаємодія між ними відбувається через загальну пам'ять.

Багатоядерні (multi-core) комп'ютери – комп'ютерні системи, засновані на тісно зв'язаних один з одним процесорах (**ядрах**), що перебувають в одному кристалі, що розділяють асоціативну пам'ять (кеш) другого рівня й працюють на загальній

пам'яті.

Переваги паралельної комп'ютерної системи:

1. **Поліпшена продуктивність (throughput)** – очевидно, що распаралелювання алгоритму рішення завдання може зменшити сумарний час її рішення;

2. **Економічність** – у паралельній системі ОС можна доручити частину роботи іншому процесору або ядру;

3. **Підвищена надійність** – при збої або відмові одного із процесорів ОС можна перемкнути обчислення на інший процесор;

4. **"Дружнє" до користувача зниження продуктивності (graceful degradation)** – якщо один із процесорів відмовив і виведений з конфігурації, користувач, при правильній організації комп'ютера й ОС, може навіть не відчути вповільнення обчислень;

5. **Стійкість до помилок (fail-soft system)** – стабільна робота багатопроцесорної системи при помилці в апаратурі або в програмі.

Симетричні й асиметричні мультипроцесорні системи

Симетрична мультипроцесорна система - symmetric multiprocessing (SMP) – це багатопроцесорна комп'ютерна система, всі процесори якої рівноправні й використовують ту саму копію ОС. Операційна система при цьому може виконуватися на **будь-якому** процесорі. У такій системі будь-якому вільному процесору може бути доручено будь-яке завдання. Всі процесори використовують загальну пам'ять і загальні дискові ресурси. Кілька процесів (або потоків) можуть виконуватися одночасно без істотного порушення продуктивності. Більшість сучасних ОС підтримують архітектуру SMP. Після інсталяції ОС (наприклад, Linux) на симетричну мультипроцесорну систему користувач може помітити в меню boot

loader, що фактично на його комп'ютер встановилася не одна, а дві версії ОС - з підтримкою SMP і без неї.

Асиметрична мультипроцесорна система (asymmetric multiprocessing) – це багатопроцесорна комп'ютерна система, у якій процесори спеціалізовані за своїми функціями. Кожному процесору дається специфічне завдання; **головний процесор (master processor)** планує роботу **підлеглих процесів (slave processors).** У такій системі ОС, як правило, виконується на одному певному, закріпленому за нею, центральному процесорі. Подібна архітектура більш типова для дуже великих систем. Приклад – система "Ельбрус", що мала у своєму складі, залежно від конфігурації, від одного до 10 центральних процесорів, від одного до чотирьох спеціалізованих **процесорів введення-виведення (ПВВ),** від одного до чотирьох **процесорів передачі даних (ППД).**

Розподілені комп'ютерні системи й особливості їх ОС

У **розподіленій системі (distributed system)** обчислення розподілені між декількома фізичними процесорами (комп'ютерами), об'єднаними між собою в мережу.

Слабко зв'язана система (loosely coupled system) – розподілена комп'ютерна система, у якій кожний процесор має свою локальну пам'ять, а різні процесори взаємодіють між собою через **лінії зв'язку –** високошвидкісні шини, телефонні лінії, бездротовий зв'язок (Wi-Fi, EVDO, Wi-Max і др.).

Переваги розподілених систем:

1. **Розподіл (спільне використання) ресурсів**: у розподіленій

системі різні ресурси можуть зберігатися на різних комп'ютерах.

Немає необхідності дублювати програми або дані, зберігаючи їхні копії на декількох комп'ютерах.

2. Спільне завантаження (load sharing): кожному комп'ютеру в розподіленій системі можуть бути доручені певні завдання, що він виконує паралельно з виконанням іншими комп'ютерами своїх завдань.

3. Надійність: при відмові або збої одного з комп'ютерів розподіленої системи його завдання може бути перерозподілено іншому комп'ютеру, щоб збій у мінімальному ступені вплинув або зовсім не вплинув на підсумковий результат.

4. Зв'язок: у розподіленій системі всі комп'ютери зв'язані один з одним, так що, наприклад, при необхідності можливий відд лений вхід з одного комп'ютера на іншій з метою використання ресурсів могутнішого комп'ютера.

У розподіленій системі комп'ютери зв'язані в мережну інф структуру, що може бути:

1. локальною мережею (local area network - LAN);
2. глобальною або регіональною мережею (wide area network -
3. WAN).

По своїй організації розподілені системи можуть бути клієнт-серверними (client-server) або одноранговими (peer-to-peer) системами. У клієнт-**серверній** системі певні комп'ютери відіграють роль серверів, а інші – роль клієнтів, що користуються їхніми послугами. Подібна організація розподілених систем найпоширеніша, і ми розглянемо її докладніше. В **одноранговій** розподіленій системі всі комп'ютери рівноправні.

Види серверів у клієнт-серверних комп'ютерних системах

Клієнт-Серверна архітектура розподілених систем досить

широко поширена й підтримана операційними системами. Тому дуже важливо знати, які види й функції серверів пропонують сучасні розподілені системи.

Файл-Сервер (file server) – комп'ютер і програмне забезпечення, що надають доступ до підмножини файлових систем, розташованих на дисках комп'ютера-сервера, іншим комп'ютерам локальної мережі (LAN). Приклад – серверне програмне забезпечення **SAMBA (SMB –** скорочення від **Server Message Block)** для ОС типу UNIX (Linux, FreeBSD, Solaris і т.д.), що забезпечує доступ з Windows-Комп'ютерів локальної мережі до файлових систем UNIX- Машин. Samba також реалізована для платформи Macintosh / MacOS.

Сервер додатків (application server) – комп'ютер і програмне забезпечення, що надає обчислювальні ресурси (пам'ять і процесор) і необхідне оточення для віддаленого запуску певних класів (як правило, великих) додатків з інших комп'ютерів локальної мережі. Приклади серверів додатків - WebSphere (IBM), WebLogic (BEA) - найкращі з відомих серверів додатків, що працюють в Java Enterprise Edition (JEE).

Сервер баз даних (database server) – комп'ютер і програмне забезпечення, що надає доступ іншим комп'ютерам мережі до баз даних, розташованих на комп'ютері-сервері. Приклад: серверне програмне забезпечення для доступу до баз даних Microsoft SQL Server.

Веб-Сервер (Web server) – комп'ютер і програмне забезпечення, що надає доступ клієнтам через WWW до Web-Сторінок, розташованих на комп'ютері-сервері. Приклад: вільно розповсюджуваний Web-Сервер Apache.

Прокси-Сервер – комп'ютер і програмне забезпечення, що є частиною локальної мережі й підтримує ефективний обіг

комп'ютерів локальної мережі до Інтернету, фільтрацію трафіка, захист від зовнішніх атак. Proxy-Сервер звичайно вбудований в операційну систему.

Сервер електронної пошти – комп'ютер і програмне забезпечення, що виконують відправлення, одержання й "розкладку" електронної пошти для комп'ютерів деякої локальної мережі. Можуть забезпечувати також **криптування** пошти (email encryption) – шифрування електронних листів перед відправленням адресатам з певного мережевого домена (як правило, замовникові) і їхнє дешифрування після одержання від замовника.

Серверний бэк-энд (Server back-end) – група (пул) зв'язаних у локальну мережу серверних комп'ютерів, використовуваних замість одного сервера, з метою більшої надійності й надання більшого обсягу ресурсів. Інший термін, близький до цього, - **центр обробки даних (data center).** Ці поняття особливо актуальні у зв'язку з усе більшим поширенням хмарних обчислень, що є, із цього погляду, найбільш сучасною реалізацією клієнт-серверної схеми взаємодії.

Кластерні обчислювальні системи і їх ОС

Комп'ютерні кластери досить популярні для наукових обчислень. Комп'ютери в кластері, як правило, зв'язані між собою через швидку локальну мережу. Кластеризація дозволяє двом або більше системам використовувати загальну пам'ять. Кластеризація забезпечує високу надійність. Розрізняють комп'ютерні кластери двох видів:

- **асиметрична кластеризація (asymmetric clustering) –** організація комп'ютерного кластера, при якій один комп'ютер виконує додаток, а інші простоюють;

- **симетрична кластеризація (symmetric clustering) -** організація комп'ютерного кластера, при якій всі машини кластера виконують одночасно різні частини одного великого додатка.

Розрізняють також:

- **кластери з високошвидкісним доступом (high-availability clusters) –** комп'ютерні кластери, що забезпечують оптимальний **доступ** до ресурсів, які надані комп'ютерами кластера, наприклад, до баз даних;
- **кластери з балансованим завантаженням (load-balancing clusters)** – комп'ютерні кластери, які мають кілька вхідних комп'ютерів, що балансують запити (front - ends), та розподіляють завдання між комп'ютерами серверного back-end'a (серверної ферми).

Кластери часто використовуються в університетах, в дослідницьких центрах. Операційні системи для кластерів: Windows 2003 for clusters; Windows 2008 High-Performance Computing

Системи й ОС реального часу

Системи реального часу часто використовуються як управляючі пристрої для спеціальних додатків, - наприклад, для наукових експериментів; у медичних системах, пов'язаних із зображеннями; системах управління в промисловості; системах відображення (display); системах управління космічними польотами, АЕС і ін. Для таких систем характерні наявність і чітке виконання певних тимчасових обмежень (час реакції - response time; час наробітку на відмову й ін.).

Розрізняються системи реального часу видів **hard real-time** і soft real-time.

Hard real-time – системи – системи реального часу, у яких

при порушенні тимчасових обмежень може виникнути критична помилка (відмова) керованого нею об'єкта. Приклади: система керування двигуном автомобіля; система керування кардіостимулятором. У таких системах вторинна пам'ять обмежена або відсутня; дані зберігаються в оперативній пам'яті (RAM) або постійному запам'ятовувальному пристрої (ПЗУ, ROM). При використанні таких систем можливі конфлікти із системами розподілу часу, що не мають місця для ОС загального призначення. Точніше, при роботі подібних систем не допускаються переривання; всі необхідні дані для основного циклу роботи системи повинні бути попередньо завантажені у пам’ять; процес, що виконує код такої системи, не повинен піддаватися відкачці на диск. ОС для таких систем звичайно спрощені, замість віртуальної пам'яті виділяється фізична, всі інші види віртуалізації ресурсів виключені. Популярною практикою розробки ОС реального часу є практика розробки таких ОС на основі відкритих вихідних кодів ОС загального призначення шляхом "відсікання всього зайвого". Однак при цьому слід дотримуватися обережності.

Soft real-time – системи – системи реального часу, у яких порушення тимчасових обмежень не приводить до відмови керованого нею об'єкта. Звичайно це системи керування декількома взаємозалежними системами з ситуацією що постійно змінюється. Приклад - система планування рейсів на комерційних авіалініях**.** У випадку якої-небудь затримки в роботі такої системи, у найгіршому разі, пасажирам деяких рейсів прийде небагато почекати в аеропорті, але ніяких фатальних наслідків не буде. Подібні системи мають обмежену корисність для промислових систем керування. Вони також корисні в сучасних додатках (наприклад, для мультимедіа й віртуальної реальності), що вимагають розвинених можливостей ОС.

Обчислювальні середовища

У сучасному світі IT має місце тенденція до інтеграції описаних вище пристроїв і їхніх локальних мереж в **обчислювальні середовища** – інтегровані розподілені комп'ютерні системи для рішення завдань у різних проблемних сферах. Обчислювальні середовища підрозділяються на наступні види:

- **традиційні обчислювальні середовища** – локальні й регіональні мережі, використовувані протягом декількох десятків років;
- **Web-Орієнтовані обчислювальні середовища –** обчислювальні середовища на основі Web-Сервісів, характерні для теперішнього часу, починаючи з 1990-х рр.; до цього класу ставляться й середовища для хмарних обчислень;
- **убудовані (embedded) обчислювальні середовища –** обчислювальні середовища для спеціалізованих пристроїв, наприклад, мережі мікропроцесорів, убудованих в елементи лінії електропередач.

Всі ці види обчислювальних середовищ повинні адекватно обслуговуватися операційними системами, у чому й складаються найближчі завдання їхньої розробки.

Хмарні обчислення й ОС для хмарних обчислень

Хмарні обчислення (**cloud computing**) є одним з найбільш популярних напрямків розвитку IT. **"Хмара" (cloud) –** це вже десятки років використовувана метафора для зображення сервісів, надаваних через Інтернет або іншу комунікаційну мережу (наприклад, через ATM-Мережу). **Хмарні обчислення** – модель

обчислень, заснована на **динамічно масштабованих (scalable)** і **віртуалізованих** ресурсах (даних, додатках, ОС і ін.), які доступні й використовуються як **сервіси** через Інтернет і реалізуються за допомогою високопродуктивних **центрів обробки даних (data centers)**

Недолік хмарних обчислень у тім, що користувач виявляється повністю залежним від використовуваної ним "хмари" (у якій доступні використовувані ним дані й програми) і не може управляти не тільки роботою "хмарних" комп'ютерів, але навіть резервним копіюванням своїх даних. У зв'язку із цим виникає цілий ряд важливих питань про безпеку хмарних обчислень, збереження конфіденційності користувальницьких даних і т.д.; далеко не всі з них на даний момент вирішені.

Серйозною проблемою організації хмарних обчислень із погляду апаратури центрів обробки даних є економія електроенергії й проблема розподілу завантаження, тому що хмарні обчислення в кожному центрі обробки даних мають (або в найближчому майбутньому будуть мати) **мільйони** віддалених користувачів.

Найбільш популярна "хмарна" платформа - Microsoft Windows Azure (хмарна ОС) і Microsoft Azure Services Platform (реалізована на основі Microsoft.NET). Windows Azure можна розглядати як "ОС у хмарі". Користувачеві немає необхідності турбуватися про її інсталяцію на його комп'ютері, який може не мати для цього необхідних ресурсів. Усе, що потрібно, це мати Web- Браузер і мінімальний пакет надбудов (plug - ins) для запуску й використання через браузер хмарних сервісів.

У цей час всі великі компанії (Microsoft, IBM, HP, Dell, Oracle і ін.) розробляють свої системи хмарних обчислень; є тенденція до інтеграції цих корпоративних систем у єдину доступну користувачеві "хмару".

Периферійні пристрої:

- *Області застосування ПЕОМ*

Електронні обчислювальні машини (ЕОМ), на і вказує їх назву, спочатку використовували тільки для трудомістких обчислень, наприклад, для визначення траєкторії виходу супутника на навколоземну орбіту, дослідження процесів в ядерному реакторі, визначення форми профілю корпусу атомного підводного човна, літака, ракети і т.д. .д.

Подальший розвиток обчислювальної техніки призвело до можливості використання персональних електронно-обчислювальних машин (ПЕОМ) у напрямі виконання технічних розрахунків, а й у інших сферах людської діяльності - у сфері економіки та планування, у криміналістиці, видавничій діяльності, зв'язку тощо.

В даний час ПЕОМ використовується як засіб обчислення тільки на ~ 10 - 15 %, а основне їхнє призначення стосується великої кількості інших можливих застосувань. Розглянемо основні сфери використання ПЕОМ.

Підготовка та редагування текстових документів

В даний час у світовій практиці друкарські машинки майже не застосовуються. Видавничі системи використовують ПЕОМ, які дозволяють як готувати текстові документи, а й легко їх редагувати, зберігати, і навіть передавати на значні відстані електронною поштою.

Крім того, підготовлені документи, наприклад листи, можуть використовуватися багаторазово, як «макети-шаблони», при цьому, достатньо лише змінити адресу та ім'я адресата, дату відправлення та ін.

Сучасні текстові редактори мають різні способи обробки документів, що забезпечує можливість виконання обчислень, побудова графіків та діаграм, сортування елементів за вибраними критеріями.

Найбільш поширеним та зручним прикладним пакетом-редактором, який

використовується для підготовки документації, належить редактор MS WORD у середовищі Windows.

Збереження та перетворення графічних даних, фотографій, малюнків

Окрім видавничої діяльності, ПЕОМ знайшли широке застосування, наприклад, у роботі правоохоронних органів багатьох країн світу для створення фотороботів та розшуку злочинців. Для цього використовують як чорно-білі, так і кольорові сканери. До найпоширеніших графічних прикладних пакетів програмування, які використовуються при скануванні образів та при подальшій обробці відносяться програми FINE READER 3.0, ADOBE FOTOSHOP, IMAGING for Windows'95 та ін. Вони широко відомі у світі мистецтва та знайшли популярність у дизайнерів інтер'єру та моди.

Створення та редагування динамічних графічних образів

За допомогою ПЕОМ можна створювати мультиплікаційне зображення, включати його відео. Можна і навпаки - включати реальні об'єкти та людські образи у мультиплікацію та ін.

Сучасний стан розвитку проектування потребує створення машинобудівних та будівельних креслень за допомогою ПЕОМ, при цьому широко використовують програму AUTOCAD. Виведення креслень на друк виконують за допомогою плоттеру.

ПЕОМ як зв'язку

ПЕОМ як зв'язку - телекомунікації широко використовується у світовій практиці. При цьому використовують FAX-MODEM (модулятор-демодулятор), який включають до звичайної телефонної мережі. Найбільш поширеними програмами зв'язку є INTERNET EXPLORER, INTERNET MAIL та н.

ПЕОМ - професійний та гральний тренажер

ПЕОМ широко використовується як тренажерів у різних галузях

промисловості, наприклад, на ТЕС та АЕС для запобігання аваріям при експлуатації обладнання, а також при підготовці фахівців високого рангу - пілотів авіації, водіїв автотранспорту, судноводіїв у навчальному процесі.
Гральні тренажери використовують для адаптації користувачів ПЕОМ молодшого віку до роботи з професійними програмами, вивчення іноземних мов, розвитку математичних та логічних здібностей, підвищення швидкості реакції при зміні зовнішніх обставин.

ПЕОМ як зв'язку з об'єктами

ПЕОМ дозволяє перетворювати аналогові сигнали на цифрові (при цьому використовують АЦП - аналого-цифрові перетворювачі) і, навпаки, цифрові сигнали на аналогові (за допомогою ЦАП - цифро-аналогових перетворювачів), що дозволяє керувати діючими об'єктами, впливати на показники їх роботи, запобігати аварійність.

ПЕОМ як обчислення

ПЕОМ широко застосовуються до виконання обчислень. При цьому широко використовують як алгоритмічні мови програмування різного рівня, так і сучасні прикладні пакети ПЕОМ MS WORD, MS EXCEL, MS ACCESS та ін. Можливості деяких з них розглянуті в подальшому матеріалі курсу.

Резюме

Мета навчання комп'ютерним наукам полягає в тому, щоб виробити уявлення про конкретні процеси і механізми обробки інформації та зрозуміти відносини, які існують між прикладними програмами та обчислювальними системами.

В даному розділі розглядається організація функціонування обчислювальної системи, основні її компоненти та архітектура, розкриваються механізми управління, що дозволяють ефективно

координувати роботу обчислювальних ресурсів при проведенні обчислень. Розглядаються різні типи комп’ютерних систем та особливості їх операційних систем.

Контрольні запитання:

1. Перелічіть складові комп’ютерних систем. Опишіть загальну картину функціонування комп'ютерної системи.
2. З якою метою організована таблиця стану пристроїв?
3. Опишіть структуру пам'яті комп’ютера.
4. Перелічіть елементи ієрархії пам’яті.
5. Як організовано апаратний захист пам'яті й процесора?
6. Як класифікуються сучасні комп'ютерні системи?
7. Що представляють собою кластери комп’ютерів?
8. Перелічіть особливості CISC-архітектури.
9. Перелічіть особливості RISC-архітектури.
10. Перелічіть особливості VLIW-архітектури.
11. Перелічіть особливості EPIC-архітектури.
12. Які основні функції операційної системи? Чи немає між ними протиріч?
13. Перелічіть основні компоненти операційної системи.
14. Що спільного й у чому відмінності між мережною і розподіленою операційними системами? Яка з них складніша в реалізації і чому?
15. Перелічіть види серверів у клієнт-серверних операційних системах.

Розділ ІІ

РОЗРОБКА КОРИСТУВАЛЬНИЦЬКОГО ІНТЕРФЕЙСУ

Проектування інтерфейсу

Інтерфейс — це зовнішня оболонка додатка разом із програмами керування доступом та іншими схованими від користувача механізмами керування, яка дає можливість працювати з документами, даними й іншою інформацією, що зберігається в комп'ютері або за його межами. Головна мета будь-якого додатка — забезпечити максимальну зручність і ефективність роботи з інформацією: документами, базами даних, графікою або зображеннями. Тому інтерфейс є, мабуть, найважливішою частиною будь-якого додатка.

Добре розроблений інтерфейс гарантує зручність роботи із додатком і в остаточному підсумку його комерційний успіх. У цьому розділі описано види інтерфейсів і процес створення інтерфейсу з усіма його основними елементами керування: меню, контекстним меню, панелями інструментів, рядком стану.

Проектування інтерфейсу — процес циклічний. На цьому етапі розробки додатка бажано частіше спілкуватися з користувачами й

замовниками додатка для вироблення найбільш прийнятних за ефективністю, зручністю й зовнішнім виглядом інтерфейсних

рішень. Вибір того або іншого типу інтерфейсу залежить від складності розроблювального додатка, оскільки кожний з них має деякі недоліки й обмеження й призначений для вирішення певних завдань. При цьому необхідно відповісти на ряд запитань: яка кількість типів документів обробляється в додатку, чи мають дані деревоподібну ієрархію, чи буде потрібна панель інструментів, яка кількість документів обробляється за певний інтервал часу (наприклад, за день) і т.д. Тільки після цього можна вибирати конкретний тип інтерфейсу.

Головні принципи при розробці інтерфейсу.

Розглянемо можливі типи інтерфейсів та їх характерні особливості, що впливають на вибір інтерфейсного рішення додатка. Загальні поради по розробці інтерфейсу

При розробці інтерфейсу потрібно керуватися наступними принципами:

- **Стандартизація.** Рекомендується використати стандартні, перевіреними багатьма програмістами й користувачами інтерфейсні рішення. Для Visual Basic це, зрозуміло, рішення Microsoft. Причому як стандарт (зразок для "наслідування") може служити кожний з додатків — Word, Excel або інші додатки Microsoft. Під рішеннями маються на увазі дизайн форм, розподіл елементів керування у формах, їхнє взаємне розташування, значки на кнопках керування, назви команд меню.
- **Зручність і простота роботи.** Інтерфейс повинен бути інтуїтивно зрозумілим. Бажано, щоб всі дії легко запам'ятовувалися й не вимагали стомлюючих процедур: виконання додаткових команд, зайвих натискань на кнопки, виклику проміжних діалогових вікон.

- **Зовнішній дизайн.** Не можна, щоб інтерфейс стомлював зір. Він повинен бути розрахований на тривалу роботу користувача з додатком протягом дня.
- **Не перевантаженість форм.** Форми повинні бути оптимально завантажені елементами керування. При необхідності можна використати вкладки або додаткові сторінки форм.
- **Угруповання**. Елементи керування у формі необхідно групувати за змістом, використовуючи елементи угруповання: рамки, фрейми.
- **Розрідженість об'єктів форм.** Елементи керування варто розташовувати на деякій відстані, а не ліпити один на одного. Для виділення елементів керування можна організувати порожні простори у формі.

  Тут перераховані основні принципи, які варто враховувати при проектуванні інтерфейсу додатка, але вони не є догмою. Згодом у процесі роботи з користувачами й накопиченням практичного досвіду будуть вироблятися й свої оптимальні принципи побудови інтерфейсу.

Контрольні запитання:

1. Поняття користуваницького інтерфейсу та етапи його розробки?

2. Від чого зависит вибір інтерфейсу?

3.Типи інтерфейсу та їх особливості?

4. Якими принципами користуются при розробці інтерфейсу?

5. Що входит в поняття оптимальних принципів побудови інтерфейсу?

6. Що уявляють собою принципи:

- стандартізації

- зовнішний дізайн

- не перевантаженість форм

- угруповання

- розрідженість об'єктів форми?

Розділ ІІІ

АЛГОРИТМІЧНІ МОВИ ТА СИСТЕМИ ПРОГРАМУВАННЯ

Алгоритми та мови програмування

Керування комп'ютером здійснюється відповідно деяким *алгоритмам.* Під алгоритмом розуміють певний опис способу рішення завдання у вигляді кінцевої (за часом) послідовності дій. Для подання алгоритмів у вигляді, який розуміє комп'ютер, використовують *мови програмування.* Спочатку розробляється алгоритм дій, потім алгоритм записується на одній з таких мов. У результаті отримують текст програми - повний, закінчений і детальний опис алгоритму мовою програмування. Текст програми переводиться в машинний код або виконується під управлінням спеціальних службових програм, які називаються *трансляторами*.

Самому написати програму в машинному коді досить складно, причому складність різко зростає зі збільшенням розміру програми й трудомісткістю рішення потрібного завдання. Умовно можна

вважати, що машинний код прийнятний, якщо розмір програми не перевищує декількох десятків байтів і нема потреби в операціях ручного уведення/виведення даних. Сьогодні практично всі програми створюються за допомогою мов програмування. Теоретично програму можна написати засобами звичайної людської (природної) мови - це називається програмуванням на *метамові* (подібний підхід зазвичай використовується на етапі складання алгоритму), але автоматично перевести таку програму в машинний код поки неможливо, через високу неоднозначність природної мови.

Мови програмування - штучні мови. Від природних вони відрізняються обмеженим числом «слів», значення яких зрозуміло транслятору, і дуже суворими правилами запису команд *(операторів).* Сукупність подібних вимог утворює *синтаксис* мови програмування, а *зміст* кожної команди й інших конструкцій мови - її *семантику.*

Порушення форми запису програми приводить до того, що транслятор не зрозуміє призначення оператору й видає повідомлення про синтаксичну помилку, а правильно написане, але не відповідаюче алгоритму використання команд мови приводить до семантичних помилок (логічних помилок або помилок часу виконання). Процес пошуку помилок у програмі називається *тестуванням,* процес усунення помилок - *налагодженням.*

Компілятори й інтерпретатори

За допомогою мови програмування створюється не готова програма, а тільки її текст, що описує раніше розроблений алгоритм. Щоб одержати працюючу програму, треба цей текст або автоматично перевести в машинний код (для цього служать *програми- компілятори)* і потім використати окремо від вихідного

тексту, або відразу виконувати команди мови, зазначені в тексті програми (цим займаються *програми-інтерпретатори).*

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). Тільки після того як поточний оператор успішно виконаний, інтерпретатор перейде до наступного. При цьому якщо той самий оператор повинен виконуватися в програмі багаторазово, інтерпретатор щораз буде виконувати його так, начебто зустрів уперше. Внаслідок цього, програми, у яких потрібно здійснювати великий обсяг повторюваних обчислень, можуть працювати повільно. Крім того, для виконання такої програми на іншому комп'ютері там також повинен бути встановлений інтерпретатор - адже без нього текст програми є просто набором символів.

По-іншому, можна сказати, що інтерпретатор моделює якусь віртуальну обчислювальну машину, для якої базовими інструкціями є не елементарні команди процесора, а оператори мови програмування.

Компілятори повністю обробляють весь текст програми (він іноді називається *вихідний код).* 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. Нерідко при цьому виконується *оптимізація* за допомогою набору методів, що дозволяють підвищити швидкодію програми (наприклад, за допомогою інструкцій, орієнтованих на конкретний процесор, шляхом виключення непотрібних команд, проміжних обчислень і т.д.). У результаті закінчена програма виходить більш компактною та 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 мінливої під час роботи програми. Тоді в машинний код доводиться вставляти безліч додаткових перевірок, аналізувати наявність ресурсів операційної системи, динамічно їх захоплювати й звільняти, формувати й обробляти в пам'яті комп'ютера складні об'єкти, що на рівні жорстко заданих машинних інструкцій здійснити досить важко, а для ряду завдань практично неможливо.

За допомогою інтерпретатора, навпаки, припустимо в будь-який момент припинити роботу програми, дослідити вміст пам'яті, організувати діалог з користувачем, виконати як завгодно складні перетворення даних і при цьому постійно контролювати стан навколишнього програмно-апаратного середовища, завдяки чому досягається висока надійність роботи. Інтерпретатор при виконанні кожного оператора перевіряє безліч характеристик операційної системи й при необхідності максимально докладно інформує розроблювача про виникаючі проблеми. Крім того, інтерпретатор дуже зручний для використання в якості інструменту вивчення програмування, тому що дозволяє зрозуміти принципи роботи будь-якого окремого оператора мови.

У реальних системах програмування перемішані технології і компіляції, і інтерпретації. У процесі налагодження програма може виконуватися по кроках, а результуючий код не обов'язково буде машинним - він навіть може бути вихідним кодом, написаним на іншій мові програмування (це істотно спрощує процес трансляції, але вимагає компілятора для кінцевої мови), або проміжним

машинно- незалежним кодом абстрактного процесора, що у різних комп'ютерних архітектурах стане виконуватися за допомогою інтерпретатора або компілюватися у відповідний машинний код.

Рівні мов програмування

Різні типи процесорів мають різні набори команд. Якщо мова програмування орієнтована на конкретний тип процесора й ураховує його особливості, то він називається *мовою програмування низького рівня.* У цьому випадку «низький рівень» не значить «поганий». Мається на увазі, що оператори мови близькі до машинного коду й орієнтовані на конкретні команди процесора.

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

С допомогою мов низького рівня створюються дуже ефективні й компактні програми, тому що розроблювач одержує доступ до всіх можливостей процесора. З іншого боку, потрібно дуже добре розуміти пристрої комп'ютера, ускладнюється налагодження великих додатків, а результуюча програма не може бути перенесена на комп'ютер з іншим типом процесора. Подібні мови звичайно застосовують для написання невеликих системних додатків, драйверів пристроїв, модулів стикування з нестандартним устаткуванням, коли найважливішими вимогами стають компактність, швидкодія й можливість прямого доступу до

апаратних ресурсів. У деяких областях, наприклад у машинній графіці, мовою асемблера пишуться бібліотеки, що ефективно реалізують потребуючі інтенсивних обчислень алгоритми обробки зображень.

Мови програмування високого рівня значно ближчі й зрозуміліші людині, ніж комп'ютеру. Особливості конкретних комп'ютерних архітектур в них не враховуються, тому створювані програми на рівні вихідних текстів легко переносяться на інші платформи, для яких створений транслятор цієї мови. Розробляти програми на мовах високого рівня за допомогою зрозумілих і потужних команд простіше, а помилок при створенні програм допускається набагато менше.

Покоління мов програмування

Мови програмування прийнято ділити на п'ять *поколінь.* У перше покоління входять мови, створені на початку 50-х років, коли перші комп'ютери тільки з'явилися на світ. Це була перша мова асемблера, створена за принципом «одна інструкція - один рядок».

Розквіт другого покоління мов програмування прийшовся на кінець 50-х - початок 60-х років. Тоді був розроблений символічний асемблер, у якому з'явилося поняття змінної. Він став першою повноцінною мовою програмування. Завдяки його виникненню помітно зросли швидкість розробки й надійність програм.

Появу третього покоління мов програмування прийнято відносити до 60-м років. У цей час народилися універсальні мови високого рівня, з їхньою допомогою вдається вирішувати завдання з будь-яких галузей. Такі якості нових мов, як відносна простота, незалежність від конкретного комп'ютера й можливість використання потужних синтаксичних конструкцій, дозволили різко підвищити продуктивність праці програмістів. Зрозуміла більшості

користувачів структура цих мов залучила до написання невеликих програм (як правило, інженерного або економічного характеру) значне число фахівців з некомп'ютерних галузей. Переважна більшість мов цього покоління успішно застосовується й сьогодні.

З початку 70-х років по теперішній час триває період мов четвертого покоління. Ці мови призначені для реалізації великих проектів, підвищення їхньої надійності й швидкості створення. Вони звичайно орієнтовані на спеціалізовані галузі застосування, де гарних результатів можна домогтися, використовуючи не універсальні, а проблемно-орієнтовані мови, що оперують конкретними поняттями вузької предметної галузі. Як правило, у ці мови вбудовуються потужні оператори, що дозволяють одним рядком описати таку функціональність, для реалізації якої на мовах молодших поколінь потрібні були б тисячі рядків вихідного коду.

Народження мов п'ятого покоління відбулося в середині 90-х років. До них відносяться також системи автоматичного створення прикладних програм за допомогою візуальних засобів розробки, без знання програмування. Головна ідея, що закладається в ці мови, - можливість автоматичного формування результуючого тексту на універсальних мовах програмування (який потім потрібно відкомпілювати). Інструкції ж уводяться в комп'ютер у максимально наочному виді за допомогою методів, найбільш зручних для людини, не знайомої із програмуванням.

Інформатика, як основна дисципліна програмування.

Інформатика - це заснована на використанні комп'ютерної техніки дисципліна, що вивчає структуру та загальні властивості інформації, а також закономірності та методи її створення, зберігання, пошуку, перетворення, передачі та застосування у різних сферах людської діяльності.

Її основні напрями:

- розробка обчислювальних систем та програмного забезпечення;
- теорія інформації, що вивчає процеси, пов'язані з передачею, прийомом, перетворенням та зберіганням інформації;
- методи штучного інтелекту, дозволяють створювати програми на вирішення завдань, потребують певних інтелектуальних зусиль під час виконання їх людиною (логічний висновок, навчання, розуміння мови, візуальне сприйняття, гри та інших.);
- системний аналіз, що полягає в аналізі призначення проектованої системи та у встановленні вимог, яким вона повинна відповідати;
- методи машинної графіки, анімації, засоби мультимедіа;
- засоби телекомунікації, зокрема, глобальні комп'ютерні мережі, що об'єднують все людство у єдине інформаційне співтовариство;
- різноманітні додатки, що охоплюють виробництво, науку, освіту, медицину, торгівлю, сільське господарство та всі інші види господарської та громадської діяльності

Інформатику зазвичай представляють що складається із двох частин:

- технічні засоби;
- програмні засоби.

Технічні засоби, тобто апаратури комп'ютерів. Програмні засоби, які підкреслюють рівнозначність програмного забезпечення та самої машини та водночас підкреслюють здатність програмного забезпечення модифікуватися, пристосовуватися, розвиватися.

Програмне забезпечення— це сукупність усіх програм, що використовуються комп'ютерами, а також уся сфера діяльності щодо їх створення та застосування.

Яким є інформація?

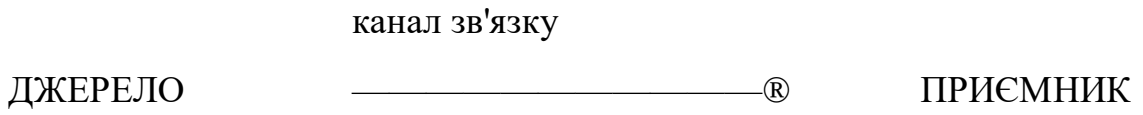
Інформація може існувати у найрізноманітніших формах:

- як текстів, малюнків, креслень, фотографій;
- у вигляді світлових чи звукових сигналів;

- у вигляді радіохвиль;
- у вигляді електричних та нервових імпульсів;
- у вигляді магнітних записів;
- у вигляді жестів та міміки;
- у вигляді запахів та смакових відчуттів;
- як хромосом, з яких передаються у спадок ознаки та властивості організмів тощо.

Як передається інформація?

Інформація передається у вигляді повідомлень від деякого джерела інформації до її приймача за допомогою каналу зв'язку між ними. Джерело посилає передане повідомлення, яке кодується в сигнал, що передається. Цей сигнал надсилається каналом зв'язку. В результаті в приймачі з'являється сигнал, який декодується і стає прийнятим повідомленням.



Приклад:

Жива істота своїми органами почуттів(око, вухо, шкіра, мова тощо.) приймає інформацію із зовнішнього світу, переробляє їх у певну послідовність нервових імпульсів, передає імпульси по нервовим волокнам, зберігає у пам'яті вигляді стану нейронних структур мозку, відтворює як звукових сигналів, рухів тощо, використовує у процесі своєї життєдіяльності.

Як вимірюється кількість інформації?

Як одиниця інформації домовилися прийняти один біт.

*Біт*теоретично інформації — кількість інформації, необхідне

розрізнення двох рівноймовірних повідомлень.

А в обчислювальній техніці бітом називають найменшу "порцію" пам'яті, необхідну для зберігання одного з двох знаків "0" та "1", що використовуються для внутрішньомашинного представлення даних та команд.

Біт - надто дрібна одиниця виміру. Насправді частіше застосовується більша одиниця — байт, рівна восьми бітам. Саме вісім бітів потрібно закодувати будь-який з 256 символів алфавіту клавіатури комп'ютера (256=28).

Широко використовуються також ще більші похідні одиниці інформації:

- 1 Кілобайт (Кбайт) = 1024 байт = 210 байт,
- 1 мегабайт (Мбайт) = 1024 Кбайт = 220 байт,
- 1 гігабайт (Гбайт) = 1024 Мбайт = 230 байт.

Останнім часом у зв'язку зі збільшенням обсягів оброблюваної інформації входять у вжиток такі похідні одиниці, як:

- 1 Терабайт (Тбайт) = 1024 Гбайт = 240 байт,
- 1 Петабайт (Пбайт) = 1024 Тбайт = 250 байт.

За одиницю інформації можна було б вибрати кількість інформації, необхідне розрізнення, наприклад, десяти рівноймовірних повідомлень. Це буде не двійкова (біт), а десяткова (дит) одиниця інформації.

Що робити з інформацією?

Інформацію можна:

- створювати;
- передавати;
- сприймати;
- використовувати;
- запам'ятовувати;
- приймати;
- копіювати;
- формалізувати;
- поширювати;
- перетворювати;
- комбінувати;
- обробляти;
- ділити на частини;
- спрощувати;
- збирати;
- зберігати;
- Шукати;
- вимірювати;
- руйнувати;
- та ін.

Усі ці процеси, пов'язані з певними операціями над інформацією, називаються інформаційними процесами.

Які властивості має інформація?

Властивості інформації:

- достовірність;
- повнота;
- цінність;
- своєчасність;
- зрозумілість;
- доступність;
- стислість;
- та ін.

Методи представлення інформації в ЕОМ

З практичної точки зору інформація подається у вигляді інформаційних повідомлень, що передаються від джерела до приймача каналами зв'язку. При цьому важливо оцінити кількість інформації, що передається або одержується - іншими словами, виміряти її.

У повсякденному житті інформація оцінюється за інтересом, виявленим до неї, за рівнем її корисності. З цього погляду, що цікавіше відомості, то більше інформації вони містять. Невипадково найзмістовніші телепередачі називають інформаційними програмами. Проте в різних людей інтереси значно різняться, у життєвому сенсі цінність інформації залежить від людського сприйняття, тобто. суб'єктивного фактора, який оцінює інформацію щодо рівня її розуміння та інтересу до неї. Спираючись на таке розуміння інформації, не можна точно визначити її кількість.

Розвиток засобів обробки інформації вимагало введення її кількісної міри, що не залежить від суб'єктивного фактора.

Вона ґрунтується на поданні інформації у вигляді послідовності шести символів (знаків); вважається, кожен новий символ збільшує кількість інформації. Ця кількість вимірюється шляхом порівняння з деяким еталоном, подібно до того, як довжина шляху порівнюється з еталоном відстані —

метром.

Для створення такого еталона в інформатиці використовується двох символьний алфавіт, що складається з цифр 0 і 1. Як еталонна вибирається послідовність, що складається з одного символу цього алфавіту Кількість інформації, що міститься в еталонній послідовності, приймають за одиницю, звану 1 біт (англ. binary degit – двійковий розряд). Таким чином, 1 біт - це одиниця двійкової інформації, що міститься в повідомленні типу 0 або 1. Маючи зразок, легко оцінити кількість інформації, що міститься в будь-якому повідомленні. І тому нею записують у тому ж двійковому алфавіті, як і еталонну послідовність. Тоді кількість інформації у цьому повідомленні вважають рівною кількості двійкових символів, що становлять нею, тобто довжині символьної послідовності.

Контрольні запитання:

1. Що таке мова програмування?
2. Що таке транслятор?
3. Чим відрізняється компіляція від інтерпретації?
4. Поясніть терміни „мова низького рівня” й „мова високого рівня”, чим вони відрізняються?.
5. Розкажіть про покоління мов програмування.
6. Які мови програмування активно використовуються сьогодні?
7. Укажіть основні компоненти, які необхідні для створення програми.
8. Що таке інтегроване середовище програмування?
9. Дати визначення поняттю „візуальне програмування”.
10. Охарактеризуйте основні напрямки розвитку мов програмування.
11. Розкажіть про основні системи програмування.

12. Перелічіть складові інтегрованого середовища розробки програм.
13. Основні напрямки «Інформатики». Як передається, вимірюється кількість інформації та які властивості вона має?
14. Методи представлення інформації.

Розділ IV

АРИФМЕТИЧНІ ОСНОВИ КОМП'ЮТЕРІВ

Системи числення

Що таке система числення?

Система числення - це спосіб запису чисел за допомогою заданого набору

спеціальних знаків (цифр).

Існують позиційні та непозиційні системи числення.

У непозиційних системах вага цифри(Тобто той внесок, який вона вносить у значення числа) не залежить від її позиції у записі числа. Так, у

римській системі числення в числі XXXII (тридцять два) вага цифри X у будь-якій позиції дорівнює просто десяти.

У позиційних системахчислення вага кожної цифри змінюється в залежності від її положення (позиції) у послідовності цифр, що зображають число. Наприклад, серед 757,7 перша сімка означає 7 сотень, друга – 7 одиниць, а третя – 7 десятих часток одиниці.

Сама ж запис числа 757,7 означає скорочений запис виразу

700 + 50 + 7 + 0,7 = 7 · 102 + 5 · 101 + 7 · 100 + 7 · 10-1 = 757,7.

Будь-яка позиційна система числення характеризується своєю основою.

Основа позиційної системи числення – це кількість різних знаків або символів, що

використовуються для зображення цифр у даній системі.

За основу системи можна прийняти будь-яке натуральне число – два, три, чотири тощо. Отже, можливо безліч позиційних систем: двійкова, трійкова, четвіркова і т.д. Запис чисел у кожній із систем числення з основою q означає скорочений запис виразу

an-1 qn-1 + an-2 qn-2+ ... + a1 q1 + a0 q0 + a-1 q-1 + ... + am qm,

де ai – цифри системи числення; n і m – число цілих та дробових розрядів, відповідно.

Цілі числа у позіційних системах числення.

Як породжуються цілі числа у позиційних системах числення?

У кожній системі числення цифри впорядковані відповідно до їх значень: 1 більше 0, 2 більше 1 і т.д.

*Просуванням*цифри називають заміну її наступної за величиною.

Просунути цифру 1 означає замінити її на 2, просунути цифру 2 означає замінити на 3 і т.д. Просування старшої цифри (наприклад, цифри 9 у десятковій системі) означає заміну на 0. У двійковій системі, використовує лише дві цифри – 0 і 1, просування 0 означає заміну їх у 1, а просування 1 – заміну їх у 0.

Цілі числа в будь-якій системі числення породжуються за допомогою Правила рахунку:

Для утворення цілого числа, наступного за будь-яким даним цілим числом, потрібно

просунути найправішу цифру числа; якщо будь-яка цифра після просування стала

банкрутом, потрібно просунути цифру, що стоїть ліворуч від неї.

Застосовуючи це правило, запишемо перші десять цілих чисел

- у двійковій системі: 0, 1, 10, 11, 100, 101, 110, 111, 1000, 1001;
- у троїчній системі: 0, 1, 2, 10, 11, 12, 20, 21, 22, 100;
- у п'ятирічній системі: 0, 1, 2, 3, 4, 10, 11, 12, 13, 14;
- вісімковій системі: 0, 1, 2, 3, 4, 5, 6, 7, 10, 11.

Системи числення використовують фахівці спілкування з комп'ютером.

Крім десяткової широко використовуються системи з основою, що є цілим ступенем числа 2, а саме:

- двійкова(використовуються цифри 0, 1);
- вісімкова(використовуються цифри 0, 1, ..., 7);
- шістнадцяткова(Для перших цілих чисел від нуля до дев'яти використовуються цифри 0, 1, ..., 9, а для наступних чисел - від десяти до п'ятнадцяти - як цифри використовуються символи A, B, C, D, E, F).

Корисно запам'ятати запис у цих системах численних перших двох десятків цілих чисел:

З усіх систем числення особливо проста і тому цікава технічної реалізації в комп'ютерах двійкова система числення.

Чому люди користуються десятковою системою, а комп'ютери двійковою?

Люди віддають перевагу десятковій системі, мабуть, тому, що з давніх часів рахували на пальцях, а пальців у людей по десять на руках і ногах. Не завжди і скрізь люди користуються десятковою системою числення. У Китаї, наприклад, тривалий час користувалися п'ятирічною системою числення.

А комп'ютери використовують двійкову систему тому, що вона має низку переваг перед іншими системами:

- для її реалізації потрібні технічні пристрої з двома стійкими станами (є струм — немає струму, намагнічений — не намагнічений тощо), а не, наприклад, із десятьма, — як у десятковій;
- подання інформації за допомогою лише двох станів надійно та завадостійке;
- можливе застосування апарату булевої алгебри для виконання логічних перетворень інформації;
- двійкова арифметика набагато простіша за десяткову.

Недолік двійкової системи — швидке зростання кількості розрядів, необхідні записи чисел.

Система числення – це певний спосіб представлення чисел і відповідні правила дії над числами.

Як ми вже знаємо, люди у своїй практиці використовують позиційну (арабський спосіб) та непозиційну (римський спосіб) системи числення.

Десятична система числення – ця позиційна система числення в якій маємо 10 цифр. Алфавіт системи (число цифр) називається основою системи. Будь-яке число ми записуємо з допомогою цих цифр, причому цифра в числі має так званий «вага» тобто значимість цифри залежить від цього у якому розряді вона.

Наприклад:

Що більше 1 чи 9? (Зрозуміло що 9). Але в числі 13509 важливіше грає роль 1. І це всім зрозуміло так як ми звикли з початкової школи читати це число наступним чином: починаючи зліва цифру множимо на відповідний розряд, в якому вона складається з наступною цифрою, помноженою на свій відповідний розряд і т.д. л., доки не дійдемо до останньої цифри. Якщо в числі зустрінеться цифра «0», то, природно, ніяких дій у цьому розряді не робимо, просто переходимо до наступного розряду. Тобто.

10000 1000 100 10 1-розряди

1 3 5 0 9 = 1*10000 + 3*1000 + 5*100 + 9*1 – розгорнута форма

Виникає питання: а чому розряди саме в десять разів більші за свій «сусід», який знаходиться правіше? А тому, ми маємо 10 цифр і якщо до 13509 додати 7, то в розряді одиниць настає переповнення. 9+7=16, «16» у розряд не міститься, у розряд десяток записуємо 1, а залишок від 16, тобто. 6 залишається в розряді одиниць.
Число 13509 записується в десятковій системі числення єдиним способом!
Давайте аналогічно діяти в іншій системі числення, в якій всього дві цифри «0» і «1». Кожен розряд у числі буде вдвічі більше за свого сусіда який знаходиться правіше, тобто:

16 8 4 2 1 - розряди

1 0 1 1 0

Для того щоб зрозуміти що це за число у зрозумілому для нас кодуванні, тобто в десятковій системі числення, проробимо ті ж дії, які ми робили раніше. Починаємо із старшої лівої цифри. Примножуємо її на той розряд, у якому вона знаходиться і складаємо з наступним відповідним твором. Але оскільки тут множити припадати лише на «1», ми можемо легко скласти величини тих розрядів у яких перебуває «1». І так:

16 8 4 2 1 - розряди

1 0 1 1 0 = 16 + 4 + 2 = 2210.

У позиційній двійковій системі числення основа дорівнює 2, а алфавіт складається з двох цифр. Отже, числа у двійковій системі записуються у

вигляді кінцевої послідовності одиниць та нулів. Розряд у двійковому числі більший за свого сусіда, який перебуває правіше вдвічі.

Оскільки представлення чисел в ЕОМ використовується набір із двох символів (0,1), все комп'ютери працюють у двійковій системі числення, тобто. у системі з основою 2). Тому кожен символ, що вводиться в машину, чи то буква, цифра, знак операції тощо, кодується послідовністю із символів двійкового алфавіту, тобто. перетворюється на числову форму. Вибір двійкової системи пояснюється лише тим, що електронні елементи обчислювальної машини можуть у двох стійких станах. По суті, це звичайні вимикачі, які можуть бути включені або вимкнені. Один із станів вимикача позначають 1, інший – 0. Час перемикання дуже мало, порядку 10 с.

Усі необхідні перетворення даних на комп'ютері виконуються за певними правилами (алгоритмами) автоматично за допомогою встановлених програм.

Так, для запису числа у двійковій системі числення необхідно подати її у вигляді суми ступенів двійки та виписати коефіцієнти такого уявлення, які будуть цифрами двійкового числа.

Правило переведення чиселз десяткової системи в двійкову систему

Правило переведення чиселз десяткової системи в двійкову системуможна сформулювати так: ділимо вихідне число на 2 основу двійкової системи числення. Залишки, що отримуються при цьому, дають цифри числа спочатку останню, потім передостанню і т.д. Останній поділ дає нам другу (останній залишок) та першу цифру числа (останнє приватне).

Продемонструємо цей алгоритм на наступному прикладі:

Подати в двійковій системі число 2710.

Послідовність дій буде такою:

27: 2 = 13 (1 у залишку, остання цифра числа – 1);

13 : 2 = 6 (1 у залишку, передостання цифра числа – 1):

6: 2 = 3 (0 у залишку, третя цифра з кінця – 0);

3.2 = 1 (1 у залишку, друга цифра числа – 1).

Останнє приватне 1 дає першу цифру числа.

Таким чином, розкладання числа 27 за ступенями двійки має вигляд

$$2710 = 1 \cdot 24 + 1 \cdot 23 + 0 \cdot 22 + 1 \cdot 21 + 1 \cdot 20$$

Отже, шукане уявлення числа 27 є 110 112 .

Для запису двійкового числа в десятковій системі числення слід подати його у вигляді суми добутку цифр цього числа на відповідні ступені двійки та виконати арифметичні операції за правилами десяткової системи числення. Результат і буде представляти шукане число.

Використовуємо це правило, наприклад, для перекладу числа 11011012.

Отримаємо

$$1101101 = 1 \cdot 26 + 1 \cdot 25 + 0 \cdot 24 + 1 \cdot 23 + 1 \cdot 22 + 0 \cdot 21 + 1 \cdot 20 = 64 + 32 + 8 + 4 + 1 = 10910 .$$

Нижче представлені вирази деяких десяткових чисел у двійковій системі числення (див. табл.2.1).

В інформатиці крім двійкової використовуються також вісімкова та шістнадцяткова системи числення.

Їх підстави, відповідно 8 (вісімкова) і 16 (шістнадцяткова), є ступенем підстави двійкової системи числення 2.

У вісімковій системі числа записуються за допомогою цифр

$$0, 1, \ldots, 7$$

У шістнадцятковій системі

$$0, 1, \ldots, 9,$$

а для позначення шести інших використовуються літери латинського алфавіту

$$A,B,C,D,E,F$$

У десятковій системі їм відповідають числа 10,11,12,13,14,15.

Для цих систем дуже простий перехід до восьми та шістнадцяткової форми запису чисел від двійкової та назад.

Для представлення двійкового числа у вісімковій формі необхідно розбити його цифри на групи з трьох цифр кожна справа наліво та замінити кожну трійку цифр відповідною вісімковою цифрою.

Наприклад, число

$$10101101011112 = 126578$$

оскільки 1 010 110 101 111

1 2 6 5 7

Зворотний перехід від вісімкового уявлення до двійкового здійснюється заміною кожної вісімкової цифри відповідною до трійки двійкових.

Для представлення двійкового числа у шістнадцятковій формі необхідно розбити його цифри на групи з чотирьох цифр кожна праворуч наліво та замінити кожну чотирірку цифр відповідною шістнадцятковою цифрою.

Наприклад, число

$$101101012 = B516$$

тому що 1011 0101

B 5

Шістнадцяткова форма чисел часто використовується для позначення адрес розміщення даних у пам'яті.ЧІП. 1 B

Десяткова система числення	Двійкова система числення
10000 1000 100 10 1 1 3 5 0 9 = 1*10000 + 3*1000 + 5*100 + 9*1	16 8 4 2 1 1 0 1 1 0 = 16 + 4 + 2 = 2210

Переведіть самостійно: 1110012 в десяткову систему числення.

Тепер переведемо число із десяткової системи до двійкової. Для цього записуватимемо розряди двійкової системи доти, поки розряд не перевищує

дане десяткове число. Н-р: число 78 перевести на двійкову систему числення. Записуємо двійкові розряди праворуч наліво, починаючи з одиниці, отримуємо:

128 64 32 16 8 4 2 1

Розряд 128 не потрібен т.к. у нас всього 78. Тепер починаємо зі старшого розряду проставляти 1. Якщо записати так:

64 32 16 8 4 2 1
1 0 0 0 0 0 0

то ми маємо тільки 64, треба добрати ще 78 - 64 = 14. Проставляємо 1 у найстарший розряд, але не перевищує 14. Отримуємо,
64 32 16 8 4 2 1
1 0 0 1 0 0 0.
Це ми отримали 72, аналогічно добираємо ще 6. У результаті отримуємо:
64 32 16 8 4 2 1
1 0 0 1 1 1 0.
Число 78 в двійковій системі числення можна тільки єдиним чином.
Далі пояснюю інший спосіб переведення десяткового числа у двійкове. Методом розподілу на 2 та збирати залишки від розподілу у зворотному порядку.

Використання в комп'ютерах вісімкової та шістнадцяткової систем числення.

Чому в комп'ютерах використовуються також вісімкова та шістнадцяткова системи числення?

Двійкова система, зручна для комп'ютерів, для людини незручна через її громіздкість та незвичний запис.

Переведення чисел із десяткової системи у двійкову і навпаки виконує машина. Проте, щоб професійно використовувати комп'ютер, слід навчитися розуміти слово машини. Для цього і розроблено вісімкову та шістнадцяткову системи.

Числа в цих системах читаються майже так само легко, як десяткові, вимагають відповідно у три (вісімкова) та в чотири (шістнадцяткове) рази менше розрядів, ніж у двійковій системі (адже числа 8 і 16 – відповідно, третій та четвертий ступеня числа 2) .

Правило переведення вісімкових і шістнадцяткових чисел у двійкову систему числення.

Переведення вісімкових і шістнадцяткових чисел у двійкову систему дуже простий:

досить кожну цифру замінити еквівалентною їй двійковою тріадою (трійкою цифр)

або зошитом (четвіркою цифр).

Наприклад:

Щоб перевести число з двійкової системи у вісімкову або шістнадцяткову, його

потрібно розбити вліво і вправо від коми на тріади (для вісімкової) або зошити

(для шістнадцяткової) і кожну таку групу замінити відповідною вісімковою

(шістнадцятковою) цифрою.

Наприклад,

Правило переведення цілого числа з десяткової системи в іншу позиційну систему числення.

Як перевести ціле число з десяткової системи в іншу позиційну систему числення?

При переведенні цілого десяткового числа в систему з підставою q

його необхідно

послідовно ділити на q доти, доки залишиться залишок, менший чи рівний q–1.

Число в системі з підставою q записується як послідовність залишків від розподілу,

записаних у зворотному порядку, починаючи з останнього.

Приклад: Перевести число 75 з десяткової системи в двійкову, вісімкову та шістнадцяткову:

Відповідь:7510 = 1001 0112 = 1138 = 4B16.

Правило переведення десяткової дробі в будь-яку іншу позиційну систему числення.

Як перевести правильну десяткову дроб в будь-яку іншу позиційну систему числення?

При перекладі правильної десяткової дроби в систему числення з основою q

необхідно спочатку сам дріб, а потім дробові частини всіх наступних творів

послідовно множити на q, відокремлюючи після кожного множення цілу частину

твору. Число в новій системі числення записується як послідовність отриманих

цілих частин твору.

Арифметичні операції у позіційних системах числення.

Множення виробляється до тих пір, поки дрібна частина твору не стане рівною нулю. Це означає, що зроблений точний переклад. У протилежному випадку переклад здійснюється до заданої точності. Досить тієї кількості цифр в результаті, яке поміститься в комірку.

Приклад:Перевести число 0,35 з десяткової системи в двійкову, вісімкову та шістнадцяткову:

Відповідь:0,3510 = 0,010112 = 0,2638 = 0,5916.

Як перевести число з двійкової (вісімкової, шістнадцятковій) системи в десяткову?

При переведенні числа з двійкової (вісімковій, шістнадцятковій) системи в десяткову

треба це число подати у вигляді суми ступенів його системи числення.

Приклади:

Розглянемо лише ті системи числення, що застосовуються в комп'ютерах – десяткову, двійкову, вісімкову та шістнадцяткову.

Як виробляються арифметичні операції у позиційних системах числення?

Розглянемо основні арифметичні операції: складання, віднімання, множення та розподіл. Правила виконання цих операцій у десятковій системі добре відомі - це додавання, віднімання, множення стовпчиком і розподіл кутом. Ці правила застосовні і для всіх інших позиційних систем числення. Тільки таблицями складання та множення треба користуватися спеціальними для кожної системи.

- ***Додавання***

Таблиці складання легко скласти, використовуючи Правило Рахунку.

Додавання в шістнадцятковій системі

При додаванні цифри сумуються за розрядами, і якщо при цьому виникає надлишок, він переноситься вліво.

приклад 1.Складемо числа 15 та 6 у різних системах числення.

Шістнадцяткова:F16+616 Відповідь:15+6 = 2110 = 101012 = 258 = 1516.

Перевірка.Перетворимо отримані

суми до десяткового виду:

101012 = 24 + 22 + 20 = 16 +4 +1

= 21, 258 = 2 * 81 + 5 * 80 = 16 + 5

= 21, 1516 = 1 * 161 + 5 * 160 =

16 +5 = 21.

приклад 2.Складемо числа 15, 7 та 3.

Шістнадцяткова:F16+716+316 Відповідь:5+7+3 = 2510 = 110012

= 318 = 1916.

Перевірка:110012 = 24 + 23 + 20 =

16 +8 +1 = 25,318 = 3 * 81 + 1 * 80

= 24 + 1 = 25, 1916 = 1 * 161 + 9 *

160 = 16 +9 = 25.

Приклад 3.Складемо числа 141,5 та 59,75.

Відповідь:141,5 + 59,75 = 201,2510 = 11001001,012 = 311,28 = C9,416

Перевірка.Перетворимо отримані суми до десяткового виду:

11001001,012 = 27 + 26 + 23 + 20 + 2-2 = 201,25 311,28 = 3 * 82 + 1 · 81 + 1 * 80 + 2 * 8-1 = 201, 25 C9,416 = 12 * 161 + 9 * 160 + 4 * 16-1 = 201,25

- ***Віднімання***

Приклад 4.Віднімемо одиницю з чисел 102, 108 та 1016

Приклад 5.Віднімемо одиницю з чисел 1002, 1008 та 10016.

Приклад 6.Віднімемо число 59,75 з числа 201,25.

Відповідь:201,2510 - 59,7510 = 141,510 = 10001101,12 = 215,48 = 8D,816.

Перевірка.Перетворимо отримані різниці до десяткового виду: 10001101,12 = 27 + 23 + 22 + 20 + 2-1 = 141,5; 215,48 = 2 * 82 + 1 * 81 + 5 * 80 + 4 * 8-1 = 141,5; 8D,816 = 8 * 161 + D * 160 + 8 * 16-1 = 141,5.

- ***Розмноження***

Виконуючи множення багатозначних чисел у різних позиційних системах числення, можна використовувати звичайний алгоритм перемноження чисел у стовпчик, але при цьому результати перемноження та складання однозначних чисел необхідно запозичувати з відповідних аналізованої системі таблиць множення та додавання.

Зважаючи на надзвичайну простоту таблиці множення в двійковій системі, множення зводиться лише до зрушень множення і додавань.

Приклад 7.Перемножимо числа 5 та 6.

Відповідь:5 * 6 = 3010 = 111102 = 368.

Перевірка.Перетворимо отримані твори до десяткового виду: 111102 = 24 + 23 + 22 + 21 = 30; 368 = 3 · 81 + 6 · 80 = 30.

Приклад 8.Перемножимо числа 115 та 51.

Відповідь:115 * 51 = 586510 = 10110111010012 = 133518.

Перевірка.Перетворимо отримані твори до десяткового виду: 10110111010012 = 212 + 210 + 29 + 27 + 26 + 25 + 23 + 20 = 5865; 133518 = 1 * 84 + 3 * 83 + 3 * 82 + 5 * 81 + 1 * 80 = 5865.

- ***Поділ***

Поділ у будь-якій позиційній системі числення проводиться за тими самими правилами, як і поділ кутом у десятковій системі. У двійковій системі розподіл виконується особливо просто, адже чергова цифра частки може бути лише нулем або одиницею.

Приклад 9.Розділимо число 30 на 6.

Відповідь:30: 6 = 510 = 1012 = 58.

Приклад 10Поділимо число 5865 на 115.

Вісімкова:133518 :1638

Відповідь:5865: 115 = 5110 = 1100112 = 638.

Перевірка.Перетворимо отримані частки до десяткового виду: 1100112 = 25 + 24 + 21 + 20 = 51; 638 = 6 * 81 + 3 * 80 = 51.

Приклад 11.Розділимо число 35 на 14.

Вісімкова:438 : 168

Відповідь:35: 14 = 2,510 = 10,12 = 2,48.

Перевірка.Перетворимо отримані часткові до десяткового виду: 10,12 = 21 + 2 -1 = 2,5; 2,48 = 2 * 80 + 4 * 8-1 = 2,5.

Контрольні запитання:

1. Яким є інформація?
2. Як передається інформаці?
3. Як вимірюється кількість інформації?
4. Що таке система числення?
5. Як породжуються цілі числа у позіційних системах числення?
6. Які системи числення числення використовуют фахівці спілкування з компютером?
7. Чому люди користуються десятковою системою, а комп'ютери двійковою?
8. Як перевести ціле число з десяткової системи в іншу позиційну систему числення?
9. Як перевести правильну десяткову дроб в будь-яку іншу позиційну систему

числення?

10. Як перевести число з двійкової (вісімкової, шістнадцятковій) системи в десяткову?
11. Як виробляються арифметичні операції у позиційних системах числення (додвання, віднімання, розмноження, поділ?

Розділ V

ОСНОВНІ ПРИНЦИПИ РОЗРОБКИ АЛГОРИТМІВ І ПРОГРАМ

Етапи рішення завдання на ЕОМ

Варто виділити такі етапи підготовки та розв'язання завдань за допомогою ЕОМ: постановка завдання, вибір методу, розробка алгоритму, складання програми, введення її в пам'ять ЕОМ, налагодження програми, її тестування та підготовка документації. Не можна ігнорувати жодного з цих етапів.

Рішення будь-якого завдання на ЕОМ складається з декількох

етапів, серед яких варто виділити основні:

1. постановка завдання;
2. формалізація (математична постановка задачі);
3. вибір (або розробка) методу рішення;
4. розробка алгоритму (алгоритмізація);
5. складання програми (програмування), введення її в пам'ять ЕОМ ;
6. налагодження програми, її тестування;
7. обчислення й обробка результатів.

Послідовне виконання вказаних етапів становить повний цикл розробки, налагодження й обчислення програми. Наведений розподіл є умовним, але не варто ігнорувати жодного з цих етапів. Розглянемо найбільш загальні й необхідні етапи. Разом із зазначеними користувач ЕОМ у процесі рішення завдання може виконувати також такі етапи, як вибір мови програмування, опис структури даних, оптимізація програми, тестування, документування й ін.

Постановка завдання.

При постановці завдання першорядну увагу треба приділити з'ясуванню кінцевої мети й виробленню загального підходу до досліджуваної проблеми; з'ясуванню, чи існує рішення поставленого завдання й чи єдине воно; вивченню загальних властивостей розглянутого фізичного явища або об'єкта, аналізу можливостей конкретної ЕОМ і даної системи програмування. На цьому етапі потрібне глибоке розуміння сенсу поставленого завдання. Правильно сформулювати завдання іноді не менш складно, ніж її вирішити.

Формалізація.

Формалізація, як правило, полягає у побудові математичної моделі розглянутого явища, коли в результаті аналізу сутності завдання визначаються обсяг і специфіка вихідних даних, вводиться система умовних позначок, встановлюється приналежність

розв'язуваного завдання до одного з відомих класів завдань і вибирається відповідний математичний апарат. При цьому потрібно вміти сформулювати мовою математики конкретні завдання фізики, механіки, економіки, технології й т. п. Для успішного подолання цього етапу потрібні не тільки солідні відомості з відповідної предметної галузі, але й гарне знання обчислювальної математики, тобто тих методів, які можуть бути використані при рішенні завдання на комп'ютері.

Повна постановка багатьох складних завдань нездійсненна засобами обчислювальної техніки. Тому ці завдання потрібно спрощувати. Грамотне спрощення завдання неможливе без гарного подання про те, які фактори й параметри найбільш важливі для досліджуваного завдання, а які - менш істотні. При цьому дуже важливо знати, яка з можливих розрахункових схем може привести до спрощення обчислювального характеру, обумовлених вибором обчислювального методу. Якщо наявних засобів недостатньо, тоді необхідно розробити новий підхід, нові методи дослідження.

Вибір методу рішення.

Після того як визначено математичне формулювання завдання, треба вибрати метод його рішення. Загалом, застосування будь-якого методу приводить до побудови ряду формул і формулювання правил, що визначають зв'язки між цими формулами. Все це розбивається на окремі дії так, щоб обчислювальний процес міг бути виконаний машиною. При виборі методу треба враховувати, *по-перше,* складність формул і співвідношень, пов'язаних з тим або іншим чисельним методом, *по-друге,* необхідну точність обчислень і характеристики самого методу. На вибір методу рішення великий вплив мають смаки й знання самого користувача.

Цей етап - найважливіший у процесі рішення задачі. З ним

зв'язані численні невдачі, які є результатом легковажного підходу до помилок обчислень. При рішенні завдання на ЕОМ необхідно пам'ятати, що будь-який одержуваний результат є наближеним! Якщо відомо алгоритм точного рішення, то крім випадкових помилок (збоїв у роботі ЕОМ), можливі помилки, пов'язані з обмеженою точністю подання чисел в ЕОМ. При обчисленнях, що полягають у знаходженні результату із заданим ступенем точності, виникає додаткова погрішність, яку, якщо можливо, оцінюють на даному етапі (до виходу безпосередньо на ЕОМ). Ця погрішність визначається обраним чисельним методом рішення завдання.

Розробка алгоритму.

Даний етап полягає в розкладанні обчислювального процесу на можливі складові частини, установленні порядку їхнього проходження, описі змісту кожної такої частини в тій або іншій формі й наступній перевірці, що повинна показати, чи забезпечується реалізація обраного методу. У більшості випадків не вдається відразу одержати задовільний результат, тому складання алгоритму проводиться методом «спроб і усунення помилок» і для одержання остаточного варіанту потрібно кілька кроків корекції й аналізу.

Як правило, у процесі розробки алгоритм проходить кілька етапів деталізації. Спочатку складається укрупнена схема алгоритму, у якій відбиваються найбільш важливі й істотні зв'язки між досліджуваними процесами (або частинами процесу). На наступних етапах розкриваються (деталізуються) виділені на попередніх етапах частини обчислювального процесу, що мають деяке самостійне значення. Крім того, на кожному етапі деталізації виконується багаторазова перевірка й виправлення (відпрацьовування) схеми алгоритму. Подібний підхід дозволяє уникнути можливих помилкових рішень.

Орієнтуючись на великоблочну структуру алгоритму, можна швидше й простіше розробити кілька різних його варіантів, провести їхній аналіз, оцінку й вибрати найкращий (оптимальний).

Ефект поетапної деталізації алгоритму багато в чому залежить від того, як здійснюється його структуризація: розчленовування алгоритмічного процесу на складові частини, що повинне визначатися не сваволею користувача (програміста), а внутрішньою логікою самого процесу. Кожний елемент великоблочної схеми алгоритму повинен бути максимально самостійним і логічно завершеним у такому ступені, щоб подальшу його деталізацію можна було виконувати незалежно від деталізації інших елементів. Це спрощує процес проектування алгоритму й дозволяє здійснювати його розробку вроздріб одночасно кількома виконавцями.

У процесі розробки алгоритму можуть використовуватися різні способи його опису, що відрізняються за простотою, наочністю, компактністю, ступенем формалізації, орієнтацією на машинну реалізацію й іншими показниками. У практиці програмування найбільшого поширення набули:

1. словесний запис алгоритмів;
2. схеми алгоритмів;
3. псевдокод (формальні алгоритмічні мови);
4. структурограми (діаграми Нассі - Шнейдермана) . Розробка алгоритмів є в значній мірі творчим,

евристичним процесом і, як правило, вимагає великої ерудиції, винахідливості, нестандартних і нетрадиційних підходів до рішення завдання.

Складання програми.

Подання алгоритму у формі, що допускає введення в машину, переклад на машинну мову є завданнями етапу складання програми (програмування). Тобто розроблений алгоритм завдання необхідно

викласти мовою, що буде зрозуміла ЕОМ безпосередньо або після попереднього машинного перекладу. Від вибору мови програмування залежить процес налагодження програми, під час якого програма набуває остаточного робочого вигляду.

Налагодження програми.

Складання програми являє собою трудомісткий процес, що вимагає від виконавця напруженої уваги. Практика показує, що в обчисленнях варто уникати поспішності й дотримуватися золотого правила: «краще менше, та краще». Але на попередніх етапах стільки можливостей припуститися помилки, і як би ми ретельно не діяли, спочатку складена програма звичайно містить помилки. Машина або не може дати відповіді, або наводить неправильне рішення.

Налагодження починається з того, що програма, акуратно записана на бланку, перевіряється безпосередньо особою, що здійснила підготовку й програмування завдання. З'ясовується правильність написання програми, виявляються змістовні й синтаксичні помилки й т.п. Потім програма вводиться у пам'ять ЕОМ і помилки, що залишилися непоміченими, виявляються вже безпосередньо за допомогою машини.

Досвідчений користувач ЕОМ знає, що необхідний діючий контроль над процесом обчислень, який дозволяє вчасно виявляти й запобігати помилки. Для цього використовуються різного роду інтуїтивні міркування, правдоподібні міркування й контрольні формули. Користувач - початківець часто вважає налагодження зайвим, а одержання контрольних точок - неприємною додатковою роботою. Однак дуже скоро він переконується, що пошук пропущеної помилки вимагає значно більшого часу, ніж час, витрачений на контроль.

Гарантією правильності рішення, наприклад, може служити:

а. перевірка виконання умов завдання (наприклад, для алгебраїчного рівняння знайдені коріння підставляються у вихідне рівняння й перевіряються розходження лівої й правої частин);

б. якісний аналіз завдання;

в. перерахування (по можливості іншим методом).

Для деяких складних за структурою програм процес налагодження може зажадати значно більше машинного часу, ніж саме рішення на ЕОМ, тому що погано сплановані процеси алгоритмізації, програмування і налагодження приводять до помилок, які можуть бути виявлені лише після багаторазових перевірок.

Обчислення й обробка результатів.

Тільки після того як з'явиться повна впевненість, що програма забезпечує одержання правильних результатів, можна приступати безпосередньо до розрахунків по програмі. Безпосереднє рішення завдання на ЕОМ не вимагає обов'язкової участі користувача. Ця робота виконується оператором ЕОМ. Порядок роботи на машині при рішенні завдання докладно описується в інструкції до програми. Після завершення розрахунків наступає етап використання результатів обчислень у практичній діяльності або, як говорять, етап впровадження результатів. Інтерпретація результатів обчислень знову належить до тієї предметної галузі знань, звідки виникло завдання.

Алгоритми

Алгоритм – опис послідовності дій (план), суворе виконання яких призводить до вирішення поставленої задачі за кінець кількох кроків.

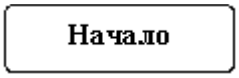
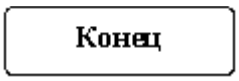
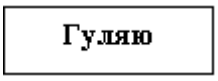
Властивості алгоритмів:

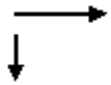
1. Дискретність (алгоритм повинен складатися з конкретних дій, що випливають у порядку);2. Детермінованість (будь-яка дія має бути строго і недвозначно визначено в кожному випадку);3. Кінцевість (кожна дія та алгоритм в цілому повинні мати можливість завершення);
4. Масовість (один і той самий алгоритм можна використовувати з різними вихідними даними);5. Результативність (відсутність помилок, алгоритм повинен призводити до правильного результату для всіх допустимих вхідних значень).

Види алгоритмів:

1. Лінійний алгоритм (опис дій, що виконуються одноразово в заданому порядку);2. Циклічний алгоритм (опис дій, які повинні повторюватися вказане число разів або поки не виконано завдання);3. Розгалужувальний алгоритм (алгоритм, у якому залежно від умови виконується або одна, або інша послідовність дій) 4. Допоміжний алгоритм (алгоритм, який можна використовувати в інших алгоритмах, вказавши лише його ім'я).

Для наочного уявлення алгоритму широко використовується графічна форма - блок-схема, що складається зі стандартних графічних об'єктів.

Вид стандартного графічного об'єкта	Призначення
	Початок алгоритму
	Кінець алгоритму
	Дія, що виконується, записується всередині прямокутника
	Умова виконання дій записується всередині ромбу

	Лічильник у повторів
	Послідовність виконання дій.

Стадії створення алгоритму:

1. Алгоритм має бути представлений у формі, зрозумілій людині, яка його розробляє.2. Алгоритм має бути представлений у формі, зрозумілій тому об'єкту (у тому числі й людині), який виконуватиме описані в алгоритмі дії.

Об'єкт, який виконуватиме алгоритм, зазвичай називають виконавцем. Виконавець – об'єкт, який виконує алгоритм. Ідеальними виконавцями є машини, роботи, комп'ютери. Комп'ютер – це автоматичний виконавець алгоритмів. Алгоритм, записаний на "зрозумілому" комп'ютері мовою програмування, називається програмою.

. Які поняття використовують алгоритмічні мови?

Кожне поняття алгоритмічного мови має на увазі деяку синтаксичну одиницю (конструкцію) і зумовлені нею властивості програмних об'єктів чи процесу обробки даних.

Поняття мови визначається у взаємодії синтаксичних та семантичних правил. Синтаксичні правила показують, як утворюється дане поняття

з інших понять та букв алфавіту, а семантичні правила визначають властивості даного поняття

Основними поняттями в алгоритмічних мовах

Основними поняттями в алгоритмічних мовах є такі.

Імена(ідентифікатори) - вживаються для позначення об'єктів програми (перемінних, масивів, функцій та ін.).

Операції. Типи операцій:

- арифметичніоперації + , - , * , / та ін. ;
- логічніоперації та, або, не;
- операції відносини < , > , <=, >= , = , <> ;
- операція зчіпки (інакше, "приєднання", "конкатенації") символьних значень друзів з одним з утворенням одного довгого рядка; зображується знаком "+".

Дані- величини, що обробляються програмою. Є три основних види даних: константи, перемінні і масиви.

- ***Константи***
- Константи— це дані, які зафіксовані у тексті програми та не змінюються у процесі її виконання.

Приклади констант:

- числові7.5, 12;
- логічнітак (істина), ні (брехня);
- символьні"A", "+";
- літерні"abcde", "інформатика", "" (порожній рядок).

- Перемінніпозначаються іменами і можуть змінювати свої значення в ході виконання програми. Перемінні бувають цілі, речові, логічні, символьні та літерні.
- Масиви - послідовності однотипних елементів, число яких фіксоване і яким надано одне ім'я.Положення елемента в масиві однозначно визначається його індексами (одним, у разі одновимірного масиву, або кількома, якщо багатомірний масив). Іноді масиви називають таблицями.

Вирази — призначені для виконання необхідних обчисленьскладаються з констант, змінних, покажчиків функцій (наприклад, exp(x)), об'єднаних знаками операцій.

Вирази записуються як лінійних послідовностей символів (без підрядкових і надрядкових символів, " багатоповерхових " дробів тощо.), що

дозволяє вводити в комп'ютер, послідовно натискаючи на відповідні клавіші клавіатури.

Розрізняють вирази арифметичні, логічні та рядкові.

- Арифметичні висловлювання служать визначення одного числового значення. Наприклад, (1+sin(x))/2. Значення цього виразу при x=0 дорівнює 0.5, а за x=p/2 - одиниці.
- Логічні висловлювання описують деякі умови, які можуть задовольнятися чи не задовольнятися. Таким чином, логічне вираження може набувати лише двох значень - "істина" або "брехня" (так чи ні). Розглянемо як приклад логічний вираз x*x + y*y < r*r , що визначає приналежність точки з координатами (x,y) внутрішньої області кола радіусом rc центром на початку координат. При x=1, y=1, r=2 значення цього виразу - "істина", а при x=2, y=2, r=1 - "брехня".
- Значення рядкових (літерних) виразів — тексти. Вони можуть входити літерні константи, літерні змінні і літерні функції, розділені знаком операції зчіпки. Наприклад, A + означає приєднання рядка B до кінця рядка A. Якщо A = "кущ ", а B = "зелений", то значення виразу A + B є "кущ зелений".

Оператори(Команди).

Оператор - це найбільше і змістовне поняття мови: кожен оператор є закінченою фразою мови і визначає певний цілком закінчений етап обробки даних. До складу операторів входять:

- ключові слова;
- дані;
- вираження і т.д.

Оператори поділяються на виконувані та невиконані. Невиконані оператори призначені для опису даних і структури програми, а виконувані - для виконання різних дій (наприклад, оператор присвоєння, оператори введення і виводу, умовний оператор, оператори циклу,

оператор процедури і .).

Стандартна функція алгоритмічної мови

Що таке стандартна функція?

При вирішенні різних завдань за допомогою комп'ютера буває необхідно обчислити логарифм або модуль числа, синус кута і т.д.

Обчислення функцій, що часто вживаються, здійснюються за допомогою підпрограм, званих стандартними функціями, які заздалегідь запрограмовані і вбудовані в транслятор мови.

- ***Таблиця стандартних функцій алгоритмічної мови***

SIN(X)	синус X	sinX, аргумент в радіанах
COS(X)	обчислення косинуса X	*cosX,* аргумент в радіанах
TAN(X)	тангенс X	tgX, аргумент в радіанах
ATN(X)	арктангенс X	ArctgX, -□/2<X<□/2,
ABS(X)	модуль X	|X|
SQR(X)	квадратний корінь X	□X, X>0
EXP(X)	експонента X	e^X
LOG(X)	натуральний логарифм X	lnX, X>0
CDBL(X)	перетворення в число подвійної точності	
CINT(X)	округлення до цілого	CINT(15.5)=16 CINT(-6.7)=-7
CLNG(X)	(довгого цілого) значення	CINT(-6.2)=-6
CSNG(X)	перетворення в число простої точності	
FIX(X)	усікання до цілого (відкидання цілої частини)	FIX(-5.6)=-5 FIX(24.07)=24.00
INT(X)	знаходження найбільшого цілого, що не перевищує X	INT(15.5)=15 INT(-6.2)=-7 INT(-6.7)=-7
RND(X)	генерація псевдовипадкових чисел від 0 до 1	
SGN(X)	знак числа X:	SGN(X) = -1, якщо X < 0 SGN(0) = 0, якщо X = 0 SGN(X) = +1, якщо X > 0

Як аргументи функцій можна використовувати константи, змінні та вирази. Наприклад:

sin(3.05) sin(x) sin(2*y+t/2) sin((exp(x)+1)**2)

min(a, 5) min(a, b) min(a+b, a*b) min(min(a,b),min(c,d))

Кожна мова програмування має власний набір стандартних функцій.

Числові функції символьних аргументів

Значеннями цих функцій є числа, а аргументами - символьні вирази або функції.

ASC, CVI, CVS, CVD, INSTR, LEN, VAL і ін.

Символьні функції

Значеннями цих функцій є ланцюжки символів. CHR$,
LEFT$, RIGHT$, MID$, STRING$, LTRIM$ і ін.

Арифметичні вирази

Як записуються арифметичні вирази?

Арифметичні вирази відповідають загальноприйнятим алгебраїчним виразам. До них можуть входити константи, змінні, функції, з'єднані знаками арифметичних операцій. Результатом арифметичного виразу є число. Число або змінна також вважаються арифметичним виразом. Для позначення арифметичних операцій використовуються знаки

Арифметичні вирази записуються за такими правилами:

- Не можна опускати знак множення між співмножниками та ставити поруч два знаки операцій.
- Індекси елементів масивів записуються у квадратних круглих (Basic) дужках.
- Для позначення змінних застосовуються літери латинського алфавіту.
- Операції виконуються у порядку старшинства: спочатку обчислення функцій, потім зведення в ступінь, потім множення та розподіл і в останню чергу - складання та віднімання.

- Операції одного старшинства виконуються зліва направо. Наприклад, a/b*c відповідає a/b*c. Однак, у шкільному АЯ є один виняток із цього правила: операції зведення у ступінь виконуються праворуч наліво. Так, вираз 2**(3**2) у шкільному АЯ обчислюється як 2**(3**2) = 512. У мові QBasic аналогічний вираз 2^3^2 обчислюється як (2^3)^2 = 64 А в мові Pascal взагалі не передбачена операція зведення в ступінь, у Pascal x^y записується як exp(y*ln(x)), а x^y^z як exp(exp(z*ln(y))*ln (x)).
- *Приклади запису арифметичних виразів*

Операція \ означає ділення націло (дробова частина відкидається)

Приклад: 17\2=8.

Операція MOD означає обчислення залишку (ділення помодулю)

Приклад: 25MOD8=1.

Якщо операнди цих операцій дійсні, то вони попередньо округляються.

Приклади:

$$\frac{ab}{} \quad A*$$

$$B/Cc$$

$$\frac{x \quad 2}{} \quad (X \quad 2)$$

$$/(C \quad D)c \quad d$$

$$\frac{a \quad d}{}$$

$$2x \quad (A \quad SQR(D))/(2*X)$$

$$\sin^2 x \quad \cos x^2 \quad SIN(X)\text{^}2 \quad COS(X\text{^}2)$$

$$3a^2 \quad \underline{b} \quad \frac{7}{4 \quad 1} \quad 3*A\text{^}2 \quad B/4 \quad 7/(1 \quad A)$$

$y_j \; y_j \square 1 \qquad \square \quad Y(J) * Y(J \; \square \; 1) / SQR(J)$

Пріоритет виконання операцій

Всі операції в арифметичному виразі виконуються в певній послідовності: порядок їхнього виконання задається правилами пріоритету

1) обчислення числових функцій;
2) унарний мінус;
3) піднесення до степеня;
4) множення / ділення;
5) ділення націло;
6) обчислення залишку;
7) додавання / віднімання.

Послідовність операцій одного пріоритету виконується зліва направо. Для зміни цього порядку можна використати круглі дужки. Вирази , що знаходяться в дужках, обчислюється в першу чергу.

Приклад: A =8B =4 C =2

A+B/C^2 =9 (A+B)/C^2 - =3
(A+B/C)^2 =100 A+(B/C)^2 - =12

Якщо у виразі кілька операцій мають однаковий пріоритет, то вони виконуються один по одному зліва направо. Виключенням є піднесення до степеня

$X^{Y}Z$ =X^(Y^Z).

Для обчислення кореня довільного степеня використовується еквівалентний вираз

Приклад: =COS(X)^(1/3).

Логарифм за довільною основою обчислюється за формулою

$\log_a b = \frac{\ln b}{\ln a}$.

Приклад: $\log_{10}(x+1)$ =LOG(X+1)/LOG(10).

Зважаючи на те, що LOG(10)=2.3..., а 1/ LOG(10)=0.434, можна записати

$$\log_{10}(x+1) = 0.434 * \log(x+1).$$

Не можна ставити два знаки арифметичних операцій підряд – треба використовувати дужки, наприклад, a^{-b} =>A/(-B).

Від'ємні значення підносити до дробового степіня забороняється, тому що $x^a = e^{a \ln x}$.

У зв'язку з наближеним поданням дійсних чисел в ПК рівність X/Y*Y=X не виконується.

Відзначимо випадок так званих особливих ситуацій при обчисленні арифметичних виразів:

- ділення на нуль - видається відповідне повідомлення; обчислення тривають, а в результаті виходить 1.701412E+38 машинна нескінченність;
- переповнення - видається відповідне повідомлення, і виконання програми припиняється.

Логічні вирази

Логічні вирази складаються з логічних операцій і логічних відношень. В них присутні два операнди, які QBasic розглядає як шістнадцяткові бінарні ланцюжки, над якими побітно зліва направо виконуються дії за допомогою таких операторів:

NOT	заперечення (НІ)	*XOR*	АБО, що виключає
AND	кон'юнкція (І)	*EQV*	еквівалентність
OR	диз'юнкція (АБО)	*IMP*	імплікація

Всі вони повертають значення "істина" (не-нуль) або "хибність" (нуль), що використовуються при ухваленні рішення про подальший хід обчислювального процесу.

Приклад: 63 AND 16=16 4 OR 2=6

111111 AND 10000=010000 100 OR 010=110

Порядок виконання логічних операцій задається пріоритетом (NOT, AND, OR) і круглими дужками.

Логічні відношення

Операції відношення порівнюють два числових або символьних значення і якщо умова виконується, то результат -1, інакше - 0.

Приклад: b2-4ac>0 B^2-4*A*C>0 i№j I<>J рядок a =рядку b A$=B$

46=41 - 0 10<8 - 0 15<=20 - 1 15<>20 - 1 2>1 -1 3>=2 – 1

Якщо порівняння робиться над числами, то припустиме порівняння цілого і дійсного типів.

Символьні величини можна порівнювати тільки з символьними, при цьому порівняння здійснюється посимвольно зліва направо із урахуванням кінцевих пропусків. Більш короткий рядок вважається меншим. Порівняння засноване на відносних значеннях їх шістнадцяткових кодів. При розбіжності символів більшим вважається рядок, що містить символ з більшим кодом.

Приклад: "AVZ">"AGN"; код"V"=086; код"G"=071.

Якщо логічні вирази містять логічні відношення і логічні операції, то спочатку виконуються логічні відношення, а потім логічні операції відповідно до пріоритету.

Приклад: -4<= X <=4 (-4<=X) AND (X<=4)

X=0 або X=1 (X=0) OR (X=1)

Символьні вирази

Складаються із символьних констант, змінних, функцій або будь-яких їхніх комбінацій, розділених знаками логічних відношень або спеціальних символьних операцій.

Спеціальна символьна операція називається **конкатенація.** Вона позначає об'єднання зазначених значень і позначається символом " + ".

Приклад:

A$="студент" B$=" гр. 5-І-В", тоді A$+B$="студент гр. 5-І-В"

Довжина рядка, що є результатом, не повинна перевищувати 255 символів.

Типові помилки у записі виразів:

5x+1 Пропущений знак множення між 5 і x

a+sin x Аргумент x функції sin x не укладений у дужки

((a+b)/c**3

Як записуються логічні вирази?

У записі логічних виразів крімарифметичних операцій складання, віднімання, множення, поділу і зведення в ступінь використовуються операції відносини < (менше), <= (менше або одно), > (більше), >= (більше або одно), = (рівно), <> (не дорівнює), а також логічні операції та, або, не.

• *Приклади запису логічних виразів, істинних під час зазначених умов.*

Умова	Запис шкільною алгоритмічною мовою
Дробна частина речовинного числа a дорівнює нулю	int(a) = 0
Ціле число a - парне	mod(a,2) = 0
Ціле число a - непарне	mod(a,2) = 1
Ціло число k кратно семи	mod(a,7) = 0
Кожне з чисел a,b позитивно	(a>0) та (b>0)
Тільки одне з чисел a,b позитивно	((a>0) та (b<=0)) або ((a<=0) та (b>0))
Хоча б одне із чисел a,b,c є негативним	(a<0) або (b<0) або (c<0)
Число x задовольняє умові a<x<b	(x>a) та (x<b)

Число x має значення у проміжку [1,3]	(x>=1) та (x<=3)
Цілі числа a і b мають однакову парність	((mod(a,2)=0) і (mod(b,2)=0) або ((mod(a,2)=1) і (mod(b,2)=1))
Крапка з координатами (x,y) лежить у колі радіуса r із центром у точці (a,b)	(xa)**2+(yb)**2<r*r
Рівняння ax^2+bx+c=0 не має дійсних коренів	b*b-4*a*c<0
Крапка (x, y) належить першому чи третьому квадранту	((x>0) та (y>0)) або ((x<0) та (y>0))
Крапка (x,y) належить зовнішності одиничного кола з центром на початку координат або його другої чверті	(x*x+y*y>1) або ((x*x+y*y<=1) та (x<0) та (y>0))
Цілі числа a та b є взаємно протилежними	a = -b
Цілі числа a та b є взаємно зворотними	a*b = 1
Число a більше середнього арифметичного чисел b, c, d	a>(b+c+d)/3
Число a не менше середнього геометричного чисел b, c, d	a>=(b+c+d)**(1/3)
Хоча б одна з логічних змінних F1 та F2 має значення так	F1 або F2
Обидві логічні зміни F1 і F2 мають значення так	F1 та F2
Обидві логічні зміни F1 і F2 мають	неF1 і не F2

значення немає	
Логічна змінна F1 має значення так, а логічна змінна F2 має значення ні	F1 і не F2
Тільки одна з логічних змінних F1 та F2 має значення так	(F1 і не F2) або (F2 і не F1)

Схеми алгоритмів

Схема - це графічне подання алгоритму, доповнене елементами словесного запису. Кожний пункт алгоритму відображається на схемі деякою геометричною фігурою-блоком (блоковим символом), причому різним за типом виконуваних дій блокам відповідають різні геометричні фігури, зображувані за ДСТ.

Застосовувані графічні символи, що відбивають основні операції процесу обробки даних, установлює ДЕРЖСТАНДАРТ 19.003-80 (позначення символів відповідає міжнародному стандарту ИСО 1028-73).

Структурне програмування

Практика програмування показала необхідність науково обґрунтованої методології розробки й документування алгоритмів і програм. Ця методологія повинна стосуватися аналізу вихідного завдання, поділу його на досить самостійні частини й програмування цих частин по можливості незалежно одна від одної. Такою методологією є *структурне програмування*, що зародилося на початку 70-х років, а в останній час набуло поширення. За своєю суттю воно втілює принципи системного підходу в процесі створення й експлуатації програмного забезпечення ЕОМ. В основу структурного програмування покладені досить прості положення:

1. алгоритм і програма повинні складатися поетапно (по кроках);
2. складне завдання повинне розбиватися на досить прості, частини, що легко сприймаються, кожна з яких має тільки один вхід і один вихід;
3. логіка алгоритму й програми повинна спиратися намінімальне число досить простих *базових управляючих структур.*

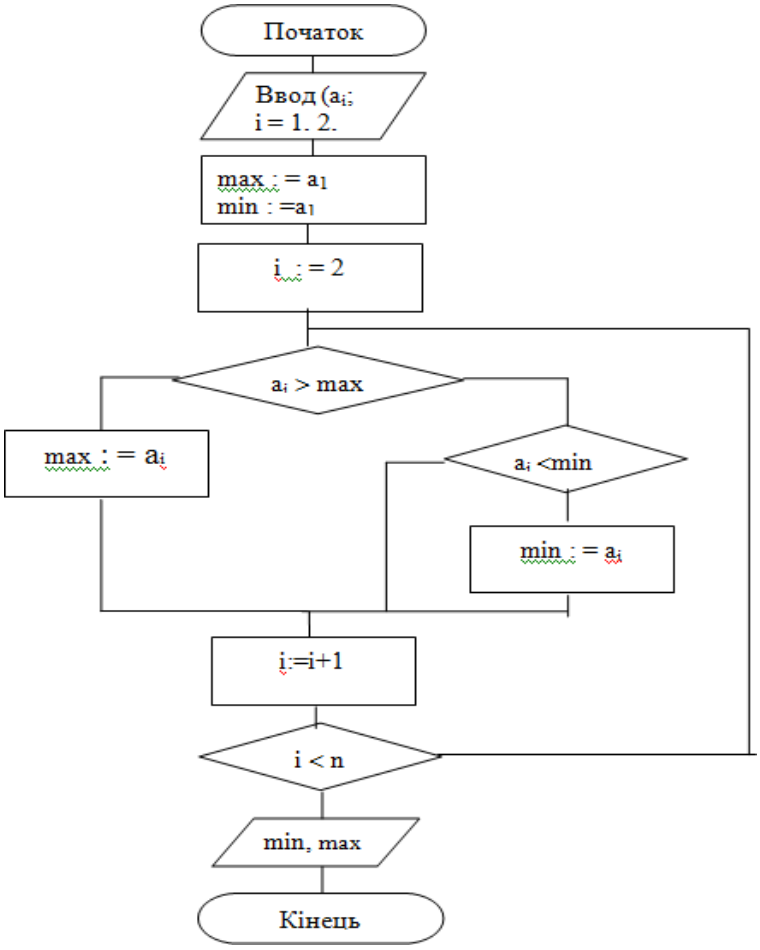


Рис. 2. Схема алгоритму пошуку максимуму й мінімуму

Використання цих положень дозволяє внести певну систему в працю програміста й отримувати алгоритми (і програми), які зручно читати, легко вивчати й перевіряти. Фундаментом структурного програмування є *теорема про структурування***.**

Контрольні запитання:

1. Назвіть основні етапи підготовки та розв'язання задач за допомогою ЕОМ?
2. Прокоментуйте кожний етап розробки програми. Які етапи тав яких ситуаціях можуть бути пропущені?
3. Що враховують при виборі методу вирішення задачі?
4. Поняття алгоритм. Назвіть основні форми запису алгоритмів. Наведіть графічніелементи, які застосовуються при складанні схем алгоритмів?
5. 5.Які поняття використовуютьалгоретмічні мови?
6. Що покладено в основу структурного програмування?
7. Назвіть основні базові структури структурного програмування?
8. .Що таке стандартна функція?
9. Як записуються аріфметичні та логічні вирази? Наведіть приклади запису аріфметичних та логічних виразів?
10. Складіть блок-схему алгоритму переведення числа з будь-якої системи числення у десяткову?

Розділ VI

ПРОГРАМУВАННЯ МОВОЮ QBASIC

QBASIC є програмою MS-DOS.

Система QBASIC може зберігатися як на гнучкому диску, так і жорсткому диску типу «Вінчестер». Якщо для роботи з системою QBASIC використовується гнучкий диск (A або B) з необхідними системними файлами, для завантаження системи QBASIC необхідно запрошення MS-DOS ввести команду: A:\> QBASIC та натиснути клавішу ENTER. Ця програма забезпечить завантаження системних програм. При використанні жорсткого диска системні програми зберігаються у спеціальній директорії, наприклад, під назвою QB. В цьому випадку для завантаження системи необхідно ввести команди:

З: \CD QB «ENTER»

C : \QB\ QBASIC "ENTER".

Після запуску QBASIC з'явиться вікно. Щоб зняти заставку та перейти в головне вікно редактора, натисніть {Esc}.

Вікно QBASIC складається з 4-х частин:

- Головне меню;
- вікно редагування;
- вікно швидких обчислень;
- вікно для допоміжної інформації

Запуск

1.Запуск {Shift+F5}.Запуск програми з першої команди, що виконується.

2.Перезапустити.Підготовка програми до запуску з першої команди, що виконується. Ця команда видаляє всі введені вами дані і висвічує першу команду, що виконується.

3.Продовжити {F5}.Робота поновлюється з команди, на якій програму було зупинено.

Вікно середовища QBASIC

4. СЕРЕДОВИЩЕ QBASIC

- **Загальні відомості про середовище QBASIC**

Назва мови програмування Basic - це перші літери англійських слів **B**eginner's **A**ll - purpose **S**ymbolic **I**nstruction **C**ode (багатоцільова мова програмування для початківців). Часстворення першої версії - 1964 р.

В 1981 р. з'явилася розширена версія Basic-A, що підтримувала текстовий і графічний режими.

Розвитком Basic-A стала версія Quick-Basic, що включала підпрограми і функції з локальними й глобальними змінними, засобами підтримки графіки й звуку, алфавітно-цифрові мітки іт. ін.

Простота граматики й синтаксису, легкість освоєння зробили Basic популярним серед користувачів-непрофесіоналів для математичних і науково-технічних розрахунків. Тому він і досі широко застосовується для навчання основам програмування та для розв’язання відносно простих задач програмування.

Усіченим варіантом Quick-Basic є система програмування QBasic.

QBasic – це інтегрована система програмування, що має власну управляючу оболонку, редактор текстів програм мовою програмування QBasic, засоби пуску, налагодження й прихованої зборки програм, потужний електронний довідник із системи.

Введений за допомогою екранного редактора текст програми

необхідно перетворити в машинний код, що складається із двійкових даних і інструкцій процесора. Таке завдання вирішує спеціальна програма - транслятор. Найбільш простий спосіб трансляції полягає в негайному перекладі в машинні коди кожного слова програми - інтерпретація (тлумачення).

Інтерпретатор QBasic може працювати у двох режимах: безпосередньому й режимі виконання програми.

У безпосередньому режимі дія, задана службовим словом, виконується відразу ж після його введення з клавіатури.

При роботі в режимі виконання текст програми спочатку повністю записується операторами мови, далі програма заноситься до пам’яті і запускається на виконання. У цьому випадку інтерпретатор слово за словом зчитує з пам'яті текст програми, переводить його на мову команд процесора й пропонує готові коди процесору для їхнього виконання.

В інтерпретатора є велика перевага - простота відлагодження програм: виконання програми можна в будь-який момент зупинити і відредагувати її текст. Але є й істотні недоліки: по-перше, програма здатна працювати тільки при наявності в пам'яті самого інтерпретатора (середовища) і, по- друге, програма, що працює з інтерпретатором, робить всі дії дуже повільно. Час іде на розпізнавання слів мови, на пошук відповідних послідовностей команд процесора й на багато чого іншого. Ці недоліки усуваються за допомогою іншого способу трансляції - компіляції (зборка): дії не виконуються негайно, а збираються з наявних у компіляторі стандартних послідовностей у програму в кодах, що не залежать від яких-небудь трансляторів. Головне в компіляторі - зробити програму максимально швидкодіючою.

QBasic має тільки інтерпретатор, а от Quick Basic дає можливість компілювати програми - створювати файли, що виконуються (exe-файли).

Після запуску QBasic на екрані з’являється перше вікно,

(рисунок 1.1).

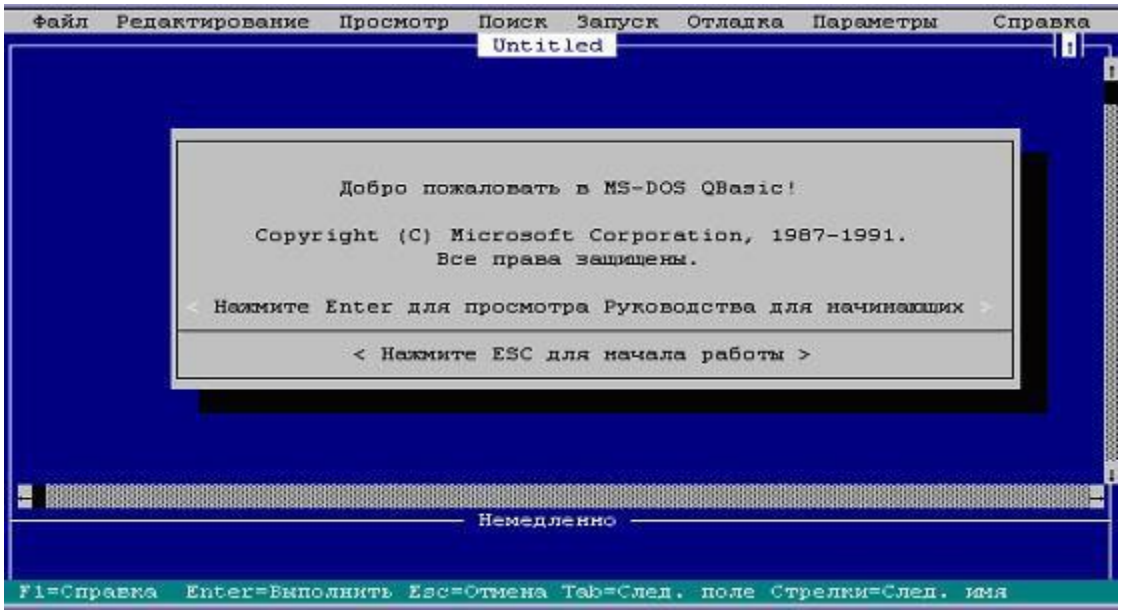


Рисунок 3.

Головне меню

У верхній частині екрана розташовано головне меню QBASIC, що містить назви основних функцій середовища (таблиця 1.1).

Таблиця 1.1

F1	Виведення підказки для того елемента програми, наякий указує курсор
F2	Виведення списку підпрограм. Розбивка екрана на дві частини для одночасного перегляду більших програм
F3	Пошук у тексті
F4	Перегляд екрана виведення
F5	Продовження виконання програми з поточного оператора
F6	Переміщення у вікно швидкого виконання (Immediate)
F7	Виконання програми до поточного положення курсора
F8	Виконання наступного оператора програми (покрокове виконання)

F9	Установка або видалення контрольної точки
F10	Виконання наступного оператора програми (пропускаючи процедуру)
Shift+F1	Перехід у режим допомоги
Shift+F2	Перехід до наступної процедури
Shift+F5	Виконання програми спочатку
Shift+F6	Перемикання у вікно перегляду
Ctrl+F1	Перегляд наступної теми в режимі допомоги
Ctrl+F2	Перехід до попередньої процедури
Ctrl+F10	Перемикання між багатовіконним і повноекранним режимами
Alt+F1	Перегляд попередньої теми в режимі допомоги

- **Введення і редагування програм**

Головне вікно QBasic розділене на дві основні частини - вікно редагування й вікно безпосереднього виконання.

Вікно редагування

На початку роботи QBasic розташовує курсор у вікні редагування. Це дозволяє вводити текст нової програми (Опція Головного меню **Файл** команда **Новый**), завантажувати в це вікно тексти існуючих програм (Файл **Открыть** клавіша Tab - **вибрати зі списку**), редагувати введені тексти (**Редактирование**), запускати програми на виконання (**Запуск Запуск**), зберігати їх у файлах на диску (**Файл Сохранить**).

Вікно безпосереднього виконання

Це вікно розташовується в нижній частині екрана. У ньому можна безпосередньо одержувати результати виконання команди після її введення й натискання клавіші **Enter**. Перехід у це вікно здійснюється після натискання клавіші F6.

- ***Редактор QBASIC***

Розглянемо роботу редактора на прикладі простої програми, що має один оператор.

PRINT "Hello УКРДАЗТ"

Набравши на клавіатурі оператор, натиснемо клавішу **Enter**. Якщо при введенні оператора припустилися помилки, то її можна виправити, використовуючи клавіші переміщення курсора й клавіші видалення символу:

- **Delete** - видаляє символ над курсором;
- **Backspace** - видаляє символ ліворуч від курсора, крімтого, клавіші
- **Shift** - перемикач режимів введення великих літер (верхній регістр) і малих (нижній регістр);
- **Caps Lock** - фіксація режиму введення великих літер іскасування його.

Текст програми можна вводити в будь-якому регістрі. Однак після натискання клавіші **Enter** всі ключові (службові) слова програми будуть виведені великими літерами.

Робота із блоками тексту

У боротьбі з помилками й для підвищення швидкості введення програм необхідно освоїти роботу із блоками тексту:

- виділення блока;
- копіювання;
- переміщення;
- видалення.

Тільки виділений текст можна скопіювати, перемістити або знищити. Виділений текст легко відрізняється від навколишнього тексту більш яскравим підсвічуванням.

Існують кілька способів виділення тексту:

- невеликий блок (слово, кілька слів, рядок, кілька рядків, екран) зручніше виділити одночасним натисканням Shift і однієї із клавіш керування курсором ↑, ↓, ←, →, **Home, End, PgUp, PgDn**. Зняти виділення можна натисканням миші на невиділеній частині тексту. Якщо не все виділено або прихоплено зайве й уже відпущено клавішу керування - не відпускайте **Shift** і натискуйте клавішу керування у зворотному напрямку. Комбінації **Ctrl**+**Shift**+← та **Ctrl +Shift**+→ виділяють слово ліворуч і праворуч від курсора;
- комбінації **Shift+Ctrl+Home** і **Shift+Ctrl+End** дозволяють виділити текст програми, відповідно від початку до поточного рядка й від кінця до поточного рядка. Спочатку натискається й утримується **Shift**, а потім послідовно натискаються й відпускаються інші клавіші.

Натисканням клавіші **Del** виділений блок видаляється без запам'ятовування в буфері обміну. Комбінацією клавіш **Ctrl+ Del** виділений блок зберігається в буфері.

Виділений блок можна скопіювати в буфер сполученням **Ctrl + Ins**, після чого сполученням **Shift + Ins** його можна вставляти в будь-які місця будь-яке число раз у будь-які програми;

- все вище назване можна робити за допомогою опцій меню <Редагування>.
- **Запуск програми на виконання і перегляд результатів**

Для виконання введеної програми треба викликати команду **Запуск** з меню **Запуск** або використати комбінацію клавіш **Shift+F5**.

На екран буде виведений результат виконання програми і

повідомлення:

"Press any key to continue" (Для продовження натисніть будь- яку клавішу).

- **Відлагодження програм**

Транслятор QBASIC виявляє помилки двох типів:

- синтаксичні, які виникають в результаті порушення правил написання конструкцій мови;
- семантичні,пов'язані з неприпустимими значеннями параметрів, неприпустимими діями над параметрами і т. ін.

При виявленні помилки видається відповідне повідомлення на екран монітора і відбувається підсвічування місця її знаходження в тексті.

Крім того, на етапі безпосереднього виконання відслідковуються деякі помилкові ситуації: ділення на нуль, обчислення квадратного кореня з від'ємного числа, переповнення - вихід за праву границю діапазону типу числа і ряд інших. У цих випадках виконання програми або припиняється, або триває з видачею повідомлення про помилку.

Помилки в логіці роботи алгоритму не приводять до зупинки програми. Виявити їх можна тільки аналізом проміжних і остаточних результатів. Для покрокового аналізу проміжних результатів у текст рекомендується включати виведення таких результатів, практикувати виключення фрагментів програми з її роботи шляхом перетворення їх у коментар, використовуючи апостроф.

Переглянути результати можна, натиснувши **F4**.

Зупинити виконання програми можна, натиснувши **Ctrl+Break**, а продовжити натиснувши **F5**.

Для призупинення виконання програми можна поставити у відповідному місці оператор **STOP**.

Значно легше виявити логічні помилки, використовуючи спеціальні клавіші або опцію Головного меню **Отладка**.

Буває корисно виконати програму до рядка, зазначеного курсором - **F7**. Аналогічні результати досягаються установкою в опції **<Отладка>** Контрольная точка остановка программы - **F9** (вона ж - зняття).

Після натискання клавіші **F8** або **F10** (команди **Шаг**, **Процедура на шаг**) - реалізується виконання одного оператора, що дозволяє виконувати програму крок за кроком.

Команда **Трассировка** дозволяє відстежити послідовність виконання операторів програми.

Команда **<Установить следующее значение>** використовується для зміни послідовності виконання програми так, що наступним виконується оператор, на якому встановлений курсор.

- **Збереження програми на диску**

Для зберігання набраної програми на диску у вигляді файла треба виконати такі дії:

Файл □ команда **Сохранить** □ клавіша **Enter**

Якщо програма ще не має імені, то вона в середовищі QBasic буде позначена як **Untitled**. Введіть ім'я файла, з яким ви бажаєте зберегти свою програму, і натисніть клавішу **Enter.**

Ім'я файла повинне бути унікальним, рекомендується використати для цієї мети послідовність літер та цифр довжиною до 8 символів. Розширення файла **.bas** буде записано автоматично.

- **Довідкова система QBASIC**

У вікні **СПРАВКА** можна побачити таку інформацію:

теми довідки;

- основні ключові слова.

Використання довідки QBasic

Щоб одержати довідку по ключовому слову QBasic, встановіть на ньому курсор і натисніть **F1** або праву кнопку миші.

Щоб одержати довідку щодо меню, команд або діалогового вікна QBasic, установіть курсор на темі меню або на кнопці **СПРАВКА** і натисніть **F1**.

Щоб переглянути теми **СПРАВКА** QBasic, натисніть **Alt+З**,для вибору теми - натисніть букву, виділену підсвічуванням.

Щоб перемістити курсор у вікно довідки, натисніть **Shift+F6**.

Щоб переглянути всю довідкову інформацію, використайте **PgDn** або **PgUp**.

Щоб скопіювати інформацію з **СПРАВКА** (приклади програм) у вікно редагування, використайте команди з меню **Редактирование** QBasic.

Щоб закрити вікно довідки, натисніть **Esc**.

Щоб перемістити курсор до теми **СПРАВКА**, використовують клавішу табуляції або натискають першу літеру теми. Щоб побачити інформацію з теми або ключового слова, треба встановити курсор на темі або ключовому слові й натиснути **F1** або праву кнопку миші.

QBasic зберігає останні 20 тем довідки, які були переглянуті. Щоб переглянути попередні теми, треба натиснути **Alt+F1** або натиснути кілька разів кнопку миші на кнопці <**Назад**>.

- **Вихід із середовища QBASIC**

Для виходу із середовища QBASIC необхідно виконати послідовно такі дії:

Головне меню: опція **Файл** команда **Выход** клавіша **Enter**.

Формальні відомості про QBASIC

- **Алфавіт QBASIC**

Алфавіт мови BASIC включає:

- Всі латинські великі та малі літери;
- Арабські цифри 0-9;
- Знаки арифметичних операцій: ^; *; /; +; -; \; MOD.
- Знаки операцій відношення: =; >; <; >=; <=; <>.
- Розділювачі та інші символи:

.- крапка;

,- кома;

;- крапка з комою;

:- двокрапка;

_- пробіл;

!– ознака речової величини;

#- ознака речової величини подвійної точності;

%- ознака цілої величини;

&- ознака довгої цілої величини

$- ознака текстової величини;

()- круглі скобки;

“- лапки;

'– апостроф.

Використовуються також літери російського алфавіту, але у текстових константах чи коментарях.

- ***Інформація у мові QBASIC***

Пам'ятайте! Розділювачем цілої та дробової частин є точка.

Наприклад:
102! - Константа, числова, речова, звичайної точності.
95966.46789# - константа, числова, речова, подвійний точності.
4326% - константа, числова, ціла.
"План на тиждень" – константа текстова. Полягає в лапки.

- ***Змінні***

Змінна – це величина, значення якої може змінюватися у виконання програми, позначається ім'ям (ідентифікатором).

Ім'я змінної – це довільний набір символів, який може містити до 40 символів. Перший символ має бути латинською літерою, а решта – латинські літери чи цифри. Регістр букв не має значення.

Приклади імен: A; dN; GodRozd; k1%; dlina!

Тип змінної BASIC розрізняє п'ять типів даних.

Коли записано z=a+b+2, BASIC повинен знати, який тип чисел мають на увазі. BASIC розпізнає тип змінної суфіксу, тобто. за останнім символом у імені змінної.

Застосування суфіксів не завжди зручне: вони захаращують текст, та й помилитись у них легко.

Тому BASIC передбачений інший спосіб опису типу змінної.

- ***Загальний формат команди опису типу:***

- **DEFINT XX**(цілі числа, INTeger)

DEFLNG XX(Довгі цілі числа, LoNG)

DEFSNG XX(речові числа звичайної точності, SiNGle)

DEFDBL XX(речові числа подвійної точності, DouBLe)

DEFSTR XX(Рядки символів, STRing)

Комбінація XX – діапазон літер. Замість діапазону можна вказати одну літеру.

Наприклад:

DEFLNG AD, всі змінні, імена яких починаються з літер, які у зазначеному діапазоні, тобто. з A до D будуть вважатися довгими цілими (aRc, BT, DLINA і т.п.)

DEFSTR STR, змінний STR-рядок символів.

Контрольні запитання:

1. Програма QBASIC. Головне меню програми (з яких опцій вона складається?.

2. З яких 2-х частин складає головне меню програми QBASIC?

3. Перерахуйте назвини основних середовищ QBASIC?

4. Призначення редактору програми QBASIC?

5. Що включає опція «відлагодження» программ та яке призначення команди «трассировка”?

6. Формальні відомості про програму QBASIC.

7. З чого складає алфавіт мови QBASIC?

8. Які імена змінних використовують у програмі QBASIC та їх зальний формат опису ? Наведіть приклади.

Розділ VII

ОСНОВНІ КОМАНДИ ПРОГРАМИ QBASIC

Команди у мові QBASIC

Програма, написана будь-якою мовою програмування, є набір інструкцій, що описує процес виконання алгоритму завдання. Інструкції записуються за певними правилами, передбаченими вимогами мови. Інструкції вихідної програми називаються командами чи операторами.

•

Правила іменування

Є декілька простих правил, яким треба слідувати, щоб мати добрий стиль програмування, вони стосуються оформлення тексту програм. Більшість із них досить прості й по ходу справи ми про них неодноразово говорили. Ось основні:

- використовуйте коментарі;
- дотримуйтесь правил іменування;
- структуруйте текст;
- будуйте програми з модулів "оптимального розміру".

Поговоримо про одне технічне, але практично важливе питання — як правильно давати імена константам, змінним і іншим об'єктам у наших програмах. Ім’я відіграє важливу роль у розумінні програми, тому в повсякденній практиці при написанні імен змінних треба слідувати правилу, яке полягає в такому: ім'я повинне відбивати сенс і складатися з одного або декількох разом написаних слів, кожне з яких починається з великої букви.

Розробники від Microsoft рекомендують дотримуватися суворіших правил. Ім'я повинне відбивати не лише сенс, але і тип

змінної і її зону дії. Тому ім'я повинне складатися з префікса і власного імені. Префікс також є складеним, дві його частини відбивають зону дії і тип змінної. Наведемо можливі значення префікса.

Таблиця 5.2 Можливі значення префікса згідно з правилами Microsoft

Префікс, який задає область дії	
Перша частина префікса	**Зона дії**
g	Global — увесь проект
m	Module — Для Private змінних модуля
відсутній	Procedure — Для локальних змінних
Префікс, який задає тип змінної	
Друга частина префікса	**Тип змінної**
str	string
int	Integer
byt	Byte
lng	Long
sng	Single
dbl	Double
cur	Currency
var	variant
obj	Object
bln	Boolean

От кілька прикладів правильно побудованих імен: gstrOneWord, mintNumberOne, strAnswer, curSalary. Також як за значенням константи можна відновити її тип, по правильно побудованих іменах можна однозначно відновити їхнє оголошення. Зробимо це в нашому прикладі:

```
Public gstrOneWord As String Private
mintNumberOne As IntegerDim strAnswer As
```

String

Dim curSalary As CurrencyЗгідно з цими же рекомендаціями імена констант варто будувати тільки із заголовних букв. Якщо ім'я константи складається з декількох слів, то для їхнього об'єднання використовується знак підкреслення, наприклад: MY_DIRECTORY_PATH. Зверніть увагу, при побудові констант Office 2000 використовується префікс, що вказує, якому з додатків належить константа.

Таблиця 1.2.Відповідність додатка та префікса констант

Додаток	Префікс констант
Access	ac
Excel	xl
FrontPage	fp
Office	mso
OfficeBinder	bind
Outlook	ol
Power Point	pp
Word	wd
VBA	vb

Префікси варто використовувати й при іменуванні об'єктів, для форм звичайно використовується префікс frm, для командних кнопок — cmd і так далі. Варто розуміти, що якщо вже користуватися префіксами, то загальновживаними, не слід займатися самодіяльністю в цьому питанні.

Оператори

Оператори забезпечують привласнення і зміну значень об'єктів, з якими працює програма.

Оператори та рядки

При записі тексту програми для спрощення читання,

налагодження та її модифікації зручніше кожен оператор розташовувати в окремому рядку. Проте в одному рядку можна розміщувати і декілька операторів. Символом розподілу операторів в рядку служить *двокрапка.*

Іноді виникає ситуація, коли оператор не уміщається в рядку і його слід продовжити в декількох рядках. Для продовження (переносу) оператора на наступний рядок використовується пара символів *пропуск-підкреслення*.

Перед оператором в рядку може стояти мітка — послідовність символів, що починається з букви і закінчується двокрапкою. Мітки можна розміщувати і в окремих рядках перед тими операторами, які вони повинні позначати. Мітки потрібні для операторів переходу типу GoTo, але їх використання вважається "поганим тоном" в структурному програмуванні.

Оператор коментарю

Коментарі на виконання програми не впливають, але потрібні як "ознака хорошого стилю". Ви можете заощадити на коментарях і написати, а потім налагодити невеликий модуль без них. Але вже через тиждень ніхто, у тому числі і ви, не зможете зрозуміти його дію і модифікувати потрібним чином.

Будь-який рядок тексту програми може закінчуватися коментарем. Коментар в VB починається апострофом (') і включає будь-який текст, розташований в рядку правіше. Зазвичай в коментарях описують завдання, що вирішуються модулями, функції, що виконуються процедурами, сенс основних змінних, алгоритми роботи процедур.

Інше застосування коментарі знаходять при налагодженні

програм. Якщо потрібно виключити з програми деякі оператори, то досить перед ними поставити апостроф.

У VB є ще один спосіб виділення коментарів за допомогою ключового слова REM. Такий коментар (на відміну від коментарю, що починається апострофом) повинен відділятися в рядку від попереднього оператора двокрапкою:

weight=weight+z : Rem Збільшення ваги value=weight*price 'Нова вартістьБудь-яку команду в програмі можна забезпечити пояснюючим текстом - коментарем. Коментарі полегшують розуміння тексту програми. Коментарі можна записати двома способами:

a) REM довільний текст
b) ' довільний текст

На перебіг обчислень команда REM жодного впливу не має.

Наприклад:

REM обчислення функції
' обчислення функції

- **Оператор привласнення**

Формат:

Ім'я змінної = вираз

Ця команда виконується за два кроки:

1) обчислюється значення арифметичного чи символьного виразу, зазначеного у правій частині команди;
2) це значення надається змінної, записаної в лівій частині команди.

Наприклад:

$A = Y+X/Z^2$

-
- **Оператор введення даних**
- **Динамічне введення даних**

Формат:

INPUT "Повідомлення", список змінних

При виконанні команди INPUT обчислення призупиняються і на екрані дисплея виводиться пояснювальне повідомлення, якщо ви його написали. У списку змінних через кому вказуються імена змінних, які приймають дані, що вводяться.

Наприклад:

Команда: INPUT"Введіть коефіцієнт та код режиму",B,KR$

*Екран:*Введіть коефіцієнт та код режиму

Курсор встановлюється за останнім символом виведеного тексту, і програма очікує введення даних. Ви повинні через кому набрати всі дані та натиснути клавішу {Enter}.

Якщо рядок символів замкнений не комою, а точкою з комою, INPUT виводить слідом за текстом символ "?" і встановлює курсор через пропуск після "?".

Наприклад:

Команда: INPUT "Рік народження"; GR$

*Екран:*Рік народження?_

Повідомлення в INPUT можна опустити. Тоді на екран буде просто видано питання.

Тип і кількість даних, що вводяться повинні відповідати типу і кількості описаних змінних.

Наприклад:

Команда: INPUT"Введіть № школи, клас, прізвище";N,K,F$

*Екран:*Введіть № школи, клас, прізвище? 2,10,Петрів

Види помилок:

1. Тип даних не збігається із типом змінних.

2. Ввели дуже багато даних.

- **Оператор виведення результатів програми**

Формат:

PRINT список_виразів

У полі операндів через кому або через точку з комою перераховуються вирази, значення яких треба вивести. Значення даних виводяться з позиції курсора.

1. Щільний висновок - роздільник ";"

Наприклад:

а) Команда: PRINT "Мені"; К; "років"

*Екран:*Мені 16 років (якщо К = 16)

Перед значенням числа виводиться пробіл, або знак "мінус".

b) Команда: PRINT "Пра"; "бабуся"

*Екран:*Прабабуся

2. Зональний висновок – роздільник ","

Якщо вказана кома, QBASIC виводить дані щодо зон, кожна зона - 14 позицій.

Наприклад:

Команда: PRINT "Мені", К, "років"

*Екран:*Мені 16 років (якщо К = 16)

3. Висновок у різних рядках

Наприклад:

Команда: PRINT "Зарплата"

PRINT ZP; "руб"

*Екран:*Зарплата

2000 руб

приклад. Дано катет прямокутного трикутника і прилеглий кут. Визначити площу трикутника та гіпотенузу. Довжину та кут у радіанах ввести динамічно.

Результат обчислення вивести у різних зонах.

```
INPUT"Введіть катет прямокутного трикутника та прилеглий кут";A,U
C=A/COS(U) 'Обчислення гіпотенузи
B=SQR(C^2-A^2) 'Обчислення катета
S= 0.5*B*A 'Обчислення площі трикутника
PRINT"C=";C,"S=";S
End
```

Команди керування ходом виконання програми

Найчастіше, залежно від деяких умов, доводиться змінювати послідовність виконання команд дії.

Команди управління ходом виконання програми поділяються на три групи:

1. Команди безумовного переходу;
2. Команди умовного переходу;
3. Команди організації циклів.

Безумовна передача керування

Формат:

GOTOномер рядка або мітка

Управління передається команді, позначеній номером рядка або міткою.

Наприклад:

```
GOTO W
…
W: PRINT"Рішення отримано"
```

*Примітка.*Серед програмістів вживання GOTO прийнято вважати поганим тоном. Надмірна пристрасть до GOTO заплутує програму.

Умовна передача керування

Конструкцію умовної передачі управління можна записати в одному з двох форматів - блоковому (у кілька рядків) або лінійному (в одному рядку).

Лінійний формат:

IFлогічний вираз THEN <блок команд 1> [ELSE <блок команд 2>]

Блоковий формат:

IFлогічний вираз THEN

<блок команд 1>

[ELSE

<блок команд 2>]

END IF

Виконання. Якщо логічний вираз приймає значення "Істина", виконується блок команд 1, якщо логічний вираз - "Брехня", виконується блок команд 2. Якщо ELSE відсутня, то виконується команда, що стоїть за END IF.

Приклад1

```
INPUT "Скільки буде 2x2", X
IF X=4 THEN PRINT "Правильно" ELSE PRINT "Неправильно"
END
```

Приклад2

Обчислити значення функції Y

Лінійний формат:

```
INPUT "Введіть значення аргументу X", X
IF X<=1 THEN Y=X-1/2*X^2+3 ELSE Y=1.05*(X-1)^2
```

```
PRINT "При значенні X=";X;" функція Y=";Y
END
```

Блоковий формат:

```
INPUT "Введіть значення аргументу X", X
IF X<=1 THEN
Y=X-1/2*X^2+3
ELSE
Y=1.05*(X-1)^2
END IF
PRINT "При значенні X=";X;" функція Y=";Y
END
```

Можлива вкладеність одного умовного оператора до іншого.

Формат запису вкладених умовних операторів

IFлогічний вираз THEN
<блок команд 1>
[ELSEIFлогічний вираз THEN
<блок команд 2>]
……..
[ELSE
<блок команд n>]
END IF

Приклад3

Скласти програму обчислення функції:

1 варіант

```
IF Z<0 THEN
F=Z-5/Z^2
ELSEIF Z<=1 THEN
F=Z^2-1
ELSE
F=1/(1-Z)
ENDIF
PRINT "F=";
END
```

2 варіант

```
IF Z<0 THEN F=Z-5/Z^2
IF Z<=1 AND Z>=0 THEN F=Z^2-1
IF Z>1 THEN F=1/(1-Z)
PRINT "F=";
END
```

Приклад: 1

Визначити вид трикутника залежно з його сторін А, У, З.

```
' Програма з оператором IF для визначення виду трикутника в залежності від його сторін
Декларація змінних
DIM A!, B!, З!
'Блок операторів
INPUT"Введіть а, в, з"; A!, B!, З!
PRINT "а="; A!; "в="; B!; "з="; З!
IF (A! > B! + C!) OR (B! > A! + C!) OR (C! > A! + B!) THEN
PRINT "Ця фігура не трикутник!"
ELSE
```

```
IF (A! = B!) AND (A! = C!) THEN
PRINT "Трикутник рівносторонній!"
ELSE
IF (A! = B!) OR (A! = C!) OR (B! = C!) THEN

PRINT "Трикутник рівнобедрений!"
ELSE
PRINT "Трикутник різносторонній!"
END IF
END IF
END IF
'Кінець програми
```

Приклад 2:

```
INPUT "Скільки буде 2x2", X
IF X=4 THEN PRINT "Правильно" ELSE PRINT "Неправильно"
END
```

Приклад3

Обчислити значення функції Y

Лінійний формат:

```
INPUT "Введіть значення аргументу X", X
IF X<=1 THEN Y=X-1/2*X^2+3 ELSE Y=1.05*(X-1)^2
PRINT "При значенні X=";X;" функція Y=";Y
```

```
END
```

Блоковий формат:

```
INPUT "Введіть значення аргументу X", X
IF X<=1 THEN
Y=X-1/2*X^2+3
ELSE
Y=1.05*(X-1)^2
END IF
PRINT "При значенні X=";X;" функція Y=";Y
END
```

Можлива вкладеність одного умовного оператора до іншого.

Формат запису вкладених умовних операторів

IFлогічний вираз THEN
<блок команд 1>
[ELSE IFлогічний вираз THEN
<блок команд 2>]
……..
[ELSE
<блок команд n>]
END IF

Оператор вибору

Формат

SELECT CASE арифм_вираз або симв_вираз

CASE умова 1

<блок команд 1>

CASE умова 2

<блок команд 2>

…….

[CASE ELSE

<блок команд n>]

END SELECT

Умову оператора CASE можна вказати в одному із трьох форматів:

1) CASE константа 1, константа 2, …
2) CASE IS знак_відносини константа
3) CASE константа 1 TO константа 2

Константи в умові повинні бути того ж типу, що і вираз у SELECT CASE.

Алгоритм множинного вибору полягає в наступному:

- обчислюється значення виразу, записаного у SELECT CASE.
- перевіряється, чи задовольняє це значення одному із зазначених у CASE умов
- якщо значення відповідає якійсь умові, виконується блок команд, що йде за даними CASE.

Приклад

INPUT "Введіть значення ",A

SELECT CASE A

CASE 1,5

PRINT "A одно 1 або 5"

CASE IS >5

PRINT "A більше 5"

CASE-8 TO 2.5

```
    PRINT "А не менше -8, але не більше 2.5"
    CASE else
    PRINT "Жодна умова не виконується"
    END SELECT
```

Приклад:

Визначити день тижня у січні 1994р згідно з введеним числом, значення якого знаходиться в діапазоні: 1….31

```
'Програма з оператором SELECT CASE
DIM chislo%
CASE 3, 10, 17, 24, 31
PRINT "Понеділок-день важкий"
CASE 4, 11, 18, 25
PRINT "Вівторок - кафедральний день"
CASE 5, 12, 19, 26
PRINT "Середа-бібліотечний день"
CASE 6, 13, 20, 27
PRINT "Четвер-рибний день"
CASE 7, 14, 21, 28
PRINT "П'ятниця-банний день"
CASE 8, 15, 22, 29
PRINT "Субота-театральний день"
CASE ELSE
PRINT "Обід у друзів"
END SELECT
PRINT "Лютий-місяць канікули"
'Кінець програми
```

Приклад

```
INPUT "Введіть значення ",A
SELECT CASE A
CASE 1,5
PRINT "A одно 1 або 5"
CASE IS >5
PRINT "A більше 5"
CASE-8 TO 2.5
PRINT "A не менше -8, але не більше 2.5"
CASE else
PRINT "Жодна умова не виконується"
END SELECT
```

Після виконання того чи іншого блоку команд керування передається команді, що настає за END SELECT.

Статичне введення даних

(оператори: DATA, READ, RESTORE)

Якщо програма постійно працює з деяким набором числових або символьних констант, можна оголосити такий набір блоком даних:

DATAсписок констант

У списку констант через кому вказуються значення констант, наприклад:

10 DATA 5,25,19.6,30,12,"ABC"

20 DATA "BASIC", "Pascal", 25.9

У програмі можна записати довільну кількість операторів DATA. До блоку даних по порядку включаються всі константи і в пам'яті створюється спеціальний покажчик блоку даних. Під час роботи програми цей покажчик містить порядковий номер константи у блоці даних. При запуску програми вказівник вказує першу константу з блоку даних.

Для присвоєння значень констант із блоку даних змінним використовується оператор READ:

Дані з оператора DATA зчитуються і надаються певним змінним чи масивам з допомогою оператора READ:

READсписок змінних

У списку змінних через кому вказуються імена змінних, яким надаються значення констант із блоку даних. Типи змінних у списку READ повинні відповідати типу констант з блоку даних. Наприклад:

READ a%, m%, t

Змінним a%, m%, t присвояться значення 5, 25 та 19.6. Покажчик переміщається на 4 константу. Оператор

READ c, sc%, a$

Надасть змінним c, sc%, a$ значення 30, 12, "ABC" і покажчик переміститься на "BASIC" і т.д.

Під час виконання оператора READ дані зчитуються
По-порядку в необхідній кількості починаючи з першого оператора DATA, якщо оператор READ зустрічається кілька разів, то зчитування продовжується з наступного порядку CONST в поточному операторі DATA.
Якщо кількість зчитувань оператора READ перевищує кількість
CONST в операторі DATA, то з'являється повідомлення про помилку.
Якщо необхідно проводити зчитування CONST повторно, то для повернення до оператора DATA використовується RESTORE з міткою.
Якщо мітку не використовувати, повторне зчитування починається з першого оператора DATA.

Як кілька разів прочитати ті самі дані з блоку даних? Для відновлення покажчика блоку даних використовується оператор RESTORE:

RESTOREномер рядка або мітка

де номер рядка повинен вказувати оператор DATA.

Якщо не вказано номер рядка, наступний після RESTORE, READ почне читати з першої константи блоку даних. Наприклад:

```
RESTORE
READ vk%, s%
RESTORE 20
READ z1$, z2$
```

vk%, s% присвояться значення 5 і 25, а змінним z1$, z2$ - значення "BASIC" та "Pascal".

Блок даних зручний у тих випадках, коли кілька разів використовується та сама послідовність констант.

приклад. Скласти програму знаходження висоти рівнобедреної трапеції, яка має підстави 5м і 11м, а бічна сторона 4м. Дані ввести статично.

```
DATA 5,11,4
READ BC, AD, AB
AE=(AD-BC)/2 'Підстава трикутника
HT=SQR(AB^2-AE^2)
PRINT"Висота трапеції =";HT
END
```

Контрольні запитання:

1. Оператори та рядки.Формат запису оператору коментарю?

2. Формат запису оператору привласнення та як він працює?

3. Динамічне введення даних. За допомогою якого оператора вводять дані в програму?

4. Формат запису оператора виведення результатів програми та як він працює?

5. Команди керування ходом виконання програми. Оператори безумовного та умовного керування програмою. Як вони працюють та формат їх запису?

6. Формат запису вкладених операторів та принцип їх роботи?

7. Які оператори використовуют для ввода статичних даних в програму?

8. Призначення оператору вибору та його формат запису?

Масиви

Одним з найважливіших інструментів програміста є можливість роботи з масивами змінних.

Масив - набір однотипних даних, що зберігаються разом та мають спільне ім'я.

Можливість об'єднання груп елементів масив дозволяє, з одного боку, полегшити масову обробку даних, з другого - спростити ідентифікацію елементів масиву.

Кожному елементу масиву може бути надано одне числове або символьне значення, тому різняться масиви числові та символьні. Крім того, масиви можуть бути одновимірними та багатовимірними.

- **Одновимірні масиви**

- *Основні поняття:*

Масив позначається одним іменем. Так усю сукупність дійсних чисел
1.6, 14.9, -5.0, 8.5, 0.46
можна вважати масивом і позначити одним ім'ям, наприклад, А. Утворюючі масив змінні називаються елементами масиву. Кожен елемент масиву позначається ім'ям масиву з індексом, укладеним у круглі дужки.
A(1), A(2), A(3), ..., A(n).

Індекс визначає положення елемента масиву даних щодо його початку.

Для розглянутого вище прикладу елементами масиву є:
A(1)=1.6, A(2)=14.9, A(3)=-5.0, A(4)=8.5, A(5)=0.46

- ***Визначення масивів***

Перш ніж скористатися масивом, у програму треба включити оператор DIM, який задає максимально допустимий індекс. Це дозволить системі з

Бейсіком зарезервувати в пам'яті область достатнього розміру.

Масиви, що містять від 1 до 10 елементів, можна використовувати без їхньої оголошення в операторі DIM. Максимальна кількість елементів будь-якого масиву визначається лише обсягом вільної ГП.

Формат запису оператора DIM:

DIM ім'я_масиву (максимальний_індекс)

"Имя_масива" зазвичай вибирається за тими самими правилами, як і імена простих змінних.

"Максимальний_індекс" вказує максимально допустимий у програмі індекс і має бути позитивним.

Приклади опису масивів:

DIM X(50) ' оголошення одновимірного числового масиву X для 50 чисел;

DIM V$(12) 'Оголошення одновимірного масиву V для 12 символьних елементів.

Оголошення масиву із змінним розміром.

```
INPUT K
DIM A(K)
.....
```

- *Види помилок*

Якщо в програмі елемент масиву з індексом більшим, ніж значення розмірності, оголошене в операторі DIM або прийняте за замовчуванням, то видається повідомлення про помилку 9:

Subscript out of range (вихід межі масиву).

Якщо оператор DIM вказано після використання імені масиву або масив повторно оголошено, з'являється повідомлення 10:

Redimensioned array (повторне завдання розмірності масиву).

- ***Заповнення масиву***

Існує два способи привласнення значень елементам масиву:

1) статичний, з використанням операторів DATA, READ та оператора привласнення;

2) динамічний, з використанням оператора INPUT та функції RND.

Працюючи з масивами дуже зручно користуватися оператором циклу FOR...NEXT. Як індекс масиву використовують параметр циклу.

1. Приклад статичного наповнення масиву.

```
DATA зливу, ананас, груша
DATA яблуко, вишня, абрикос
DIM A $ (6)
FOR I=1 TO 6
READ A$(I)
NEXT I
```

Цикл FOR...NEXT послідовно надає значення всім змінним зі списку.

2. Приклад динамічного заповнення масиву

```
INPUT "Введіть кількість елементів масиву";
DIM A(N)
FOR I=1 TO N
INPUT A(I)
NEXT I
```

У цьому прикладі використовується змінне завдання розмірності масиву.

3. Приклад заповнення масиву за допомогою стандартної функції RND

```
DIM V(12)
FOR I=1 TO 12
V(I)=INT(RND(1)*10)
PRINT V(I);
NEXT J
```

- ***Порядок роботи з масивами***

1. оголошення масиву (завдання максимального розміру масиву);
2. заповнення масиву;

3. обробка елементів масиву;
4. виведення результату обробки.

- ***Події над елементами одновимірного масиву***

Приклад 1. Підрахувати загальну суму 10 чисел, записаних статично в масив та кількість негативних чисел у цьому масиві.

```
DATA 3,4,5,-8,10.67,2.7,765,-6.98,9,-1
DIM A(10)
FOR I=1 TO 10
READ A(I)
S=S+A(I)
IF A(I)<0 THEN N=N+1
NEXT I
PRINT " сума=";S,"N=";N
END
```

Приклад 2. Задано масив К(45). Визначити мінімальний елемент масиву та його індекс.

```
DIM K(45)
FOR I=1 TO 45 ' Заповнення масиву
INPUT K(I)
NEXT I
MIN=A(1) ' Змінної MIN надається перше значення масиву
FOR I=2 TO 45
IF A(I) <MIN THEN MIN=A(I):K=I
NEXT I
PRINT "Мінімальний елемент=";MIN, "його індекс=";K
```

```
END
```

Приклад 3. Задано масив A(18). У масиві поміняти місцями значення 1-го та 2-го елемента, 3-го та 4-го тощо. Змінений масив вивести на екран.

```
DIM A(18)
FOR I=1 TO 18
A(I)=INT(RND(1)*10)
PRINT A(I);
NEXT I
FOR I=1 TO 18 STEP 2
SWAP A(I),A(I+1) ' Змінюються місцями значення елементів масиву
NEXT I
PRINT
FOR I=1 TO 18
PRINT A(I); 'Виведення зміненого масиву
NEXT I
END
```

Приклад 4. Даний масив M(30). Елементи масиву – довільні числа. Видати значення кожного п'ятого елемента на екрані. Вказані елементи видати у рядок.

```
DIM M(30)
FOR I=1 TO 30
M(I)=INT(RND(1)*15)
PRINT M(I);
NEXT I
PRINT
FOR I=5 TO 30 STEP 5
PRINT M(I); ' Виведення кожного п'ятого елемента масиву
NEXT I
```

Приклад 5. Сформувати з елементів N одномірний масив A і вивести його на друк у вигляді K стовпців.

```
INPUT "Введіть кількість елементів масиву";
DIM A(N)
FOR I=1 TO N
A(I)=INT(RND(1)*10)
NEXT I
INPUT "Введіть кількість стовпців";
FOR I=1 TO N
PRINT A(I);
IF I MOD K = 0 THEN PRINT
NEXT I
END
```

Приклад 6. Скласти програму обчислення та друкування значень функції

Y=(SIN X+1) COS 4X

Значення аргументу задані статично у масиві X(10). Значення функції записати масив Y(10) і роздрукувати в п'ять рядків.

```
DATA 5,6.78,56,7.34,678,89,5,23.9,10,34.7
DIM X(10),Y(10)
FOR J=1 TO 10
READ X(J)
NEXT J
FOR J=1 TO 10
Y(J)=(SIN(X(J))^2+1)*SQR(COS(4*X(J))^2)
PRINT Y(J);
IF J MOD 2 = 0 THEN PRINT
NEXT J
END
```

- **Двовимірні масиви**

- ***Основні поняття***

Бейсик дозволяє працювати з масивами, що мають до 255 вимірювань і містять до 32767 елементів кожного з них. Проте слід добре подумати, чи варто використовувати ці можливості у повному обсязі. Якщо ви зробите свої масиви занадто великими, то довго згадуватимете, як отримати доступ до тієї чи іншої інформації. У той самий час іноді вирішення завдання можливе лише за наявності багатовимірних масивів, чи матриць. Найчастіше застосовуються двомірні масиви, оскільки описувати інформацію з її позиції у одномірному списку дуже незручно. Уявіть собі, як важко було б знайти своє місце на стадіоні, якби на квитку вказувався лише порядковий номер сидіння – один із багатьох тисяч! Квитки ж, у яких проставлено і номер ряду, і номер місця, значно спрощують це завдання. Укладачі географічних карток теж користуються таким прийомом. Координати Парижа, Осло, Рима позначаються не як 1, 2 або 300, а в термінах градусів широти та довготи. Ці два числа вказують розташування міста щодо екватора та початкового (Грінвічського) меридіана.

У Бейсику передбачені засоби, за допомогою яких ви можете організувати інформацію так само. Складаючи програму, уявіть собі прямокутну таблицю на кшталт бджолиних сот, або матрицю, де зберігатимуться дані. Кожне значення можна "покласти в комірку" та "вийняти з неї", якщо вказано номер відповідних рядка та стовпця.

Ім'я та розмірність матриці визначається оператором

DIM A(3,4)

Цей оператор зарезервує 12 осередків:

A(1,1), A(1,2), A(1,3), A(1,4)

A(2,1), A(2,2), A(2,3), A(2,4)

A(3,1), A(3,2), A(3,3), A(3,4)

Кожен елемент описується двома індексами: перший означає номер рядка, а другий - номер стовпця.

	Стовпець 1	стовпець 2	стовпець 3	стовпець 4
рядок 1	1,1	1,2	1,3	1,4
рядок 2	2,1	2,2	2,3	2,4
рядок 3	3,1	3,2	3,3	3,4

Коли матриця визначена, змінними з цієї області пам'яті можна скористатися нарівні з іншими.

- ***Заповнення масиву***

Приклад статичного заповнення та друкування двовимірного масиву.

```
DATA Настя, Уляна, Женя
DATA Щербакова, Євланова, Клочко
DIM A $ (2,3)
FOR I=1 TO 2 ' Заповнення масиву
FOR J=1 TO 3
READ A $ (I, J)
NEXT J,I
FOR I=1 TO 2
FOR J=1 TO 3 ' Друк значень масиву за рядками
PRINT A $ (I, J);
NEXT J
PRINT
NEXT I
```

Приклад динамічного заповнення та друку двовимірного масиву.

```
DIM A(5,4)
FOR I=1 TO 5 ' Заповнення масиву
FOR J=1 TO 4
INPUT A(I,J)
```

```
NEXT J,I
FOR I=1 TO 5 ' Друк значень масиву за рядками
FOR J=1 TO 4
PRINT A(I,J);
NEXT J
PRINT
NEXT I
```

Приклад наповнення масиву за допомогою стандартної функції RND.

```
DIM M(2,8)
FOR I=1 TO 2
FOR J=1 TO 8
M(I,J)=INT(RND(1)*10) ' Заповнення масиву випадковими
PRINT M(I,J); ' числами та печатка значень масиву
NEXT J' за рядками
PRINT
NEXT I
```

- ***Дії над елементами двовимірного масиву***

Приклад 1. Задано матрицю U(7,3). Негативні елементи матриці замінити на 0. Змінений масив вивести на екран.

```
DIM U(7,3)
FOR I=1 TO 7 ' Заповнення масиву негативними та
FOR J=1 TO 3 ' позитивними числами
U(I,J)=INT(RND*20-10)
PRINT U(I,J);
NEXT J
PRINT
NEXT I: PRINT
FOR I=1 TO 7
```

```
FOR J=1 TO 3
IF U(I,J)<0 THEN U(I,J)=0
PRINT U(I,J);
NEXT J
PRINT
NEXT I
END
```

Приклад 2. Визначити суму позитивних елементів парних рядків матриці C(5,3).

```
DIM C(5,3)
FOR I=1 TO 5
FOR J=1 TO 3
3(I,J)=INT(RND(1)*20)
PRINT (I,J);
NEXT J
PRINT
NEXT I: PRINT
S=0
FOR I=2 TO 5 STEP 2
FOR J=1 TO 3
S=S+C(I,J)
NEXT J
NEXT I
END
```

Приклад 3. Дано матрицю U(7,3). Отримати масив Q(7) елементами якого є кількість негативних елементів відповідних рядків матриці U(7,3).

```
DIM U(7,3),Q(7)
FOR I=1 TO 7 ' Заповнення масиву негативними та
```

```
FOR J=1 TO 3 ' позитивними числами
U(I,J)=INT(RND(1)*20-10)
PRINT U(I,J);
NEXT J
PRINT
NEXT I: PRINT
FOR I=1 TO 7
K=0
FOR J=1 TO 3
IF U(I,J)<0 THEN K=K+1
NEXT J
Q(I)=K: PRINT Q(I);
NEXT I
END
```

Приклад 4. Задано масив С(5,3). У масив К(5) записати суми елементів відповідних рядків масиву С(5,3). Вивести на екран значення елементів масиву К(5).

```
DIM C(5,3),K(5)
FOR I=1 TO 5
FOR J=1 TO 3
З(I,J)=INT(RND(1)*20)
PRINT (I,J);
NEXT J
PRINT
NEXT I: PRINT
FOR I=1 TO 5
S=0
FOR J=1 TO 3
S=S+C(I,J)
```

```
NEXT J
K(I)=S: PRINT K(I); ' Формування та друк масиву К
NEXT I
END
```

Приклад 5. Задано матрицю M(6,3), підрахувати кількість рядків, у яких перший елемент рядка менший за нуль.

```
DIM M(6,3)
FOR I=1 TO 6
FOR J=1 TO 3
U(I,J)=INT(RND(1)*20-10)
PRINT M(I,J);
NEXT J
PRINT
NEXT I: PRINT
FOR I=1 TO 6
IF M(I,1)<0 THEN K=K+1 ' Підрахунок кількості рядків
NEXT I
PRINT "Кількість рядків=";K
```

Приклад 6. У заданій матриці N(4,6) поміняти місцями рядки L та K. Змінену матрицю вивести на екран.

```
DIM N(4,6)
FOR I=1 TO 4
FOR J=1 TO 6
N(I,J)=INT(RND(1)*20)
PRINT N(I,J);
NEXT J
PRINT
NEXT I
```

```
INPUT " Введіть рядки, які потрібно замінити місцями ";L,K
FOR J=1 TO 6
SWAP N(L,J),N(K,J) ' Змінюються місцями рядки
NEXT J
FOR I=1 TO 4 ' Друк зміненого масиву
FOR J=1 TO 6
PRINT N(I,J);
NEXT J
PRINT
NEXT I
```

Оголошення розмірності масиву буває змінним, тобто. розмірність з'ясовується у процесі роботи програми.

Приклад 7. Задано матрицю A(n,m), отримати масив B(n), елементами якого є сума найменшого та найбільшого елементів відповідного рядка матриці A(n,m).

```
INPUT "Введіть кількість рядків та стовпців матриці A"; N,M
DIM A(n,m),B(n)
FOR I=1 TO N ' Динамічне заповнення
FOR J=1 TO M' масиву
INPUT A(I,J)
NEXT J,I
FOR I=1 TO N ' Друк елементів масиву A
FOR J=1 TO M
PRINT A(I,J);
NEXT J
PRINT
NEXT I: PRINT
FOR I=1 TO N
MI=A(I,1): MA=A(I,1) ' Перший елемент кожного рядка
```

```
FOR J=2 TO M' запам'ятовується у змінних MI,MA
IF A(I,J) < MI THEN MI=A(I,J)
IF A(I,J) > MA THEN MA=A(I,J)
NEXT J
B(I)=MI+MA: PRINT B(I); ' Формування масиву B
NEXT I
END
```

Матриця, у якої кількість рядків дорівнює кількості стовпців, називається квадратною.

Приклад 8. Задано квадратну матрицю A порядку N. Отримати масив B(n), елементами якого є перший позитивний елемент відповідного рядка квадратної матриці A.

```
INPUT "Введіть кількість рядків та стовпців матриці A"; N
DIM A(n,n),B(n)
FOR I=1 TO N
FOR J=1 TO N
INPUT A(I,J)
NEXT J,I
FOR I=1 TO N
FOR J=1 TO N
PRINT A(I,J);
NEXT J
PRINT
NEXT I: PRINT
FOR I=1 TO N
FOR J=1 TO N
IF A(I,J)>0 THEN B(I)=A(I,J):J=N ' Формування масиву B
NEXT J
PRINT B(I);
```

```
NEXT I
END
```

Елементи квадратної матриці, у яких номер рядка співпадає з номером стовпця, розміщуються на головній діагоналі матриці.

У квадратній матриці B(4,4)

B(1,1), B(1,2), B(1,3), B(1,4)

B(2,1), B(2,2), B(2,3), B(2,4)

B(3,1), B(3,2), B(3,3), B(3,4)

B(4,1), B(4,2), B(4,3), B(4,4)

елементи B(1,1), B(2,2), B(3,3), B(4,4) розташовуються на головній діагоналі матриці.

Приклад 9 Дана квадратна матриця A порядку 5. Замінити нулями всі елементи матриці, розташовані на головній діагоналі. На екран вивести змінену матрицю як таблиці.

```
DIM A(5,5)
FOR I=1 TO 5
FOR J=1 TO 5
A(I,J)=INT(RND(1)*20)
PRINT A(I,J);
NEXT J
PRINT
NEXT I: PRINT
FOR I=1 TO 5 ' Заміна нулями елементів, розташованих на
A(I,I)=0' головної діагоналі
NEXT I
FOR I=1 TO 5
FOR J=1 TO 5
PRINT A(I,J); ' Друк зміненої матриці
NEXT J
```

```
PRINT
NEXT I
```

Приклад 10. Дано квадратну матрицю порядку N. У рядках з негативним елементом на головній діагоналі знайти суму всіх елементів рядка. На екрані вивести суму елементів та номер рядка.

```
INPUT "Введіть кількість рядків та стовпців"; N
DIM C(N,N)
FOR I=1 TO N
FOR J=1 TO N
З(I,J)=INT(RND(1)*20)
PRINT (I,J);
NEXT J
PRINT
NEXT I
FOR I=1 TO N
S=0
FOR J=1 TO N
IF C(I,I)<0 THEN S=S+C(I,J)
NEXT J
IF S<>0 THEN PRINT " Сума елементів =";S," Номер рядка =";I
NEXT I
```

Цикли

У мові BASIC передбачено два основні способи організації циклів:

- повторення блоку команд задану кількість (число) разів (цикл з лічильником);
- циклічне повторення блоку команд, поки виконується (або виконується) деяке умова.

У мові BASIC передбачено два основні способи організації циклів:

повторення блоку команд задану кількість (число) разів (цикл з лічильником -FOR…NEXT) ;

циклічне повторення блоку команд, поки виконується (або виконується) деяке умова.

Цикл з лічильником (FOR…NEXT)

Цей оператор використовується, коли певний блок команд необхідно виконати задану кількість разів.

Формат

FORлічильник = E1 TO E2 [STEP E3]

Блок команд

NEXTлічильник

лічильник (параметр) - числова змінна;

E1 – початкове значення лічильника;

E2 – кінцеве значення лічильника;

E3 – крок зміни лічильника. За замовчуванням цей крок дорівнює 1;

Блок команд – набір операторів, призначених для повторення.

Виконання. Якщо E3 >0, то цикл триває постійно, поки лічильник < або = E2. Якщо E3 <0, то цикл триває весь час, доки лічильник > або = E2.

Організація циклічних обчислень можлива, якщо відомі початкове, кінцеве

значення та крок(УП).

Якщо крок (УП) не задано, то за замовчуванням він дорівнює 1.

Якщо крок позитивний, то початкове значення (УП) має бути меншим ніж кінцеве значення (УП) і навпаки.

(УП) може бути змінена в циклі іншими операторами. Нормальним є закінчення циклу після його виконання n разів, де

N = кінцеве значення (УП) - початкове значення (УП)
крок (УП)

при цьому n-змінна цілого типу.

Допустиме дострокове завершення циклічних обчислень під час виконання деяких умов за допомогою оператора EXIT FOR

Приклади

1) FOR I=1 TO 5
PRINT I;
NEXT I
Результат: 1 2 3 4 5

Тіло циклу виконуватиметься 5 разів. Крок зміни лічильника за замовчуванням дорівнює 1.

2) FOR I=1 TO 5 STEP 2
PRINT I;
NEXT I
Результат: 1 3 5

3) Зворотний цикл
FOR I=5 TO 1 STEP -1

PRINT I;

NEXT I

Результат: 5 4 3 2 1

4) FOR I=5 TO 1

PRINT I;

NEXT I

Перевірка виходу із циклу проводиться на початку циклу, тому цикл не виконається жодного разу.

Обмеження під час використання оператора циклу FOR … NEXT:

1. Не змінюйте значення параметра циклу всередині циклу

FOR I=1 TO 5

I=I+1' НЕ МОЖНА!

NEXT I

2. Ніколи не передавайте керування всередину циклу, це може призвести до непередбачуваних результатів.

GOTO 10' НЕ МОЖНА!

FOR I=1 TO 5

10 INPUT A

S=S+A

NEXT I

Достроковий вихід із циклу

Для циклу FOR передбачена можливість безумовного виходу з циклу (до завершення). Управління передається команді, що настає за NEXT.

Формат команди виходу:

EXIT FOR

Наприклад

FOR I=1 TO 10

INPUT "Введіть параметр";F

```
IF F=100 THEN EXIT FOR
S=S+F
NEXT I
PRINT "Значення"
```

Підрахунок:

Сума	**Кількість**	**твір, добуток**
S=0	K=0	P=1
S=S+A	K=K+1	P=P*A

Приклад1. Підрахувати суму 10 значень змінної А.

```
S=0
    FOR I=1 TO 10
    INPUT "Введіть значення змінної";
    S=S+A 'Підрахунок суми значень змінної А
    NEXT I
    PRINT "Сума ="; S
```

Приклад 2. Підрахувати кількість позитивних чисел і добуток негативних чисел N введених.

```
INPUT “Введіть кількість значень”;
        K=0
        P=1
    FOR I=1 TO N
    INPUT "Введіть значення змінної";
    IF A>0 THEN K=K+1
    IF A<0 THEN P=P*A
    NEXT I
    PRINT"Кількість позитивних чисел =";K
    PRINT"Твір негативних чисел =";P
```

Практичне заняття

Приклад 1. Розрахувати та вивести на друк значення функції Y=5/X за зміни аргументу X від -5 до 5 з кроком 2.

```
FOR X=-5 TO 5 STEP 2
Y= 5/X
PRINT "X=";X, "Y=";Y
NEXT X
```

Приклад 2. Даний ряд чисел 100, 55, -1000, 20, 5, 8, 33, 48. Знайти кількість пар чисел квадратних різниць яких <100.

```
DATA 100, 55, -1000, 20, 5, 8, 33, 48
K=0
FOR I=1 TO 8 STEP 2
READ A,B
IF (AB)^2 < 100 THEN K=K+1
NEXT I
PRINT "Кількість пар чисел квадрат різниці яких <100 =";
```

Приклад 3. Знайти максимальне значення N введених. Вивести максимальне значення та його індекс.

```
INPUT "Введіть кількість значень";
INPUT "Введіть перше значення";
MAX=A: IN=1
FOR I=2 TO N
PRINT "Введіть"; I; "-е значення"
INPUT A
IF A>MAX THEN MAX=A: IN=I
NEXT I
PRINT "Максимальне значення =";
PRINT "Індекс максимального значення ="; IN
```

Приклад 1:

```
'Програма з оператором FOR
'Декларація змінних
DIM S!, I%, n%
'Введіть дані
INPUT "Введіть n"; n%
S=0
FOR i%=1 TO n%
S!= S!+ i%
NEXTi%
'Виведення результату
PRINT "S="; S!
'Кінець обчислення
```

Цикл WHILE … WEND

Формат

```
WHILEлогічний вираз
 Блок команд
WEND
```

Цей оператор дозволяє виконувати блок команд доти, доки значення логічного виразу "Істина". Після завершення циклу управління передається команді, що настає за WEND. Наприклад:

```
WHILE PAS$<>"Петунія"
INPUT"Введіть пароль";PAS$
WEND
PRINT "Ласкаво просимо"
```

Цей цикл буде працювати, доки не буде введено правильний пароль.

У блоці команд повинен бути оператор, що впливає на значення логічного виразу.

Наприклад:

1) I=0

```
WHILE I<=10
I=I+1 'оператор, який впливає на значення логічного виразу
WEND
PRINT I
```

2) Приклад нескінченного циклу

```
I=0
WHILE I<=10
INPUT A
S=S+A
WEND
PRINT S
```

У блоці команд немає оператора, який змінював значення I.

Якщо за першої перевірки умови виявиться, що значення логічного висловлювання - " Брехня " , блок команд нічого очікувати виконано жодного разу. Наприклад:

```
I=10
WHILE I<10
I=I+1
WEND
```

Приклад 1. Обчислити суму парних чисел в інтервалі від 1 до 10 включно.

```
'I - парні числа
'Sum - сума парних чисел
I=2
Sum=0
```

```
WHILE I<=10
Sum=Sum+I
I=I+2
WEND
PRINT"Сума парних чисел в інтервалі від 0 до 10 =";Sum
```

Приклад 2. У під'їзді N сходинок. Скільки кроків буде зроблено, якщо крокувати через 3 сходинки.

```
KS - кількість сходинок
' KH - кількість кроків
INPUT "Введіть кількість сходів";
KS=0
KH=0
WHILE KS<=N
KS=KS+3
KH=KH+1
WEND
PRINT"Кількість кроків=";KH
```

Приклад 3 Відома сума номерів сторінок. Визначити номер сторінки.

```
'NS - номер сторінки
'S - сума номерів сторінок
INPUT"Введіть суму номерів сторінок";
NS=0
S=0
WHILE S<Q
NS=NS+1
S=S+NS
WEND
PRINT"Номер сторінки = ";NS
```

Цей оператор дозволяє виконувати блок команд доти, доки значення

логічного виразу "Істина". Після завершення циклу управління передається команді, що настає за WEND.

Робота циклу починається з перевірки логічної умови, якщо умова виконується, обробляється група операторів. Ця послідовність дій повторюється доти, доки умова перестане виконуватися. У такому випадку керування передається оператору після WEND.

Якщо умова не виконується під час першої перевірки, то група операторів не виконується жодного разу і керування переходить до наступного оператора.

Приклад:

Обчислити таблицю з 2-х значень електричного струму за допомогою формули:

i = U / R1+R2 (1-EXP(-P*t))

Для t є [0, tmax],

де tmax = LOG(1/e)/ABS(P)

dt = tmax/19;

e=0,001; P = 1250;

R1 = 25 ОМ; R2 = 45 ОМ; U = 220 В

```
'Програма циклічного обчислення
'Використання оператора WHILE
'Декларація констант та змінних
CONST E!=0.001
DIM U!, R1!, R2!, P!, tm!, dt!, i!, t!
'Введіть початкові дані
INPUT"Введіть дані"; U!, R1!, R2!, P!
tmax = LOG(1/e)/ABS(P)
dt = tmax/19: t! = 0.0
'Організація циклу
```

```
WHILEt <= tmax
i = U / R1+R2 (1-EXP(-P*t))
PRINT "t! = "; t!; TAB(10); "i="; i!
t! = t! + dt!
WEND
'Кінець програми
```

Оператор циклу DO …… LOOP

Цей оператор має три форми реалізації.
Синтаксична діаграма першої форми має вигляд:

DO група операторів EXIT DO LOOP

WHILE
Умова
UNTIL

Така форма дає назву з умовою, так як група операторів виконується до перевірки умови.
Повторне виконання групи операторів залежить:
- від вибору ключового слова WHILE чи UNTIL
У разі вибору WHILE повторне виконання групи операторів можливе, якщо перевірка умови дала позитивний результат, інакше управління переходить до наступного оператора за циклом.

При виборі UNTIL, навпаки, повторне виконання групи операторів можливе, якщо результат перевірки умови негативний. Позитивний результат означає завершення циклічних обчислень і переходить до іншого оператора.
Оператор дозволяє включити до циклу додаткові умови, за допомогою яких можливе дострокове завершення обчислення та вихід із циклу.

Запишемо організацію циклу для попереднього прикладу за допомогою 1-ї форми оператора DO

```
i = U / R1+R2 (1-EXP(-P*t))
PRINT "t! = "; t!; TAB(10); "i="; i!
t! = t! + dt!
LOOP UNTIL t!> tmax!
```

Інша форма оператора DO визначається такою синтаксичною діаграмою

DO WHILE Група операторів

UNTIL

EXIT DO LOOP

Ця конструкція аналогічна оператору циклу з передумовою, коли може статися, що група операторів не виконується жодного разу. Це можна при виборі ключового слова WHILE та негативному результаті перевірки умови.
Запишемо організацію циклу для попереднього прикладу за допомогою 2-ї форми оператора DO

```
'Обчислення в циклі з DO 2-ї форми
```

```
DO WHILE t! <tmax!
i = U / R1+R2 (1-EXP(-P*t))
PRINT "t! = "; t!; TAB(10); "i="; i!
t! = t! + dt!
LOOP
```

3-я форма оператора DO взагалі не передбачає перевірки умови на базі WHILE/UNTIL. Вона описується такою діаграмою

DO Група операторів EXIT DO LOOP

Ця форма оператора передбачає вихід із циклу за допомогою оператора EXIT DO.

```
'Обчислення в циклі з DO 3-ї форми
DO
i = U / R1+R2 (1-EXP(-P*t))
PRINT "t! = "; t!; TAB(10); "i="; i!
t! = t! + dt!
IF t! > tmax! THEN
EXIT DO
END IF
LOOP
```

Наприклад:

```
WHILE PAS$<>"Петунія"
INPUT"Введіть пароль";PAS$
WEND
PRINT "Ласкаво просимо"
```

Цей цикл буде працювати, доки не буде введено правильний пароль.

У блоці команд повинен бути оператор, що впливає на значення логічного виразу.

Наприклад:

```
11.                 I=0
    WHILE I<=10
I=I+1 'оператор, який впливає на значення логічного виразу
    WEND
    PRINT I
```

```
12.                 Приклад нескінченного циклу
I=0
    WHILE I<=10
    INPUT A
    S=S+A
    WEND
    PRINT S
```

У блоці команд немає оператора, який змінював значення I.

Якщо за першої перевірки умови виявиться, що значення логічного висловлювання - " Брехня " , блок команд нічого очікувати виконано жодного разу. Наприклад:

```
I=10
WHILE I<10
I=I+1
WEND
```

Приклад 1. Обчислити суму парних чисел в інтервалі від 1 до 10 включно.

```
'I - парні числа
'Sum - сума парних чисел
I=2
Sum=0
```

```
WHILE I<=10
Sum=Sum+I
I=I+2
WEND
PRINT"Сума парних чисел в інтервалі від 0 до 10 =";Sum
```

Приклад 2. У під'їзді N сходинок. Скільки кроків буде зроблено, якщо крокувати через 3 сходинки.

```
KS - кількість сходинок
' KH - кількість кроків
INPUT "Введіть кількість сходів";
KS=0
KH=0
WHILE KS<=N
KS=KS+3
KH=KH+1
WEND
PRINT"Кількість кроків=";KH
```

Приклад 3 Відома сума номерів сторінок. Визначити номер сторінки.

```
'NS - номер сторінки
'S - сума номерів сторінок
INPUT"Введіть суму номерів сторінок";
NS=0
S=0
WHILE S<Q
NS=NS+1
S=S+NS
WEND
PRINT"Номер сторінки = ";NS
```

Вкладені цикли

Можна організувати складні повторення, використовуючи цикли всередині циклів (Вкладені цикли).

Формат:

FOR

…

FOR ---------

 тіло внутрішнього циклу

NEXT --------

…

NEXT

Приклад 1. Надрукувати таблицю множення до 12 у вигляді

1*2=2

2 * 2 = 4 і т.д.

```
FOR I=2 TO 12
FOR J=1 TO 12
PRINT J;”*”;I;”=”;J*I
NEXT J
NEXT I
```

Приклад 2. Надрукувати таблицю множення до 5 як таблиці.

1 2 3 4 5

2 4 6 8 10 і т.д.

```
FOR I=2 TO 12
FOR J=1 TO 12
PRINT J*I;
NEXT J
PRINT
```

```
NEXT I
```

Приклад 3. Надрукувати всі чотиризначні натуральні числа в десятковому записі яких не мають двох однакових цифр.

```
FOR T=1 TO 9 '
FOR S=0 TO 9
FOR D=0 TO 9
FOR E=0 TO 9
IF T<>S AND T<>D AND T<>E AND S<>D AND S<>E AND D<>E THEN
M=T*1000+S*100+D*10+E
PRINT M,
ENDIF
NEXT E
NEXT D
NEXT S
NEXT T
```

Контрольні запитання:

1. Основне поняття масиви. Що таке одномірний та двовимірний масиви?
2. Заповнення масиву. які дії відбуваются над елементами двовимірних

масивів?

3. Що таке цикл в програмуванні? Формат запису циклу з лічильником?

4. Пояснити призначення оператора цикл з лічильником? Якізначення має крок циклу? Чи завжди він вказується?

5. Як підрахувати кількість циклів?

6. Чи завжди можна замінити оператор циклу з лічильникомоператором умовного циклу і навпаки?

7. Які ще види циклів використовують в програмуванні та як вони працюють?

8. У яких випадках використовуються оператори умовногоциклу?

9. У чому полягає відмінність у вживанні ключових слів While iUntil

10. Поняття вкладені цикли. Як їх використовують у програмуванні?

Оператори для роботи з файлами

Результати обробки програм як дані можна зберігати на магнітних дисках. Це дає можливість використовувати дані для обробки нових програм, завантажуючи їх з диска в оперативну пам'ять. І тому створюють файли даних. За принципом організації файли поділяють на два типи:

- Файли з послідовним доступом;
- Файли з прямим доступом

У файлах з послідовним доступом дані записують у порядку, у якому вони прийшли з програми, а виймаються у зворотному (за принципом: перший зайшов, останній вийшов). У файлах з прямим доступом дані можна розміщувати будь-де, визначені показником запису. Таким чином можна зчитувати дані з файлу в програму.

Утворення дискового файлу.

Для утворення (відкриття) файлу на магнітному диску мовою QBASIC використовують оператор, формат якого має вигляд:

OPEN повне ім'я FOR Режим AS
файлу

номер LEN = цілісне
Файла вираз

OPEN п.і.ф. [FOR Реж.] AS # Н.Ф. [LEN=n],
Де
п.і.ф. - повне ім'я файлу - наприклад "A: REZULT";
реж. - Режим відкриття файлу;

OUTPUT – послідовне виведення даних у зазначений файл із програми;
INPUT - послідовне введення у програму з файлу;
APPEND – послідовне дописування в кінець програми;
Відсутній – прямий доступ до файлу (I/0);
Н.Ф. - Номер файлу; набуває значення від 1 до максимально можливого;
n - вираз типу INTENGER, що визначає довжину запису файл з прямим доступом від 1 до 32767 бат. За замовчуванням це 128 байт. Великий буфер (512 байт) збільшує швидкість обміну.
Оператор OPEN відкриває (утворює) на диску файл даних, відкриває для нього буфер, визначає режим доступу та пов'язує з файлом номер (# Н.Ф.), який використовується в операторах вводу/виводу.

Приклад відкриття файлу прямого доступу:

OPEN "A: \ REZULT" FOR OUTPUT AS #1

У цьому прикладі оператор відкриває на диску А директорії файл під ім'ям REZULT і призначає йому номер 1 для виведення в нього з програми даних в режим послідовного доступу.
Приклад: Відкриття файлу прямого доступу.

OPEN "A: \SUMME" AS #2 LEN=256

У цьому прикладі оператор відкриває на диску А директорії файл з ім'ям SUMME і призначає йому номер 2 для введення/виведення даних в режим прямого доступу з довжиною запису 256 байт.

Виведення даних у файл

Для виведення даних із програми в дисковий файл використовують такий оператор:

WRITE # номер список
Файли змінних
,

WRITE #Н.Ф., список

де # Н.Ф. - Номер відкритого файлу
Список - перелік імен числових або рядкових змінних, розділених комою або крапка з комою.

Закриття дискового файлу

Для закриття дискового файлу використовується оператор, формат якого має вигляд

CLOSE [Н.Ф.[, # Н.Ф….]]

Цей оператор розриває зв'язки між файлом та його ім'ям.
Якщо номер файлу (# Н.Ф) відсутній. То оператор закриває усі відкриті файли. За наявності номера файлу оператор закриває виділений файл. У режимі прямого доступу, крім оператора CLOUSE, відкриті файли також закривають оператори END, NEW, SYSTEM , RUN.

Приклад:

Створити дисковий файл послідовного доступу під назвою “SPISOK”, в якому зберігаються записи для визначення Прізвища, віку студентів. Файл закрити під час введення прізвища NN.

```
'Програма відкриття файлу “SPISOK”
OPEN "A: \ SPISOK" FOR OUTPUT AS #1
DO WHILE – 1!
INPUT "Прізвище"; FAM$
IF FAM$ = "NN" THEN
CLOSE #1
EXIT DO
END IF
INPUT "Вік"; VZT!
WRITE #1, FAM$, VZT!
LOOP
'Кінець програми
```

Введення даних у програму

Для введення даних у програму з файлу використовує оператор, формат якого має вигляд

INPUT # номер список

Файли змінних

,

INPUT # Н.Ф., список

Де

Н.Ф. - Номер відкритого файла;

Список - список імен числових чи малих змінних.

Оператор зчитує з файлу послідовного доступу дані та надає їм значення змінним у списку.

Приклад: З файлу СПИСОК вивести на екран дисплея Прізвища студентів віком менше 25 років.

```
'Програма вибору студентів
OPEN "A: \ SPISOK" FOR INPUT AS #1
DO WHILE – 1!
INPUT #1, FAM$, VZT!
IF ( VZT! <25) AND ( NOT EOF(1)) THEN
  PRINT FAM$

END IF
LOOP
CLOSE #1
'Кінець програми
```

Контрольні запитання:

1.Перерахувати оператори для роботи з файлами?

2.Який оператор використовують для Утворення дискового файлу?

3. За допомогою яких операторів проходить виведення даних у файл, закріття дискового файлу та введення даних у програму? Запишить їх формат запису та як вони працюють?

4. Навести приклад запису введення у програму із файла?

П Р О Ц Е Д У Р И М О В И QBASIC

Мова QBASIC підтримує процедури двох типів:

FUNCTION та SUB.

Процедурою називається група операторів, якій присвоєно ім'я та складається з набору логічно пов'язаних між собою діями. Процедура може бути виконана при зверненні до неї із програми, яка називається головною програмою або головним модулем.

Обмін даними між основним модулем і процедурою здійснюється двома способами:

1. Це використання глобального оголошення типу даних. Дані, декларовані як глобальні, будуть доступні як у програмі, так і в процедурі.

При декларації глобальних змінних та масивів записують зарезервоване слово SHARED.

Наприклад:

DIM SHARED Chifre!

DIM SHARED PRIM %(1 TO 10)

2. Цей спосіб обміну виходить з механізмі формально-фактичних параметрів. І тут процедура використовує формальні параметри, значення яких визначають з допомогою фактичних параметрів при зверненні до процедури. Обидва методи обміну даними можуть бути використані в одній процедурі.

Між фактичними параметрами головного модуля та формальними параметрами процедури необхідно встановити відповідність за 3-ма правилами:

1. Повинна збігатися кількість формальних та фактичних параметрів.
2. Повинен співпадати порядок їхнього слідування.
3. Повинен співпадати тип даних по порядку (другий-другий, третій-третій і т.д.)

Визначення процедур

Чим більше ви створюєте програм і чим більшими вони стають, тим частіше доводиться стикатися з тим, що один і той же блок коду знову і знову з'являється в різних місцях вашої програми (або в декількох програмах). Для коду, що повторюється, треба використовувати процедури.

Процедура (функція) — це програмна одиниця VB, що включає оператори опису її локальних даних і виконувані оператори. Зазвичай в процедуру об'єднують регулярно виконувану послідовність дій, яка вирішує окрему задачу або підзадачу. Тобто процедура — це сегмент коду, який виконує те або інше завдання, а потім передає управління в ту частину коду, з якої він був викликаний. Це означає, що одну і ту ж процедуру можна викликати з різних місць вашої програми і при правильному використанні застосовувати в різних програмах.

Ви вже стикалися з використанням процедур, навіть не знаючи про це. Кожного разу, вводячи код, який виконується у відповідь на якусь подію елемента управління типу Button (або якогось іншого типа), ви створювали процедури, що називаються *обробниками подій*. Обробники подій автоматично викликаються програмою при винекненні тієї або іншої події. Ви можете створювати свої власні процедури і викликати їх у міру потреби. Створені вами процедури часто називаються *процедурами користувача*. І хоча в попередніх прикладах код повністю знаходився в процедурах обробки подій, в реальних програмах, велика кількість коду може знаходитися в окремих процедурах користувача.

Види процедур

Процедури можна класифікувати за кількома ознаками: за способом використання (виклику) в програмі, за способом запуску процедури на виконання, за місцем знаходження коду процедури у проекті.

Процедури VB поділяються на підпрограми і функції. Перші описуються ключовим словом *Sub,* другі — *Function.*

За способом запуску процедур на виконання можна виділити в окрему групу процедури, що запускаються автоматично при виникненні тієї чи іншої події, — ми називаємо їх процедурами обробки подій.

Головне призначення процедур у всіх мовах програмування полягає в тому, що при їхньому виклику змінюється стан програмного проекту — змінюються значення змінних (властивості об'єктів), описаних в модулях проекту.

ПРОЦЕДУРА FUNCTION

Процедура FUNCTION складається з одного або більше операторів, але в головний модуль повертає тільки одне значення.
Ім'я процедури вибирає користувач. Останній символ імені означає тип результату, тобто. тип значення, що повертається.
Формальні параметри записуються через (,) кому.
Останнім в операторі процедури обов'язково має бути оператор присвоєння, який надає імені процедури отриманий результат. Для запуску процедури FUNCTION у головному модулі має бути оператор, який складається із звернення до процедури. Це може бути оператор присвоєння або вивод.

Існує два способи, за допомогою яких процедура отримує і

передає інформацію, змінюючи тим самим стан системи документів.

Перший і основний спосіб полягає у використанні параметрів процедури. При виклику процедури її аргументи, що відповідають вхідним параметрам, повинні отримати значення, які программа виробляє або отримує від зовнішнього середовища. В результаті роботи процедури формуються значення вихідних параметрів, що передаються програмі за посиланням.

Другий спосіб полягає у використанні процедурою глобальних змінних і об'єктів як для одержання, так і для передачі інформації.

Робота з процедурами

Основна ідея, що стоїть за процедурою будь-якого виду, полягає в розподілі програми на послідовність завдань меншого розміру. Кожне з цих завдань можна потім оформити у вигляді процедури (функції). Такий підхід має декілька переваг.

0. Кожне завдання можна тестувати окремо. Чим менше в процедурі коду, тим легше проводити налагодження і легко займатися проектом спільно з іншими розробниками.
1. Можна позбавитися від зайвого коду, викликаючи при кожному виконанні завдання відповідну процедуру, а не повторюючи в програмі один і той же код.
2. Можна створити бібліотеку процедур, щоб використовувати їх в різних програмах і тім самим економити час при розробці нових проектів.
3. Стає більш легко підтримувати програми, оскільки, по-перше, якщо код не повторюється, то редагувати блок коду потрібно один раз. Крім того, відділення один від одного ключових компонентів (наприклад, призначеного для користувача інтерфейсу і засобів роботи з базами даних) дозволяє робити

значні зміни в якійсь частині програми, не переписуючи її всієї.

Синтаксис процедур і функцій

Опис процедури Sub в VBA має такий вигляд:

```
[Private | Public] [Static] Sub ім'я([список-аргументів])
 тіло-процедури
End Sub
```

Ключове слово Public в заголовку процедури використовується для того, щоб оголосити процедуру загальнодоступною, тобто дозволити викликати її зі всіх інших процедур всіх модулів будь-якого проекту. Якщо модуль, в якому описана процедура, містить закриваючий оператор Option Private, то процедура буде доступна лише модулям свого проекту. Альтернативний ключ Private використовується, щоб закрити процедуру від всіх модулів, крім того, в якому вона описана. За умовчанням процедура вважається загальнодоступною.

Ключове слово Static означає, що значення локальних (оголошених в телі процедури) змінних зберігатимуться в проміжках між викликами процедури (використовувані процедурою глобальні змінні, описані поза її тілом, при цьому не зберігаються).

Параметр ім'я — це ім'я процедури, що задовольняє стандартним умовам VB на імена змінних.

Необов'язковий параметр список-аргументів — це послідовність розділених комами змінних, задаючих передавані процедури при виклику параметрів.

Аргументи, або, по-іншому, формальні параметри, що задаються при описі процедури, завжди представляють лише імена

(ідентифікатори). В той же час при виклику процедури її аргументи
— фактичні параметри — можуть бути не лише іменами, але і виразами.

Послідовність операторів тіло — процедури задає програму виконання процедури. Тіло процедури може включати як «пасивні» оператори оголошення локальних даних процедури (змінних, масивів, об'єктів і ін.), так і «активні», які змінюють стани аргументів, локальних і зовнішніх (глобальних) змінних і об'єктів. У тіло можуть входити також оператори Exit Sub, що призводять до негайного завершення процедури і передачі управління в зухвалу програму. Кожна процедура в VB визначається окремо від інших, тобто тіло однієї процедури не може включати опису інших процедур і функцій.

Розглянемо детальніше структуру одного аргументу зі списку аргументів.

[Optional] [ByVal | ByRef] [ParamArray] змінна[( )][As тип] [= значення-за-умовчанням']

Відзначимо одну цікаву особливість, яку не слід використовувати, але яку слід враховувати, — VB допускає, щоб фактичне значення аргументу, передаваного за посиланням, було константою або вираженням відповідного типу. В цьому випадку даний аргумент розглядається як передаваний за значенням, і не видається жодних повідомлень про помилку. Ключове слово Optional означає, що заданий ним аргумент є можливим — його необов'язково задавати у момент виклику процедури. Для таких аргументів можна задати значення за умовчанням. Необов'язкові аргументи завжди поміщаються в кінці списку аргументів.

Альтернативні ключі ByVal і ByRef визначають спосіб передачі аргументу в процедуру.

ByVal означає, що аргумент передається за значенням, тобто при виклику процедури створюватиметься локальна копія змінної з початковим передаваним значенням і зміни цій локальній змінній під час виконання процедури не відіб'ється на значенні змінної, що передала своє значення в процедуру при виклику. Передача за значенням можлива лише для вхідних параметрів, які передають інформацію в процедуру, але не є результатами. Для таких параметрів передача за значенням частенько зручніше, ніж передача по засланню, оскільки у момент виклику аргумент може бути заданий скільки завгодно складним вираженням. Відмітимо, що вхідні параметри, що є об'єктами, масивами або змінними призначеного для користувача типу, передаються по засланню, що дозволяє уникнути створення копій. Вирази над такими аргументами все одно недопустимі, тому передача за значенням втрачає сенс.

ByRef означає, що аргумент передається по засланню, тобто всі зміни значення передаваній змінній при виконанні процедури безпосередньо відбуватимуться із змінною — аргументом з тієї програми, що викликала дану процедуру. У VB за умовчанням аргументи передаються по засланню (ByRef). Це не зовсім зручно для програмістів, які звикли до інших мов (наприклад, Паскаль або C), де за умовчанням аргументи передаються за значенням. Тому при описі процедури рекомендують явно вказувати спосіб передачі кожного аргументу, навіть якщо цей аргумент зустрічається в лівій частині.

Процедура VB допускає необов'язкові аргументи, які можна опустити у момент виклику. Узагальненням такого підходу є можливість мати змінне, заздалегідь не фіксоване число аргументів. Досягається це за рахунок того, що один з параметрів (останній в списку) може задавати масив аргументів — в цьому випадку він

задається з описувачем ParamArray. Якщо список аргументів включає масив аргументів ParamArray, то ключ Optional використовувати в списку не можна. Ключове слово ParamArray може з'являтися перед останнім аргументом в списку з метою вказати, що за аргумент — масив з довільним числом елементів типу Variant. Перед ним не можна використовувати ключі ByVal, ByRef або Optional.

Змінна — це ім'я змінної, що представляє аргумент. Якщо після імені змінної задані круглі дужки, то це означає, що відповідний параметр є масивом.

Параметр тип задає тип значення, передаваного в процедуру. Він може бути одним з базисних типів VBA (не допускаються лише рядки String з фіксованою довжиною). Обов'язкові аргументи можуть також мати тип визначеного користувачем запису або класу. Якщо тип аргументу не вказаний, то за умовчанням йому приписується тип Variant. Тип може бути одним з типів Office 2000. Для необов'язкових (Optional) аргументів можна явно задати значення за умовчанням. Це константа або константне вираження, значення якого передається в процедуру, якщо при її виклику відповідний аргумент не заданий. Для аргументів типу об'єкт (Object) як значення за умовчанням можна задати лише Nothing.

Синтаксис визначення процедур-функцій схожий на визначення звичайних процедур:

```
[public | Private] [Static] Function ім'я [(список-
    аргументів)] [As тип-значення]тіло-функції
End Function
```

Відмінність лише в тому, що замість ключового слова Sub для оголошення функції використовується ключове слово Function, а після списку аргументів слід вказати тип значення, що повертається функцією. У тілі функції має бути використаний оператор

привласнення виду: ім'я = вираження

Тут, в лівій частині оператора, стоїть ім'я функції, а в правій — значення виразу, який задає результат обчислення функції. Якщо при виході з функції значення змінної ім'я явно не привласнене, то функція повертає значення відповідного типу, визначене за умовчанням. Для числових типів це 0, для рядків — рядок нульової довжини (""), для типу Variant функція поверне значення Empty, а для заслань на об'єкти — Nothing.

Щоб негайно завершити обчислення функції і вийти з неї, в тілі функції можна використовувати оператор: **Exit Function**

Основна відмінність процедур від функцій полягає в способі їх використання в програмі. Наступна функція cube повертає аргумент, піднесений в куб:

```
Function cube(ByVal N As Integer) As Longcube=
N*N*N
End Function
```

Виклик цієї функції може мати вигляд

```
Dim x As Integer, в As Integerв = 2
x =cube(y+3)
```

Вже говорилося, що будь-яку функцію можна перетворити в еквівалентну їй процедуру, при цьому з'являється додатковий параметр, необхідний для завдання результату. Отже, в еквівалентної процедури cube1 два аргументи:

```
Sub cube1 (ByVal N As Integer, ByRef 3 As Long)
C= N*N*N ' здобуття результату в змінній, заданій по засланню
End Sub
```

Її можна використовувати для такого ж піднесення в куб: cubel y+3, x.

Але це не означає, що не має значення, який вид процедур слід використовувати в програмі. Якби вираз, в якому бере участь

функція, був складніше, наприклад:

x = cube(y)+sin(cube(x))

то його обчислення за допомогою процедури cube1 потребувало б виконання декількох операторів і введення додаткових змінних:

cube1 y,z: cube1 x,u : x=z+sin(u)

Як бачите, поводитися з функціями, що володіють побічним ефектом, слід обережно. У прикладі результат обчислення суми трьох доданків залежить від порядку їх запису. Це заперечує основним принципам математики. Більш того, слід розуміти, що результат обчислення непередбачуваний, оскільки VBA може для збільшення ефективності змінювати порядок дій при обрахуванні арифметичних виразів. Тому краще не використовувати у вираженні виклик функції з побічним ефектом, що змінює значення тих змінних, що входять в нього.

Виклики процедур і функцій

Виклик звичайної процедури Sub з іншої процедури можна оформити по-різному. Перший спосіб:

Ім'я список фактичних параметрів

де Ім'я — ім'я процедури, що викликається, а список фактичних параметрів — список аргументів, передаваних процедурі, він повинен відповідати списку аргументів, заданому в описі процедури. Задати цей список можна різними способами. У простому випадку значення передаваних процедурі аргументів перераховуються через кому в тому ж порядку, що і в списку аргументів із заголовка процедури.

Може виявитися, що в одному проекті декілька модулів містять процедури з однаковими іменами. Для відмінності таких процедур потрібно при їх виклику вказувати ім'я процедури через крапку після імені модуля, в якому вона визначена. Наприклад, якщо кожен з двох модулів Mod1 і Mod2 містить визначення процедури ReadData, а в процедурі Myproc потрібно скористатися процедурою Mod2, то цей виклик буде мати вигляд:

```
Sub Myproc() Mod2.ReadDate
 End Sub
```

Якщо потрібно використовувати процедури з однаковими іменами з різних проектів, додайте до імен модуля і процедури ім'я проекту. Наприклад, якщо модуль Mod2 входить в проект MyBook, той же виклик можна уточнити так:

```
MyBook.Hod2.ReadData
```

Другий спосіб виклику процедур пов'язаний з використанням оператора Call. В цьому випадку виклик процедури виглядає так:

```
Call ім'я(список-фактичних-параметрів)
```

Зверніть увагу на те, що в цьому випадку *список фактичних параметрів* поміщено в круглі дужки, а в першому випадку — ні. Спроба викликати процедуру без оператора Call, але із завданням круглих дужок, є джерелом синтаксичних помилок, особливо для розробників з великим досвідом програмування на Паскаль або С, де списки параметрів завжди в дужках. Слід звернути увагу на одну важливу і, мабуть, неприємну особливість виклику процедур VBA. Якщо процедура VBA має лише один параметр, то вона може бути викликана без оператора Call, з використанням круглих дужок і не повідомлення про помилку виклику. Це було б не так страшно, якби повертався правильний результат. На жаль, це не так. Проілюструємо сказане прикладом:

```
Public Sub MyInc(ByRef X As Integer) X = X + 1
```

```
End Sub
Public Sub TestInc()
Dim X As Integer X = 1
'Виклик процедури з параметром в дужках 'синтаксично
допустимо, але працює некоректно'Mylnc (X)
Debug.Print X 'Коректний виклик
Mylnc X Debug.Print X
'Це теж коректний викликCall MyInc(X)
Debug.Print X
End Sub
```

Результати роботи:

1

2

3

Хоча перший раз процедура викликається нормально ізбільшує значення результату, але після закінчення її роботи значення аргумента не змінюється. У цій ситуації не діє описувач ByRef, виклик йде так, ніби параметр описаний з описувачем ByVal.

Якщо ж процедура має більш ніж один параметр, то спроба викликати її, взявши параметри в круглі дужки і без ключьового слова Call, призводить до синтаксичної помилки. Ось простий приклад:

```
Public Sub SUMXY(ByVal X As Integer, ByVal Y As Integer,
  ByRef Z As Integer)
Z = X + Y
End Sub
Public Sub TestSumXYO
Dim a As Integer, b As Integer, з As Integera = 3: b = 5
'SUMXY (a, b, з) 'Синтаксична помилкаSUMXY a, b,
```

```
з
Debug.Print з
End Sub
```

В даному прикладі некоректний виклик процедури SUMXY буде виявлений на етапі перевірки синтаксису.

Розглянемо ще одну особливість виклику VB-процедур, пов'язану з аргументами, передаваними по засланню. Як правило, в мовах програмування для таких аргументів можливе значення фактичного параметра обмежується, — воно має бути ім'ям змінної, заслання на яку передається процедурі. VB допускає можливість завдання для таких аргументів констант і виразів у момент виклику. Всі ці допуски роблять мову менш надійною і призводять до серйозних помилок.

Виклики функцій. Оформлення виклику функції залежить від того, чи потрібно використовувати її значення в процедурі. Якщо ви хочете передати обчислюване функцією значення в змінну або застосувати його у вираженні правої частини оператора привласнення, то виклик призначеної для користувача функції має той же вигляд, що і виклик вбудованої функції, наприклад, sin(x). При виклику вказується ім'я функції, а після нього йде список фактичних параметрів в круглих дужках. Наприклад, якщо заголовок функції Myfunc:

```
Func Myfunc(Name As String, Age As Integer, Newdate As Date) As
  Integer
```

то використовувати її значення можна за допомогою викликів:

```
val= Myfunc("Alex",25, "10/04/97")
```

або

```
x = sqrt(Myfunc("Alex",25, "10/04/97"))+ x
```

Якщо ж значення, що обчислюється функцією, вас не цікавить

і потрібно скористатися лише її побічними ефектами, то виклик функції може мати ту ж форму, що і виклик процедури Sub. Наприклад:

Myfunc "Alex",I, "10/04/97"

або:

Call Myfunc(Myson, 25, DateOfArrival)

Використання іменованих аргументів

Аргументи масиви

Аргументи процедури можуть бути масивами. Процедурі передається ім'я масиву, а його розмірність визначається вбудованими функціями LBound і UBound. Наведемо приклад процедури, що обчислює скалярний добуток векторів:

```
public Function ScalarProduct(X() As Integer, Y() As Integer) As
  Integer •Обчислює скалярний добуток двохвекторів.
  'Передбачається, що кордони масивів збігаються.
Dim i As Integer, Sum As Integer Sum = 0
For i = LBound(X) To UBound(X) Sum = Sum + X(i)* Y(1)Next i
ScalarProduct = Sum
End Function
```

Обидва параметри процедури, передавані по засланню, є масивами. Робота з ними в тілі процедури не викликає утруднень завдяки тому, що функції LBound і UBound дозволяють встановити кордони масиву по будь-якому виміру. Наведемо програму, в якій викликається функція ScalarProduct:

```
Public Sub TestScalarProductQ Dim A(1 To 5) As Integer Dim B(1 To
  5) As Integer Dim 3 As Variant Dim Res As Integer Dim i As Integer
3 =          2, 3, 4,  5) For i = 1 Te 5 A(i)= C(i —
Array(1,                1)
Next i
3 =          4, 3, 2,  1)
Array(5,
For i = 1 Te 5
B(i)= C(i — 1) Next i Res = ScalarProduct(A, B) Debug.Print
  Res
End Sub
```

Інколи, коли в процедуру слід передати лише один масив, можна використовувати конструкцію РагатАггау. Наступна процедура PosNeg прочитує суми надходжень Positive і витрат Negative, вказаний у масиві Sums:

```
    Sub PosNeg(Posltlve As Integer, Negative As Integer,
   ParamaArray Sums() As Variant)
 Dim I As Integer
 Positive = 0: Negative = 0
 For I = 0 To UBound(Sums())' цикл по всіх елементах масиву
 If Sums(I)> 0 Then
 Positive = Positive + Sums(I) Else
 Negative = Negative — Sums(I) End If
   Next I
     End Sub
```

Виклик процедури PosNeg може мати вигляд:

```
Public Sub TestPosNeg()
Dim Incomes As Integer, Expences As Integer
PosNeg Incomes, Expences -20, 100, 25, -44, -23, -60,
```

Debug.Print Incomes, Expences End Sub

В результаті змінна Incomes набуде значення 245, а змінна Expences — 147. Відмітьте, перевагою використання масиву аргументів ParamArray є можливість безпосереднього перерахування елементів масиву у момент виклику.

Проте таке використання масиву аргументів ParamArray не вичерпує всіх його можливостей. У складніших ситуаціях передавані аргументи можуть мати різні типи. Наведемо приклад.

Завдання про медіану

Для масиву M і елемента Cand обчислити різницю між числом елементів масиву M, більших і менших, ніж Cand.

Це варіація завдання про медіану — «середній» елемент масиву. Медіану можна визначити, наприклад, таким алгоритмом: упорядкувавши масив, узяти елемент, що знаходиться в середині. Є і ефективніші алгоритми. Але обмежимося простішим завданням — перевіркою на «медіанність». Відмітимо: якщо всі елементи масиву M різні і число їх непарне, то для медіани шукана в завданні різниця дорівнює 0. У загальному випадку значення різниці є мірою наближення параметра Cand до медіани масиву M.

Але займемося аспектами програмування цього завдання. У функції, що реалізовує це завдання, на вході — масив, а на виході — скаляр. Хотілося б, щоб ця функція могла викликатися у формулах робочої сторінки, а як фактичний параметр їй могли б бути передані як offeV1 """•-' ' дана IsMediana:

Public Function IsMediana(M As Variant, Cand As Variant) As Integer

•даний масив M і елемент Cand. Як результат

повертається

•різниця між числом елементів масиву M, більших і менших ніж

Cand. Dim i As Integer, j As Integer niffi Pos As Integer, Neg As

Integer pos = 0: Neg = 0

•Аналіз типа параметра M

If TypeName(M)= "Range" Then For 1 = 1 To M.Rows.Count For j = 1

To M.Columns.Count If M.Cellsd, j) > Cand

Then

Pos = Pos + 1 Elself M.Cellsd, j) < Cand Then

Neg = Neg + 1 End If Next j Next i IsMediana = Pos — Neg Elself

TypeName(M)= "VARIANTO" Then 'TypeName is "VARIANTO"

'Це масив, але не зовсім справжній, для нього не визначені,

'наприклад, функції кордонів: LBound, UBound. Dim Val As

Variant For Each Val In M If Val > Cand Then

Pos = Pos + 1 Elself Val < Cand Then

Neg = Neg + 1 End If Next Val IsMediana = Pos — NegElself

TypeName(M)= "INTEGERO" Then

'Це справжній масив цілих VBA, для якого 'визначені функції

кордонів. For i = LBound(M) To UBound(M) IfM(i)> Cand Then

Pos = Pos + 1 Elself M(i)< Cand Then

Neg = Neg + 1 End If Next i IsMediana = Pos — Neg Else

MsgBox ("При виклику функции:IsMediana(M,Cand)" & "- M

не є масивом або об'єктом Range!") End IfEnd Function

Рекурсивні процедури

Рекурсія відбувається, якщо функція або підпрограма викликає сама себе.

Пряма рекурсія (direct recursion) виглядає приблизно так:

```
Function Factorial(num As Long) As LongFactorial =
   num * Factorial(num — 1)
End Function
```

В разі непрямої рекурсії (indirect recursion) рекурсивна процедура викликає іншу процедуру, яка, у свою чергу, викликає першу:

```
Private Sub Ping(num As Integer)Pong(num — 1)
End Sub

Private Sub Pong(num As Integer)Ping(num / 2)
End Sub
```

Стандартний приклад рекурсивної процедури – функція — факторіал Fact(N)= N!. Її визначення в VB:

```
Function Fact(n As Integer) As Long
If n <= 1 Then        ' базис індукції.
   Fact = 1           ' 0! =1
   Else               ' рекурсивний виклик в разі N > 0.
      Fact = Fact(n — 1)* nEnd If
End Function
```

Спочатку ця функція перевіряє, що число менше або дорівнює 0. Факторіал для чисел менше нуля не визначений, але ця умова перевіряється для підстраховування. Якби функція перевіряла лише умову дорівнюваності числа нулю, то для від’ємних чисел рекурсія була б нескінченною.

Якщо вхідне значення менше або дорівнює 0, функція

повертає значення 1. В інших випадках значення функції дорівнює добутку вхідного значення на факторіал від вхідного значення, зменшеного на одиницю. Те, що ця рекурсивна функція врешті-решт зупиниться, гарантується двома фактами. По-перше, при кожному подальшому виклику значення параметра num зменшується на одиницю. По-друге, значення num обмежене знизу нулем. Коли num дорівнюватиме 0, функція зупиняє рекурсію. Умова, наприклад, в даному випадку умова num<=0, називається умовою зупинки рекурсії.

При кожному виклику підпрограми система зберігає ряд параметрів в системному стеку. Оскільки цей стек відіграє важливу роль, інколи його називають просто стеком. Якщо рекурсивна функція викличе себе надто багато разів, вона може вичерпати стековий простір і аварійно завершити роботу з помилкою «Out of stack space».

Число разів, яке функція може викликати сама себе до того, як використає весь стековий простір, залежить від об'єму встановленої на комп'ютері пам'яті і кількості даних, що поміщаються програмою в стек. В одному з тестів програма вичерпала стековий простір після 452 рекурсивних викликів. Після зміни рекурсивної функції таким чином, щоб вона визначала 10 локальних змінних при кожному виклику, програма могла викликати себе лише 271 раз.

Оскільки кожен виклик процедури вимагає накладних витрат, ефективніше для факторіалу ітеративна програма:

```
Function Fact1(n As Integer) As LongDim Fact As
Long, i As Integer
Fact = 1       ' 0! =1.
If n > 1 Then     ' цикл замість рекурсії.
    For i = 1 To n
```

```
        Fact = Fact * iNext i
End If
Fact1 = FactEnd Function
```

Наведемо процедуру, що оцінює час виконання рекурсивного і нерекурсивного варіантів:

```
Private Sub cmStart_Click()
' Порівняння за часом рекурсивної і не рекурсивної реалізації
  факторіалу.
Dim i As Long, Res As LongDim n As
Integer
Dim Start As Single, Finish As Single
'Рекурсивне обчислення факторіалу
n = txInput.TextStart = Timer
For i = 1 To 100000
 Res = Fact(n)
 Next i
 Finish = Timer
 lbOutput.Caption = lbOutput.Caption & vbCrLf & _ "Час
 рекурсивних обрахувань:" & (Finish — Start) 'Не рекурсивне
 обчислення факторіалу
 Start = Timer
 For i = 1 To 100000
 Res = Fact1(n)Next i
 Finish = Timer
 lbOutput.Caption = lbOutput.Caption & vbCrLf & _ "Час
 нерекурсивних обрахувань:" & (Finish — Start)End Sub
```

Результати обчислень рис.5.1 наведені для двох запусків тестів процедури.

Як бачите, в даному випадку нерекурсивний варіант працює в

три рази швидше. Окрім проблем з часом виконань, рекурсивні процедури можуть легко вичерпати і стекову пам'ять, в якій розміщуються аргументи кожного рекурсивного виклику. Тому уникайте неконтрольованого розмноження рекурсивних викликів і замінюйте рекурсивні алгоритми на ітеративні там, де використання рекурсії по суті не потрібне.

Обчислення найбільшого загального дільника

Найбільшим загальним дільником (greatest common divisor, GCD) двох чисел називається найбільше ціле, на яке діляться два числа без залишку. Наприклад, найбільший загальний дільник чисел 12 і 9 дорівнює 3. Два числа називаються взаємно простими (relativelyprime), якщо їхній найбільший загальний дільник дорівнює 1.

Математик Ейлер, що жив у вісімнадцятому столітті, виявив цікавий факт: якщо A без остачі ділиться на B, то GCD(A, B)= A.

Інакше GCD(A, B)= GCD(B Mod A, A).

Цей факт можна використовувати для швидкого обчислення найбільшого загального дільника. Наприклад:

GCD(9, 12) = GCD(12 Mod 9, 9) = GCD(3, 9) = 3

На кожному кроці числа стають все менше, оскільки 1<=B Mod A<A, якщо A не ділиться на B без залишку. У міру зменшення аргументів, врешті-решт, A набуде значення 1. Оскільки будь-яке число ділиться на 1 без залишку, на цьому кроці рекурсія зупиниться. Таким чином, в якій-небудь момент B розділиться на A без залишку, іробота процедури завершиться.

Відкриття Ейлера закономірним чином наводить до рекурсивного алгоритму обчислення найбільшого загального дільника:

```
public Function GCD(A As Integer, B As Integer) As Integer
  If B Mod A = 0 Then   ' Чи ділиться B на A без залишка?
      GCD = A           ' Так. Процедура завершена.
  Else
      GCD = GCD(B Mod A, A)   ' Немає. Рекурсія.
  End   If End Function
```

Щоб проаналізувати час виконання цього алгоритму, необхідно визначити, наскільки швидко зменшується змінна A. Оскільки функція зупиняється, коли A доходить до значення 1, то швидкість зменшення A дає верхній кордон оцінки часу виконання алгоритму. Виявляється, при кожному другому виклику функції GCD параметр A зменшується, принаймні, в 2 рази.

Припустимо, A < B. Ця умова завжди виконується при першому виклику функції GCD. Якщо B Mod A <= A/2, то при наступному виклику функції GCD перший параметр зменшиться, принаймні, в 2 рази, і доказ закінчений.

Передбачимо зворотне. Допустимо, B Mod A > A / 2. Першим рекурсивним викликом функції GCD буде GCD(B Mod A, A).

Підстановка у функцію значення B Mod A і A замість A і B дає наступний рекурсивний виклик GCD(B Mod A, A).

Але ми передбачили, що B Mod A > A / 2. Тоді B Mod A розділиться на A лише один раз, із залишком A – (B Mod A). Оскільки B Mod A більше, ніж A / 2, то A – (B Mod A) повинне бути менше, ніж A / 2. Значить, перший параметр другого рекурсивного виклику функції GCD менший, ніж A / 2, що і потрібно було довести.

Передбачимо тепер, що N — це вихідне значення параметра A. Після двох викликів функції GCD значення параметра A повинне зменшитись, принаймні, до N / 2. Після чотирьох викликів це

значення буде не більше, ніж (N / 2) / 2 = N / 4. Після шести викликів значення не перевершуватиме (N / 4) / 2 = N / 8. У загальному випадку після 2 * K викликів функції GCD значення параметра A буде не більше, ніж N / 2K.

Оскільки алгоритм повинен зупинитися, коли значення параметра A дійде до 1, він може продовжувати працю лише до тих пір, поки не виконається рівність N/2K=1. Це відбувається, коли N=2K або коли K=log2(N). Оскільки алгоритм виконується за 2*K кроків, це означає, що алгоритм зупиниться не більш, ніж через 2*log2(N) кроки. З точністю до постійного множника це означає, що алгоритм виконується за час порядку O(log(N)). Алгоритми порядку O(log(N)) зазвичай виконуються дуже швидко, і алгоритм знаходження найбільшого загального дільника не є виключенням з цього правила. Наприклад, щоб знайти, що найбільший загальний дільник чисел 1.736.751.235 і 2.135.723.523 дорівнює 71, функція викликається всього 17 разів. Фактично алгоритм практично миттєво обчислює значення, що не перевищують максимального значення числа у форматі long , — 2.147.483.647. Функція Visual Basic Mod не може оперувати значеннями, більшими за це, тому це практична межа для даної реалізації алгоритму.

Користь від рекурсивних процедур більшою мірою може виявитися при обробці даних, що мають рекурсивну структуру (скажімо, ієрархічну або мережеву).

Небезпеки рекурсії

Рекурсія може служити потужним методом розбиття великих завдань на частини, але вона таїть у собі кілька небезпек. У цьому розділі ми розглянемо деякі з цих небезпек і пояснимо, коли варто, а

коли не варто використовувати рекурсію.

Нескінченна рекурсія

Найбільш очевидна небезпека рекурсії полягає в нескінченній рекурсії. Якщо неправильно побудувати алгоритм, то функція може пропустити умову зупинки рекурсії й буде виконуватися нескінченно. Найпростіше зробити цю помилку, якщо просто забути про перевірку умови зупинки, як це зроблено в наступній помилковій версії функції факторіала. Оскільки функція не перевіряє, чи досягнута умова зупинки рекурсії, вона буде нескінченно викликати саму себе.

```
Private Function BadFactorial(num As Integer) As Integer
  BadFactorial = num * BadFactorial (num — 1) End Function
```

Функція також може викликати себе нескінченно, якщо умова зупинки не припиняє всі можливі шляхи рекурсії. У наступній помилковій версії функції факторіала, функція буде нескінченно викликати себе, якщо вхідне значення — не ціле число, або якщо воно менше 0. Ці значення не є припустимими вхідними значеннями для функції факторіала, тому в програмі, що використовує цю функцію, може знадобитися перевірка вхідних значень. Звичайно, при виклику підпрограми система виконує три речі. По-перше, зберігає дані, які потрібні їй для продовження виконання після завершення підпрограми. По-друге, вона проводить підготовку до виклику підпрограми й передає їй керування. По-третє, коли викликувана процедура завершується, система відновлює дані, збережені на першому кроці, і передає керування назад у відповідну кроці.

Контрольні запитання:

1. Що таке поняття процедура та які види процедур Вам відомі?
2. Як працює процедура Function та які оператори вона використовує у головному модулі програми?
3. Як рацює процедура SUB та які оператори вонавикористовує у головному модулі програми?
4. Чим відрізняються процедури: Function від SUB, та як вони працюють?
5. Чим відрізняються формальні параметри від фактичних та які параметри використовує процедура, а які головний модуль програми?
6. **Поняття «Рекурсивні процедури». Які небезпеки становить рекурсія?**
7. **Що таке нескінчена рекурсія?**

Розділ IX

ГРАФІЧНІ МОЖЛИВОСТІ QBASIC

Графічна обробка даних

У графічному режимі екран є мозаїкою точок (пікселів), кожна з яких може бути пофарбована в той чи інший колір. За допомогою програм, по-різному забарвлюючи точки, ви можете формувати геометричні фігури, малювати діаграми та графіки функцій, барвисті картинки та карикатури (пейзажі, людей, тварин) тощо. Комбінуючи програмними засобами можна створювати різні візуальні ефекти – від ошатного калейдоскопа (це найпростіше) до імітації руху людей та предметів (це називається анімацією).

- **Графічні режими екрана**

Існують два режими роботи з екраном - текстовий іграфічний.

Текстовий режим дозволяє виводити 80 символів в одному рядку й містить на екрані 25 рядків (один - службовий). Такі параметри екрана встановлюються при вмиканні комп'ютера й зберігаються при роботі QBasic.

В QBasic існують спеціальні графічні оператори для створення зображень. Вони вимагають перемикання в графічний режим роботи екрана оператором **SCREEN**.

Графічні режими характеризуються кількістю точок по вертикальній і горизонтальній осях екрана.

SCREEN режим% [,[перемик_кольору] [,активн_стор][,видима _стор]]]

Вибір графічного режиму визначається параметром

- *режим%,* що супроводжує цей оператор.

0-текстовий; 1,7,13 – *320*200*; 8 – *640*200*; 9 – *640*350*; 11,12 – *640*480*

- *перемик_кольору* – 0/1 – встановлює монохромний або кольоровий режим;
- *активн_стор* – сторінка екрана, у яку записується виведення тексту або графіки;
- *видима_стор* - – сторінка екрана, відображувана на екрані в цей момент.

Приклад

SCREEN 2

- ***Кодування графічних зображень. Принципи поданнязображень***

Комп'ютерні графіки – розділ інформатики, що вивчає роботу на комп'ютері із графічними зображеннями (рисунками, кресленнями, фотографіями, відеокадрами та ін.)

Найпоширенішими є два принципи подання зображень: векторний і растровий.

Растрова графіка – засоби й методи комп'ютерної графіки, що використовують растрове подання графічної інформації у вигляді сукупності кодів пікселів, складових зображення.

Піксель – найменший елемент зображення на екрані. Чим щільніше розташовані пікселі, тим краще виглядає зображення на екрані. Щільність пікселів виміряється як їх кількість на одиницю довжини; найпоширеніша одиниця – dpi – кількість точок на дюйм (як правило, 72 або 96 dpi).

Растр – прямокутна сітка пікселів на екрані.

Розв'язувальна здатність екрана – розмір сітки растра (**M** x **N**-добуток числа точок по горизонталі на число точок по вертикалі).

Відеоінформація – інформація про зображення, якавідтворюється на екрані комп'ютера і зберігається в пам'яті.

Відеопам'ять – оперативна пам'ять, що зберігає відеоінформацію під час її відтворення в зображенні на екрані.

Сторінка – частина відеопам'яті, що містить інформацію про один образ екрана (однієї “картинки” на екрані). У відеопам'яті можуть одночасно розміщатися кілька сторінок.

Графічний файл – файл, що зберігає інформацію про графічне зображення (.bmp, .jpeg)

Число кольорів, відтворених на екрані дисплея (**H**) можна підрахувати, знаючи число біт, що відводяться у відеопам'яті під кожний піксель (**n**)

$$H=2^n$$

де **n - бітова глибина** (довжина коду пікселя)n=1-

монохромний монітор, 2 кольор;

n=24 - кольоровий монітор, 16777216 кольорів

Приклад

Для “маленького монітора” з растровою сіткою 10 х10 і чорно-білим зображенням представимо літеру “К” у вигляді бітової матриці (рисунок 3).

	1	2	3	4	5	6	7	8	9	10	0	0	0	0	0	0	0	0	0	0
1											0	0	0	1	0	0	0	1	0	0
2											0	0	0	1	0	0	1	0	0	0
3											0	0	0	1	0	1	0	0	0	0
4											0	0	0	1	1	0	0	0	0	0
5											0	0	0	1	0	1	0	0	0	0
6											0	0	0	1	0	0	1	0	0	0
7											0	0	0	1	0	0	0	1	0	0
8											0	0	0	0	0	0	0	0	0	0
9											0	0	0	0	0	0	0	0	0	0
10																				

Рисунок 3.

Все різноманіття фарб на екрані отримують шляхом змішування трьох базових кольорів: червоного, синього й зеленого.

Це змішування аналогічно змішуванню акварельних фарб на папері, але з однією відмінністю: колір акварельних фарб виходить у результаті відбиття падаючого на них світла, у той час, як колір на екрані формується в результаті випромінювання світла. Тому, коли ви змішуєте на папері три основні фарби, то одержуєте чорний колір, а при цих же кольорах максимальної яскравості на екрані виходить білий колір.

Кожний піксель на екрані складається із трьох близько

розташованих елементів, що світяться цими кольорами. Кольорові дисплеї, що використовують такий принцип, називаються RGB - моніторами. Код кольору пікселя містить інформацію про частку кожного базового кольору.

Якщо всі три складові мають однакову інтенсивність (яскравість), то з їхніх сполучень можна одержати 2^3=8 різних кольорів. Таблиця показує кодування 8-кольорової палітри за допомогою 3-розрядного двійкового коду. У ній наявність базового кольору позначена 1, а відсутність - 0.

Таблиця

Двійковий код 8-кольорової палітри			
к	**з**	**с**	**Колір**
0	0	0	Чорний
0	0	1	Синій
0	1	0	Зелений
0	1	1	Блакитний
1	0	0	Червоний
1	0	1	Рожевий
1	1	0	Коричневий
1	1	1	Білий

Таблиця показує кодування 16-кольорової палітри за допомогою 4-розрядного двійкового коду..

Таблиця

Двійковий код 16-кольорової палітри					
И	**к**	**з**	**с**	**10-й код**	**Колір**
0	0	0	0	0	Чорний
0	0	0	1	1	Синій
0	0	1	0	2	Зелений
0	0	1	1	3	Блакитний
0	1	0	0	4	Червоний

0	1	0	1	5	Рожевий
0	1	1	0	6	Коричневий
0	1	1	1	7	Сірий
1	0	0	0	8	Темно-сірий
1	0	0	1	9	Яскраво-синій
1	0	1	0	10	Яскраво-зелений
1	0	1	1	11	Яскраво-блакитний
1	1	0	0	12	Яскраво-червоний
1	1	0	1	13	Яскраво-рожевий
1	1	1	0	14	Яскраво-жовтий
1	1	1	1	15	Білий

Перший стовпець- код інтенсивності, він управляє яскравістю всіх трьох кольорів одночасно

Ця таблиця відповідає номерам кольорів при використанні графіки в QBasic. Наприклад, жовтому відповідає шістнадцяткове число 1110H або 14 в десятковій системі.

Більша кількість кольорів виходить при роздільному управлінні інтенсивністю базових кольорів. Причому інтенсивність може мати більше 2 рівнів, якщо для кодування кожного з базових кольорів виділяти більше одного біта.

Приклад

При використанні бітової глибини 8 біт/піксель кількість кольорів 2^8=256. Біти такого коду розподілені в такий спосіб:

КККЗЗЗСС.

Червоний і зелений компоненти мають 2^3=8 рівнів яскравості, а синій – 4.

Відзначимо, що растрова графіка працює з реалістичними (фото) зображеннями.

Графічні растрові файли мають великий обсяг (потрібен стиск при зберіганні).

При зміні розмірів, обертанні й інших перетвореннях рисунка відбувається його перекручування.

Векторна графіка – засоби й методи комп'ютерної графіки, що використовують таке подання графічної інформації у вигляді графічних примітивів - сукупності простих елементів: прямих ліній, дуг, кіл, прямокутників, зафарбувань та ін. Положення й форма графічних примітивів задаються в системі графічних координат, пов'язаних з екраном. Звичайно початок координат розташований у верхньому лівому куті екрана.

Сітка пікселів збігається з координатною сіткою. Горизонтальна вісь X спрямована ліворуч праворуч, вертикальна вісь Y - зверху вниз.

Відрізок прямої лінії однозначно визначається вказівкою координат його кінців; коло - координатами центра й радіусом; багатокутник - координатами його кутів; зафарбована область - граничною лінією й кольором зафарбування та ін.

Приклад

Літера “К”, зображена растровими засобами (див. вище), у векторному поданні може бути подана трьома лініями. Якщо формат команди, що задає лінію, умовно має вигляд

ЛІНІЯ(X1,Y1,X2,Y2),

то наша літера описується в такий спосіб:

ЛІНІЯ(4,2,4,8)
ЛІНІЯ(5,5,8,2)

ЛІНІЯ(5,5,8,8)

Ці три оператори й подають векторний код літери “К”. Графічні файли векторного типу мають невеликий обсяг.

Векторні зображення легко масштабуються без втрати якості.

Варто пам'ятати:

- при введенні зображень у комп'ютер за допомогою сканера формуються графічні файли растрового типу;
- при виведенні на екран монітора будь-якого зображення у відеопам'яті формується інформація растрового типу;
- розходження в поданні графічної інформації в растровому й векторному форматах існує лише для графічних файлів.

Крім мов програмування, графіка використовується в графічних редакторах - прикладних програмах, призначених для створення й обробки графічних зображень на екрані.

Приклади векторних редакторів: CorelDraw, AdobeIllustrator.

Приклади растрових редакторів: Paint, CorelPHOTO-PAINT, AdobePhotoshop.

- **Графічні примітиви в мові QBASIC**

Для зображення графічних примітивів в QBasic використовуються відповідні оператори:

- PSET, PRESET - рисування точки;
- LINE - рисування відрізка;
- CIRCLE - рисування кола.

Існують *три різних типи координат графічного екрана*.

- абсолютні координати;
- координати на основі Точки Останнього Посилання, у скороченні — ТОС (*Last Point Referenced- LPR);*
- відносні координати.

Абсолютні координати

З огляду на систему координат екрана, просто вказується місце, у якому треба нарисувати точку, наприклад, PSET(100,120).

Значення в дужках (100 і 120) вказують на положення точки, що рисується, по осях x і y відповідно.

Точка Останнього Посилання (ТОП)

Координати точки, що була нарисована останньою, зберігаються в пам'яті комп'ютера. Ця точка й називається *Точкою Останнього Посилання* й часто використовується в графічних операторах.

Приклад: при рисуванні лінії за допомогою оператора LINE -(300,120) досить указати координати тільки однієї точки, і на екрані буде проведений відрізок від ТОП до зазначеної точки, що після цього стане ТОП.

Відразу після включення графічного режиму ТОП є точка в центрі екрана.

Відносні координати

Ці координати показують величину переміщення щодо положення ТОП. Щоб нарисувати нову точку, використовуючи відносні координати, буде потрібно ключове слово STEP.

Приклад

PSET STEP (-5,8)

При цьому з'явиться точка, положення якої буде ліворуч на5 і нижче на 8 точок відносно ТОП. Інакше кажучи, якщо ТОП має координати (100,100), то даний оператор означає рисування точки з координатами (95,108).

Абсолютні координати повинні бути завжди додатними, авідносні можуть бути як додатними, так і від’ємними.

- ***Оператори PSET І PRESET***

Оператор PSET призначений для рисування точки на екранішляхом зміни її кольору з фонового (чорного) на білий.

Оператор може мати такі форми:

PSET ***(X, Y)*** — абсолютна форма; **PSET**
STEP ***(X, Y)*** - відносна форма; **PRESET**
(X,Y) — абсолютна форма; **PRESET**
STEP ***(X,Y)*** - відносна форма,

де ***X, Y*** - абсолютні координати або зсув точки відносно ТОП.

Оператор PRESET призначений для зміни кольору відповідної точки на фоновий, тобто виконує дію, зворотну PSET.

- ***Прямі лінії, відрізки, прямокутники***

Оператор LINE призначений для рисування відрізка, щоз'єднує дві довільні точки екрана.

Загальна форма запису:

LINE [(X_початок, B_початок)] - (X_кінець, B_кінець)[,[колір][,[B/BF],[,стиль%]]],

де **X_початок, B_початок** — координати початку відрізка (необов'язкові параметри);

X_кінець, B_кінець — координати кінця відрізка (обов'язкові параметри);

колір – колір лінії;

B/BF – рисує прямокутник або зафарбований прямокутник замість лінії;

стиль% - чи будуть рисуватися точки растра.

Якщо координати початку відрізка не зазначені, то відрізок буде починатися в ТОП. В операторі LINE можна використати відносні координати початку й/або кінця відрізка.

Для рисування прямокутників можна вибрати більш простий шлях.

Приклад

```
LINE    (50,50)-(150,155) , ,B
```

У випадку пропуску якого-небудь параметра або параметрів потрібно зберегти необхідну кількість розділовими комами.

Команда DRAW дозволяє:

- рисувати лінії;
- розфарбовувати області;
- обертати зображення;
- змінювати розміри (масштаб).

Загальний вид команди:

DRAW ***текстова постійна***

текстова постійна - послідовність параметрів-кодів, які означають напрямок руху (команди). Після кожного параметра стоїть число, що вказує довжину лінії. Команда визначається однією або двома літерами, що задають конкретну дію.

Приклад: Ln - перемістити вліво P n, m - зафарбувати область

2. переміщення

DRAW "M 20,20" - рисує лінію від ТОП до точки з координатами (20,20)

(Аналог Line -(20,20)

DRAW "M +25,-40" - рисує лінію від ТОП до точки на25 точок праворуч й на 40 вище ТОП;

б) відносний рух

використовується для рисування ліній різних кольорів у вісьмох припустимих напрямках або переміщення в кожному із цих напрямків.

Кожна з команд супроводжується цілочисловим аргументом, що вказує довжину лінії в точках

DRAW "R16"

U

H

E

Корисними також є такі параметри:

B - переміщення без рисування;

- **N** - переміщення без зміни ТОП;

L R □ **C** - якщо він знаходиться в середині рядка символів разом із G F числовим значенням кольору, то D колір ліній зміниться на зазначений.

Приклад

REM Рисування літери "B"

SCREEN 1

DRAW "BM+10, +0 R20 E10 U10 H5 E5 U10 H10 L20 D50"DRAW "BM+10, -10 R10 U10 L10 D10"

DRAW "BM+0, -20 R10 U10 L10 D10"

в) обертання

Команда An обертає зображення на кут, кратний 90 градусам, де n=0,1,2,3

Команда TAn дозволяє повертати зображення на довільний кут від -360 до +360 (- означає поворот за годинниковою стрілкою; + проти годинникової)

Приклад

REM Рисування літери "B" з поворотом на 90 градусівDRAW "A1"

г) масштабування

Команда Sn змінює розмір зображення, збільшуючи або зменшуючи його залежно від значення n.

Зображення масштабується в n/4 разів, тому команда "S12" збільшує

лінійний розмір зображення в 3 рази, а команда "S2" - зменшує вдвічі (n може мати значення від 1 до 225).

Приклад

```
REM Рисування літери "B", збільшеної вдвічі
DRAW "S8";
```

д) колір в операторі DRAW

Колір може бути визначений за допомогою команди CnREM
Рисування літери "B" рожевого кольору
DRAW "C2"
DRAW P ***колір, контур***
зафарбовує замкнуту область.

- ***Оператор*** **CIRCLE**

Дозволяє рисувати коло у будь-якому місці екрана:

CIRCLE ***(X_центр,B_центр), радіус*** **—** абсолютна форма;
CIRCLE STEP ***(X_центр,B_центр), радіус*** **—** відносна форма,

де ***X_центр, B_центр*** — координати або зсув центра кола;
радіус — радіус кола.

Приклад

```
REM Рисування кола
SCREEN  2
CLS
CIRCLE     (100,100),25Коло із центром у точці з координатами
(100,100) має радіус25 точок.
```

- ***Використання кольору***

Команда COLOR

COLOR A [,B,C]

A - задає колір зображення, **B** - задає колір тіла, **C** - задає колір границі.

Колір, що використовується як параметр в операторах PSET, PRESET, LINE I CIRCLE, впливає тільки на зображення, залишаючи тіло без змін.

У графічних операторах QBasic колір визначається в такий спосіб:

PSET (X, Y) , колір;

PRESET (X,Y) ,колір;

LINE (X_початок,B_початок) - (X_кінець,B_кінець) , колір;

CIRCLE (X_центр, B_центр), радіус, колір, де колір - значення колірного параметра.

У режимі екрана, що задається оператором SCREEN 2, можливі тільки два кольори чорний і білий, тому даний параметр марний.

Режим SCREEN 1 підтримує 4 кольори, яким відповідають значення від 0 до 3

Щоб “стерти” який-небудь елемент зображення без очищення всього екрана, можна просто перерисувати цей елемент кольором тіла.

Команда PAINT

використовується для розфарбовування замкнутих ділянок(не тільки прямокутних).

PAINT (X,Y),D,C

X,Y - координати будь-якої точки в середині замкнутої ділянки, що розфарбовують; **D** - колір розфарбовування; **C** - колірграниці ділянки.

- ***Дуга, еліпс і сектор***

Щоб нарисувати дугу, еліпс або сектор кола, необхідно додати нові параметри в оператор **CIRCLE**, повна форма якого має такий вигляд:

CIRCLE (X,Y), радіус, колір, початок, кінець, коефіцієнт

6. **X, Y** — координати центра кола;
7. **радіус** — радіус кола;
8. **колір** — його колір;
9. **початок** — початкова точка дуги, задана в радіанах;
10. **кінець** — кінцева точка дуги, задана в радіанах;
11. **коефіцієнт** — відношення значень Y-радіуса й X - радіуса.

Для рисування кола використовуються тільки параметри X, Y і радіус.

Для рисування дуги необхідно додати значення параметрів початкової й кінцевої точок. Дуга визначається кутом, що вирізається з відповідного кола. Значення параметрів **початок** і **кінець** задаються в радіанах і повинні мати значення між 0 і 2□.

QBasic при рисуванні дуг веде відлік від початкової точки дуги до кінцевої проти годинникової стрілки.

Приклад

```
REM Рисування кола, дуги й сектора
SCREEN 2
CLS
CIRCLE (100,100), 30
CIRCLE (180,100), 30, 3, 1, 2
CIRCLE (260,100), 30, 3, -2, -1
```

Якщо одному з параметрів ***(*початок або кінець*)*** значення не присвоюється, то воно вважається рівним нулю.

При від’ємних значеннях цих параметрів QBasic з'єднує початкові кінцеві точки дуги із центром відповідного кола. Таким чином, на екрані виходить зображення сектора кола. Якщо від’ємним є значення тільки одного параметра, то й з'єднуватися із центром кола буде тільки одна точка дуги.

Для рисування еліпса потрібно задати **коефіцієнт** відношень радіусів по осях Y і X. Цей параметр визначає ступінь стиску еліпса й може мати будь-яке додатне значення.

Якщо параметр **коефіцієнт** опущений або дорівнює 1, ви одержуєте зображення кола. При від’ємному значенні параметра ви одержите повідомлення про помилку.

Приклад

```
REM Рисування еліпсів
SCREEN 2
CLS
CIRCLE (50,90), 30
CIRCLE (150,90),30, , , ,0.3
CIRCLE (250,90,30, , , ,1.5
```

- **Імітація руху на екрані**

Для імітації руху зображення об'єкта на екрані необхідно виконати такий алгоритм:

12. нарисувати об'єкт у заданій точці;
13. знищити об'єкт, зафарбувавши його кольором тіла;
14. змінити координати об'єкта;
15. перейти до п.1.
16. ***Приклад***

```
REM рух униз зеленого кола на чорному тілі
SCREEN 8
COLOR 2,1
Y=10: SY=4
50 CIRCLE (120,Y),40,2          'рисування кола
CIRCLE (120,Y),40,1             'стирання кола
Y=Y+SY          ' зміна координати центраIF Y<10 OR
Y>190 THEN SY=-SY
GOTO 50
```

Варто організовувати штучні паузи між рисуванням і стиранням об'єктів для підтримки правильних зображень на екрані ("холості" цикли)

Оператори GET і PUT

З їхньою допомогою можна домогтися ефекту мультиплікації: створювати багаторазові копії зображення або рухати графічну картинку.

GET дає можливість зберегти будь-яку прямокутну область екрана в числовому масиві, а **PUT** відтворює зображення в довільному місці екрана.

Загальна форма запису

GET [STEP] (x1!,y1!) - [STEP] (x2!,y2!), *ім'я* [(*індекс%*)]

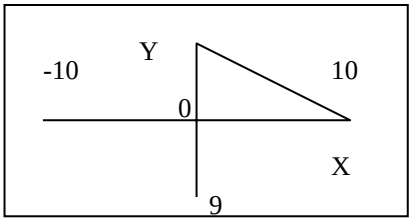
PUT[STEP] (x1!,y1!), *ім'я* [(*індекс%*)] [,*режим*]

17. **STEP** - задає відносну форму графічних координат;
18. **x1!,y1!** - координати верхньої лівої точки зображення, що зберігається оператором GET або точка на екрані, у якій оператор PUT поміщає це зображення;
19. **x2!,y2!** - координати нижньої правої точки зображення, що зберігається;
20. **ім'я_**- ім'я масиву, у якому зберігається зображення;
21. **індекс%** - індекс елемента масиву, починаючи з якого зберігається зображення;
22. **режим** - ключове слово, що визначає режим відтворення збереженого зображення. За замовчуванням - *XOR*;

- *AND* - об'єднання збереженого зображення з існуючого;
- *OR* - накладення збереженого зображення на існуюче;
- *PSET* - рисування збереженого зображення, знищуючи існуюче;
- *PRESET* - рисування збереженого зображення кольором тіла, знищуючи існуюче;
- *XOR*- рисування збереженого зображення або знищення попереднього, зі збереженням і відновленням тіла, відтворюючи ефект анімації.

Побудова графіків

Приклад



Знайдемо формули для переходу від математичних координат (X,Y) до графічного (XD,YD):

16. Різний початок координат:

XD=x+100:YD=Y+90;

17. Масштаб:1 до 10

XD=x*10+100:YD=Y*10+90;

18. Осі Y і YD протинаправлені

Будемо змінювати X від -10 до 10 з маленьким кроком, атакож додамо побудову осей (LINE).'Програма - графікCLS

```
SCREEN 12
LINE (0, 90)-(200, 90)
LINE (110, 0)-(100, 180)
FOR x = -10 TO 10 STEP .01
y = x * x
xd = x * 10 + 100yd
= y * 10 + 90
IF yd > 0 AND yd < 180 THEN PSET (xd, yd)NEXT x
```

•

Контрольні запитання:

- 1. Які графічні режими використовують у програмуванні, та як проходить обробка даних?
- 2. Принципи подання зображень. кодування графічних зображень.
- 3. Які оператори використовують у графічному режимі роботи екрану для малювання різних предметів: прямої лінії, відрізка, прямокутника, еліпса, дуги, круга, сектора та ін.
- 4. Використання операторів:GET та PUT та їх призначення?
- 5. Наведіть приклади побудови графіків за допомогою графічних операторів?
- 6. Для малювання чого використовують оператори: PSET та PRESET.

Розділ X ОПЕРАЦІЙНА СИСТЕМА WINDOWS

Операційна система Windows.

Основні принципи роботи з об'єктами

Із вдосконаленням апаратної частини сучасного комп'ютера програмне забезпечення вдосконалюється в двох напрямках. З одного боку, це максимальне використання нових можливостей, що пропонуються апаратною частиною, з іншої – максимальна простота і зручність роботи інтерфейсу користувача.

Операційна система Windows являє собою сукупність програм, керуючих роботою обладнання і що здійснюють взаємодію з користувачем.

Windows XP - це удосконалений і модернізований варіант попередньої операційної системи Windows 2000. В операційній системі Windows XP реалізовані як засоби роботи з файлами і пристроями, так і виправленні й удосконалені • засоби, що вже були доступні користувачеві в ранніх версіях Windows. Операційна система Windows XP забезпечує поліпшену сумісність з апаратними і програмними засобами, усталену роботу системи, відновлення системи після збою, використання нових програм, широкі можливості для роботи в локальній і глобальній комп'ютерних мережах.

Windows — це 32-розрядна операційна система, яка забезпечує одночасну роботу кількох додатків (програм). Ця ОС **багатозадачна, графічна, багатовіконна**. Порівняно з іншими операційними системами у ній використовують сучасніші механізми забезпечення такого режиму роботи.

Простий та зручний інтерфейс системи забезпечує природність спілкування користувача з комп'ютером. Потрібно від-мітити, що система має широкий набір засобів, який дозволяє настроїти інтерфейс так, як подобається користувачеві.

При вмиканні комп'ютера на екрані монітора з'являється **головне вікно**,

яке називають *Робочим столом.* Якщо клацнути мишею по піктограмам на ньому, їх можна перетворити у меню, папки, файли та ін.

Головна особливість інтерфейсу — наявність панелі завдань, на якій розміщена кнопка Пуск. Натискування цієї кнопки відкриває доступ до **головного меню системи**. При запуску будь-якого додатку або відкритті будь-якої папки на панелі завдань з'являється кнопка, яка представляє цей додаток чи папку.

Вікна додатків можна згорнути, але їх кнопки залишаться на панелі завдань. Як тільки будуть натисненні кнопки будь-якого додатку, він стає активним, а якщо його вікно було згорнуте, то відбувається відновлення вікна.

Друга важлива особливість інтерфейсу - наявність так званого контекстного меню. Відкривається це меню натискуванням правої кнопки миші. Зміст контекстного меню залежить від положення вказівки миші у відкритих вікнах чи головному вікні системи. Використання меню прискорює процес роботи з об'єктами, оскільки користувач звільняється від необхідності шукати команди у меню вікон.

Основні компоненти Windows:

1-й **Робочий стіл.** Екран комп'ютера після завантаження Windows називається Робочим столом. Елементи Робочого стола включають значки програм, що часто використовуються, і інших необхідних для роботи інструментів. Робочий стіл - старша папка в ієрархічній структурі Windows, вона залишається відкритою протягом усього сеансу роботи.

2-й **Вікно.** Частина екрана, що відображає певну програму або документ. На екрані можуть одночасно бути присутні декілька вікон.

3-й **Ярлик.** Тип значка, який забезпечує швидкий доступ до файлів, папок і програм. Ярлик відрізняється від значків тим, що помічений зігненою стрілкою в нижньому лівому кутку. Ярлики можна створювати, копіювати, переміщувати і видаляти, не впливаючи при цьому на відповідний

йому файл, папку або програму.

4-й **Панель задач.** Панель задач являє собою горизонтальну смугу внизу екрана. Вона звичайно містить кнопку Пуск, Панель швидкого запуску, значки деяких службових програм і час. На панелі задач відображаються кнопки запущених програм, їх натискання відкриває вікно відповідної програми.

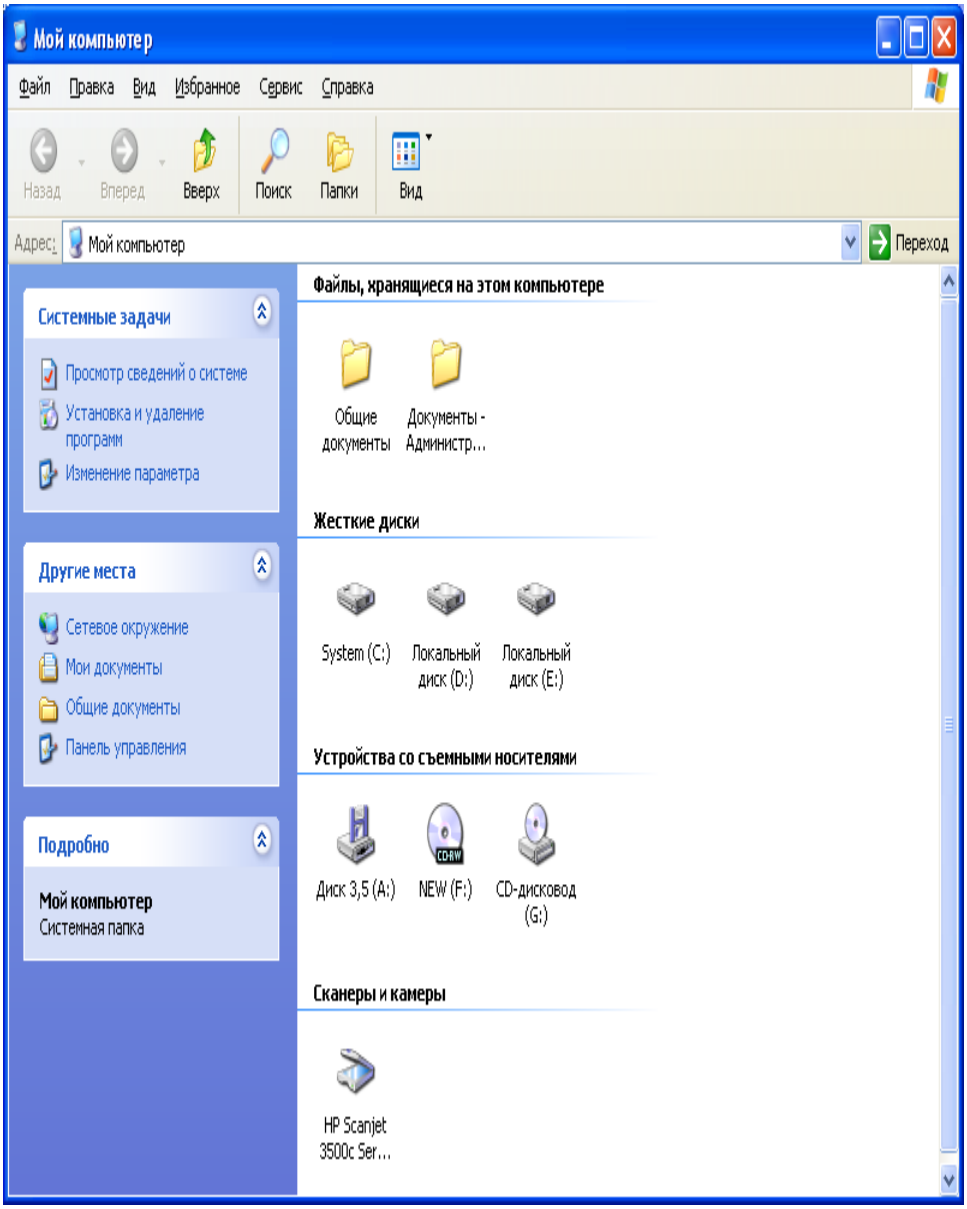
5-й Кнопка **Пуск.** Кнопка **Пуск** розташована у лівого краю **Панелі задач** і дозволяє запустити будь-яку програму, встановлену на комп'ютері, і відкрити будь-яке доступне вікно. Натиснення на кнопці **Пуск** відкриває послідовність меню для запуску програм, пошуку файлів, отримання довідок, настройки параметрів, установки обладнання і програмного забезпечення, а також для завершення роботи комп'ютера.

6-й **Панель швидкого запуску.** Являє собою перший набір кнопок зліва на **Панелі задач**, розташований відразу за кнопкою **Пуск,** клацнувши на яких можна запустити відповідні програми. Панель швидкого запуску включає кнопку **Свернуть все окна**, що дозволяє скрутити всі відкриті вікна до кнопок на панелі задач і звільнити робочий стіл.

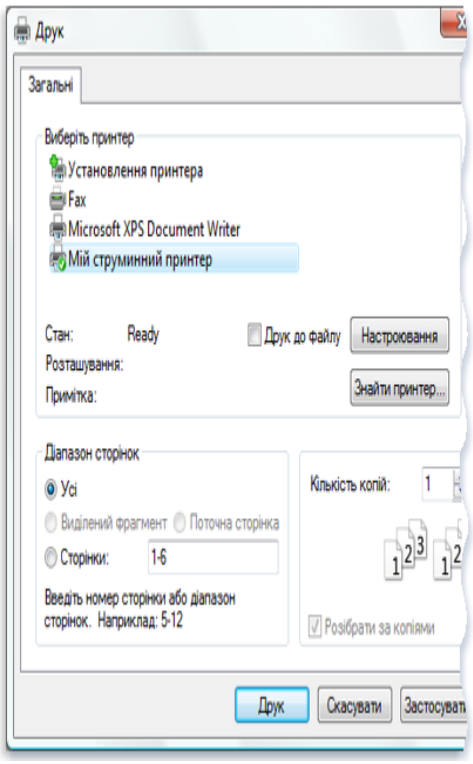
7. Час. Час відображається праворуч на панелі задач. Якщо вмістити покажчик миші на час, на екрані відобразиться поточна дата. За допомогою подвійного натиснення на часі можна відкрити діалогове вікно **Свойства: Дата и время**, яке відображає календар і дозволяє змінити дату, час і часовий пояс.

Вся робота з Windows та її додатками проходить у **вікнах, які бувають типовими і діалоговими.**

Типове вікно:



Діалогове вікно:



Діалогове вікно з'являється на екрані, коли необхідно задати або вибрати параметри для виконання певної команди. Типове діалогове вікно включає наступні елементи: поля введення, кнопки вибору, прапорці і меню.

У правому верхньому кутку діалогового вікна розташовується кнопка **Закрыть**, а внизу, в більшості випадків, кнопки **Ок** і **Отмена**.

Якщо клацнути на кнопці **Ок,** то Windows виконає вказані в діалоговому вікні дії, використовуючи задані параметри. Натиснення на кнопках **Закрыть** і **Отмена** закриває діалогове вікно без виконання яких-небудь дій і збереження значень параметрів.

Елементи типового вікна:

1-й **Рядок заголовка.** Крім того, що рядок заголовка містить назву програми і звичайно найменування документа, його можна використати для переміщення вікна по екрану. Для цього потрібно клацнути на рядку заголовка і перетягнути вікно, не відпускаючи кнопку миші.

2-й Кнопка **Свернуть.** При натисненні на цій кнопці вікно зникає з екрана, але не закривається, на панелі задач залишається відповідна йому кнопка.

3-й Кнопка **Развернуть**. Якщо клацнути на цій кнопці, вікно збільшиться і заповнить весь екран, за винятком панелі задач.

4-й Кнопка **Восстановить.** Коли вікно розгорнене до повного екрана, кнопка **Свернуть** замінюється на кнопку **Восстановить**, яка стискає вікно до розміру, який воно мшю раніше. Можна також розвернути вікно або відновити його розмір, клацнувши двічі на рядку заголовка.

5-й Кнопка **Закрыть**. Ця кнопка закриває вікно програми і видаляє її з оперативної пам'яті комп'ютера.

6-й Область зміни розміру вікна. Це позначена трьома косими штрихами область в нижньому правому кутку вікна, перетягши яку, можна змінити його розміри.

Робота з об’єктами в ОС Windows

В ОС Windows існую три основних типи об’єктів: папки, файли, ярлики. На відміну від MS-DOS в операційної системи Windows імена об’єктів складаються з 256 символів, але 3-х символьні розширення збереглися і виконують ту ж саму функцію – вказують на тип файлу.

Windows зберігає імена останніх п'ятнадцяти файлів. Щоб швидко відкрити такий файл, клацніть на кнопці **Пуск**. У меню **Пуск** укажіть на пункт **Документы**, а потім клацніть на потрібному файлі в меню, що відкрилося.

Дії над об’єктами:

- *виділення групи об’єктів* – „Обвести” мишею, або клацати по об’єктах, утримуючи Ctrl;
- *створення об’єктів* – Файл/Создать/... або Создать/... в контекстному меню;
- *копіювання об’єктів* – Правка/Копировать + Прака/Вставить або Копировать + Вставить в контекстному меню;
- *переміщення об’єктів* – Правка/Вырезать + Прака/Вставить або Вырезать + Вставить в контекстному меню;

- *перейменування об'єктів* – Файл/Переименовать або Переименовать в контекстному меню;
- *знищення об'єктів* – Файл/Удалить або Удалить в контекстному меню;
- *створення ярликів об'єктів* (вони забезпечують швидкий доступ до об'єктів) – перетягнути об'єкт до місця створення ярлика, утримуючи клавіші Ctrl та Shift;
- *запуск програми на виконання* – двічі клацнути по її ярлику лівою кнопкою миши.

В меню **Пуск** знаходиться пункт **Справка и поддержка**, який відкриває довідкову службу Windows XP - центр довідки і підтримки. Довідкова система дозволяє одержати довідку з питання, яке нас цікавить, консультацію і підтримку в інтерактивному режимі. Довідковий розділ містить інформацію, згруповану за темами. При виборі теми відкривається меню, що пропонує користувачеві конкретизувати питання. Довідкова система є свого роду навчальною програмою. Наявність гіперпосилань (фрагментів тексту довідки, виділених кольором) дозволяє одержати досить повну довідку з питання, яке нас цікавить. Пошук інформації робиться також у локальній довідковій системі та в Internet. Також для отримання довідкової інформації кожне вікно ОС Windows містить кнопку „?" або „Справка"; також в будь-якому вікні можна натиснути функціональну клавішу F1.

В меню **Пуск** знаходиться пункт **Найти**, який надає можливість відкрити пошукову систему Windows. У меню **Что вы хотите найти?** вікна **Результаты поиска**, яке з'являється після вибору пункту контекстного меню Файлы и папки..., потрібно вказати тип об'єкта пошуку. Для пошуку файлу припустиме завдання шаблону. Для підвищення точності пошуку використовуйте Дополнительные параметры, що дозволяють використовувати при пошуку такі параметри, як розмір файлу, дата його останньої зміни тощо.

Тобто, для того, щоб знайти будь-який об’єкт в ОС Windows, треба виконати таку послідовність дій: **Пуск / Найти / Файлы и папки** і вказати умови пошуку у відповідних рядках введення.

Для того, щоб в ОС Windows вимкнути комп’ютер, треба скористатися кнопкою **Пуск / Выключить компьютер / Выключение.**

Контрольні запитання:

1. Поясніть смисл фрази: „Windows – графічна ОС.
2. Що розуміється під багатозадачністю ОС?
3. Чому ОС Windows має таку назву?
4. Наведіть приклади об’єктів ОС Windows.
5. Для чого використовується контекстне меню об’єкта?
6. Які додаткові можливості надає ОС Windows користувачу?
7. В чому схожість і розбіжність між іменами об’єктів в MS-DOS та Windows?
8. Як за допомогою контекстного меню дізнатися про властивості об’єктів?
9. Які дії над об’єктами в ОС Windows можна виконувати і як?
10. Для чого призначена панель задач?
11. Як запустити програму на виконання?
12. Опишіть процес вимкнення комп’ютера.
13. Як отримати довідкову інформацію?
14. Опишіть структуру звичайного (типового вікна) Windows.
15. Назвіть елементи діалогового вікна Windows.
16. Як знайти папку чи файл в ОС Windows?

Розділ IX

МЕТОДИЧНІ ВКАЗІВКИ ДО ПРАКТИЧНИХ РОБІТ З КУРСУ «ОСНОВИ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ПРОГРАМУВАННЯ»

ВСТУП

Методичні вказівки до практичних робіт містять 8 робіт, які виконуються відповідно до програми дисципліни «Основи інформаційних технологій і програмування» для здобувачів теплоенергетичного напрямку, які навчаються за програмою бакалавра.

Даний матеріал має допомогти швидше почати роботу у середовищі програмування QBASIC і освоїти основні прийоми написання програм. Велика кількість прикладів і практичних завдань допоможе добре вивчити нові визначення, постійно контролюючи якість засвоєння матеріалу.

Перед виконанням практичних робіт здобувач має ознайомитися із завданням до роботи і відповісти на контрольні запитання керівника занять.

Звіт до практичної роботи містить:

- постановку задачі;
- результати розрахункового експерименту;
- графічні матеріали (якщо це потрібно);
- тексти програм;
- аналіз отриманих результатів.

ПРАКТИЧНА РОБОТА №1.
АЛГОРИТМИ ЛІНІЙНИХ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ

Мета роботи: Скласти алгоритм лінійних обчислювальних процесів та алгоритм для обчислення кулонівських сил.

Постановка задачі: Скласти алгоритм для обчислення кулонівських сил, що прикладені до тіл із зарядами $q_1 = 3*10^{-8}\,Кл, q_2 = \frac{1}{5}*10^{-8}\,Кл, q_3 - 5*10^{-8}\,Кл.$

Математична модель розв'язання задачі:

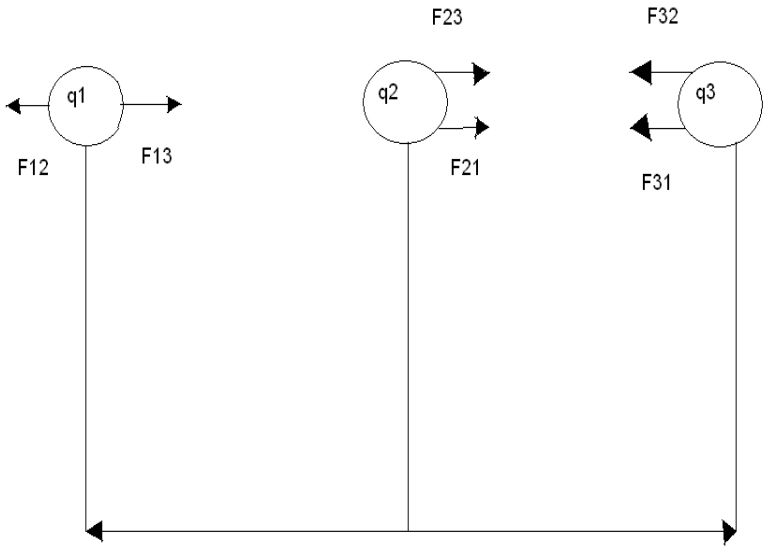


Рис.1.

Використовуючи закон Кулона відповідно до двох тіл із зарядами q_1 та q_2,
розташованих на відстані r_{12} одне від другого, визначимо кулонівські сили, з якими однетіло взаємодіє з іншим:

$$F_{12} = -F_{21} = K\frac{|q_1| \cdot |q_2|}{r_{12}^2}$$

(1.1)

де К - коефіцієнт пропорційності ($K = 9*10^9 \frac{Hm^2}{Kл^2}$).

Відповідно:

$$F_1 = F_{12} - F_{12}, F_2 = F_{21} + F_{23}, F_3 = F_{31} + F_{32.} \quad (1.2)$$

$$F_{23} = -F_{32} = K\frac{|q_2| \cdot |q_3|}{r_{23}^2} \quad (1.3)$$

$$F_{12} = -F_{31} = K\frac{|q_1| \cdot |q_3|}{r_{13}^2} \quad (1.4)$$

Визначити F_1, F_2, F_3 використовуючі типові блоки, запишемо схему алгоритму для обчислення кулонівських сил.

Контрольні запитання:

1. Що вивчає предмет Інформатика?
2. Що таке програмування і програма ?
3. Що таке мова програмування ?
4. Що таке змінна (визначення)
5. Алгоритм (визначення) та його властивості.
6. Операція присвоєння (визначення)

Приклад:

Програма для обчислення площі прямокутника

```
CLS
DIM a%, b%
INPUT “Довжину”; a%
INPUT “Ширину”; b%
Square%  =  a% * b%
PRINT “Площина дорівнює”; square%
END
```

ПРАКТИЧНА РОБОТА №2.
ПОБУДОВА ЛІНІЙНОГО АЛГОРИТМУ ОБЧИСЛЕННЯ ПАРАМЕТРІВ ЕЛЕКТРИЧНИХ СХЕМ

Мета роботи: Скласти алгоритм обчислення спаду напруги на резисторах,силу струму для схеми.

Постановка задачі: Скласти алгоритм обчислення спаду напруги на резисторах, силу струму для схеми, якщо: E,R1,R2,R3 – відомі дані. Визначити I1,I2,I3 та U1,U2,U3.

Математична модель розв'язання задачі:

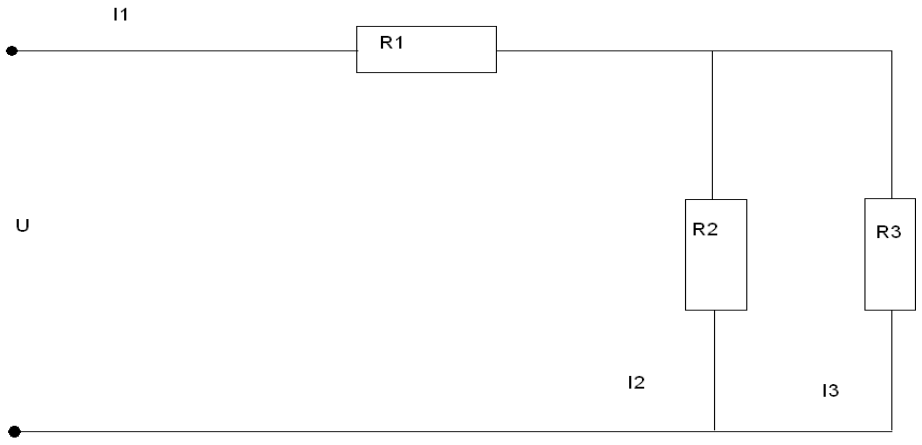


Рис.2.

Використовуючи закон Ома, запишемо I = E/R,

де R = R1 + R23, а R23 = R2*R3/(R2 + R3).

Тоді I1 = I, а U1 = I1 * R1. Далі маємо U3 = U2, де U2 = I * R23. Далі I2 = U2/R2, а I3 = U3/R3.

Використовуючи типові блоки складаємо алгоритм.

Контрольні запитання:

1. Блок-схема (визначення).
2. Оператор опису типу змінних

(форма запису, що означає кожен символ)

3. Змінна (визначення)
4. Типи змінних (перерахувати)
5. Оператор вводу даних

(форма запису, що означає кожен символ)

6. Оператор виводу даних

(форма запису, що означає)

7. Головне меню QBASIC

(з чого складається, які виконує функції кожен елемент меню)

Приклад:

(див. приклад до лабораторної роботи №1)

ПРАКТИЧНА РОБОТА №3.
ПОБУДОВА АЛГОРИТМІВ ГІЛКОВИХ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ

Мета роботи: Скласти алгоритми гілкових обчислювальних процесів для обчислення спаду напруги та струмів на резисторах у схемі.

Постановка задачі: Скласти алгоритм обчислення спаду напруги та струмів на резисторах у схемі при різних положеннях перемикача K, де R_1, R_2, R_3, E, K відомі дані.

Визначити $U_1, U_2, U_3, I_1, I_2, I_3$ при $K = 1$ та $K = 2$.

Математична модель розв'язання задачі:

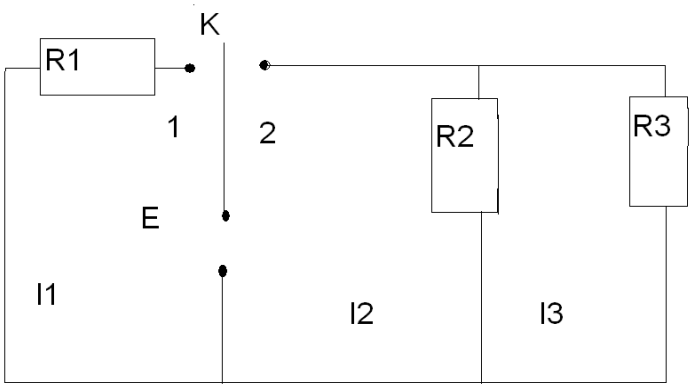


Рис.3.

У даній схемі ключ К може бути у положенні $K = 1$ або $K = 2$. У положенні ключа $K = 1$, згідно закону Ома запишемо: $U_1 = E, U_1 = I_1 * R_1$, звідти знайдемо $I_1 = U_1 / R_1$. У положенні ключа $K = 2$, $U_2 = U_3 = E, I_2 = U_2 / R_2, I_3 = U_3 / R_3$. Використовуючи типові блоки скласти алгоритм розв'язування задачі.

Контрольні запитання:

1. Оператор умовного переходу (визначення).
2. Призначення оператору умовного переходу.
3. Коротка форма запису оператора умовного переходу
(що означає кожен символ в запису даного оператора, як він працює).
4. Повна форма запису оператора умовного переходу
(що означає кожен символ в запису даного оператора, як він працює).
5. Логічне "І"(AND), призначення, форма запису, коли воно істинно.
6. Логічне "ИЛИ"(OR), призначення, форма запису, коли воно істинно.

Приклад:

В цьому прикладі показана вкладеність операторів IF

```
CLS
DIM a!, b!, c!
INPUT “ввести a”; a!
INPUT “ввести b”; b!
INPUT “ввести c”; c!
IF a! < 10 THEN
PRINT “a меньше 10”
IF a! = c! THEN
PRINT “a дорівнює c”
ELSE
PRINT “a не дорівнює c”
```

```
END IF
END IF
END
```

ПРАКТИЧНА РОБОТА №4.
ПОБУДОВА АЛГОРИТМІВ РОБОТІ З ОПЕРАТОРАМИ ЦИКЛІВ ТА МАСИВАМИ ДАНИХ.

Мета роботи: Скласти алгоритм роботи з операторами циклів та масивами даних для квадратної матриці розміром 4х4

Постановка задачі:

1.Для квадратної матриці розміром 4х4 визначити суму від'ємних елементів, які знаходяться нижче головної діагоналі матриці.

2.Знайти середнє значення додатних елементів на головній діагоналі.

Контрольні запитання:

1. Цикл (визначення).
2. Призначення циклів.
3. FOR…NEXT форма запису
4. FOR…NEXT принцип дії
5. Визначення масиву.
6. Визначення одномірного/багатомірного масиву.
7. Призначення масивів.

Приклад:

```
CLS
DIM i % , j%
FOR i % = 5 TO 104
PRINT i%
```

```
j % =  j%  +  1
NEXT i%
PRINT
PRINT “Кількість виконань циклу: “ ; j%
END
```

Після запуску програми першим на екран буде виведено число 5, останнім – 104. Цикл виконується 100 разів (104-5+1)/1. Змінить початкове та кінцеве значення змінної циклу і
наступним чином: «FOR i = 6 TO 5 “ та запустіть програму заново. В результаті виводу на екран не буде, тому що цикл FOR…NEXT не виконується жодного разу.

ПРАКТИЧНА РОБОТА №5.
ПОБУДОВА АЛГОРИТМІВ
З ВИКОРИСТАННЯМ ПРОЦЕДУР QBASIC

Мета роботи: Скласти алгоритм з використанням процедур QBASIC для квадратної матриці розміром 4x4.

Постановка задачі:

1. Взяти за основу постановку задачі та вихідні дані лаб. Роботи №4
2. Суму від’ємних елементів, які знаходяться нижче головної діагоналі, розрахувати, використовуючи процедуру SUB

3. Середнє значення додатних елементів на головній діагоналі розрахувати використовуючи процедуру FUNCTION.

Контрольні запитання:

1. Процедура SUB
(визначення, форма запису, що означає кожен символ в запису даного оператора)
2. Процедура FUNCTION
(визначення, форма запису, що означає кожен символ запису даного оператора)
3. Виклик процедури із головної програми
(форма запису, що означає кожен символ в запису даного оператора)
4. DIM SHARED
(визначення, форма запису, що означає кожен символ в запису даного оператора)

Приклад:

Программа на якій зображено принцип дії процедури SUB

```
CLS
PRINT “A”
   PRINT “B”
   PRINT “B1”
   PRINT “B2”
PRINT “C”
   PRINT “B”
   PRINT “B1”
   PRINT “B2”
```

```
PRINT “ert”
  PRINT “B”
  PRINT “B1”
  PRINT “B2”
END
```

Такі повторення збільшують програму та ускладнюють роботу з нею. Процедури дозволяють об’єднувати під одним ім’ям фрагменти, які повторюються. Процедуру можна визвати із головної програми

```
SUB beispiel
  PRINT “B”
  PRINT “B1”
  PRINT “B2”
END SUB
```

ПРАКТИЧНА РОБОТА № 6.
ПОБУДУВАННЯ АЛГОРИТМІВ З ВИКОРИСТАННЯМ ОПЕРАТОРІВ РОБОТИ З ФАЙЛАМИ І ДАНИМИ.

Мета роботи: Скласти алгоритм з використанням операторів роботи з файлами і даними для квадратної матриці розміром 4х4

Постановка задачі:

1. Для квадратної матриці розміром 4х4 (дані для лаб. роб. № 4) виповнити присвоєння значень елементів використовуючи оператори DATA и READ.
2. Створити дисковий файл (на диску Z:) з имям MATRITSA, в який помістити значення элементів массива.
3. Із файла з именем MATRITSA вивести на дісплей значення элементів головної діагоналі матрици.

Контрольні запитання:

1. Оператори роботи з даними (DATA, READ).
(визначення, форма запису, що означає кожен символ в запису даного оператора).
2. Взаємодія операторів DATA та READ.
3.Оператор роботи з файлами OPEN
(визначення, форма запису, що означає кожен символ в запису даного оператора)
4. Режими роботи оператора OPEN

(перерахувати, дати визначення кожному)

5. Оператор роботи з файлами WRITE

(визначення, форма запису, що означає кожен символ в запису даного оператора)

6. Оператор роботи з файлами CLOSE

(визначення, форма запису, що означає кожен символ в запису даного оператора)

Приклад:

```
Програма для запису адресу

CLS
DIM surname$, name$, address$
OPEN “Address book” FOR APPEND AS #1
INPUT “Surname : ” , surname$
INPUT “Name : “ , name$
INPUT “Address : “ , address$
WRITE #1, surname$, name$, address$
CLOSE #1
END
```

ПРАКТИЧНА РОБОТА № 7.
ПОБУДОВА АЛГОРИТМІВ З ВИКОРИСТАННЯМ ГРАФІЧНИХ ОПЕРАТОРІВ.

Мета роботи: Скласти алгоритм з використанням графічних операторів для побудови геометричних фігур

Постановка задачі:

23. Вивести на екран вікно у режимі SCREEN 1 з координатами 20, 20- 250, 195
24. Розділити вікно на 4 трикутника
25. В кожному трикутнику помістити еліпси, витянуті впродовж відповідних осей
26. Кожен еліпс замалювати своїм кольором

Контрольні запитання:

1. Яким оператором задається графічний екран, "разрешение" граф. екрана.
2. Де розміщується початок координат графічного екрану.
3. Лінія (LINE) - всі види запису даного оператора
(що означає кожен символ в запису даного оператора).
4. Окружність (CIRCLE) - всі види запису даного оператора
(що означає кожен символ в запису даного оператора).
5. Малювання точки (PSET, PRESET).
(що означає кожен символ в запису даного оператора).
6. Яким оператором виконується заливка кольором, форма запису цього оператора
7. Скільки кольорів має стандартна палітра мови QBASIC, які номера мають ці кольори.

Приклад:

```
CLS
SCREEN 12
COLOR 3,4
‘Оператор, який задає колір тексту та фону
LINE (30,40) – (130,140), 5, B
‘Оператор малювання прямокутної рамки
CIRCLE (65,70), 20, 8
‘Оператор малювання окружності
END
```

ПРАКТИЧНА РОБОТА №8.
ВИВЧЕННЯ КОНФІГУРАЦІЇ ПАКЕТУ WINDOWS.

Мета роботи: Освоєння практичних прийомів роботы з програмою «Панель управління».

Постанова задачі:

- Активізувати меню «Пуск» - «Налаштування» - «Панель управління». Відмітити в протоколі назви найчастіше використовуваних програм.
- Активізувати програму «Екран».
- Змінити фон «Робочого столу»
- Змінити час ввімкнення «Заставки»
- Змінити колір вікна
- Змінити заставку
- Активізувати програму «Клавіатура»
- Змінити швидкість реакції на натискання клавиші
- Змінити швидкість мигання курсору
- Додати в мовне меню будь-яку мову
- Перевірити та уточнити дату і час.
- Активізувати програму «Миша».
- Змінити довжину шлейфу «миши»
- Відклюити або включити «пісочний годинник».
- Викласти порядок виконання завдань.
- Зробити висновки по роботі.

Контрольні запитання:

Вміти детально розказати про кожну з програм, яка входить до «Панелі управління»

ДОДАТОК 1.1

Вихідні дані до практичної роботи №1:

№ вар.	q1, кл	q2, кл	q3, кл	r12, м	r23, м
1	3.00E-08	1.00E-09	-5,00E-08	0,1	0,5
2	3.50E-08	2.00E-09	-5,50E-08	0,2	0,6
3	4.00E-08	3.00E-09	-6,00E-08	0,3	0,7
4	4.50E-08	4.00E-09	-6,50E-08	0,4	0,8
5	5.00E-08	5.00E-09	-7,00E-08	0,5	0,9
6	5.50E-08	6.00E-09	-7,50E-08	0,6	1
7	6.00E-08	7.00E-09	-8,00E-08	0,7	1,1
8	6.50E-08	8.00E-09	-8,50E-08	0,8	1,2
9	7.00E-08	9.00E-09	-9,00E-08	0,9	1,3
10	7.50E-08	1.00E-08	-9,50E-08	1	1,4
11	8.00E-08	1.10E-08	-1,00E-07	1,1	1,5
12	8.50E-08	1.20E-08	-1,05E-07	1,2	1,6
13	9.00E-08	1.30E-08	-1,10E-07	1,3	1,7
14	9.50E-08	1.40E-08	-1,15E-07	1,4	1,8
15	1.00E-07	1.50E-08	-1,20E-07	1,5	1,9
16	1.05E-07	1.60E-08	-1,25E-07	1,6	2
17	1.10E-07	1.70E-08	-1,30E-07	1,7	2,1
18	1.15E-07	1.80E-08	-1,35E-07	1,8	2,1
19	1.20E-07	1.90E-08	-1,40E-07	1,9	2,3
20	1.25E-07	2.00E-08	-1,45E-07	2	2,4
21	1.30E-07	2.10E-08	-1,50E-07	2,1	2,5
22	1.35E-07	2.20E-08	-1,55E-07	2,2	2,6
23	1.40E-07	2.30E-08	-1,60E-07	2,3	2,7
24	1.45E-07	2.40E-08	-1,65E-07	2,4	2,8

Вихідні дані до практичної роботи №2:

№ вар.	E, В	R1, Ом	R2, Ом	R3, Ом
1	200	50	1,00E+02	10
2	200	55	1,00E+02	20
3	200	60	1,00E+02	30
4	200	65	1,00E+02	40
5	200	70	1,00E+02	50
6	220	75	1,10E+02	60
7	220	80	1,10E+02	70
8	220	85	1,10E+02	80
9	220	90	1,10E+02	90
10	220	95	1,10E+02	100
11	240	100	1,20E+02	110
12	240	105	1,20E+02	120
13	240	110	1,20E+02	130
14	240	115	1,20E+02	140
15	240	120	1,20E+02	150
16	260	125	1,30E+02	160
17	260	130	1,30E+02	170
18	260	135	1,30E+02	180
19	260	140	1,30E+02	190
20	260	145	1,30E+02	200
21	280	150	1,40E+02	210
22	280	155	1,40E+02	220
23	280	160	1,40E+02	230
24	280	165	1,40E+02	240

Вихідні дані до практичної роботи №4:

№ вар					№ вар				
	24	20	27	86		36	44	26	18
1	69	18	71	47	7	34	82	92	37

	55	9	33	77		2	1	66	14
	-71	-89	-82	-35		-63	-66	-65	-86
	10	89	10	5		11	75	65	49
	46	64	2	28		91	76	67	82
	88	49	94	37		14	49	88	64
2	-66	-37	-24	-87	8	-14	-88	-96	-96
	3	20	37	17		21	60	76	20
	76	41	16	79		59	94	62	43
	38	56	86	71		18	20	52	11
3	-94	-92	-59	-99	9	-85	-22	-16	-22
	57	38	28	32		82	79	1	22
	32	31	74	59		49	6	79	32
	16	19	67	35		15	77	19	28
4	-51	-39	-73	-75	10	-11	-23	-94	-11
	7	83	62	93		99	11	82	92
	45	71	24	6		34	19	78	0
	26	2	79	20		74	72	38	76
5	-94	-34	-55	-18	11	-65	-10	-36	-48
	94	68	39	8		23	65	82	38
	66	81	11	76		5	70	31	31
	91	13	92	15		94	42	61	62
6	-97	-11	-30	-50	12	-23	-21	-48	-64

№ вар					№ вар				
	2	60	94	19		77	73	93	51
	94	35	95	58		97	11	78	44
	2	34	48	9		60	66	4	26
13	-90	-55	-22	-55	19	-74	-93	-90	-98
	12	89	63	98		46	48	3	56
	30	48	82	41		90	16	99	24
	60	17	2	76		41	44	22	40
14	-93	-61	-63	-76	20	-66	-58	-89	-51
15	16	39	3	94	21	59	0	7	70

	14	96	50	19		22	78	14	50
	93	35	83	74		78	29	41	87
	-17	-72	-79	-55		-19	-1	-93	-35
	84	100	73	6		94	81	95	68
	78	66	55	60		78	96	25	28
	82	15	10	46		61	40	43	10
16	-25	-95	-79	-95	22	-4	-11	-45	-6
	15	9	24	77		11	28	51	69
	14	31	98	94		24	58	25	42
	16	8	12	94		17	44	83	73
17	-93	-99	-3	-66	23	-16	-65	-61	-30
	64	1	22	85		27	81	79	69
	43	84	72	64		45	53	55	58
	36	39	96	89		84	96	97	10
18	-79	-69	-88	-42	24	-17	-67	-53	-72

Ключові терміни

Boot loader - завантажувач однієї з декількох ОС, встановлених на деякому комп'ютері, керований спеціальним меню при вмиканні комп'ютера.

Bluetooth – інтерфейс для бездротового підключення (за допомогою радіозв'язку) до комп'ютера мобільних телефонів, органайзерів, навушників, плеєрів та інших видів пристроїв.

BluRay – диск – різновид компакт-дисків великої ємності (25 – 50 ГБайт).

CISC (Complicated Instruction Set Computer – комп'ютер з ускладненою системою команд) – історично перший підхід до комп'ютерної архітектури, сенс якого полягає в ускладненості в системі команд внаслідок реалізації в них складних за семантикою операцій, що реалізують типові дії, часто використовувані при програмуванні й при реалізації мов (наприклад, групове пересилання рядків).

COM (communication port, serial port, послідовний порт) – порт для підключення до комп'ютера різних комунікаційних пристроїв, наприклад, модемів.

DMA (Direct Memory Access) – контролери із прямим доступом до оперативної пам'яті, минаючи використання спеціалізованої пам'яті пристрою.

Double bootable system - комп'ютер, на якому встановлені дві (або більше) операційні системи, при включенні якого користувачеві видається початкове меню для уточнення, яку саме ОС потрібно запустити.

EPIC (Explicit Parallelism Instruction Computers – комп'ютери з явним розпаралелюванням команд) – підхід до архітектури комп'ютера, аналогічний **VLIW**, але з додаванням ряду вдосконалень, наприклад, спекулятивних обчислень – паралельного виконання обох гілок умовної конструкції з обчисленням умови.

EPP (Extended Parallel Port) – двонаправлений режим роботи порту **LPT**, у якому він може працювати не тільки для виведення, але й для введення інформації.

Hard real-time – система реального часу, у який при порушенні тимчасових обмежень може виникнути критична помилка (відмова) керованого нею об'єкта.

HDMI (High Definition Multimedia Interface) – інтерфейс і порт, що дозволяє підключити до комп'ютера телевізор або інше відеоустаткування, що забезпечує найкращу якість відтворення (HD – High Definition).

IEEE 1394 (FireWire) – порт для підключення до комп'ютера цифрової відеокамери або фотоапарата.

LPT (від line printer), або **паралельний порт -** застарілий вид порта для підключення принтера або сканера з товстим у перерізі кабелем і великим рознімачем, що вимагає попереднього вимикання комп'ютера й пристрою для їхнього безпечного з'єднання.

Original Equipment Manufacturer (OEM) - фірма-розроблювач якогось зовнішнього пристрою, що звичайно розробляєй **драйвер** до нього.

PCI (Personal Computer Interface) – найбільш поширений тип системної шини, до якої приєднані процесор, пам'ять, диски, принтер, модем та інші зовнішні пристрої комп'ютерної системи.

RISC (Reduced Instruction Set Computer – комп'ютер зі спрощеною системою команд) – спрощений підхід до архітектури

комп'ютерів, що характеризується такими принципами: спрощення семантики команд; відсутність складних групових операцій; однакова довжина команд (32 або 64 біта, за розміром машинного слова); виконання арифметичних операцій тільки в регістрах і використання спеціальних команд запису й зчитування регістрів пам'яті; відсутність спеціалізованих регістрів; використання великого набору регістрів загального призначення (реєстрового файлу); передача при виклику процедур параметрів через регістри.

RS-232 - інша (більш застаріла) назва порту **COM**.

SCSI (Small Computer System Interface) – інтерфейс, адаптери й порти для підключення широкого спектра зовнішніх пристроїв – жорстких дисків, сканерів і ін. з можливістю обслуговування **гірлянди пристроїв**, що мають різні **(SCSI** IDs) номери та підключені до одного SCSI-Порту.

SCSI ID – номер пристрою (від 0 до 9), що є частиною **гірлянди SCSI-Пристроїв,** підключених до одного SCSI-Порту.

Soft real-time – система реального часу, порушення тимчасових обмежень якої не приводить до відмови керованого нею об'єкта.

TV-Тюнер - пристрій для прийому телевізійного сигналу з антени й показу телевізійного зображення на комп'ютері.

USB (Universal Serial Bus) – найпоширеніший універсальний порт комп'ютера (з характерним плоским рознімачем, розміром порядку 1 см, із зображенням тризубця), до якого можуть підключатися клавіатура, миша, зовнішній диск, принтер, сканер та інші зовнішні пристрої.

VLIW (Very Long Instruction Word – комп'ютери із широким командним словом) – підхід до архітектури комп'ютерів, заснований на таких принципах: статичне планування паралельних

обчислень компілятором на рівні окремих послідовностей команд і підкоманд; подання команди як "широкої" - утримуючої декілька підкоманд, виконуваних паралельно за той самий машинний такт на декількох однотипних пристроях процесора, наприклад, двох пристроях додавання й двох логічних пристроях.

Автоматизація управління *(Computer Aided Controlling)* — комплекс технічних, організаційних, економічних дій і заходів, реалізація яких забезпечує зниження або виключення особистої участі людини у здійсненні процесу управління. По суті це впровадження нової технології обробки інформації із застосуванням сучасних засобів обчислювальної техніки, створенням автоматизованої системи управління.

Автоматизована система (в інформаційних технологіях) *(Automated System (In Information Techology)* — система, що реалізує інформаційну технологію виконання встановлених функцій за допомогою персоналу та комплексу засобів автоматизації.

Адреса *(address)* — об'єкт, що визначає розміщення даних.

Асиметрична кластеризація (asymmetric clustering) – організація комп'ютерного кластера, коли один комп'ютер виконує додаток, а інші простоюють.

Асиметрична мультипроцесорна система (asymmetric multiprocessing) – багатопроцесорна комп'ютерна система, у якій процесори спеціалізовані за своїми функціями; є головний процесор, що планує роботу підлеглих процесорів.

Асинхронне введення-виведення – введення-виведення, виконуване паралельно з виконанням програми користувача.

Асоціативна пам'ять (кеш – cache) – ділянка пам'яті, розташована в більш швидкодійній системі пам'яті, яка зберігає елементи більш повільної пам'яті разом з їхніми адресами, що

найбільш часто використовуються, з метою оптимізації звертань до них.

Багатоядерний комп'ютер (multi-core computer) – найпоширеніша в наш час (2010 р.) архітектура комп'ютерів, при якій кожний процесор має кілька ядер (cores), об'єднаних в одному кристалі й такі, що працюють на одній і тій же загальній пам'яті, що дає широкі можливості для паралельних обчислень.

Багатоцільові комп'ютери (комп'ютери загального призначення, mainframes) – традиційна історична назва для комп'ютерів, розповсюджених в 1950-х – 1970-х рр., що використовувалися для вирішення будь-яких завдань.

Базовий регістр (base register) – системний регістр, використовуваний для захисту пам'яті та такий, що утримує початкову адресу ділянки пам'яті, виділену користувальницькій програмі.

Біт режиму – біт, що зберігається в системному регістрі й задає поточний режим виконання команд: дорівнює 0 для системного режиму й 1 – для користувальницького режиму.

Браузер *(browser)* — вікно перегляду — програма (в системах програмування з багатовіконним доступом), що дає змогу переглядати в групі виділених вікон текстові уявлення програм і даних.

Буфер (buffer) — область оперативної пам'яті, що тимчасово містить дані у процесі їхнього одержання, передавання, читання або запису. Використовується для згладжування швидкісних або часових характеристик пристроїв, застосовується у терміналах, периферійних пристроях, зовнішніх запам'ятовуючих пристроях і центральному процесорі.

Алгоритм *(algorithm)* — формальна процедура, що гарантує

одержання оптимального або коректного розв'язку задачі.

Аналіз вимог ***(requirementi analyses)*** — дослідження вимог користувача для специфікації системи.

Аналіз даних ***(data analyses)*** — дослідження даних у реальній або проектованій системі.

Атрибут ***(attribute)*** — найменування властивостей одного або кількох об'єктів. Атрибут іменує властивість, а його значення визначає конкретну властивість.

Веб-Сервер (Web server) – комп'ютер і програмне забезпечення, що надає доступ клієнтам через WWW до Web-Сторінок, розташованих на комп'ютері-сервері.

Вектор переривань (interrupt vector) – резидентний масив в оперативній пам'яті, у якому зберігаються доступні по номерах переривань адреси підпрограм-оброблювачів переривань (модулів ОС).

Віртуальний ***(virtual)*** — концептуальний чи можливий, але не той, що реально існує. Вказує на те, що елементи даних, структури або обладнання подають програмісту або користувачеві у виді, який відрізняється від реального. Перетворення елементів із реальних на віртуальні виконується програмним забезпеченням.

Визначення задачі ***(problem definition)*** — формулювання задачі, яке може містити опис даних, а також метод, процедури й алгоритм її розв'язання.

Внутрішня схема ***(internal schema)*** — фізична структура даних.

Вторинна пам'ять ***(secondary storage)*** — пам'ять, що не є складовою частиною обчислювальної машини, але безпосередньо з нею пов'язана і керована нею (вінчестер, гнучкі диски, CD—ROM та ін.).

Віртуальний COM-Порт – уявлюваний COM-Порт (такий, щоу дійсності не існує і не має рознімача), що ОС наче інсталює в систему при установці, наприклад, драйвера для взаємодії через Bluetooth або через кабель комп'ютера з мобільним пристроєм. Зазвичай має великий номер, наприклад, 18.

Гібридний процесор – новий підхід до архітектури комп'ютерів, який дуже поширюється, при цьому підході процесор має гібридну структуру – складається з (багатоядерного) центрального процесора **(CPU)** і (також багатоядерного) графічного процесора (GPU – Graphical Processor Unit).

Гірлянда SCSI-Пристроїв – ланцюжок пристроїв, підключених до одного SCSI-**Порту, які** мають різні **SCSI IDs** (номери).

Дані *(data)* — інформація, подана у формалізованому виді, придатному для пересування, інтерпретації або оброблення за участі людини чи автоматичними засобами.

Двійковий пошук *(binary search)* — метод пошуку в упорядкованій таблиці або файлі. Процедура полягає у поділі розглядуваної ділянки навпіл та виборі її верхньої або нижньої частини. Вибір ґрунтується на аналізі значення ключа всередині ділянки. Вибрана частина потім знову поділяється навпіл і так триває доти, поки не буде знайдено потрібний елемент.

Дескриптор (**descriptor**) – адресне слово в системах з теговою архітектурою; містить тегдескриптор, **адресу** початку адресованого масиву в пам'яті, **довжину** масиву й 4 біти зашиті – **від читання, від запису, від виконання** й **від запису адресної інформації**.

Діалоговий режим *(interactive mode)* — режим оперативної взаємодії користувача з обчислювальною системою.

Динамічний розподіл *(dynamic resource allocation)* — розподіл, за

якого ресурси, призначені для виконання програм, визначаються критеріями, що застосовуються у потрібний момент.

Динамічний розподіл пам'яті *(dynamic storage allocation)* — надання пам'яті процедури згідно з оперативним запитом на противагу наданню пам'яті процедури на підставі наперед передбачуваного запиту.

Доріжка (track) – частина **жорсткого диска**, розташована між двома концентричними колами на одному із магнітних дисків, що його становлять.

Драйвер – низькорівнева системна програма для керування якимось зовнішнім пристроєм (наприклад, жорстким диском).

Жорсткий диск (hard disk) - різновид **зовнішньої пам'яті, що** фізично складається із твердих пластин з металу або скла, покритих магнітним шаром для запису, шпинделя й головок зчитування – запису.

Елемент *(element)* — об'єкт (матеріальний, енергетичний, інформаційний), яким слід оперувати в дослідженні системи, але внутрішня будова (зміст) якого безвідносна до мети розгляду. Елементами можуть бути книга, верстат, працівник, документ, підприємство, сила, маса, енергія, файл, матриця, запис файлу тощо.

Елемент даних *(data item, data element)* — одиниця даних, щов певних контекстах розглядається як неподільна.

Запит *(record)* — звернення до програми з метою одержання інформації, що міститься в базі даних, яке формулюється у виді пропозиції або команди.

Зовнішні пристрої - див. **Пристрої введення-виведення**

Зовнішня (вторинна) пам'ять – розширення основної пам'яті, що забезпечує функціональність стійкої (що зберігається) пам'яті великого обсягу.

Ідентифікатор об'єкта або типу сутності *(entity identifier)* — атрибут або сукупність атрибутів, які однозначно ідентифікують об'єкт або тип сутності проблемної сфери.

Ієрархія *(hierarchy)* — різновид структури системи, елементи якої наділяються властивістю підпорядкування. Ієрархія називається сильною (ієрархія типу "дерево"), якщо між рівнями ієрархічної структури існують взаємовідносини суворої підпорядкованості компонентів нижчого рівня одному з компонентів вищого рівня. Ієрархія називається слабкою, якщо один і той самий вузол нижчого рівня ієрархії може бути підпорядкований кільком вузлам вищого рівня. Основне призначення ієрархічної організації у створюваній системі — розподіл функцій оброблення інформації та здійснення процедур вибору між окремими елементами системи.

Ініціатива щодо надійних і безпечних обчислень (trustworthy computing initiative) – ініціатива корпорації Microsoft (2002), метою якої є підвищення надійності й безпеки програмного забезпечення, насамперед – операційних систем.

Індекс (index) — таблиця, що використовується для визначення місця знаходження запису.

Інтерпретатор *(interpretive routine)* — програма, що розшифровує команди у псевдокодах і негайно виконує їх на відміну від компілятора, який розшифровує псевдокоди та формує програму машинною мовою для подальшого виконання.

Інтерфейс, дружній до користувача *(friendly user interface)* — властивість, що забезпечує комфортне робоче середовище при взаємодії користувача з інформаційною системою.

Інтерфейс користувача *(user interface)* — частина програми, що відповідає за діалог з користувачем і може мати форму природно- мовної системи (для інтелектуальних баз даних та

експертних систем).

Інфрачервоний порт (IrDA) – порт для підключення ноутбука до мобільного телефона (або двох ноутбуків один до одного) через інфрачервоний зв'язок.

Інформаційна система *(information system)* — система, що організує пам'ять і маніпулювання інформацією про проблемну сферу.

Інформація *(information)* — знання про предмети, факти, поняття тощо проблемної сфери, якими обмінюються користувачі системи оброблення даних.

Кластери з балансуванням завантаження (load-balancing clusters) – комп'ютерні кластери, які мають кілька вхідних комп'ютерів, що балансують запити (front- ends) та розподіляють завдання між комп'ютерами серверного бек - енда.

Кластери з високошвидкісним доступом (high-availability clusters, HAC) – комп'ютерні кластери, що забезпечують оптимальний доступ до ресурсів, наданих комп'ютерами кластера, наприклад, до баз даних.

Кластери комп'ютерів – групи комп'ютерів, фізично розташовані поряд і з'єднані один з одним високошвидкісними шинами й лініями зв'язку.

Клієнт-Серверна система – розподілена комп'ютерна система, у якій певні комп'ютери відіграють роль спеціалізованих серверів, а інші – роль клієнтів, що користуються їхніми послугами.

11. Кишеньковий портативний комп'ютер (КПК, органайзер) мініатюрний комп'ютер, що поміщається на долоні або в кишені, за своїми параметрами майже порівнюваний з ноутбуком, він призначений для повсякденного використання з метою запису, зберігання й читання інформації, у тому числі –

мультимедійної, і комунікації через Інтернет.

Кеш-пам'ять *(cache memory)* — надоперативна пам'ять, кеш. Запам'ятовуючий пристрій з високою швидкістю доступу (в кілька разів більшою, ніж швидкість доступу до основної оперативної пам'яті), що застосовується для тимчасового зберігання проміжних результатів і вмісту комірок пам'яті, які часто використовуються.

Кільцева структура *(ring structure)* — організація даних у ланцюжки, в яких кінець ланцюжка вказує на його початок, утворюючи таким чином кільце.

Класифікація *(classification, від лат.classis — розряд, fasio — роблю)* — один із найпростіших різновидів моделювання, розподіл предметів певного роду на взаємозв'язані класи згідно з найсуттєвішими ознаками, властивими предметам одного роду, які відрізняють їх від предметів інших родів.

Ключ *(key)* — ідентифікатор, що міститься в середині набору елементів даних.

Ключ вторинний *(secondary key)* — ключ, що може ідентифікувати щонайменше один запис; ключ, який містить значення атрибуту (елемента даних), відмінного від унікального ідентифікатора(пошуковий ключ).

Ключ зовнішній *(foreign key)* — атрибут підпорядкованого логічного запису, що є первинним ключем породжувального логічного запису і слугує для з'єднання записів за цим атрибутом.

Комп'ютер переносний - надмініатюрний комп'ютер, убудований в одяг або імплантований у тіло людини, призначений для обробки інформації від датчиків, управління спеціалізованими пристроями (наприклад, кардіостимулятором), або видачі рекомендацій з навігації й виконання інших типових дій людини.

Контролер пристрою – спеціалізований процесор (пристрій керування) для якогось пристрою комп'ютерної системи - основної пам'яті або зовнішнього пристрою.

Контрольна точка/рестарт *(check point /restarti)* — спосіб повторного пуску програми з точки, що відрізняється від початкової. Використовується після виникнення аварійної ситуації чи переривання. Контрольні точки можуть бути створені через певні проміжки часу роботи прикладної програми. В них запам'ятовуються записи, що містять достатню кількість інформації про стан програми і дають змогу здійснити її рестарт із зазначеної точки.

Контрольний приклад *(check problem)* — приклад із відомим розв'язком, що використовується для перевірки правильності роботи функціонального блока.

Користувальницький (непривілейований) режим (user mode) – стандартний режим виконання програм, у якому виконуються програми користувачів. У даному режимі заборонені деякі привілейовані операції (наприклад, зміна системних ділянок пам'яті й регістрів).

Лічильник команд – адреса поточної виконуваної або перерваної команди процесора.

Локальна мережа *(local area network(lan))* — мережа передачі даних, що об'єднує кілька цифрових пристроїв, розташованих на обмеженій площі, яка не перевищує десятки квадратних кілометрів. Як пристрої можуть бути робочі місця, міні - та мікрокомп’ютери, інтелектуальна інструментальна апаратура тощо.

Математичне забезпечення *(mathematical support)* — сукупність економіко-математичних методів, моделей та алгоритмів оброблення інформації в автоматизованій інформаційній системі.

Материнська плата (motherboard) – основна друкована

плата комп'ютера, на якій змонтовані процесор і пам'ять.

Мережний адаптер (мережна карта) – пристрій для підключення комп'ютера до локальної мережі.

Мобільний пристрій (мобільний телефон, комунікатор) – кишеньковий пристрій, призначений для голосового зв'язку, обміну короткими повідомленнями, а також для читання, запису й відтворення мультимедійної інформації й комунікації через Інтернет.

Модем (абревіатура від **модулятор – демодулятор**) – пристрій для виходу в Інтернет і передачі інформації по аналоговій або цифровій телефонній лінії.

Мова опису даних *(data description language)* — штучна мова для опису даних та їхніх взаємозв'язків у базі даних.

Модель *(model, від лат. "зразок")* — штучно створений об'єкт (реальний або ідеальний), який, будучи аналогічним (подібним) досліджуваному об'єкту, відображає і відтворює у простішому вигляді структуру, зв'язки та взаємозв'язки між елементами досліджуваного об'єкта, безпосереднє вивчення якого пов'язано з певними труднощами, тим самим полегшує процес одержання інформації про нього.

Моделювання *(modeling)* — вид творчої діяльності людини, що містить такі процедури: конструювання моделі на підставі попереднього вивчення об'єкта і виділення його основних характеристик; експериментальний та теоретичний аналіз моделі; зіставлення результатів з даними про об'єкт; корекція моделі; її використання в пізнавальному, творчому або виробничому процесах.

Модель математична *(mathematical model)* — абстрактна або знакова модель, побудована засобами математики (у виді системи рівнянь, графа, формули логіки).

Модуль *(module)* — цілісна група елементів системи, описана

тільки своїми входами і виходами. Модуль як частина системи відображує зв'язок між елементами, тобто обмін речовиною, енергією, інформацією. Модульний підхід дає змогу значно спростити опис системи і зробити видимими та доступними для розуміння найскладніші системи.

Мультипрограмування *(multiprogrammihg)* — режим роботи, що передбачає почергове виконання двох або більше програм одним процесором.

Настільний комп'ютер – персональний комп'ютер, розташовуваний на робочому столі й використовуваний на роботі або дома.

Обчислювальне середовище – інтегрована розподілена комп'ютерна система для рішення завдань у якихось проблемних сферах.

Операційна система – базове системне програмне забезпечення, що управляє роботою комп'ютера і є посередником (інтерфейсом) між апаратурою, прикладним програмним забезпеченням і користувачем комп'ютера.

Опитування пристроїв (polling) – дії операційної системи по періодичній перевірці стану всіх портів і зовнішніх пристроїв, що можуть мінятися із часом.

Основна (оперативна) пам'ять – швидкодіюча пам'ять, до якої процесор має безпосередній доступ під час виконання програми, що зберігає програми й дані, інформація в якій не зберігається після вимикання комп'ютера або перезапуску системи.

Пам'ять – частина комп'ютера, що зберігає дані й програми.

Паралельна комп'ютерна система – мультипроцесорна система, що складається з декількох безпосередньо взаємодіючих процесорів.

Паралельний порт, або LPT (абревіатура від Line PrinTer) – порт для підключення застарілих моделей принтерів. Для підключення принтера через даний порт потрібно попередньо відключити й принтер, і комп'ютер.

Переривання за таймером – періодичні переривання через певний квант часу, призначені для **опитування пристроїв** і інших необхідних періодичних дій ОС.

Переповнення (*overflow*) — ситуація, коли запис (або сегмент) не може бути розміщений за його власною адресою, тобто адресою пам'яті, яку логічно було присвоєно йому в процесі завантаження.

Підсистема *(sybsystem)* — частина системи, що вивчається або досліджується самостійно. Є компонентом більшим, ніж. окремий елемент системи, але меншим, ніж система загалом. Повинна мати властивості системи, але для неї має бути сформульована своя локальна мета функціонування.

Підсистема управління ресурсами – компонент операційної системи, що управляє обчислювальними ресурсами комп'ютера.

Покажчик *(pointer)* — адреса запису (або іншого групування даних), що міститься в іншому записі. Прочитавши один запис, програма, використовуючи покажчик, може одержати доступ до іншого запису. Адреса може бути абсолютною, відносною або символічною.

Поле даних *(date field)* — складова частина запису, що відповідає певному атрибуту.

Порт – пристрій з рознімачем і контролером для підключення до комп'ютера зовнішніх пристроїв.

Портативний комп'ютер (ноутбук, лаптоп) – мініатюрний комп'ютер, за своїми параметрами не уступає настільному, але за своїми розмірами вільно поміщається в невелику сумку й

призначений для використання в поїздці, дома, на дачі.

Прикладна задача ***(application problem)*** — задача, ініційована користувачем, що потребує для її розв'язання оброблення інформації.

Проблема ***(problem)*** — ситуація, яка має наявний і бажаний стани. Кожен з них містить набір об'єктів, властивостей та зв'язків, об'єднаних у процес. Проблеми поділяють на кількісні, якісні, слабо-структуровані та мішані.

Прикладне програмне забезпечення – програми, призначені для рішення різних класів завдань.

Пристрої введення-виведення – пристрої комп'ютера, що забезпечують введення інформації в комп'ютер і виведення результатів роботи програм у формі, сприйманої користувачем або іншими програмами.

Програма, керована перериваннями (interrupt-driven program) – програма, що запускається автоматично при виникненні переривання центрального процесора (наприклад, операційна система).

Програмувальне переривання (trap; дослівно – пастка) – переривання, яке явно генерується за допомогою спеціальної команди процесора (зазвичай для обробки помилки в програмі).

Прокси-Сервер – комп'ютер і програмне забезпечення, що є частиною локальної мережі й підтримує ефективне звернення комп'ютерів локальної мережі до Інтернету, фільтрацію трафіка, захист від зовнішніх атак.

Процес ***(process)*** — послідовність станів системи, що відповідає впорядкованій зміні конкретного параметра, який визначає характерні властивості системи.

Реальний час ***(real time)*** — час, що збігається з фактичним часом перебігу фізичного процесу. Стосовно режиму оброблення

даних це означає, що потрібні розрахунки виконуються під час фактичного перебігу відповідного фізичного процесу так, що результати обчислень можна використовувати для керування цим процесом. Стосовно прикладних програм це значить, що в них відповідь на інформацію, яка вводиться, формується досить швидко, так що вона може впливати на подальше введення даних (діалоговий режим).

Регістр межі (limit register) – системний регістр, використовуваний для захисту пам'яті й такий, що утримує довжину ділянки пам'яті, виділеної користувальницькій програмі.

Рекурсія *(recursion)* — властивість структури даних або правила, що полягає в можливості звернення до самої себе один чи більше разів.

Робоча пам'ять *(working storag)* — частина оперативної пам'яті ЕОМ, що резервується для розміщення тимчасових результатів операцій.

Робоча станція *(workstation)* — підключений до мережі персональний комп'ютер користувача, якому доступні її ресурси.

Розв'язування задачі *(problem solve)* — процес одержання підсумкового показника (документа, відеокадра), що містить інформацію для прийняття рішень під час управління діяльністю суб'єкта господарювання.

Розподілена система (distributed system) – комп'ютерна система, у якій обчислення розподілені між декількома фізичними процесорами (комп'ютерами), об'єднаними між собою в мережу проводну або безпроводну мережу.

Сектор – частина **жорсткого диска**, обмежена **доріжкою** й двома радіусами.

Сервер баз даних (database server) – комп'ютер і програмне

забезпечення, що надає доступ іншим комп'ютерам мережі до баз даних, розташованих на комп'ютері-сервері локальної мережі.

Серверний бек-енд (Server back-end) – група (пул) зв'язаних у локальну мережу серверних комп'ютерів, використовуваних замість одного сервера, з метою більшої надійності й надання більшого обсягу ресурсів.

Сервер додатків (application server) – комп'ютер і програмне забезпечення, що надає обчислювальні ресурси (пам'ять і процесор) і необхідне оточення для віддаленого запуску певних класів (як правило, більших) додатків з інших комп'ютерів локальної мережі.

Сервер електронної пошти – комп'ютер і програмне забезпечення, що виконують відправлення, одержання й "розкладку" електронної пошти для комп'ютерів деякої локальної мережі. Може забезпечувати також криптування пошти (email encryption).

Синхронне введення-виведення – операція введення-виведення, виконання якої приводить до переходу програми в стан очікування, доти, поки операція введення-виведення не буде повністю закінчена.

Системний виклик (system call) – явний запит користувальницької програми до ОС шляхом виклику системної підпрограми.

Система реального часу – обчислювальна система, призначена для управління технічним, військовим або іншим об'єктом у режимі реального часу.

Системний (привілейований) режим (system mode, kernel mode, monitor mode) – особливий режим виконання команд, у якому виконуються модулі ядра ОС, що допускають виконання ряду привілейованих операцій, наприклад, зміну системних ділянок пам'ятій регістрів

Системна шина (system bus) – комунікаційний пристрій, що з'єднує між собою всі модулі комп'ютерної системи - центральний процесор, пам'ять і контролер пам'яті, зовнішні пристрої і їхні контролери, які через системну шину обмінюються сигналами.

Слабо зв'язана система (loosely coupled system) –розподілена комп'ютерна система, у якій кожний процесор має свою локальну пам'ять, а різні процесори взаємодіють між собою через лінії зв'язку.

Сканер – пристрій для оцифрування паперових зображень, наприклад, підписаних або рукописних документів.

Симетрична кластеризація (symmetric clustering) - організація комп'ютерного кластера, при якій всі машини кластера виконують одночасно різні частини одного великого додатка.

Симетрична мультипроцесорна система (symmetric multiprocessing - SMP) - багатопроцесорна комп'ютерна система, всіпроцесори якої рівноправні й використовують ту саму копію ОС; операційна система при цьому може виконуватися на будь-якому процесорі.

Сервер мережі *(server)* — комп'ютер, підключений до мережі, що надає певні послуги користувачам.

Система *(system)* — сукупність елементів, зв'язки між якими з'єднують їх у структуру; об'єктивно існуючий комплекс процесів і явищ, а також зв'язків. Основні особливості систем: цілісність, відносна відокремленість від навколишнього середовища, наявність зв'язків з ним, наявність частин і зв'язків між ними (структурованість), підпорядкованість всієї системи конкретній меті.

Система збирання даних *(data acquisition (collection)* — найпростіша система телеоброблення, призначена вона для передачі інформації від абонентських систем до процесора. Наприклад,

диспетчерська служба, що збирає інформацію від абонентських систем, обробляє її та передає на центральний диспетчерський пункт.

Система колективного користування *(multiaccess system)* — універсальна система, орієнтована на інтерактивний режим роботи віддалених користувачів.

Система оброблення даних *(data processiong system)* — система, що складається з сукупності технічних і програмних засобів, а також обслуговуючого персоналу, які забезпечують оброблення даних.

Система підтримки прийняття рішень *(decision support system)* — інтерактивна прикладна система, що забезпечує кінцевим користувачам, які приймають рішення, легкий та зручний доступ до даних і моделей з метою прийняття рішень у погано структурованих та неструктурованих ситуаціях різних сфер людської діяльності.

Система реального часу *(real-time system)* — інформаційно-керуюча система, що забезпечує передачу та оброблення даних зі швидкістю, яка відповідає швидкості перебігу керованого або контрольованого процесу.

Система розподіленого оброблення даних *(data distributed processing system)* — система оброблення даних, у якій база даних є централізованою (корпоративною), а обчислювальні потужності та програмне забезпечення розподілені між взаємозв'язаними ЕОМ.

Складність *(complexity)* — властивість явища (об'єкта, процесу, системи), що проявляється в несподіваності, непередбачуваності, незрозумілості, випадковості, "антиінтуїтивності" його поведінки. Виникає внаслідок непростої взаємодії великої кількості об'єктів, поведінка одного або кількох об'єктів впливає на поведінку інших.

Сортування *(sort)* — упорядкування файлу згідно з заданим

ключем.

Список *(list)* — упорядкована множина елементів даних.

Стан процесора – значення регістрів і значення **лічильника** команд .

Структура *(structure, від лат. "будова")* — суттєві взаємозв’язки між елементами та їхніми групами (підсистемами), які мало змінюються при змінах у системі й забезпечують її існування та основні властивості. Зображають її у виді схеми з комірками, що відповідають елементам або групам елементів, і лініями, які з'єднують їх.

Суперкомп'ютер – потужний багатопроцесорний комп'ютер з продуктивністю до декількох петафлопс (1015 дійсних операцій у секунду), призначений для рішення завдань, що вимагають великих обчислювальних потужностей, наприклад, моделювання, прогнозування погоди.

Сутність *(entity)* — будь-який конкретний або абстрактний об'єкт зі зв'язками між іншими об'єктами.

Таблиця стану пристроїв – таблиця, яку зберігає та використовує операційна система, у якій кожному пристрою відповідає елемент, що містить тип пристрою, його адресу й стан, а для зайнятого пристрою – посилання на чергу оброблюваних запитів до нього.

Таймер – системний регістр, що містить деяке встановлене спеціальною командою значення часу, що зменшується через кожний квант (такт) процесорного часу. Коли значення таймера дорівнює нулю, відбувається переривання.

Тісно зв'язана (tightly coupled) система – паралельна комп'ютерна система, у якій процесори розділяють загальну пам'ять і таймер (такти); взаємодія між ними відбувається через загальну пам'ять.

Технічне забезпечення *(hardware)* — комплекс технічних засобів, що забезпечують роботу автоматизованої інформаційної системи.

Технологічне забезпечення *(technology support)* — сукупність організаційних, методичних і технологічних документів, що регламентують процес людино-машинного оброблення інформації в автоматизованій інформаційній системі.

Технологія електронної пошти *(e-maii technology)* — технологія комп'ютерного способу пересилання та оброблення інформаційних повідомлень, що забезпечує оперативний зв'язок між різноманітними користувачами.

Транзакція *(transaction)* — вхідне повідомлення, що належить до наявного файлу. Описує подію, яка зумовлює або формування нового запису файлу, або зміну чи вилучення наявного запису. Транзакція в сітьовому середовищі обчислень (розподілена транзакція) — логічна одиниця роботи, а також одиниця відновлення, паралелізму і цілісності.

Трафік *(traffic)* — потік інформаційного обміну, робоче навантаження лінії зв'язку.

Управління *(direction)* — процес організації цілеспрямованого впливу на об'єкт (систему), внаслідок якого він переходить у потрібний цільовий стан.

Управляюча програма – компонент операційної системи, управляючої виконанням інших програм і функціонуванням пристроїв вводу-виводу.

Файл *(file)* — множина записів одного типу.

Файл-Сервер (file server) – комп'ютер і програмне забезпечення, що надають доступ до підмножини файлових систем, розташованих на дисках комп'ютера-сервера, іншим комп'ютерам

локальної мережі.

Флеш-Пам'ять (флешка) – модуль зовнішньої пам'яті компактного розміру, що підключається через USB-Порт і має ємність до 128 ГБайт.

Хмарні обчислення – модель обчислень, заснована на **динамічно масштабованих (scalable)** і **віртуалізованих** ресурсах (даних, додатках, ОС і ін.), які доступні й використовуються як **сервіси** через Інтернет і реалізуються за допомогою високопродуктивних **центрів обробки даних (data centers).**

Центральний процесор – центральна частина комп'ютера, що виконує його команди (інструкції).

Циліндр – частина **жорсткого диска**, що представляє собою сукупність **доріжок** одного діаметру, що перебувають на всіх його паралельно розташованих магнітних дисках.

Черга переривань – системна структура ОС, що забезпечує почергову обробку всіх викликаючих переривань.

Ядро – основний низькорівневий компонент будь-якої операційної системи, виконуваний апаратурою в привілейованому режимі, що завантажується при запуску ОС і резидентно перебуває в пам'яті.

ЛІТЕРАТУРА

1. Зарецька Т.І., Колодяжний Б.Г. та ін. Інформатика. Навч. посіб..К., Навчальна книга, 2002. – 425 – 438 с.

2. Збірник задач з курсу „Інформатика та програмування” для студентів механіко-математичного

факультету.: Вакал Є.С., Личман В.В., Обвінцев О.В., Бублик В.В., Попов В.В. - К.: ВПЦ

„Київський університет”, 2004.

3. Лабораторний практикум з інформатики та комп’ютерних технологій / В.В. Браткевич та ін./ З

ред. О.І. Пушкаря: Навч. посібник. – Х.: Видавничій дім «ІНЖЕК», 2003. – 424 с .

4. Інформатика: Комп'ютерна техніка. Комп'ютерні технологи / За ред. О. І. Пушкаря, - К.: Видав.

центр «Академія», 2002. – 704 с.

5. Ковалюк Т.В. Основи програмування. – К.: Видавнича група BHV, 2005. – 384 с.

6. Інформатика, комп’ютерна техніка, комп’ютернітехнології: Посібник/ За ред. О.І. Пушкаря - К.: Академія, 2001. – 696 с.

7. Філіппенко І.Г., Гончаров В.О., Меркулов В.С. Основи побудови ПК: Конспект лекцій з курсу "Обчислювальна техніка та програмування”. – Харків: УкрДАЗТ, 2005. –Ч. 1.

8. Філіппенко І.Г., Гончаров В.О., Меркулов В.С. Основи алгоритмізації: Конспект лекцій з курсу "Обчислювальна техніка та програмування”. – Харків: УкрДАЗТ, 2005. –Ч. 2.

9. Методичні вказівки до лабораторних робіт з дисциплін “Основи інформатики”, „Обчислювальна техніка та програмування” (основи проектування алгоритмів). – Харків: ХарДАЗТ, 2000. – Ч. 1

10. Методичні вказівки до лабораторних робіт з дисциплін “Основи інформатики”, „Обчислювальна техніка та програмування” (програмування мовою BASIC). – Харків: ХарДАЗТ, 2000. –Ч.2

11. Методичні вказівки до лабораторних робіт з дисциплін “Основи інформатики”, „Обчислювальна техніка та програмування” (програмування мовою BASIC). – Харків: ХарДАЗТ, 2000. –Ч. 3.