А. Ф. ВЕРЛАНЬ, Н. В. АПАТОВА

ІНФОРМАТИКА

Підручник
для учнів 10—11 класів
середньої загальноосвітньої
школи

Допущено
Міністерством освіти України

Київ «Форум» 2001

ББК 32.973я721
В33

Допущено
Міністерством освіта Україна
(Рішення Колегії Міністерства освіти України
протокол № 16/3-18 від 12.08.97 р.)

За редакцією академіка АПН України,
д-ра пед. наук, проф. М.І. Жалдака

Верлань А.Ф., Апатова Н.В.

В33 Інформатика: Підруч. для учнів 10—11 кл. серед, загальноосвіт. шк.- К.: Форум, 2001.-255 с.

ISBN 966-7786-33-1.

ББК 32.973я721

ISBN 966-7786-33-1

Вступ

Інформатика — це наука про методи та засоби отримання, обробки, зберігання, передавання, подання інформації.

Протягом усієї історії свого існування люди, спілкуючись з природою та між собою, набували знань про навколишню дійсність і про самих себе. Від кількості цих знань, від ефективності їх використання залежали і дедалі більше залежать умови життя і окремих людей, і людства в цілому. Знання як інформація або як відомості про явища та різноманітні за своєю природою — фізичною, хімічною, біологічною, соціальною, економічною і т. ін. — об'єкти набуваються шляхом спостереження. Природними засобами спостереження є органи чуттів людини, передусім зір і слух. Спочатку різноманітні відомості просто накопичувались. А вже маючи їх у достатній кількості, люди помічали, що існують певні закономірності, які можна узагальнити і обґрунтувати. У результаті такі закономірності набували вигляду прикмет, наприклад прикмет, пов'язаних із прогнозами врожаю. Прикмети ставали характерними ознаками явищ, тобто виділялися як їхні властивості.

Властивості можуть бути різноманітними не тільки за своєю назвою, а й за своїм значенням. Кожний об'єкт або явище має групу властивостей, що вивчаються різними науками.

Проілюструємо наведене на прикладі літака, який розглядається як об'єкт руху. У механіці вивчаються його властивості як матеріальної точки, яка має швидкість і напрямок, в аеродинаміці розглядаються його форми, профілі крила та інші властивості, а для пасажира основною властивістю літака є своєчасні виліт і прибуття. Така властивість літака, як швидкість, може мати нескінченну кількість значень. Значення можуть вибиратися з неперервної (наприк-

лад, точки прямої) або з дискретної (наприклад, натуральне число або прізвище зі списку) множини. Так, неперервну зміну швидкості автомобіля показує стрілка спідометра, але його шкала має лише кілька позначок. Тому, визначаючи швидкість, водій фіксує її значення приблизно, одержуючи конкретне число. Коли ж ідеться про характеристику автомобіля в цілому, то зазначається, при якій конкретній швидкості чи діапазоні швидкостей витрата палива мінімальна. Таким чином, говорячи про певний об'єкт, ми визначаємо якісні й кількісні значення його різноманітних властивостей. Для додаткової характеристики об'єкта також використовуються його атрибути. Атрибутами автомобіля, наприклад, можуть буги марка, колір, форма кузова та ін.

Інформація як відомості про об'єкт або явище відображається у вигляді конкретних даних. Зв'язок реального світу, інформації і даних показано на рис. 1.

Рис. 1. Зв'язок реального світу, інформації і даних

Число або слово, що відображає якісну ознаку, — це дані, що можуть передаватися, оброблятися, зберігатися.

Інформація може існувати в різноманітних формах, у вигляді різних сигналів. Перехід від природних для людини сигналів (звуків, жестів) до їх письмових позначень — це приклад існування різних форм подання інформації. Наочним прикладом перетворення форми подання інформації може бути переклад текстів з однієї природної мови спілкування (української, російської та ін.) на іншу (англійську, німецьку та ін.).

Діяльність сучасної людини постійно пов'язана з отриманням певних відомостей, необхідністю збереження їх у часі, перетворення з однієї форми на іншу, переміщення у просторі. Діяльність людини, пов'язана з процесами отримання, перетворення, накопичення, зберігання, передавання, подання інформації, називається інформаційною діяльністю.

До початку двадцятого століття для передавання та зберігання інформації використовувалися книги, а її опрацюванням займалася сама людина. Однак науково-технічна революція зробила людину безсилою в сучасному потоці друкованої, кіно-, теле- та іншої продукції. Щорічно в світі друкується близько 100 тис. назв журналів на 60 мовах, 5 млн наукових статей, книг, брошур, 250 тис. дисертацій і звітів. Управління виробництвом, використання складних технологій вимагає від керівників швидкого аналізу ситуацій, що виникають, прийняття правильних рішень.

Складна наукова задача вже не може бути розв'язана вручну, навіть якщо талановитий учений витратить на неї все своє життя. Опрацьовувати всі ці дані людині допомагає комп'ютер. В результаті такої взаємодії можлива ефективна обробка первинної інформації та одержання інформації нової якості.

Якщо розглядати комп'ютер як об'єкт, що має складові, які пов'язані між собою і функціонують згідно правил, тоді комп'ютер можна назвати системою. Оскільки комп'ютер використовується здебільшого як засіб збереження, пошуку та подання інформації, то його можна розглядати як інформаційну систему.

Інформаційні системи допомагають аналізувати проблеми і розробляти нові вироби. Інформація є одним з найцінніших ресурсів суспільства поряд з такими природними багатствами, як нафта, газ та інше. Отже, методи і засоби переробки інформації як і переробки матеріальних ресурсів теж можна визначити як технологію.

За визначенням одного із засновників вітчизняної інформатики В.М.Глушкова, інформаційні технології — це процеси, пов'язані з обробкою інформації. Уточнюючи це поняття, можна сказати, що інформаційні технології — це методи, які реалізуються засобами обчислювальної техніки і забезпечують виконання заданих вимог до пошуку, подання, перетворення, передавання інформації.

Комп'ютерні засоби та інформаційні технології дозволили значно підвищити можливості й віддачу такого всеохоплюючого методу пізнання та творення, як моделювання об'єктів, явищ та процесів, як тих, що існують у природі, так і тих, що створюються людиною штучно.

У процесі виробничої та інтелектуальної еволюції, у боротьбі за виживання та зростання добробуту людина навчилася моделювати велику кількість об'єктів і явищ. Споруди — мости, будинки, дамби тощо — вивчались за допомогою макетів, зменшених копій цих об'єктів. Це так звані фізичні моделі, чи більш узагальнено — натурні моделі.

З часом збільшується кількість об'єктів, які досліджуються; об'єкти ускладнюються, і натурне моделювання стає невигідним, не економним. Потім для дослідження об'єктів починають застосовувати математику. Математичні рівняння дають можливість описувати і досліджувати більш складні об'єкти і явища — від найпростіших механізмів до біологічних процесів та логічних міркувань. При цьому все це виконується на папері без побудови макетів, без проведення натурних експериментів, що дорого коштують.

У першій половині XX ст. з'явились і набули значного поширення в наукових та інженерних колах електричні моделі багатьох видів фізичних об'єктів. Наприклад, електричний коливальний контур здатний моделювати механічні маятники.

Складні механізми моделюються електричними ланцюгами як більш доступними для побудови та вимірювання. Було розроблено багато інших методів та засобів моделювання.

Застосування математичних моделей — рівнянь, нерівностей, формул, за допомогою яких описують об'єкти — і є математичне моделювання. Для реалізації складних математичних моделей недостатньо аналітичного апарату досліджень, а потрібні ефективні чисельні методи.

Використання чисельних методів для реалізації математичних моделей на комп'ютері потребує механізму їх перенесення на комп'ютер, тобто потребує такого засобу, за допомогою якого б комп'ютер «зрозумів» як розв'язувати задачі. Таким засобом є спеціально створені мови. їх називають *мовами програмування.*

Реалізувати великий потенціал математичного моделювання неможливо без потужних засобів автоматизації обчислень, якими є сучасні комп'ютери. Так, без комп'ютерів неможливо було б детально промоделювати складні фізичні процеси при розробці джерел атомної енергії, розрахувати

траєкторії супутників та космічних кораблів, розв'язати багато інших задач.

«Налаштування» комп'ютера користувачем на розв'язання якоїсь задачі дослідження конкретного об'єкта і означає створення комп'ютерної моделі об'єкта. Використання такої моделі для визначення стану, поведінки, параметрів, характеристик об'єкту являє собою процес комп'ютерного моделювання.

Комп'ютерне моделювання — це самостійний науковий напрям, що не завжди зводиться до простої автоматизації обчислень, реалізації математичних моделей. Сучасні можливості засобів комп'ютерного моделювання вражаючі, вони допомагають робити нові відкриття у фізиці, грають в шахи на рівні чемпіонів світу.

Завдяки появі потужних комп'ютерів і розвитку інформаційних технологій створюються методи та засоби комп'ютерного моделювання, здатні розв'язувати складні та надскладні практичні задачі, такі як складання достовірних прогнозів погоди чи врожаю, керування великими енергетичними системами, моделювання регіональних та загальнодержавних систем, проектування літаків, кораблів тощо.

Предмет «Інформатика» розглядає теоретичні і практичні аспекти інформаційних технологій.

1. Що таке інформатика?
2. Що вивчає інформатика?
3. Чим характеризується об'єкт чи явище?
4. Наведіть приклади різних якісних характеристик об'єкта, що вивчаються різними науками (об'єкти: людина, повітря, вода, земля, космос та ін.).
5. Що ми розуміємо під словом «інформація»?
6. Який зв'язок між даними і реальним світом?
7. Що таке дані? Яких основних форм вони набувають?
8. Обґрунтуйте необхідність використання комп'ютера при розв'язанні сучасних задач опрацювання інформації.
9. Що таке інформаційна технологія?
10. Заповніть порожній прямокутник на рис. 2 і зробіть спробу доповнити структуру.

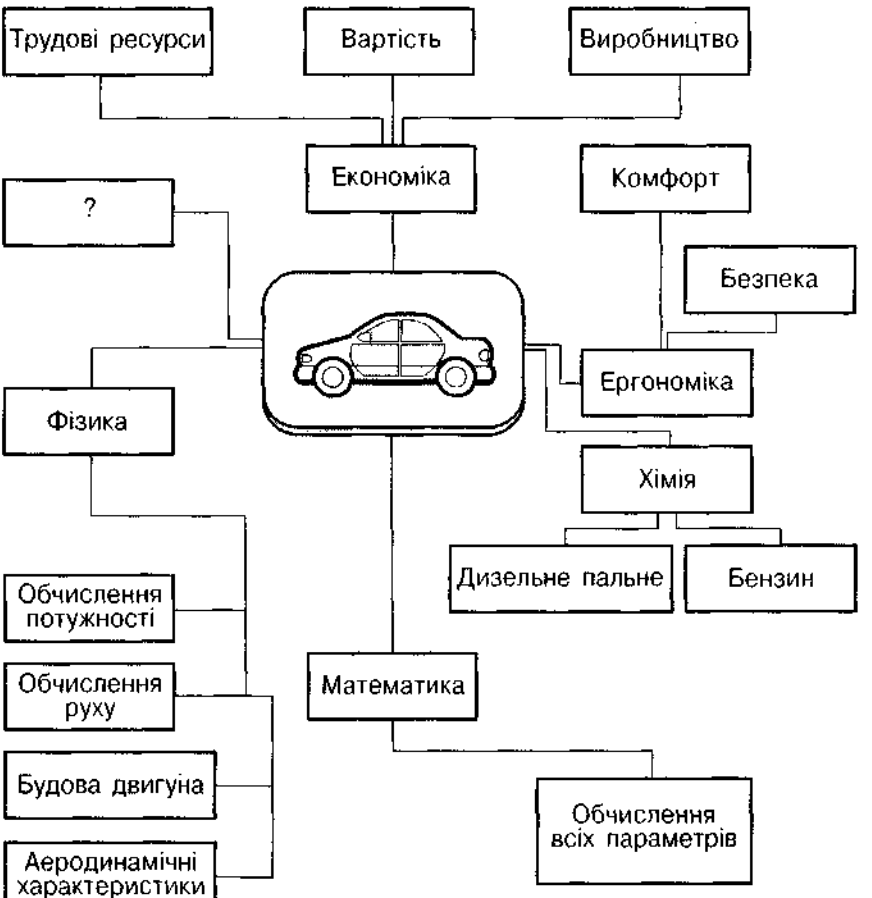

Рис. 2. Об'єкт «автомобіль» та множини атрибутів, що його характеризують

Частина 1

Персональний комп'ютер

1 . 1. Історія створення комп'ютерів

Перші спроби створення інструментів для обробки інформації пов'язані з прагненням спростити та прискорити виконання дій над числами. У Стародавньому Китаї близько чотирьох тисяч років тому була винайдена рахівниця. Греки та римляни більше двох тисячоліть тому почали використовувати абак — рахункову дошку, на якій числа зображувались певною кількістю камінців, а дії над числами виконувалися пересуванням камінців.

У 1642 р. відомий французький фізик і математик Б. Паскаль винайшов арифмометр — механічний прилад для виконання додавання та віднімання чисел. А через двадцять років німецький математик Г. Лейбніц удосконалив цю машину, додавши до її можливостей дії множення і ділення. Арифмометри кілька століть вірно слугували людям, будучи незамінними помічниками в бухгалтерському обліку, наукових розрахунках та в багатьох інших сферах. Однак можливості арифмометра були обмежені, швидкість обчислень була незначною, а «пам'ять» давала змогу зберігати лише результат попередньої операції з числами.

Наприкінці XIX ст. в США проводився черговий перепис населення. Для опрацювання результатів було застосовано табулятор Г. Холлерита — пристрій для обробки перфокарт. Перфокарти, невеликі аркуші тонкого картону розміром з паперовий долар, мали 12 горизонтальних рядів, в кожному з яких можна було пробити до 20 отворів. Комбінації отворів відповідали' таким даним, як вік, стать, місце народження, кількість дітей та ін. Г. Холлерит у 1924 р. заснував фірму IBM (International Business Mashines Corporation), — відомого в наш час виробника сучасних комп'ютерів.

Ідеї, реалізовані в сучасних комп'ютерах, були сформульовані ще в 1830 р. англійським ученим Ч. Беббіджем.

Розроблена ним Аналітична машина повинна була мати пам'ять, що вміщує до 100 сорокарозрядних чисел, та арифметичний пристрій. Результати операцій також мали зберігатися в пам'яті. Ч. Беббідж хотів побудувати свою машину з механічних елементів із використанням парового двигуна, але тогочасний технічний рівень так і не дозволив йому цього зробити. Тільки згодом, більш як через сто років, ці ідеї знайшли своє реальне втілення.

Головна ідея Аналітичної машини — виконання обчислювальних операцій за інструкціями, що наперед задаються. Іншими словами, ця машина мала працювати за програмою.

Розробка сучасних обчислювальних машин, або комп'ютерів (англ. computer — обчислювач), почалася в різних країнах у ЗО—40-х рр. XX ст.

У 1943 р. у США було створено обчислювальну машину Марк-1, основними елементами якої були електромеханічні реле. Ця машина мала довжину 17 м і висоту 2,5 м, містила близько 750 тис. реле, з'єднаних проводами загальною довжиною понад 800 км. Вона пропрацювала в Гарвардському університеті майже 17 років. За день машина виконувала таку кількість обчислень, на яку раніше потрібно було півроку.

Першою електронною обчислювальною машиною (ЕОМ), зібраною на радіолампах, стала ENIAC, створена в США у 1946 р.

У 1951 р. у Києві групою українських учених під керівництвом академіка С.О. Лебедєва було створено першу на території нинішніх країн СНД електронну обчислювальну машину МЗСМ (рос. «Малая электронно-счетная машина»).

Для подання чисел і символів у пам'яті комп'ютера використовуються послідовності всього з двох цифр — 0 та 1. Це дає змогу використовувати найпростіші електронні запам'ятовуючі пристрої, які можуть перебувати тільки в одному з двох станів (наприклад, у стані «ввімкнено» і в стані «вимкнено»). Послідовності з нулів та одиниць називаються *двійковими кодами.* За їх допомогою можна представити не лише дані, потрібні для розв'язання задачі комп'ютером, а й послідовність команд, які треба виконати, щоб розв'язати задачу. Двійкові коди є основою *машинної мови,* тобто мови, якою подаються команди комп'ютеру.

Комп'ютери першого покоління «розуміли» тільки машинну мову. (Коли говорять про покоління обчислювальної техніки, то виділяють дві головні ознаки: елементну базу комп'ютера та рівень спілкування між комп'ютером і людиною. Чим складніша будова комп'ютера, тим складніші засоби, потрібні для того, щоб ним керувати.) Сьогодні для швидкого й правильного створення таких засобів використовують спеціальні мови, близькі до природної.

Для спілкування людини із сучасними комп'ютерами вчені створили та активно розвивають «дружні» та інтуїтивно зрозумілі засоби, в тому числі з використанням таблиць і рисунків. Тим самим розширюються сфери використання комп'ютерів і збільшується кількість задач, які вони розв'язують.

У 1976 р. у США С. Возняк і С. Джобс сконструювали перший персональний комп'ютер Apple. А в 1981 р. фірма IBM випустила свою відому модель IBM PC (Personal Computer). З її появою розпочалася нова епоха, яка характеризується багатоплановою інформаційною діяльністю людини.

Зараз простіше сказати, де не використовуються персональні комп'ютери, ніж назвати всі галузі їхнього застосування. Наприклад, у виробництві комп'ютери використовуються на всіх етапах — від конструювання окремих деталей виробу та його проектування в цілому до складання та продажу. Широко використовуються системи автоматизованого проектування (САПР), які дають можливість робити креслення, одержувати загальний вигляд об'єктів, що проектуються, швидко виконувати складні розрахунки. Гнучкі виробничі системи (ГВС) дозволяють швидко реагувати на зміни ринкової ситуації, практично миттєво розширювати або припиняти виробництво товарів, заміняти їх іншими.

За допомогою комп'ютерів може швидко оброблятися інформація від різних датчиків, у тому числі й від систем автоматизованої охорони, від датчиків температури для регулювання витрат енергії на опалення, від складної системи томографа, що дозволяє «побачити» внутрішню будову органів людини і правильно поставити діагноз тощо.

Нині на робочому столі майже кожного фахівця є комп'ютер. Він забезпечує спілкування між людьми на будь-які

відстані, дає змогу скористатися фондами великих бібліотек у різних країнах, не виходячи з дому, використовувати потужні інформаційні системи, комп'ютерні енциклопедії, вивчати нові науки і набувати різні знання за допомогою навчальних програм і тренажерів. Архітектору комп'ютер допомагає створювати споруди, видавцю — компонувати текст та ілюстрації, художнику — створювати нові картини, а композитору — музику. У багатьох випадках дорогі експерименти в науці та техніці можна повністю замінити розрахунками на комп'ютері.

Засоби та методи подання інформації, технології розв'язування виробничих, дослідницьких, побутових та інших задач із використанням комп'ютерів стали найважливішими прикладними аспектами інформатики.

ЗАПИТАННЯ І ЗАВДАННЯ

1. Хто винайшов перший арифмометр? Назвіть інші досягнення цього вченого.
2. Хто й коли вперше сформулював ідею програмно керованої обчислювальної машини?
3. Схарактеризуйте поняття машинної мови.
4. Які персональні комп'ютери випустила фірма IBM в 1981 році?
5. Розкажіть про застосування комп'ютерів.

1.2. Будова та принципи роботи комп'ютера

Комп'ютер являє собою сукупність засобів, які призначені для автоматизації процесів обробки та зберігання інформації (перша сторінка форзацу). Інша назва комп'ютера — електронна обчислювальна машина (ЕОМ).

Комп'ютер є універсальним інструментом для обробки інформації. За його допомогою можна виконувати обчислення, опрацьовувати тексти, розпізнавати і формувати зображення, перетворювати та аналізувати сигнали, керувати різноманітними об'єктами і технологічними процесами, розв'язувати логічні задачі тощо.

Технічні засоби, які входять до складу комп'ютера, називаються його *апаратним забезпеченням (hardware).* Роботою комп'ютера потрібно керувати, тобто йому треба вказати, які операції він повинен виконувати.

Сукупність чітких, однозначних, зрозумілих комп'ютеру *команд* (інструкцій), що визначають послідовність операцій, які потрібно виконати для отримання розв'язку певної задачі, називається *програмою.* Прикладом програми є вказівка: «скласти два числа 6 і 9».

За допомогою комп'ютера можна розв'язувати велику кількість різноманітних задач. Функціонування комп'ютера забезпечується цілим комплексом програм, який називається *програмним забезпеченням (software).*

Таким чином, роботу комп'ютера можна розглядати як опрацювання даних за програмою. Комп'ютер приймає вхідні дані, обробляє їх та отримує результати — вихідні дані. Наприклад, для програми, яка призначена для додавання двох чисел, вхідними даними є числа, які додаються, а вихідними даними — їх сума.

Американський вчений Дж. фон Нейман у 1946 р. сформулював загальні принципи організації та функціонування комп'ютерів, іншими словами — описав *архітектуру* комп'ютерів, яку прийнято називати фон-нейманівською.

Архітектуру сучасного комп'ютера, в якій використовуються ідеї фон Неймана, ілюструє функціональна схема, наведена на рис. 3.

Згідно з цією схемою до складу апаратної частини комп'ютера повинні входити такі основні пристрої (функціональні блоки): *пристрій управління, арифметико-логічний пристрій* (АЛП), *оперативна пам'ять* і *пристрої введення та виведення.*

Щоб розв'язати певну задачу, в пам'ять комп'ютера потрібно ввести програму, яка визначає процес розв'язування, а також потрібні дані. Команда і дані, які вона має обробити, вводяться до АЛП, після чого команда виконується і її результат записується в оперативну пам'ять. Пристрій управління визначає наступну команду в пам'яті комп'ютера, і вона в свою чергу надходить до АЛП. Процес виконання програми послідовно триває доти, доки не трапиться команда завершення роботи програми. Якщо серед команд

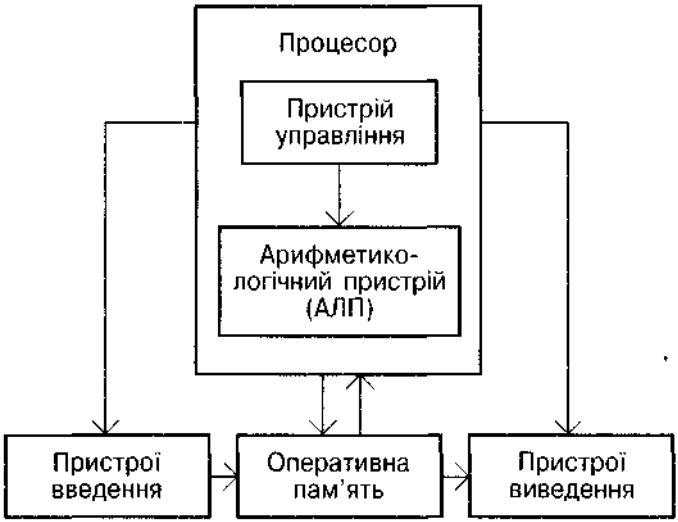

Рис. 3. Функціональна схема комп'ютера

програми є команда виведення, то опитується відповідний пристрій, наприклад пристрій друку, і якщо він увімкнений і готовий до роботи (папір заправлений), то здійснюється виведення (друкування) результатів.

Здебільшого АЛП та пристрій управління не розглядаються окремо, а об'єднуються одним словом *«процесор»*. Процесор — це пристрій для обробки даних. Саме процесор виконує всі ті дії, які необхідні для розв'язання конкретної задачі, що ставить перед комп'ютером користувач.

Процесор виконує команди одну за одною, тобто працює послідовно. Існують паралельні обчислювальні системи, які мають кілька процесорів, що працюють одночасно. Найпотужніші з них називаються суперкомп'ютерами. Вони розв'язують складні задачі керування космічними апаратами, моделюють зміну клімату Землі, роблять розрахунки складних форм літальних апаратів тощо.

Одним із принципів роботи комп'ютерів, запропонованих Дж. фон Нейманом, є принцип єдиної лінійної пам'яті. «Єдиної» означає, що і програма, і дані зберігаються в одній пам'яті. Одна й та сама комірка пам'яті для однієї задачі може зберігати команду, а для іншої — дані. «Лінійної»

означає, що всі ділянки пам'яті пронумеровано від 0 до деякого числа *N*. Номер комірки називається її адресою.

Різні комп'ютери мають різні розміри комірок пам'яті. У середині 60-х років, коли фірма IBM випустила серію обчислювальних машин третього покоління, склався стандарт одиниці вимірювання пам'яті — *байт*. Байт являє собою послідовність восьми двійкових розрядів. Один двійковий розряд називається *бітом*.

Кожний біт може приймати одне з двох значень — 0 чи 1. Двома бітами можна закодувати чотири значення: 00, 01, 10, 11. Згідно з позиційною системою числення значення байта, який складається з восьми бітів, залежить від позицій, які займають нулі та одиниці. Кількість різних комбінацій бітів у байті дорівнює $2^8 = 256$, тобто код довжиною один байт може набувати 256 різних значень.

У комп'ютері за допомогою двійкових кодів представляються як числа, так і символи (літери, цифри тощо). Для кодування символів здебільшого достатньо коду довжиною один байт. Для кодування цілого числа, як правило, використовуються два або чотири байти, а для дійсного — чотири або вісім, інколи шість байтів, залежно від конкретної задачі та можливостей комп'ютера.

Машинні команди також бувають різної довжини. Здебільшого в командах дані для обробки містяться не у вигляді конкретних чисел, а у вигляді адрес комірок пам'яті, де вони зберігаються. Програмне забезпечення сучасних комп'ютерів дозволяє користувачеві не думати про те, які команди машинної мови використовуються і де саме в пам'яті знаходяться програми і дані.

При роботі з комп'ютером використовуються зовнішні пристрої введення-виведення, які дають змогу відображати інформацію у зрозумілій користувачеві формі (текстовій, графічній, звуковій) і вводити до комп'ютера дані та команди.

ЗАПИТАННЯ І ЗАВДАННЯ

1. Що розуміється під архітектурою комп'ютера?
2. Назвіть основні функціональні блоки комп'ютера і вкажіть їх призначення.

3. Опишіть за функціональною схемою комп'ютера основні етапи проходження програми і даних від введення до виведення.
4. Що таке процесор, де він зображений на функціональній схемі?
5. Опишіть принцип єдиної лінійної пам'яті роботи комп'ютера.
6. Яких значень може набувати двійковий код довжиною один біт?
7. Що таке байт?
8. Що являє собою комп'ютерна програма?

1.3. Компоненти комп'ютера

Функціональні блоки комп'ютера можуть бути реалізовані у вигляді різних фізичних пристроїв.

Процесор сучасного персонального комп'ютера виконаний у вигляді мікросхеми — єдиного мікроелектронного пристрою, створеного на кристалі напівпровідника і вміщеного в мініатюрний корпус (тому він називається мікропроцесором).

Можливості комп'ютера значною мірою визначаються *розрядністю процесора,* тобто кількістю бітів, яку він може обробити одночасно, та його внутрішньою організацією. Спочатку персональні комп'ютери мали 8-бітові процесори. Перші комп'ютери IBM PC були вже 16-бітовими. Сучасні персональні комп'ютери є 32-бітовими і навіть 64-бітовими.

Швидкість роботи процесора характеризується тактовою частотою, що задається спеціальним генератором і вимірюється в мегагерцах (МГц); 1 МГц — це мільйон тактів за секунду. У перших процесорах такт визначав проміжок часу, за який могла бути виконана елементарна дія типу додавання двох чисел чи пересилання числа із процесора в оперативну пам'ять. Процесори сучасних персональних комп'ютерів можуть виконувати за один такт кілька операцій, а тактова частота сягає сотень та тисяч мегагерц.

Пам'ять комп'ютера —це пристрої, в яких зберігаються дані та програми. Пам'ять комп'ютера поділяється на *внутрішню* (основну) та *зовнішню.* До внутрішньої пам'яті відносяться: оперативна пам'ять, регістри процесора, постійна пам'ять і кеш-пам'ять.

Оперативна пам'ять — це пристрої де розміщені дані, які опрацьовує процесор у даний проміжок часу. При цьо-

му виконується умова, що в будь-який момент є можливість працювати з будь-якою коміркою оперативної пам'яті. Англійська абревіатура оперативної пам'яті — RAM (Random Access Memory — пам'ять із довільним доступом). В оперативній пам'яті зберігається тимчасова інформація, яка змінюється під час виконання процесором різних операцій, таких як запис, зчитування, зберігання тощо. При відключенні живлення комп'ютера вся інформація, що перебувала в оперативній пам'яті, зникає, якщо вона не була збережена на інших носіях інформації.

Оперативна пам'ять реалізована на основі спеціальних мікросхем і її обсяг легко розширюється додаванням потрібної кількості мікросхем. Обсяг оперативної пам'яті вимірюється в кілобайтах (1 Кб = 1024 байта або 2^{10} байтів) і мегабайтах (1 Мб = 2^{20} байтів).

Регістри — це надшвидка пам'ять процесора. Вони зберігають адресу команди, саму команду, дані для її виконання і результат.

Для збільшення продуктивності комп'ютера, тимчасового зберігання вмісту комірок оперативної пам'яті використовується *кеш-пам'ять* (від англійського cashe — склад, тайник). Кеш-пам'ять є проміжним запам'ятовуючим пристроєм і використовується для прискорення обміну між процесором і оперативною пам'яттю. У сучасних комп'ютерах використовуються кілька рівнів кеш-пам'яті. Кеш-пам'ять може розміщуватись як на кристалі мікропроцесора, так і на *системній платі.* Системна (або материнська) плата — це плата, на якій розміщено процесор, оперативну та постійну пам'ять, а також системну шину, через яку здійснюється зв'язок з усіма іншими пристроями комп'ютера.

Постійна пам'ять — це пристрої для довготривалого зберігання програм та даних. Використовується вона тільки для читання інформації. Як правило, ця інформація записується при виготовленні комп'ютера і слугує для початкового завантаження операційної системи (див. далі), перевірки дієздатності комп'ютера. Для назви цієї пам'яті використовується англійська абревіатура ROM (Read Only Memory — па'мять,тільки для читання)

Комп'ютер має зовнішню пам'ять,щ використовується для довготривалого збереження програм та данихВона

реалізується за допомогою спеціальних пристроїв *(накопичувачів),* які залежно від способів запису та зчитування діляться на магнітні, оптичні та магнітно-оптичні.

Основними характеристиками зовнішньої пам'яті є її об'єм, швидкість обміну інформацією, спосіб та час доступу до даних. Пристрої, в яких використовуються магнітні стрічки, належать до пристроїв з послідовним доступом. Пристрої, які використовують магнітні та оптичні диски, належать до пристроїв з прямим доступом.

У персональних комп'ютерах застосовуються магнітні диски двох типів — незмінні тверді (жорсткі) та змінні гнучкі. Вони дають змогу здійснювати як введення, так і виведення даних.

Жорсткий магнітний диск — це алюмінієва пластина, покрита шаром магнітної речовини. Накопичувач на жорстких магнітних дисках (іноді його називають вінчестером) має невеликі розміри і монтується в корпусі системного блоку комп'ютера. На жорсткому диску записується найважливіша для роботи комп'ютера інформація — програми управління комп'ютером та обслуговування різних пристроїв, «лікарі» для лікування від комп'ютерних вірусів (програм, що псують інформацію та заважають роботі ЕОМ), а також потрібні користувачеві дані та так звані інструментальні засоби (редактори текстів, засоби створення програм, баз даних тощо). Для постійного зберігання великої кількості різних програм та інформації, які завжди повинні бути під рукою, потрібний великий об'єм пам'яті. Сучасні жорсткі диски мають об'єм від кількох мегабайтів до десятків гігабайтів (1 гігабайт = 2' мегабайтів = 2^{30} байтів).

До зовнішньої пам'яті належать також накопичувачі на *гнучких* дисках *(дискетах).* Найпоширенішими є дискети діаметром 3,5" (знак «"» означає дюйм, 1" = 2,54 см). Для читання та запису інформації на дискетах у комп'ютері є спеціальні пристрої — дисководи для гнучких дисків.

Комп'ютер може мати дисководи для зчитування інформації з *оптичних* (лазерних) дисків. Нині широке застосування мають оптичні диски діаметром 5,25", їх називають компакт-дисками. Є диски, що дають змогу здійснювати тільки читання даних. Інформація записується на диск під час його виготовлення і потім не може бути змінена. Такі компакт-дис-

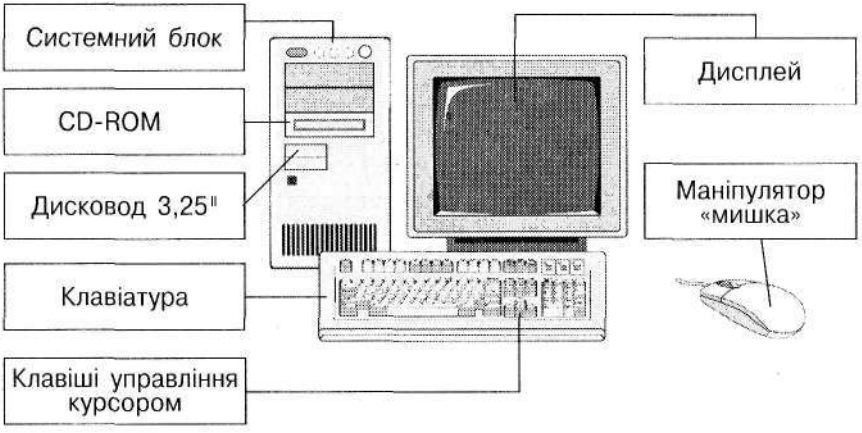

Рис. 4. Персональний комп'ютер

ки називаються CD ROM (сі-ді-ром). На них можна записувати тексти, програми, зображення, звук, фрагменти відеофільмів тощо. Обсяг пам'яті компакт-дисків — сотні мегабайтів. Існують також оптичні диски, інформація на яких може перезаписуватись. Вони називаються CD-R (сі-ді-ер). Названі компоненти комп'ютера (рис. 4) розміщені в системному блоці (див. 2 сторінку форзацу), на передній панелі якого розташований вимикач, входи дисководів, індикатор та ін. На задній панелі є гнізда, що дають можливість під'єднувати різноманітні прилади — дисплей, клавіатуру, принтер, мишку, сканер та ін.

Основним пристроєм введення інформації в комп'ютер служить клавіатура, яка містить набір клавіш із написами (с. 253). Одна група клавіш призначена для введення букв, цифр та інших знаків. Друга група — так звані функціональні клавіші та клавіші управління — призначена для управління процесами введення-виведення, обробки та зберігання даних.

Дисплей — це пристрій відображення даних. Більшість дисплеїв можуть відображати на екрані дані в різних формах — числовій, текстовій, графічній. Форма відображення даних залежить від можливостей комп'ютера, наявності відповідної апаратної частини — графічної плати (або

графічної карти). Як дисплей у принципі можна використовувати і побутовий телевізор.

Чіткість зображення та його деталізація залежать від роздільної здатності екрана дисплея — кількості точок, які можуть бути відображені на ньому. Кожна точка являє собою невеликий прямокутник і називається *пікселем* (picture element — елемент зображення). Дисплей має спеціальну пам'ять *(відеопам'ять),* що дозволяє йому швидко видавати зображення на екран, а не будувати його кожного разу по точках. Команди та значення даних, що вводяться за допомогою клавіатури, також відображаються на екрані дисплея.

Для виділення певної позиції екрана використовується вказівник, що називається курсором. *Курсор* — це позначка (фігура), яка може переміщатися по екрану. Переміщення курсора по екрану здійснюється за допомогою натискування клавіш управління курсором або пересування на столі маніпулятора «мишка». При натисканні алфавітно-цифрової клавіші клавіатури зображений на ній символ з'являється в тому місці екрана, де знаходиться курсор. Символи, що відображаються на екрані дисплея у звичайному текстовому режимі, з'являються у фіксованих прямокутниках (наприклад, розміром 8x8 чи 8x16 пікселів).

Маніпулятор «мишка» являє собою пристрій, виконаний у вигляді невеликої коробочки з кулькою знизу. При переміщенні мишки по столу курсор мишки відповідним чином переміщується на екрані. Мишка має дві або три кнопки, які використовуються для введення команд.

Для керування пристроями комп'ютера використовуються електронні схеми, які називаються контролерами. Як правило, для кожного пристрою є свій контролер. Всі контролери підтримують зв'язок з процесором та оперативною пам'яттю комп'ютера через електричний ланцюг, який задіяний для паралельного сполучення пристроїв. Електричний ланцюг називають *магістраллю* або *шиною.*

ЗАПИТАННЯ І ЗАВДАННЯ

1. Назвіть види пам'яті комп'ютера.
2. Як використовується оперативна пам'ять комп'ютера?

3. Скільки байтів міститься в кілобайті і мегабайті?
4. Що таке жорсткий диск і для чого він призначений?
5. Які види дисків використовуються у персональних комп'ютерах?
6. Що таке піксел?
7. Чим відрізняється управління курсором за допомогою клавіш клавіатури і «мишки»?
8. Назвіть пристрої введення-виведення інформації, що використовуються в персональних комп'ютерах.

. Носії інформації. Пристрої запису і зчитування

Під час роботи комп'ютера всі пристрої, хоча б тимчасово, бувають носіями інформації. Однак, як правило, цей термін використовується у разі тривалого зберігання даних.

Інформація може зберігатися на різних матеріальних носіях, які забезпечують її запис, зберігання та відтворення. Розрізняють магнітні, оптичні, перфоровані, паперові та екранні носії. До магнітних носіїв належать магнітні стрічки і диски, до оптичних — компакт-диски, до перфорованих — перфокарти і перфострічки, до паперових — папір друкарських пристроїв, до екранних — екрани дисплеїв.

Основними магнітними носіями інформації є *магнітні диски* — жорсткі та гнучкі (дискети). Перші жорсткі магнітні диски мали діаметр близько півметра і збиралися в пакети. Сучасні магнітні диски персональних комп'ютерів мають значно менші розміри. Незважаючи на різницю в розмірах і обсягах даних, що запам'ятовуються, диски мають однаковий принцип запису інформації. Дані на магнітних дисках розміщуються на концентричних доріжках, кількість яких залежить від якості дисків. Доріжки, у свою чергу, діляться на сектори (рис. 5). Магнітна головка —

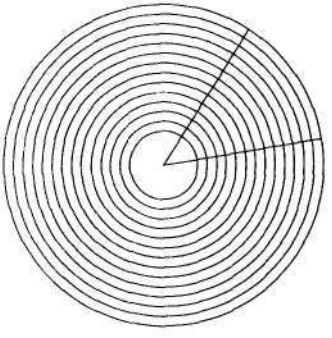

Рис.5. Доріжки диска

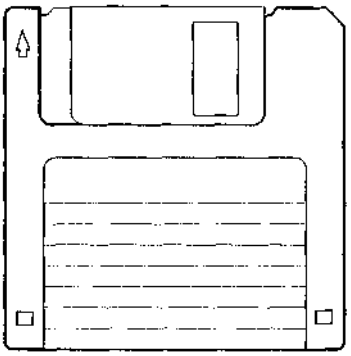

Рис.6. Дискета 3,5"

пристрій запису-читання, що рухається вздовж радіуса диска, який обертається.

Щільність запису на доріжках буває різною і відповідно до цього різним може бути об'єм пам'яті. Так, місткість дискет розміром 3,5" (рис. 6) може дорівнювати 720 Кб, 1,44 Мб, 2 Мб, 2,88 Мб.

На *оптичних дисках* (компакт-дисках) дані зберігаються на спіральних доріжках, що складаються з впадин і рівних ділянок. При зчитуванні інформації лазерним променем впадини інтерпретуються як нулі, а рівні ділянки, які добре відбивають світло, — як одиниці.

Паперові носії використовуються для відображення та зберігання текстової та графічної інформації. Запис на них здійснюється за допомогою автоматичних друкуючих пристроїв *(принтерів),* а також графопобудовників (*плотерів).* За принципом роботи принтери поділяються на пристрої контактного і безконтактного друкування.

Контактне друкування здійснюється ударом молоточка, на якому закріплена літера, або набору голок по фарбувальній стрічці, яка доторкується до паперу. Голкові принтери називають матричними, вони можуть мати різну кількість голок, наприклад 9 або 24. 9-голкові принтери можуть друкувати строку з кілька проходів головки принтера, при цьому добре видно, що буква складається з окремих крапок. Кожна голка діє як незалежний молоточок, що управляється програмою друкування. Принтери з 24 голками забезпечують значно краще за якістю друкування тексту різноманітними шрифтами.

Принтери безконтактного друкування — це струменеві та електрографічні (лазерні). Струменевий принтер будує зображення з краплинок кольорового чорнила, що розбризкуються кількома друкувальними головками з чорнилом різних кольорів. Послідовність роботи кожної головки регулюється мікропроцесором принтера. При щільності близь-

ко 100 крапок на сантиметр паперу отримуються високоякісні чорно-білі або кольорові зображення.

Високоякісне друкування забезпечують електрографічні (лазерні) принтери. Під дією лазерного променя змінюється провідність напівпровідникового покриття попередньо електризованого барабана, і із засвічених ділянок заряд стікає. До наелектризованих ділянок прилипає дрібний порошок. При цьому формується дзеркальне зображення всієї сторінки. Папір, що має протилежний барабанові заряд, проходить через пристрій і притягує порошок, який закріплюється під дією тепла й тиску. Перед друкуванням наступної сторінки барабан нейтралізується і звільняється від порошку.

Оскільки матричні, струменеві та лазерні принтери формують зображення з окремих точок, вони можуть здійснювати виведення не лише текстової, а й графічної інформації.

Для введення в комп'ютер інформації з паперових носіїв (креслень, малюнків, фотографій, текстів тощо) застосовується *сканер*. Цей пристрій сприймає зображення, перетворює його по точках у послідовність кодів, що характеризують яскравість та колір точок, і передає їх до пам'яті комп'ютера. Введені в комп'ютер зображення можуть оброблятися за допомогою спеціальних програм — графічних редакторів, зберігатися на дисках, виводитись на друк тощо.

Для виведення даних на екран потрібно висвітити відповідні пікселі в певних місцях екрана, наприклад у позиції курсора. З цією метою кожний піксел позначається в комп'ютері двома координатами і може висвічуватися фіксованою кількістю кольорів.

Координати на екрані тільки додатні, при цьому вісь X направлена зліва направо, а вісь Y — зверху вниз, як показано на рисунку (рис. 7). Таке розташування осі Y відрізняється від традиційного в математиці і по-

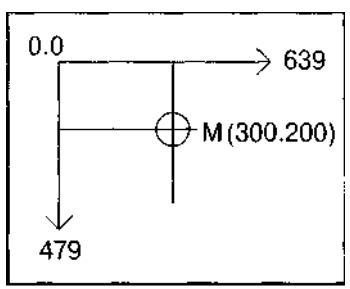

Рис. 7. Система координат екрана дисплея

требує перерахунку координат при побудові графіків функцій. На рис. 7 показано розміщення координат екрана дисплеїв розміром 640x480 пікселів. Для кожного пікселя кольорового дисплея зображення будується накладанням трьох кольорів — червоного, зеленого і синього. При 8-бітному кодуванні кольору пікселя можна отримати 256 різних кольорових відтінків, при 24-бітному — понад 16 мільйонів. Сучасні комп'ютери мають дисплеї розміром 800x600 пікселів, 1024x768 пікселів тощо. Найбільшу кількість пікселів, яка може розміститися на екрані дисплея, називають роздільною здатністю екрану. А зображення, яке будується по точках, називають растровим.

ЗАПИТАННЯ І ЗАВДАННЯ

1. Що таке носій інформації?
2. Як подаються дані на перфоносіях?
3. Як переміщується головка запису-читання по магнітному диску?
4. Яким чином записується інформація на лазерних дисках?
5. Який принцип дії матричного принтера?
6. Як струменевий принтер формує кольорове зображення?
7. Чому електрографічні принтери називають лазерними?

1.5 Файли

Комп'ютер допомагає людині систематизувати відомості і швидко знаходити потрібні дані. Для зручності пошуку інформації дані об'єднуються в групи за різними ознаками. Таке об'єднання може бути багатоетапним, як, наприклад, об'єднання учнів у класи, класів — у школу, а школа характеризується адресою населеного пункту і своїм номером. Аналогічні принципи об'єднання інформації застосовуються і при зберіганні інформації в зовнішній пам'яті ЕОМ.

Файл — це іменована сукупність інформації, яка розміщена в зовнішній пам'яті комп'ютера. У файлах на диску зберігаються і програми, і дані різних користувачів, і ігри (які часто містять кілька файлів). Донедавна при створенні імен файлів дозволялося використовувати лише латинські літе-

ри, однак у багатьох сучасних комп'ютерних системах це обмеження знімається.

Принципи іменування файлів можуть бути різними. Найчастіше ім'я файла складається з двох частин, які розділяються при написанні крапкою. Першу частину імені задає автор файла. При цьому бажано, щоб в імені відображався зміст цього файла. Наприклад, якщо це ігрова програма, то ім'я файла — назва гри. Друга частина, що називається *розширенням імені файла,* показує, до якої категорії належить файл. Розширення найчастіше складається з трьох символів. Існують *стандартні розширення,* за якими можна визначити, що містить файл — готову до запуску програму, опис або допоміжні дані програми і т.д. Прикладами стандартних розширень є:

exe, com — програма на машинній мові, готова до виконання;

bat — текстовий файл, що містить інформацію для запуску exe-, corn-файлів;

txt, doc — файли, що містять текстову інформацію;

Ьmp, pcx — файли малюнків;

arj, zip — архіви, що містять файли, спеціальним способом зменшені в обсязі;

hlp — файли довідникової системи;

drv, sys — службові програми комп'ютера;

bas, pas — файли програм на мовах програмування Бейсік і Паскаль.

Користувач сам може придумати розширення імені свого файла. Розширення є необов'язковим, однак бажаним елементом імені файла, тому що допомагає розрізняти файли за призначенням та типами.

Файли на диску об'єднуються в *каталоги.* Кожний каталог може містити в собі як файли, так і інші каталоги. Кожний диск має один головний каталог, який називається *кореневим.* Усі інші каталоги є його підкаталогами різних рівнів. Їх імена мають таку саму структуру, як і імена файлів, однак, як правило, вони не мають розширення і складаються з одного слова. Система каталогів на диску утворює складну структуру, що називається *деревовидною.*

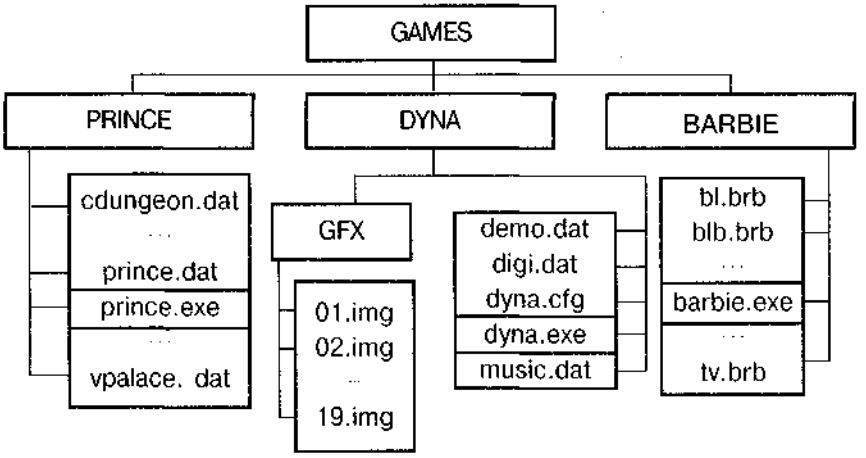

Рис. 8. Деревовидна структура каталогу

Файли, які належать до різних програмних систем, різним користувачам тощо розміщують в окремих каталогах, що значно спрощує їх пошук та доступ до них. Так, ігрові програми зручно зберігати в каталозі GAMES (імена каталогів часто позначають великими літерами). Для кожної гри також зручно мати свій підкаталог. Приклад такої деревовидної структури наведено на рис 8.

Можна бачити, що каталог GAMES, призначений для зберігання ігрових програм, містить три підкаталоги: PRINCE, DYNA і BARBIE. Підкаталоги PRINCE і BARBIE інших підкаталогів не мають, а підкаталог DYNA включає підкаталог GFX. У кожній грі є запускаючий її файл з розширенням *,exe.* Каталоги також містять інші допоміжні файли. Імена файлів позначені малими буквами і знаходяться на різних рівнях дерева. Так, файли ігор PRINCE та BARBIE знаходяться на другому рівні (на першому — GAMES), а файли підкаталогу GFX — на третьому.

Каталог, з файлами якого працюють у даний момент, називається *активним* або поточним. Щоб звернутися до файла, що знаходиться в активному каталозі, достатньо вказати його ім'я. Послідовно переходячи з каталогу в каталог, отримують доступ до файлів, що знаходяться на різних рівнях.

До файла, розташованого в будь-якому місці деревовидної структури, можна також звернутися, вказавши шлях до нього. Так, шлях до файла, що запускає гру PRINCE, має вигляд

C:\GAMES\PRINCE\prince.exe

Він складається з імені диска (літера з двокрапкою), послідовного списку каталогів, розділених зворотною похилою рискою (англ. back slash — "бек слеш"), та імені файла. Зворотна похила риска перед іменем першого каталогу вказує на те, що шлях починається від кореневого каталогу диска (інакше пошук здійснюється з поточного каталогу). Шлях до файла іноді називають маршрутом. Коли вказується шлях до файла на поточному диску, ім'я диска вказувати не обов'язково. Гнучкі магнітні диски мають імена A: та B:. Імена C:, D: і т.д. надають жорстким і оптичним дискам, а також розділам, на які поділяються жорсткі диски. Ці розділи називаються *логічними дисками*.

ЗАПИТАННЯ І ЗАВДАННЯ

1. Що таке файл? Як файли розміщуються на дисках?
2. Що таке ім'я файла?
3. Що таке каталог?
4. Яким чином можна знайти потрібний файл на диску?
5. Поясніть, чому на дисках комп'ютера має бути єдиний порядок зберігання і подання даних.
6. Яку структуру має ім'я файла?
7. Як за іменем файла можна дізнатися про його зміст?
8. Як виглядають імена файлів, що містять готові до виконання програми?
9. Чи може ім'я файла бути довільним?
10. Що таке шлях до файла?
11. На рис. 9 подано структуру каталогу на гнучкому магнітному диску. Припускаючи, що комп'ютер має один дисковод для гнучких магнітних дисків, сформуйте шлях до файлів: 2-10-4.fsh, foxhelp.arj, foxl.arj.

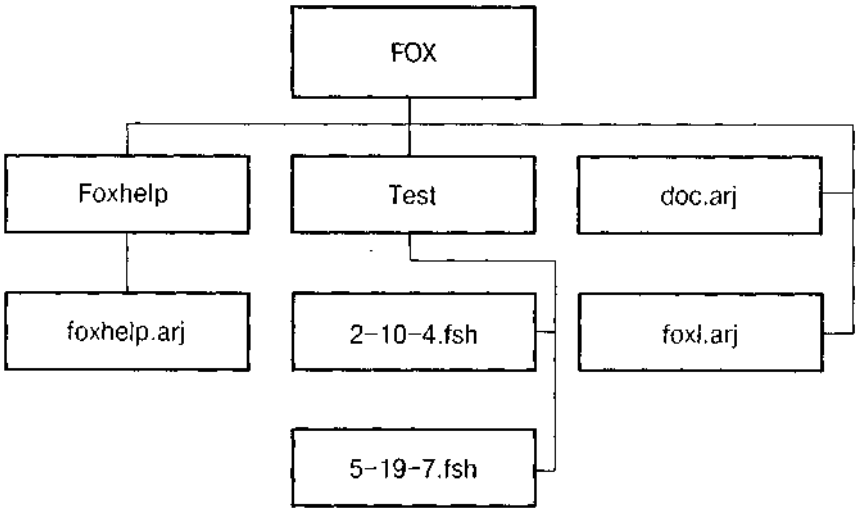

Рис. 9. Структура каталогу FOX

1.6. Операційна система MS-DOS

Комп'ютер — це обчислювальна система, яка складається з апаратної частини і програмного забезпечення. У програмному забезпеченні виділяють прикладне і системне забезпечення.

Прикладне програмне забезпечення — це сукупність програм, які використовуються безпосередньо для розв'язування конкретних задач — роботи з текстами, виконання наукових та інженерних розрахунків, управління виробничими процесами, управління інформаційно-довідковими системами тощо.

Системне програмне забезпечення включає операційні системи та інструментальні засоби (системи програмування). *Системи програмування* є засобами для розробки як прикладного, так і системного програмного забезпечення комп'ютерів.

Операційна система являє собою сукупність програм, які забезпечують управління роботою апаратної і програмної складових комп'ютера, координують їх взаємодію. Таким чином, операційна система організує роботу комп'ютера, контролює виконання всіх програм і використання всіх пристроїв комп'ютера, тобто ресурсів системи. Управління операційною системою здійснюється за допомогою спеціальної системи команд, які задаються користувачем.

Дуже поширеною операційною системою є розроблена фірмою Microsoft дискова операційна система MS-DOS для IBM-сумісних персональних комп'ютерів. Структура MS-DOS подана на рис. 10.

Основними компонентами MS-DOS є: базова система введення-виведення — BIOS (Basis Input/Output System); системний завантажувач (System Bootstrap) — розміщується в блоці початкового завантаження (Boot Record); модуль розширення BIOS — розміщується в файлі IO.SYS; модуль обробки переривань — файл MSDOS.SYS; командний процесор — файл COMMAND.COM; утиліти, які реалізують виконання зовнішніх команд MS-DOS — файли з розширенням *.com,* наприклад FORMAT.COM; драйвери пристроїв — розміщуються у вигляді файлів на дисках; інформація про бажані параметри настроювання MS-DOS — при необхідності задається у файлі конфігурації CONFIG.SYS; командний файл, який при необхідності використовується для настроювання параметрів і конфігурації MS-DOS, має ім'я AUTOEXEC.BAT.

При вмиканні комп'ютера операційна система завантажується до його пам'яті з жорсткого або гнучкого диска, після чого комп'ютер готовий до роботи. При потребі можна перезавантажити операційну систему, натиснувши комбінацію **Ctrl** + **AIt** + **Del** на клавіатурі або кнопку **RESET** на системному блоці. (Тут і далі одночасне натискування кількох клавіш позначається знаком « + » між їхніми позначеннями).

Процес завантаження проходить так: BIOS — базова система введення-виведення (комплекс програм, записаних у постійній пам'яті комп'ютера (ROM) при його виготовленні) — перевіряє працездатність пам'яті і підключеного до комп'ютера обладнання. Потім вона шукає програму почат-

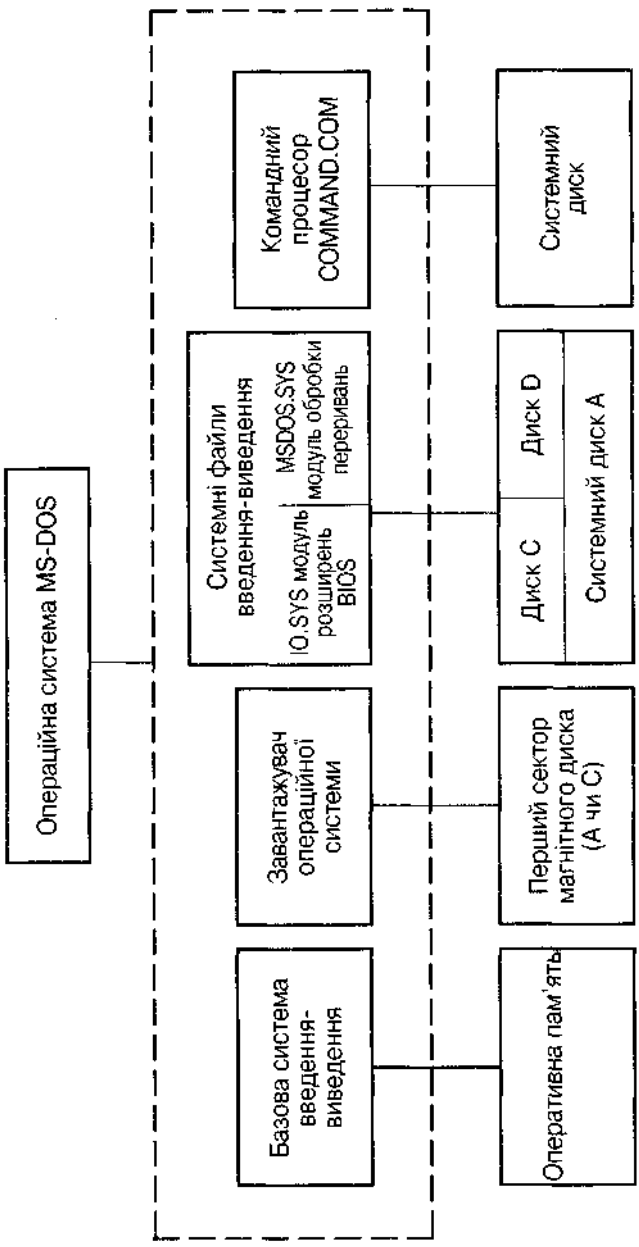

Рис. 10. Компоненти MS–DOS та їх розміщення в пам'яті комп'ютера до початку роботи операційної системи

кового завантаження (завантажувач), яка розміщується завжди в першому секторі нульової доріжки диска або дискети (завантажувальному, або BOOT-секторі), і зчитує її в оперативну пам'ять. Потім BIOS запускає цю програму і вона починає перезавантажувати в пам'ять файли операційної системи.

У системному файлі введення-виведення IO.SYS містяться додаткові програмні засоби для забезпечення введення-виведення, які після завантаження постійно знаходяться в оперативній пам'яті. IO.SYS дає можливість враховувати особливості конкретної операційної системи та організовувати роботу з деякими додатковими зовнішніми пристроями. Коли IO.SYS починає працювати, він шукає файл CONFIG.SYS, який містить допоміжну інформацію про те, які додаткові пристрої мають бути підключені при завантаженні операційної системи. Для забезпечення роботи кожного пристрою існує своя програма, яка називається *драйвером.* Драйвери стандартних пристроїв містяться у файлі IO.SYS. Команди для запуску додаткових драйверів, що зберігаються в окремих файлах, а також команди встановлення параметрів операційної системи містяться у файлі CONFIG.SYS. Якщо цей файл не знайдено, то параметри системи встановлюються автоматично (за умовчанням).

Комплекс програм, який міститься у файлі MSDOS.SYS, дозволяє управляти оперативною і дисковою пам'яттю та організацією обчислювального процесу.

Одним із головних завдань операційної системи персонального комп'ютера є розподіл дискового простору. При цьому реалізуються такі принципи:

1) усі файли даних зберігаються в секторах стандартного розміру;

2) сектори виділяються в разі потреби;

3) забезпечується логічний зв'язок між секторами, який дозволяє користувачеві не турбуватися про те, на скільки частин система поділила його файл і де ці частини розміщені;

4) кожний диск містить кореневий каталог, в якому зберігається список розміщених на диску файлів та підкаталогів.

Підпрограми, що містяться у файлі MSDOS.SYS, також управляють проходженням програм, які запускаються ко-

ристувачем комп'ютера — завантажують ці програми у пам'ять, контролюють їх виконання та завершення.

Програма COMMAND.COM, яка називається командним процесором, зчитує команди, які вводяться з клавіатури, і виконує їх. Командний процесор поділяється на три частини. Його перша частина після завантаження постійно перебуває в оперативній пам'яті. Друга частина зберігається в пам'яті тимчасово і використовується для пошуку та виконання командного файла AUTOEXEC.BAT, у якому вказані програми і команди, що викопуються при кожному запуску комп'ютера, наприклад команда підключення драйвера клавіатури для роботи з українськими літерами. Після виконання файла AUTOEXEC.BAT процес завантаження завершується. Третя частина командного процесора містить таблицю основних команд операційної системи, а також інтерпретатор команд, який перетворює їх на виклики відповідних програмних засобів. Ця частина може бути вилучена з пам'яті при завантаженні інших програм, а при потребі вона повторно завантажується з диска.

Операційна система здійснює діалог з користувачем, видаючи йому запрошення для введення команд, аналізуючи команди і коментуючи свої дії. Запрошення, що з'являється на екрані дисплея після завантаження системи, має вигляд:

A:\> або C:\>

Весь процес діалогу користувача з комп'ютером — введення команд, повідомлення операційної системи про результати їх виконання — відображається на екрані дисплея.

Для виконання команди операційної системи слід набрати за допомогою клавіатури ім'я команди та необхідні параметри, натиснути клавішу введення (Enter). Параметри, які задаються в команді, вказують, до яких об'єктів (файлів, каталогів тощо) застосовується команда, та конкретизують особливості її виконання.

Команди операційної системи поділяються на внутрішні і зовнішні. Деякі команди користувача, такі як type, dir, copy, командний процесор виконує сам. Такі команди називаються внутрішніми. Зовнішні команди реалізуються за допомогою окремих програм, які зберігаються у файлах

із розширеннями *.com, .exe* або *.bat.* Для виконання зовнішньої команди, так само як і для запуску будь-якої іншої програми, крім імені потрібно вказати шлях до відповідного програмного файла. Шлях до файла можна не вказувати, якщо він знаходиться в поточному каталозі або цей шлях міститься у списку пошуку операційної системи PATH, який задається у файлі AUTOEXEC.BAT. Якщо задану команду не знайдено ні в таблицях команд системи, ні в каталогах, то видається повідомлення: «Неправильне ім'я команди чи файла» (Bad command or file name).

Операційна система використовує при доступі до даних на диску таблицю розміщення файлів — FAT (File Allocation Table), кореневий каталог (root) та підкаталоги. FAT містить інформацію про розташування файлів, вільний простір на диску, наявність непрацездатних ділянок пам'яті, а також код формату диска. Для кожного файла в FAT створюється ланцюжок елементів, де елемент ланцюжка вказує на область диску фіксованої довжини (кластер), яку займає частина файла. У каталозі, який містить ім'я файла, є покажчик, який вказує на початок ланцюжка. При видаленні файла елементи FAT та області даних, які їм відповідають, звільняються (позначаються як вільні) і можуть бути використані для інших файлів.

Стартовий сектор (завантажувальний запис), таблиця розміщення файлів, кореневий каталог та вільний простір пам'яті, яка залишилась на диску, називають *елементами файлової структури диску.* Ці елементи створюються операційною системою в процесі підключення до роботи (ініціалізації) диску. Така структура диску дає можливість здійснення прямого доступу до даних файла.

ЗАПИТАННЯ І ЗАВДАННЯ

1. ДЛЯ ЧОГО потрібна операційна система?
2. Назвіть та схарактеризуйте основні компоненти операційної системи MS-DOS.
3. В яких випадках операційна система завантажується в пам'ять комп'ютера?

4. Які функції виконує при завантаженні базова система введення-виведення BIOS?
5. Де розташований завантажувач операційної системи?
6. Як завантажується операційна система?
7. Що таке драйвер?
8. Яку інформацію містить файл CONFIG.SYS?
9. Які задачі розв'язує комплекс програм файла MSDOS.SYS?
10. Для чого потрібний командний процесор?
11. Які допоміжні файли використовуються при завантаженні операційної системи MS-DOS?
12. Яка різниця між внутрішніми та зовнішніми командами операційної системи?

Створення і редагування файлів в операційній системі MS-DOS

ДЛЯ спрощення роботи користувачів із комп'ютером в операційній системі MS-DOS фірма Peter Norton Computing розробила програму Norton Commander (скорочено NC), яка набула великої популярності. Ця система, яка належить до так званих програм-оболонок, дала змогу замінити введення текстів команд простим натисканням однієї клавіші чи комбінації кількох клавіш. Команди в NC можна також вводити шляхом вибору їх із *меню* — списку команд, що відображається на екрані. Вибір команди з меню здійснюється встановленням курсора за допомогою клавіш управління курсором на потрібний пункт меню і натисканням клавіші **Enter.** Усі операції у NC можна також виконувати за допомогою маніпулятора «мишка».

Norton Commander наочно відображає вміст каталогів у двох прямокутних ділянках, що поділяють екран на дві рівні частини. Вони називаються *панелями.* Подібні виділені ділянки довільного розміру називаються *вікнами* і широко застосовуються при відображенні інформації на екрані.

Використання оболонки Norton Commander дає можливість легко запускати програми, копіювати, перейменовувати, переміщувати, а також переглядати та редагувати файли і виконувати інші команди. Наприклад, щоб увійти до підкаталогу поточного каталогу, потрібно встановити курсор

на ім'я каталогу, яке відображається на панелі великими літерами, і натиснути клавішу **Enter.** Так само слід діяти, аби перейти до каталогу, що знаходиться на один рівень вище (для його позначення в операційній системі застосовуються дві крапки (..); в NC вони висвічуються у першій позиції кожного підкаталогу). Для запуску програми в Norton Commander також потрібно встановити курсор на ім'я відповідного файла з розширенням *.com, .exe* або *.bat* і натиснути **Enter.** Перехід з однієї панелі на іншу, тобто переведення курсора з панелі на панель, здійснюється клавішею **Tab.**

Хоча при роботі в Norton Commander користувач може у більшості випадків обійтися без набирання текстів команд, однак при потребі команди MS-DOS можна вводити у *командному рядку,* що знаходиться під панелями NC (рис. 11).

Для контролю команд, які задаються, і усунення помилок при їх введенні Norton Commander видає у вікні (яке з'являється, як правило, в центрі екрана) детальні коментарі та запити. Для підтвердження виконання команди потрібно вибрати варіант роботи (наприклад, «виконати» або «відмінити»).

На панелях NC (рис. 11) показано кореневі каталоги двох розділів (логічних дисків) жорсткого диска — C: і D: . Над кожною панеллю вказується шлях до каталогу, зміст якого відображається у вікні. Курсором на правій панелі виділений каталог PICTURES. Останній рядок екрана містить меню, що відповідає функціональним клавішам від F1 до **F10.** Команди з цього меню можна вибирати мишкою тощо. При роботі з клавішами цей рядок може використовуватись як підказка.

Розглянемо послідовні етапи роботи з файлом: створення, збереження, перегляд і редагування.

Створення текстового файла. Файл, що містить текстову інформацію (текстовий файл), можна створити за допомогою команди операційної системи

copy CON ім'я__файла

де CON — системне ім'я клавіатури при введенні та екрана дисплея при виведенні. Фактично дається команда копіювати «клавіатурний файл» у файл на диску. Файл набирається рядками, в кінці кожного рядка натискується **Enter.** Після

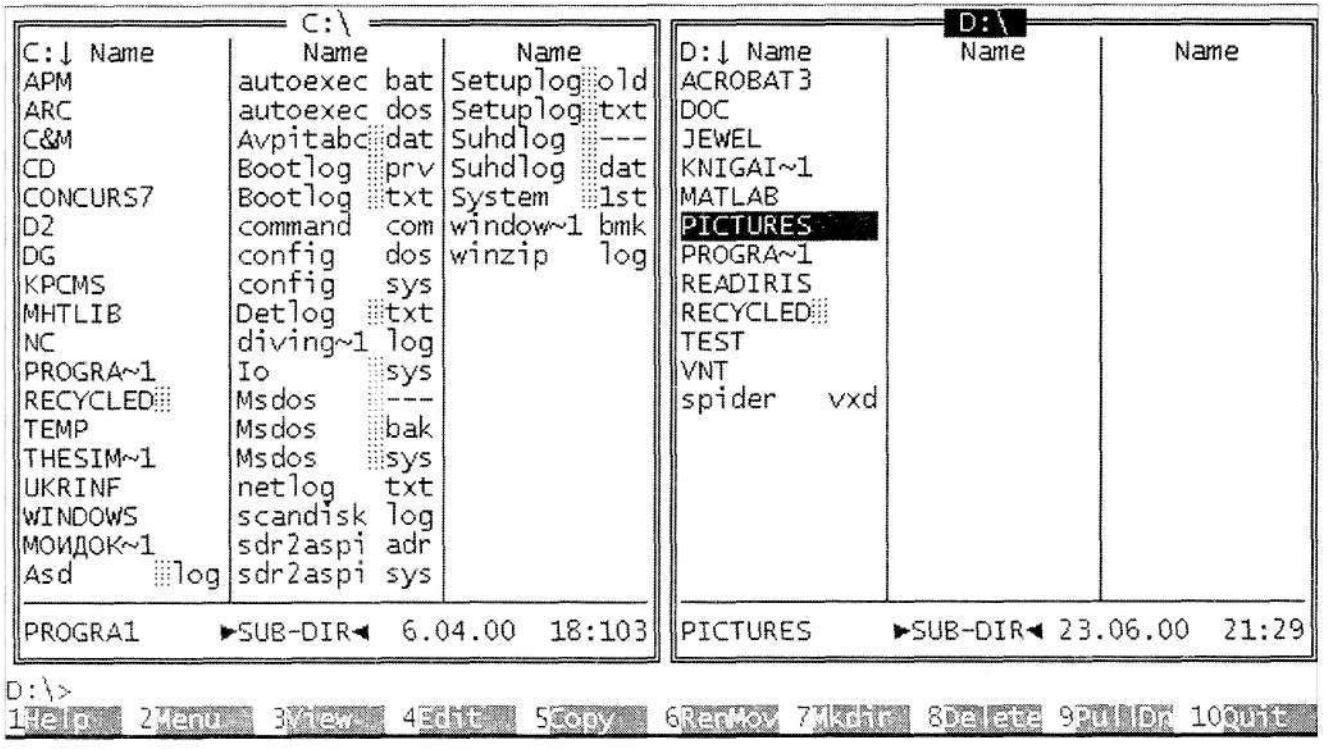

Рис. 11. Панелі системи Norton Commander

введення останнього рядка треба спочатку натиснути клавішу **F6,** а потім **Enter.** На диску в поточному каталозі з'явиться файл із вказаним іменем. Такий спосіб створення файлів має обмежене застосування через те, що він не надає змоги вносити виправлення в рядки, які вже набрані.

Створювати текстові файли будь-якого розміру та редагувати, тобто вносити зміни до них, можна за допомогою спеціальних програм — текстових редакторів.

Один із найпростіших редакторів входить до складу Norton Commander. Для створення за його допомогою нового файла треба натиснуги клавіші **Shift+ F4.** У вікні, що з'явилося після цього, необхідно набрати ім'я створюваного файла.

Екран очищається і можна набирати текст, дотримуючись таких правил:

- у кінці рядка треба натиснути **Enter;**
- для вилучення неправильного символу зліва від курсора використовується клавіша Backspace;
- для вилучення символу, на який указує курсор, використовується клавіша **Delete;**
- для переміщення по тексту можна використовувати клавіші управління курсором;
- вилучити рядок тексту можна комбінацією клавіш **Ctrl + Y,**
- вставити порожній рядок можна, натиснувши **Enter** у кінці попереднього;
- для виходу з редактора необхідно натиснути Esc або **F10** і підтвердити збереження файла, вибравши команду «Save».

Додаткові відомості про роботу з редактором можна отримати, натиснувши клавішу F1, яка в різних програмах служить для виведення довідкової інформації **(Help).**

Виклик редактора, що входить до складу MS-DOS, здійснюється за командою **edit.** При цьому з'являється вікно з поясненням подальших дій: очистити екран для набору тексту натискуванням клавіші Esc, отримати допомогу натискуванням клавіші **F1.** Натискування клавіші **Alt** дозволяє ввійти до меню, розташованого у верхньому рядку екрана. Вибір із меню здійснюється клавішами управління курсо-

ром та **Enter.** При виборі певного пункту меню з'являється список можливих операцій, вибір із якого також здійснюється за допомогою клавіш управління курсором і **Enter.** Так, після закінчення набирання тексту треба вибрати пункт меню **File,** а в ньому — підпункт **Save** (зберегти на дискові) і ввести ім'я файла. Докладно роботу текстового редактора MS-DOS описано в наступному параграфі.

Перегляд файла. Часто потрібно лише переглянути вміст файла, не змінюючи його. Для цього необхідно підвести курсор до імені файла в каталозі і натиснути клавішу F3. Текст файла з'являється на екрані. Для припинення перегляду файла слід натиснути Esc,

Переглянути на екрані вміст текстового файла можна також за допомогою команди операційної системи

type ім'яфайла

Наприклад, за командою

type read.me

буде виведено вміст файла read.me з поточного каталогу. Якщо скористатися цією командою для перегляду великих файлів, то практично нічого не можна побачити, оскільки виведення файла на екрані відбувається дуже швидко.

Щоб файл виводився на екран частинами, слід скористатися командою

type ім'яфайла | more

Для виведення наступної частини тексту треба натиснути **Enter.** Щоб вилучити з екрана панелі NC, які закривають виведений текст, треба натиснути **Ctrl + O.** Для відновлення панелей NC слід знову натиснути **Ctrl + O.**

Редагування файла. Для внесення змін у текстовий файл у редакторі NC потрібно натиснути клавішу F4, використовуючи такі самі правила, як і при створенні файла. Для редагування файла у редакторі MS-DOS потрібно задати команду операційної системи

edit ім'яфайла

або, викликавши редактор командою **edit,** увійти в меню

«File» і вибрати пункт «Open» (відкрити файл), після чого ввести ім'я файла.

ЗАПИТАННЯ І ЗАВДАННЯ

1. Яке призначення має система Norton Commander?
2. Чим відрізняється натискання комбінації клавіш **Shift+F4** від натискання тільки **F4?**
3. Як увійти в текстовий редактор Norton Commander?
4. Як вставити порожній рядок у текст?
5. Як у редакторі NC із рядка «Бондаренко Петро Іванович» зробити рядок «Бондаренко П.І.»?
6. Як вилучити рядок тексту?
7. Як вийти з редактора NC після набирання тексту і зберегти текст у файлі?
8. Як отримати довідку про роботу з програмним засобом?
9. Якою командою викликається текстовий редактор системи MS-DOS?
10. Як у редакторі MS-DOS зберегти набраний текст?
11. Як переглянути файл, використовуючи команди операційної системи?
12. Як вилучити панелі NC з екрана для перегляду коментарів і результатів роботи команд? Як потім відновити панелі?

Копіювання, переміщення, перейменування і відновлення файлів

Форматування дискети. Перш ніж говорити про переміщення файла з каталогу в каталог на одному диску чи з диска на диск, розглянемо, як підготувати до роботи нову дискету за допомогою програми форматування, що виконує спеціальну розмітку дисків. Ця програма розмічає доріжки, перевіряє їх і у разі дефекту видає повідомлення про зіпсовані доріжки й сектори. Для форматування дискети, що знаходиться в дисководі А, потрібно задати команду

```
format a:
```

У цій команді можна задати різні додаткові параметри, вони розділяються похилою рискою (це стосується всіх команд операційної системи). Так, для одержання довідки про будь-яку команду можна як параметр після риски ввести знак питання, наприклад

format /?

Щоб ознайомитися з виведеною довідкою, треба вилучити панелі NC (натиснути **Ctrl + O).**

При потребі обсяг дискети (720 Кб, 1.2Мб, 1.44Мб, ...) можна задати параметром **/f:розмір.** Наприклад:

1) обсяг 720 кілобайтів

format a: /f:720

2) обсяг 1,44 мегабайтів

format a: /f:1.44 або format a: /f:1440

Щоб створити так звану *системну дискету,* що містить операційну систему, потрібно задати команду

format a: /S

При цьому на дискету записуються системні файли IO.SYS, MSDOS.SYS і COMMAND.COM, а в перший сектор нульової доріжки — завантажувач системи.

Слід пам'ятати, що форматування повністю знищує інформацію, яка міститься на диску. Тому треба бути дуже уважним, щоб помилково не відформатувати жорсткий диск.

Norton Commander дає змогу дізнатися про об'єм диска чи дискети та про наявність на них вільного місця. Якщо зміст диска не відображається на одній із панелей NC, потрібно натиснути комбінацію AU + F1 або AU + F2 (відповідно для лівої та правої панелі) і вибрати ім'я диска зі списку у вікні, яке з'явиться при цьому (рис. 12).

Після переходу до панелі, яка відображає зміст диска, який нас цікавить, потрібно натиснути комбінацію **Ctrl + L.** При цьому на другій панелі замість змісту диска з'явиться інформація про обсяг повної та вільної оперативної пам'яті та пам'яті на диску. Щоб повернутися до змісту диска, потрібно знову натиснути клавіші **Ctrl + L.**

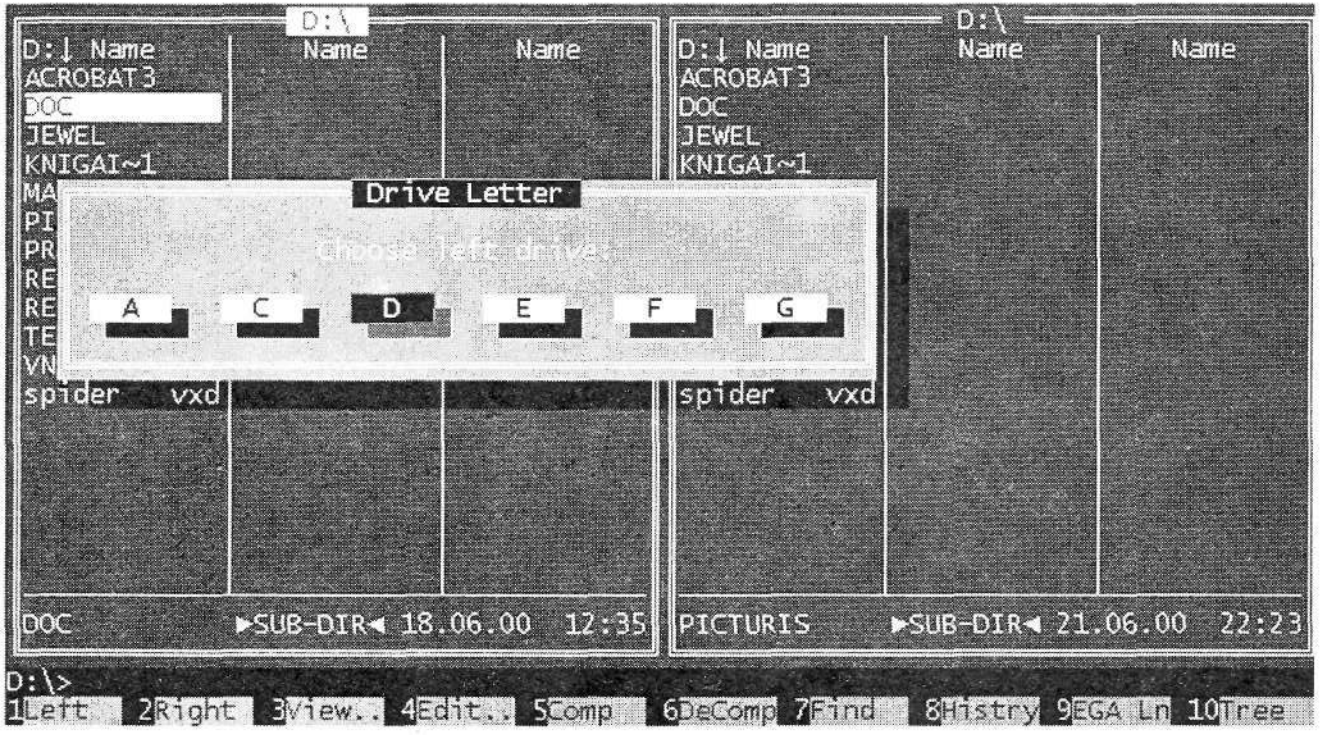

Рис. 12. Вигляд екрана після натискання клавіш Alt-F1. У вікні вибору імені диска є горизонтальне меню із шести пунктів. Використовуються клавіші управління курсором, вибирається ім'я диска і натискається Enter. На лівій панелі системи NC з'явиться зміст обраного диска

Копіювання файлів. Найчастіше використовується копіювання файлів з одного диска на інший, з каталогу в каталог, і навіть створення й виведення змісту файла на друк можна розглядати як копіювання. В Norton Commander копіювання проводиться натискуванням клавіші F5. Курсор при цьому повинен знаходитись на імені файла, що копіюється. Якщо файл знаходиться в підкаталозі, потрібно підвести курсор до імені підкаталогу і знову натиснути **Enter.** Зробити це слід для кожного з послідовних підкаталогів, якщо їх кілька, доки не з'явиться список з іменем потрібного файла. Попередньо на другій панелі треба відкрити потрібний каталог, для чого необхідно поставити курсор на ім'я каталогу в змісті диска і натиснути **Enter.** Після натискування F5 програма NC виводить на екран вікно, в якому повідомляється вікно копіюємого файла та каталог, у який вона його копіює.

Розглянемо приклади копіювання файлів.

Приклад 1. Копіювання з гнучкого диска в кореневий каталог. Для цього потрібно:

а) вставити дискету в дисковод А;

б) для появи змісту диска А натиснути клавіші **Alt** + Fl або **Alt** + F2 і вибрати з меню, що з'явилося, ім'я диска (A:);

в) у змісті дискети помістити курсор на ім'я файла, що копіюється;

г) натиснути клавішу F5; при цьому з'являється діалогове вікно команди копіювання (рис. 13);

д) ім'я файла можна змінити, набравши нове ім'я у вікні команди; для виконання команди копіювання потрібно натиснути клавішу **Enter;**

є) якщо на диску вже є файл із зазначеним іменем, то з'являється вікно з попередженням і треба або підтвердити необхідність копіювання нової версії того самого файла, вибравши варіант «Overwrite» (переписати), або відмінити копіювання («Cancel»), після чого знову натиснути **Enter.**

Файли можна копіювати також за допомогою команди copy операційної системи. Для копіювання файла з дискети команда має вигляд

copy a:\ім'я файла

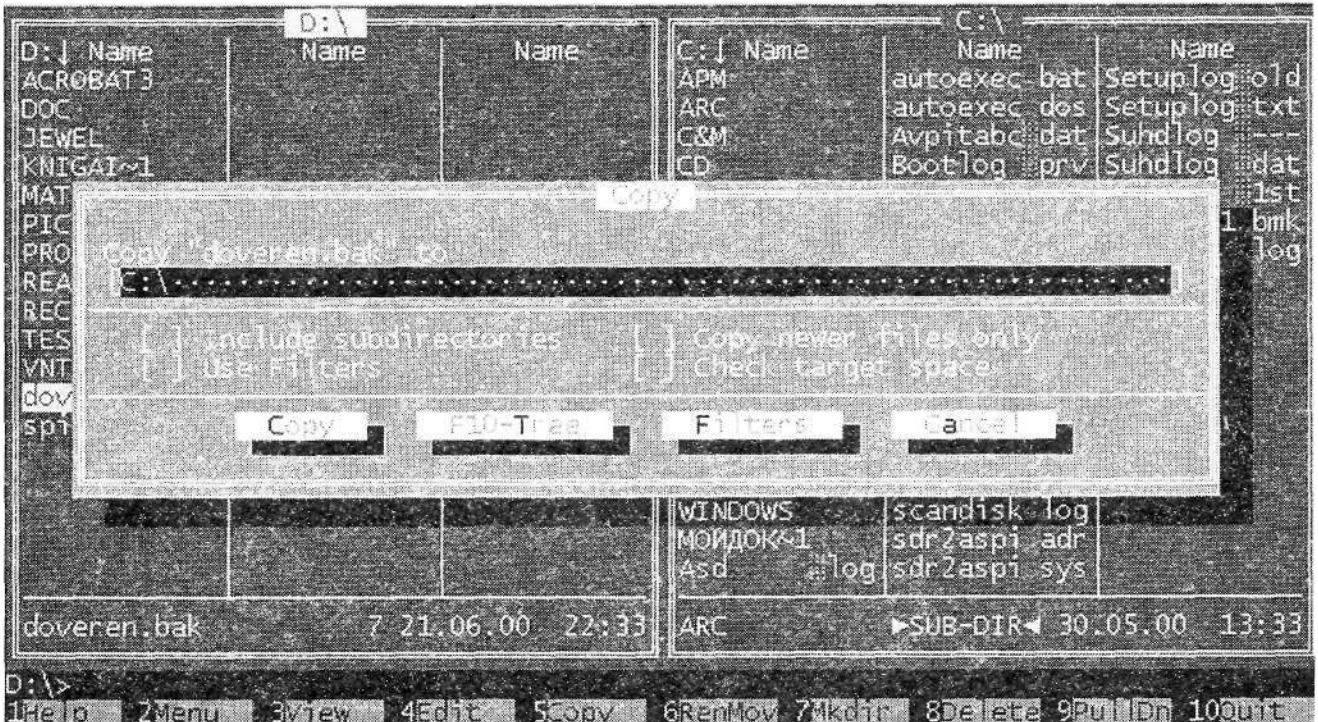

Рис. І3. Вигляд екрана при виконанні команди копіювання. У вікні команди вказане ім'я файла, який копіюється (doveren.bak) і куди він буде скопійований (до кореневого каталогу диска C)

Файл із зазначеним іменем буде скопійовано у поточний каталог. Для копіювання у поточний каталог усіх файлів з кореневого каталогу дискети А треба набрати команду

copy a:*.*

Для копіювання файлів певного типу, наприклад із розширенням *.txt,* потрібна команда

copy a:*.txt

Приклад 2. Копіювання на гнучкий диск А. Для цього потрібно:

а) — в) повторити з прикладу 1 ;

г) позначити в поточному каталозі ім'я файла, що копіюється;

д) натиснуги F5;

є) змінити або залишити ім'я файла, що копіюється, і натиснути клавішу **Enter.**

Файли рекомендується періодично переписувати на дискети, тому для кожної нової версії може виникнути ситуація копіювання файла з ім'ям, яке вже є в змісті. У такому разі, як і в прикладі 1, треба вибрати «Overwrite» і натиснути Enter. Команда, за якою буде виконано ту ж саму операцію копіювання, має вигляд

copy ім'яфайла a:

Приклад 3. Копіювання з каталогу в каталог на жорсткому диску. Для цього потрібно:

а) на одній з панелей відкрити каталог, куди буде проводитись копіювання;

б) на другій панелі ввійти до каталогу з іменем файла, що копіюється;

в) установити курсор на ім'я файла, що копіюється;

г) натиснути F5;

д) змінити або залишити ім'я файла у вікні, підтвердити команду і натиснути **Enter.**

Відповідна команда операційної системи має вигляд

copy ім'яфайла ім'я_каталогу

Приклад 4. Виведення файлів на друк.

Друкуючий пристрій (принтер) у системі має спеціальне

ім'я prn. Для друкування файла підводимо курсор до його імені і натискуємо F5. У вікні команди копіювання витираємо ім'я каталогу (скориставшись клавішею **Del),** в який має копіюватися файл, і набираємо prn. Друкування файла можна здійснити за командою

copy ім'я_файла prn

або за командою

print ім'я_файла

В останньому випадку після повідомлення, що з'являється у відповідь на цю команду, треба натиснути **Enter.**

Приклад 5. Копіювання групи файлів.

Для одночасного копіювання кількох файлів спочатку треба відмітити ці файли, для чого треба підвести курсор до імені кожного файла і натиснути клавішу **Ins.** Потім потрібно натиснути F5. У вікні, що з'являється після цього, повідомляється, скільки файлів відмічено. Далі слід натиснути **Enter.**

Створення нового каталогу. Для копіювання файлів, що належать до однієї групи, може знадобитися новий каталог (підкаталог). Створити його можна, використовуючи клавішу F7. У вікні треба набрати ім'я каталогу (не більше 8 символів) і натиснути **Enter.**

Перейменування файлів. Для здійснення операції перейменування файлів в Norton Commander потрібно встановити курсор на потрібний файл або каталог і натиснути клавішу F6. У вікні, що з'являється при цьому, слід вказати нове ім'я файла або каталогу і натиснути **Enter.** Для скасування операції слід натиснути клавішу Esc. Можна також одночасно перейменувати групи файлів, відмічаючи їх за допомогою клавіши **Ins** (найчастіше це використовується для зміни розширень імен файлів).

Перейменувати файли можна також за командою операційної системи ren. Для одного файла команда має вигляд

ren ім'яфайла новеім'яфайла

Перейменування файлів здійснюється в межах одного каталогу. При перейменуванні файла, який знаходиться не в поточному каталозі, вказується шлях до нього. Для груп

файлів команда виконується аналогічно команді копіювання. Так, команда

```
ren    *.txt    *.doc
```

перейменує всі файли в поточному каталозі з розширенням *.txt* у файли з розширенням *.doc,* зберігаючи імена файлів.

Перейменування з переміщенням. В Norton Commander натисканням клавіші F6 можна не тільки перейменовувати, а й переміщувати файли з одного каталогу в інший або з диска на диск, надаючи при потребі нові імена. Переміщення відрізняється від копіювання лише тим, що після успішного завершення пересилання початкові файли вилучаються. При переміщенні файлів у вікні, що з'являється після натискування F6, потрібно вказати шлях до нового місцезнаходження файла і, при потребі, нове ім'я.

Вилучення файла. Для вилучення файла курсор ставиться на його ім'я і натискається клавіша F8. Norton Commander видає у вікні ім'я файла, який вилучається. Якщо вилучення підтверджується, треба натиснути **Enter,** якщо ні — **Esc.** Вилучати можна також групи файлів.

Команда операційної системи для вилучення файла має вигляд

```
del ім'яфайла
```

Відновлення вилученого файла. Для відновлення помилково вилучених файлів можна скористатися командою, що реалізується *утилітою* (службовою програмою) **undelete** або **unerase:**

```
undelete ім'яфайла
```

або

```
unerase ім'яфайла
```

Краще відновлювати файл відразу після його помилкового вилучення. При виконанні команди вилучення файла він фізично не стирається, а лише змінюється перша буква його імені. Якщо ввести команду відновлення без імені файла, видається список вилучених файлів, Якщо в останньому стовпці списку проти імені файла написано «poor», то файл відновленню не підлягає.

1. Як відформатувати дискету?
2. Чи залишиться без змін інформація на диску після форматування?
3. Як створити системну дискету?
4. Які модулі операційної системи розміщуються на системному диску?
5. Як дізнатися, скільки вільного місця є на диску?
6. Як задати обсяг дискети при форматуванні?
7. Як можна скопіювати файли?
8. Чи можна скопіювати файл, ім'я якого відсутнє в поточному каталозі?
9. Як побачити зміст гнучкого диска на одній із панелей NC?
10. Як перейти з однієї панелі NC до іншої?
11. За якою командою можна скопіювати файл у поточний каталог?
12. Як скопіювати всі файли в поточний каталог?
13. Як скопіювати файли з однаковим розширенням у поточний каталог?
14. Як скопіювати файл із жорсткого диска на дискету?
15. Як скопіювати файл foxhelp.arj (рис. 9) у підкаталог TEST?
16. Чи можна змінити ім'я файла при копіюванні?
17. Назвіть системні імена клавіатури і принтера.
18. Як надрукувати файл, використовуючи команду копіювання?
19. Як надрукувати файл, не припиняючи роботи з комп'ютером?
20. Задайте команду операційної системи для копіювання файла doc.arj (рис. 9) на гнучкий диск, вказавши повний шлях до даного файла.
21. Задайте команду операційної системи для копіювання на жорсткий диск файла 5-19-7.fsh (рис. 9) , вказавши повний шлях до даного файла.
22. Як перейменувати групи файлів, що мають однакове розширення імен?
23. Як змінити розширення імен групи файлів, що мають однакове розширення імен?
24. Як вилучити файл?
25. Що робити, якщо файл було вилучено помилково?
27. Як можна вилучити файл, зберігши його одночасно в іншому каталозі?
28. За якою командою можна змінити не тільки місцезнаходження файла, а й його ім'я?

1.9. Сучасні операційні системи

Розглянута в попередніх параграфах операційна система MS-DOS належить до однозадачних операційних систем, під керуванням яких може одночасно виконуватися лише одна програма. Це означає, що користувач одночасно може запускати на виконання лише одне завдання (формування тексту, виконання обчислень тощо).

На відміну від цього, *багатозадачні* системи (Windows 95 та 98, Windows NT, Windows 2000, UNIX та ін.) забезпечують одночасне виконання кількох програм.

Існують операційні системи, які дають змогу одночасно працювати на одному комп'ютері багатьом користувачам. Такі системи називаються *багатокористувацькими.* Такими операційними системами є UNIX, Windows NT, Windows 2000.

До основних функцій операційних систем можна віднести такі:

1. *Запуск програм, контроль за їх виконанням та завершенням.* Для запуску програми потрібно розмістити її в оперативній пам'яті та настроїти процесор на виконання першої команди цієї програми, після чого потрібно контролювати процес її виконання. Після закінчення роботи програми необхідно звільнити пам'ять та інші ресурси, які попа використовувала.

2. *Розподілення часу процесора.* На одній машині може виконуватися декілька програм. У такому разі вони повинні розподіляти між собою процесорний час, тобто процесор повинен постійно переходити від виконання однієї програми до іншої. Виділення процесорного часу здійснюється відповідно до рівня *пріоритету* (першочерговості) програми.

3. *Контроль за доступом до ресурсів комп'ютера.* Програми не повинні заважати одна одній; зокрема дві програми не можуть мати одночасний доступ до певних апаратних ресурсів комп'ютера, файлів тощо.

4. *Підтримка файлової системи,* тобто забезпечення певної організації зберігання файлів на дисках та виконання базових операції з файлами (створення, копіювання, переміщення, вилучення і т.п.).

5. *Керування зовнішніми пристроями.* Операційна система повинна забезпечувати взаємодію із зовнішніми пристроями.

6. *Забезпечення взаємодії з користувачем.* Операційна система повинна підтримувати діалог із користувачем: сприймати від нього певні команди і відповідати на них. Сукупність засобів, які забезпечують таку взаємодію, називається *інтерфейсом користувача.*

Інтерфейси користувача можна розділити на два типи: *інтерфейс командного рядка* і *графічний інтерфейс.* Характерною рисою інтерфейсу командного рядка є наявність визначених команд, які користувач повинен вводити з клавіатури. Сприймання і обробка цих команд забезпечуються складовою частиною операційної системи, яка називається *командним процесором.*

Інтерфейс командного рядка є не дуже зручним, оскільки важко запам'ятовувати велику кількість потрібних для роботи команд та вводити їх до комп'ютера за допомогою клавіатури. Графічні інтерфейси є більш зручними і наочними.

Однією з найвідоміших операційних систем є UNIX. Саме з цією системою пов'язується розвиток Інтернету, програмне забезпечення якого донедавна базувалося головним чином на UNIX. Існує декілька версій UNIX, між якими можуть бути досить значні відмінності. Аналогом UNIX є популярна операційна система LINUX.

UNIX з самого початку створювалася як багатозадачна багатокористувацька операційна система, в якій завжди приділялася значна увага питанням безпеки інформації, надійності роботи користувачів на багатьох робочих місцях. UNIX забезпечує ефективну роботу комп'ютерів, об'єднаних у мережу. Лише останнім часом серйозну конкуренцію UNIX у цій галузі створює інша операційна система — Windows NT та її нова версія Windows 2000.

На більшості сучасних персональних комп'ютерів використовуються операційні системи Windows 95 та 98 (їх об'єднують назвою Windows 9x). Зручний графічний інтерфейс цих систем знайшов подальший розвиток у системі Windows 2000, яка має декілька версій. Так, Windows 2000 Professional являє собою операційну систему для настільних

і переносних комп'ютерів. Windows 2000 Server — багатоцільова мережева операційна система. Windows 2000 Advanced Server включає всі компоненти Windows 2000 Server та забезпечує підтримку великої кількості користувачів.

ЗАПИТАННЯ І ЗАВДАННЯ

1. Назвіть і схарактеризуйте основні функції операційних систем.
2. Що таке однозадачні та багатозадачні операційні системи? Наведіть приклади.
3. Що таке багатокористувацька операційна система? Наведіть приклади.

Система Windows

Розроблені фірмою Microsoft операційні системи Windows мають дуже простий для вивчення і зручний для використання графічний інтерфейс. Він дозволяє користувачеві легко налагоджувати систему відповідно до своїх потреб і працювати швидко й ефективно. У ньому реалізована ідея «поверхні письмового столу»: користувачеві видно все, що знаходиться на столі, і йому варто лише простягнути руку і «взяти» потрібний предмет. У Windows для представлення різноманітних об'єктів широко використовуються графічні зображення *(піктограми)*.

У сучасних версіях Windows значно спрощено під'єднання до комп'ютера нового обладнання. У них також зняті обмеження в назвах файлів. Тепер ім'я файла може містити до 255 символів, включаючи пропуски та розділові знаки, що дозволяє давати файлам більш зрозумілі імена.

Щоб розпочати роботу з комп'ютером, на якому встановлена операційна система Windows, досить увімкнути живлення. Невдовзі на екрані з'являється заставка, а потім — *робочий стіл* Windows (рис. 14). Тут і далі наводяться зображення екранів російськомовної версії Windows.

Зовнішній вигляд екрана може змінюватися, однак у будь-якому разі на робочому столі можна побачити кілька піктограм, які відображають різні об'єкти Windows.

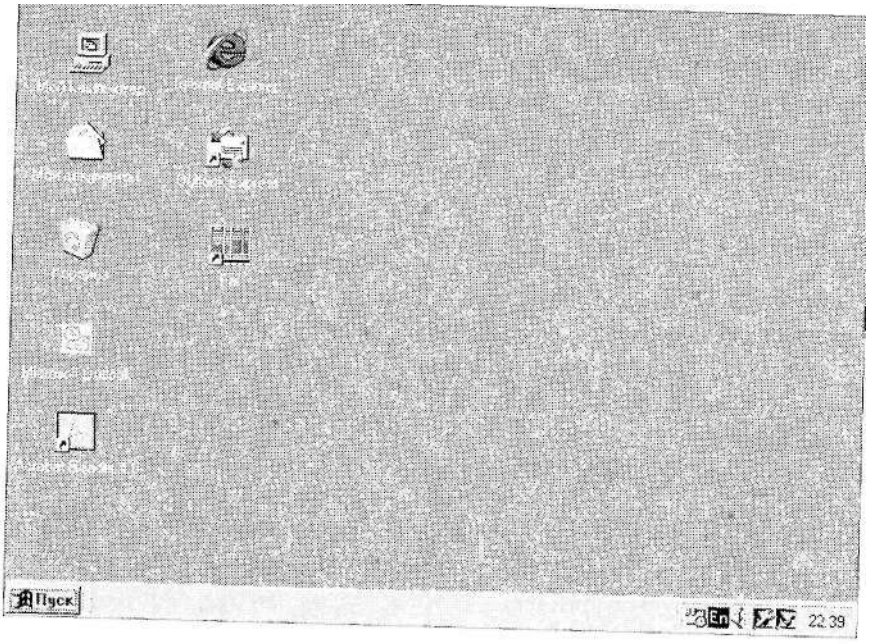

Рис. 14. Робочий стіл Windows

У нижній частині екрана розташовується панель задач у лівій частині якої знаходиться кнопка «Пуск». Вона використовується для відкриття головного меню, за допомогою якого можна здійснювати запуск програм, пошук файлів входити до довідкової системи тощо.

Відразу після запуску системи ми бачимо на екрані курсор мишки, який здебільшого має форму нахиленої вліво стрілки. Курсор мишки може змінювати свою форму залежно від того, на який об'єкт або частину об'єкта він вказує Форма курсора говорить про те, що саме в даний момент можна робити з об'єктом.

За термінологією, прийнятою у Windows, файлові каталоги називаються *папками.* Для забезпечення зручного й ефективного доступу до ресурсів комп'ютера через стандартний графічний інтерфейс поняття папки було розширене спеціальні системні папки, такі як «Мій комп'ютер» чи «Мережеве оточення», є контейнерами, що містять папки таких системних ресурсів, як диски, дисководи, принтери під єднані до мережі комп'ютери тощо.

Вікна у системі Windows. При запуску програми чи відкритті папки на робочому столі з'являється її вікно, а на панелі задач — відповідна кнопка. Вікно, в якому працює користувач, називається активним. Воно знаходиться на передньому плані. Щоб перейти до якогось вікна, слід підвести курсор до видимої частини потрібного вікна або його кнопки на панелі задач і клацнути лівою кнопкою мишки.

Вікна у Windows бувають чотирьох типів:

1) *вікна папок* вміщують значки (піктограми) інших об'єктів Windows і елементи управління вікном (рис. 15);

2) *вікна прикладних програм* вміщують робочу інформацію, із якою працюють ці програми, а також елементи управління вікном (рис. 16);

3) *діалогові вікна* вміщують тільки елементи управління (рис. 17);

4) *вікна довідкової системи* вміщують допоміжну довідкову інформацію для роботи з операційною системою та прикладними програмами, а також елементи управління довідковою системою (рис. 18).

Усі типи вікон мають подібну структуру. Заголовок, що знаходиться вгорі вікна, містить його назву, а також кілька кнопок управління в правому кінці рядка заголовку. Вікна прикладних програм та папок мають три кнопки управління. Перша з них (із горизонтальною рискою внизу) служить для згортання вікна. Друга кнопка використовується для розгортання вікна та відновлення його попереднього розміру (при цьому на ній зображується відповідно одне віконце або два). Третя кнопка (із хрестиком) служить для закриття вікна. Безпосередньо під заголовком знаходиться рядок меню. Якщо розмір вікна недостатній для розміщення всього зображення, справа та внизу з'являються смужки прокручування.

Розміри та положення вікна можна змінювати, «перетягуючи» мишкою межі рамки та «буксируючи» його в потрібне місце. Змінювати параметри вікна можна також за допомогою меню, яке з'являється при встановленні курсора в будь-якому місці заголовка і клацанні правою кнопкою мишки (можна також натиснути на клавіатурі комбінацію **Alt** + пробіл).

Слід зауважити, що використання таких пов'язаних із змістом об'єктів меню (вони називаються контекстними)

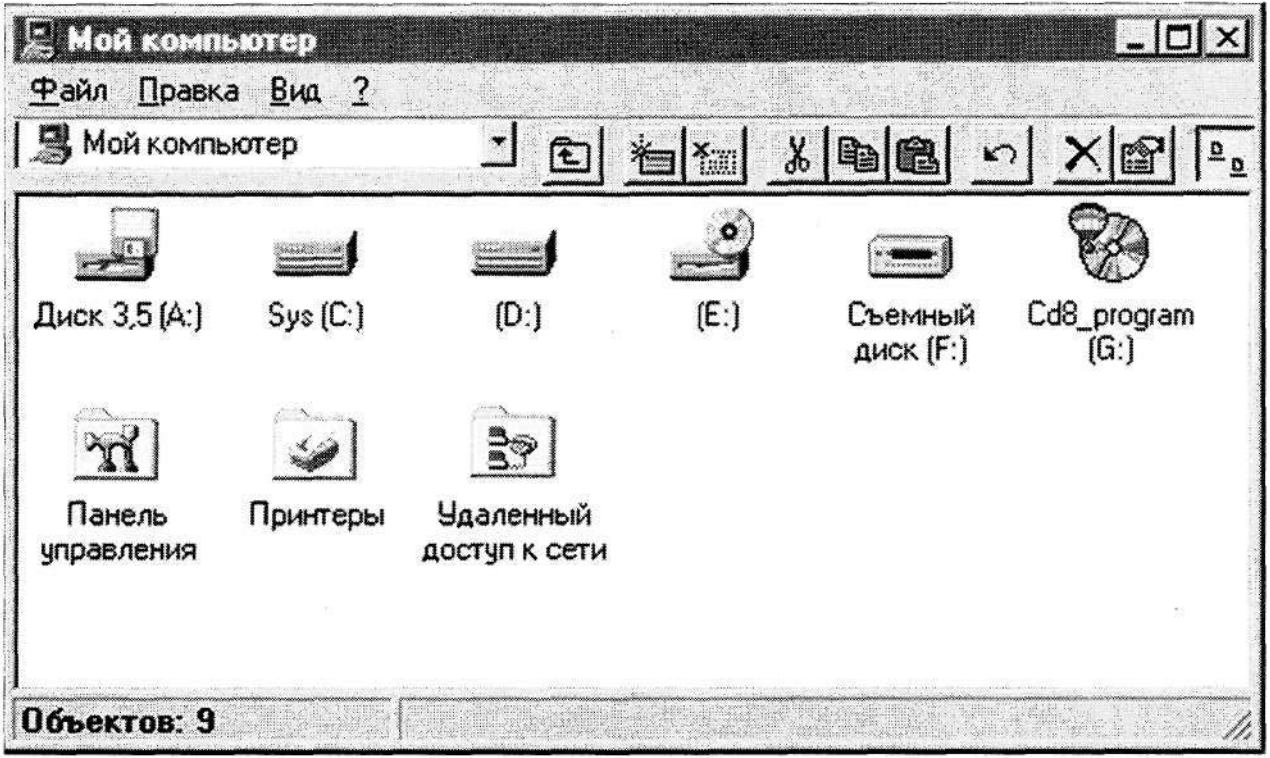

Рис. 15. Вікно папки

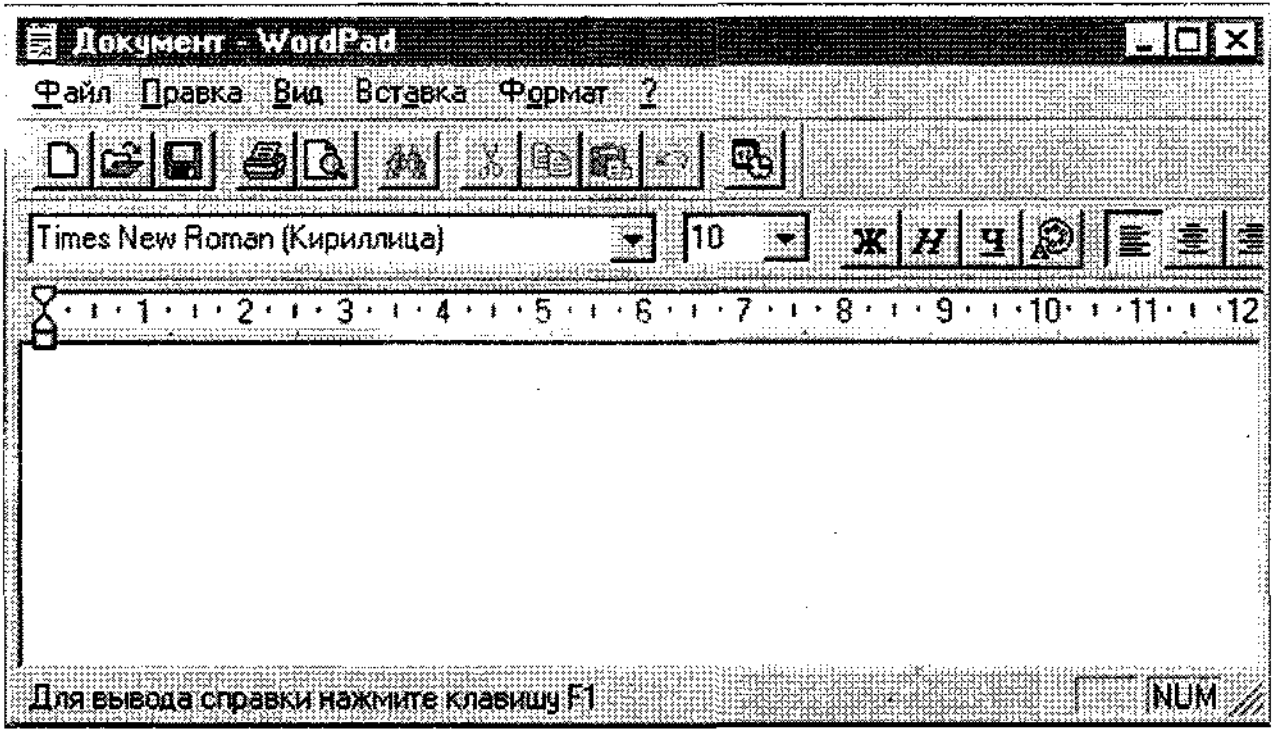

Рис. 16. Вікно прикладної програми

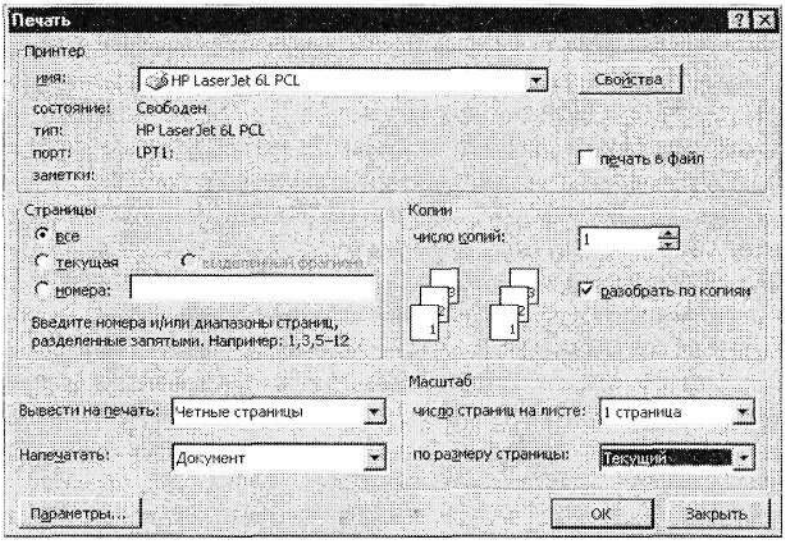

Рис. 17. Діалогове вікно

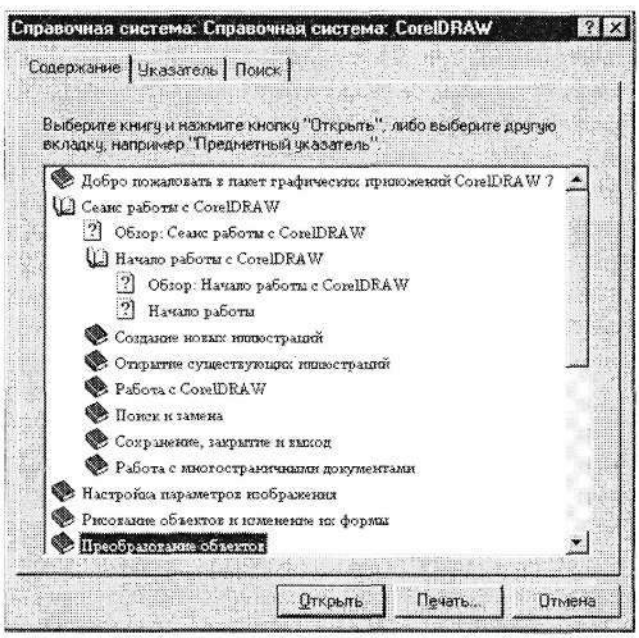

Рис. 18. Вікно довідкової системи

дає змогу спростити доступ до багатьох операцій і прискорити їх виконання. Такі меню викликаються встановленням курсора мишки на об'єкт і клацанням правою кнопкою мишки.

Використання папок. Побачити вміст будь-якої папки дає змогу подвійне клацання лівою кнопкою мишки з наведенням курсора на її зображення. Послідовно відкриваючи папки, ми можемо дістатися до будь-якого потрібного файла. Для створення нової папки у вікні або на робочому столі треба встановити курсор мишки на вільному місці і клацнути правою кнопкою мишки, а потім вибрати з контекстного меню, що з'явиться, пункт «Створити», а в ньому — пункт «Папка». В результаті з'являється піктограма нової папки. Ім'я нової папки вводиться за допомогою клавіатури, після чого треба натиснути клавішу **Enter.**

Панка «Корзина» являє собою тимчасове сховище вилучених файлів. Наявність їх у «Корзині» дає можливість відновити ці файли, якщо їх було видалено помилково.

Дуже зручним у Windows є використання технології Drag and Drop («перетягни й кинь»), яка дозволяє за допомогою мишки не лише змінювати положення практично всіх об'єктів, а й переміщувати та копіювати їх із одного вікна до іншого та на поверхню робочого столу, передавати їх на обробку до відповідних прикладних програм тощо. Як правило, при перетягуванні позначень файлів та папок у межах одного диска та на робочий стіл відбувається перенесення об'єктів, а між дисками — копіювання. Щоб явно вказати операцію копіювання, треба при перетягуванні натиснути клавішу **Ctrl,** а для перенесення — **Shift.** Якщо ж при перетягуванні утримувати праву кнопку мишки, то після його завершення з'явиться контекстне меню, з якого можна вибрати потрібну операцію.

Щоб видалити об'єкт, його піктограму перетягують до «Корзини» або користуються відповідною командою з контекстного меню.

Об'єкт можна перейменовувати, встановивши курсор на його піктограмі і клацаючи правою кнопкою мишки. Потім вибрати команду «Перейменувати» із контекстного меню.

Для виконання операції копіювання, перенесення, вилучення тощо з групою об'єктів їх потрібно вибрати (відміти-

ти), а потім, захопивши мишкою один із об'єктів, переміщувати групу як єдине ціле. Вибір групи здійснюється кількома способами. Можна за допомогою мишки окреслити її прямокутником, натиснувши на ліву кнопку і розтягуючи його по діагоналі (починати потрібно з вільного місця вікна чи робочого столу). Якщо ж відмітити один об'єкт, а потім, натиснувши клавішу **Shift,** другий, то буде здійснено виділення неперервної послідовності об'єктів — від першого з вибраних до останнього. При потребі вибору об'єктів по одному треба утримувати клавішу **Ctrl.**

Використання «Провідника». У Windows є програма-утиліта, яка називається «Провідник». У ній можна виконувати ті самі задачі, що й у вікнах папок — завантажувати програми, відкривати документи, а також копіювати, переміщувати, вилучати і перейменовувати папки і файли. Перевага використання програми «Провідник» полягає в графічному представленні ієрархії папок, за допомогою якої можна легко визначати місця розташування даних елементів і переходити від одного рівня ієрархії до іншого.

Завантажити програму «Провідник» можна після встановлення курсора на кнопці «Пуск», що знаходиться на панелі задач. Потім клацнути лівою кнопкою мишки та перейти до меню «Програми» і вибрати пункт «Провідник», або ж клацнути правою кнопкою мишки і вибрати пункт «Провідник» із контекстного меню. В обох випадках з'явиться вікно програми «Провідник», яке розділене на дві частини або панелі. Ліва панель містить діаграму ієрархії папок, права виглядає як звичайне вікно папки. При першому запуску програми «Провідник» з'являється чотири рівні ієрархії папок:

1) робочий стіл;

2) папки й інші піктограми робочого столу, серед них такі, як «Мій комп'ютер» і «Корзина»;

3) вкладені папки папок робочого столу (наприклад, під рядком «Мій комп'ютер» ви побачите піктограми різних дисків комп'ютера);

4) вкладені папки вкладених папок, що входять у папки робочого столу.

У правій частині вікна «Провідник» відображається вміст папки, яка була вибрана в даний проміжок часу. Об-

рана папка виділяється в ієрархічній градації папок, а її ім'я з'являється на панелі заголовка вікна «Провідник». Наприклад, якщо вибрана папка «Робочий стіл», то на панелі заголовка ви побачите «Провідник» — «Робочий стіл» (рис. 19). Як тільки в лівій частині вікна вами буде обрана будь-яка інша папка, то її вміст буде відображено в правій частині вікна замість вмісту попередньої папки.

Кожна з панелей вікна «Провідник» містить свої власні смужки прокручування. Переміщатися по вмісту однієї з панелей можна, клацнувши в якому-небудь місці цієї панелі, щоб вибрати її, а потім, натискаючи клавіші **Page Up** і **Page Down** для переміщення відразу на цілий екран вгору або вниз. Можна також використовувати клавішу **End,** щоб переміститися вниз списку, і клавішу **Home,** щоб переміститися в самий верх.

Розглянемо використання технології Drag and Drop при роботі з «Провідником» на прикладі операцій копіювання або переміщення. Для цього потрібно:

1) виділити елементи для копіювання чи переміщення в правій частині вікна «Провідник»;

2) перейти по сходах ієрархії в лівій частині до папки, у яку необхідно вставити елементи, і зробити її видимою;

3) клацнувши правою кнопкою мишки, перетягнути виділені елементи з правої частини в призначену папку в лівій частині;

4) коли з'явиться запитання про те, чи хочете ви перемістити чи скопіювати елементи, вказати потрібний варіант.

Якщо немає впевненості у своїх діях, слід уникати перетягування піктограм у лівій частині вікна «Провідник», тому що при цьому можна порушити ієрархію папок.

Для створення нової папки чи файла у вікні «Провідник» потрібно виділити папку, в яку треба помістити новий файл чи папку, клацнути правою кнопкою мишки при розміщенні курсора на вільному просторі правої частини вікна і вибрати команду «Створити».

Щоб отримати більш докладну інформацію про роботу Windows, можна клацнути по кнопці «Пуск» і увійти до пункту меню «Довідка». Одна з команд цього пункту — підручник для вивчення системи Windows. Довідкова система Windows дозволяє шукати тематичні сторінки за допомогою

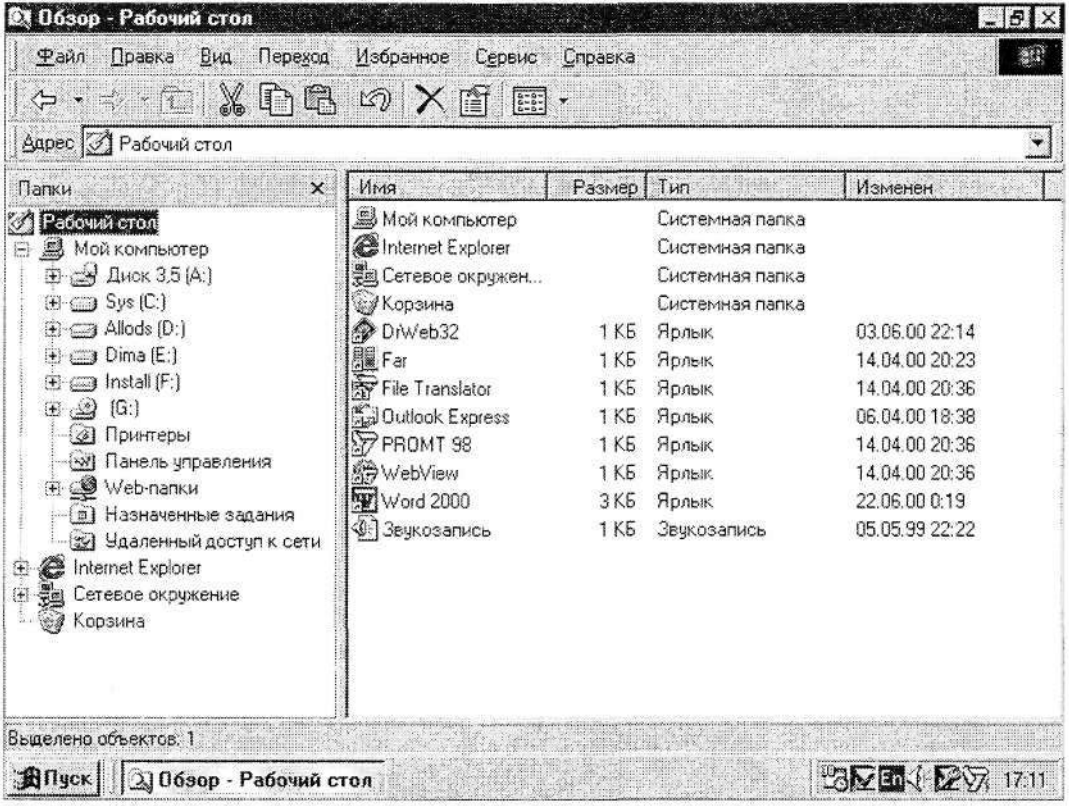

Рис. 19. Вікно «Провідника»

предметного покажчика. Попрацювати з довідковою системою рекомендується на практичних заняттях. У прикладних програмах Windows також є пункт меню «Довідка» або «?», при звертанні до якого надається докладна інформація щодо роботи з даним програмним засобом.

З метою створення більших зручностей у використанні комп'ютера фірма Microsoft пропонує в кожній наступній версії Windows більш висока надійність та ефективність, вдосконалений інтерфейс користувача, нові засоби налагодження та керування системою. Значна увага приділяється забезпеченню можливостей використання програмних продуктів, розроблених для попередніх версій Windows.

Щоб вимкнути комп'ютер або перезавантажити операційну систему, слід виконати процедуру закриття системи. Оскільки повинна існувати можливість збереження всіх змін на робочому столі, то операційна система має бути підготовленою до вимкнення. Просте вимкнення комп'ютера під час роботи Windows майже неминуче призведе до деякої втрати даних. Тому для безпечного вимкнення комп'ютера служить команда «Закінчення роботи» у головному меню. Після виконання цієї команди (натискання кнопки) треба зачекати появи на екрані повідомлення про те, що комп'ютер можна безпечно вимкнути,

ЗАПИТАННЯ І ЗАВДАННЯ

1. Що таке Windows?
2. Яка ідея реалізована в інтерфейсі Windows?
3. Яка структура вікон системи Windows?
4. Що таке піктограма?
5. Як перемістити на екрані вікно або піктограму?
6. Що таке «Провідник»?
7. Які бувають значки на екрані Windows?
8. Що таке папка в Windows?
9. Якою може бути довжина імен файлів у Windows?

Графічний редактор

Серед стандартних програм у системі Windows є прикладна програма для створення, перегляду та редагування рисунків, яку називають графічним редактором, У вікні графічного редактора, поданому на рис. 20, зліва знаходиться меню з піктограмами інструментів для побудови рисунків.

Засоби для побудови малюнків. Щоб вибрати інструмент для побудови зображення, слід встановити курсор на піктограму потрібного інструмента і клацнути лівою кнопкою мишки. При цьому курсор змінює свою форму на хрестик, олівець тощо. У нижній частині вікна графічного редактора розміщена палітра для вибору кольорів. Зліва від палітри знаходяться два зафарбованих квадратики. Під меню з піктограм може з'являтися додаткове меню для параметрів інструментів. Наприклад, при побудові ліній в додатковому меню вибирається їх товщина. На рис. 20 вибрана третя зверху лінія. Цією лінією зроблено рисунок.

Графічний редактор дає змогу одночасно зображати різні лінії і фігури. Рисунок, що потребує старанної обробки зображення, можна збільшити, скориставшись піктограмою збільшувального скла. Розмір рисунка може бути більшим, ніж розмір вікна. Побачити фрагменти рисунка, що не помістилися у вікні, дають можливість смужки зі стрілками для вертикального і горизонтального перегляду, що знаходяться внизу і справа від вікна.

На рис. 21 подані піктограми інструментів, які використовуються при створенні рисунків.

Зображена пунктиром зірочка дозволяє виділяти будь-які фрагменти рисунка для подальшого переміщення, копіювання та вилучення, а прямокутник — фрагменти прямокутної форми. Щоб виділити фрагмент рисунка, потрібно натиснути ліву кнопку мишки і обвести його. Вирізати, копіювати, вставляти або вилучати виділені фрагменти зображення можна, користуючись пунктами меню «Правка».

Для стирання зображень використовуються гумки. Розмір гумки можна змінювати.

Розпилювач будує зображення у вигляді розпиленої фарби. При швидкому переміщенні курсора слід розпилювача

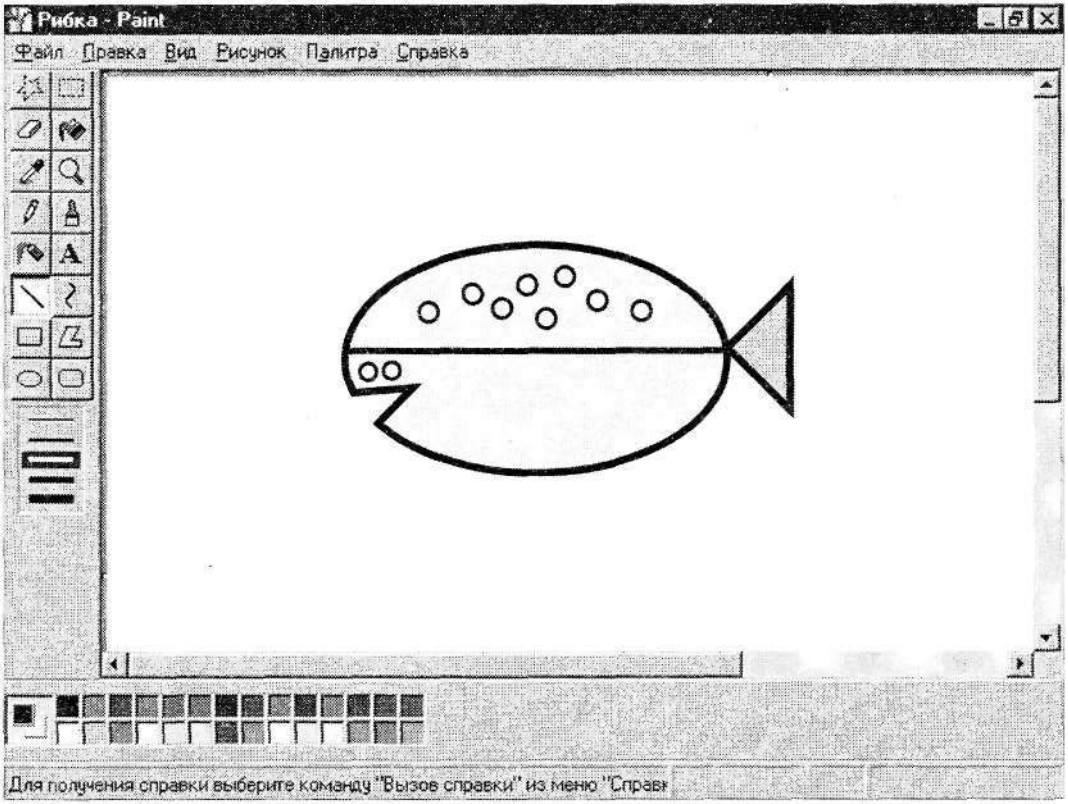

Рис.20. Вигляд вікна графічного редактора з нарисованою картинкою

Рис. 21. Піктографічне меню інструментів графічного редактора

має невелику щільність, а при повільному — більшу.

Баночка з фарбою зафарбовує вибраним кольором замкнену область, для чого на неї потрібно встановити курсор-піктограму і клацнути лівою або правою кнопкою мишки.

Для рисування довільних ліній використовуються олівець та пензлик. Пензлик проводить вибраним кольором лінію вибраної товщини та форми, для чого курсор ставиться в потрібне місце і при натиснутій кнопці мишки переміщується по екрану.

Для побудови відрізків вибирається інструмент, зображений у вигляді відрізка (палички). Курсор встановлюється на початок відрізка, що будується, і при натисненій лівій кнопці мишки переміщується до кінцевої точки відрізка. Якщо при цьому натиснути Shift, то одержимо вертикальні, горизонтальні і діагональні відрізки (під кутом 45°).

Поряд із відрізком знаходиться піктограма інструмента для побудови кривих ліній. Побудовану цим інструментом лінію можна викривити у двох місцях, для чого слід поставити курсор на відрізок або біля нього і, натиснувши кнопку мишки, відтягнути його в потрібному напрямку.

Для зображення прямокутників слід вибрати інструмент, піктограма якого має вигляд прямокутника. Курсор потрібно встановити в одну з вершин прямокутника, що будується, і при натисненій кнопці мишки перемістити його до іншої вершини по діагоналі. Додаткове меню під піктограмами інструментів дозволяє вибирати тип кольорового заповнення прямокутника. Щоб побудувати точний квадрат, треба тримати натиснутою клавішу Shift.

Овали та прямокутники з закругленими кутами зображуються також «з кута», тобто начебто вписуються в прямокутники. Для одержання точного кола або квадрата з закругленими кутами треба також натискати Shift.

У піктографічному меню графічного редактора є інструмент для зображення багатокутників. Для побудови першої сторони багатокутника відмічається курсором перша вершина і при натиснутій лівій кнопці мишки тягнеться лінія до наступної вершини. Відпускання кнопки мишки фіксує відрізок. Потім підводиться курсор до третьої вершини і робиться одноразове клацання. Так само позначаються всі наступні вершини багатокутника. Подвійне клацання на останній вершині замикає фігуру лінією.

Вибравши піктограму з літерою, можна помістити на рисунку текст. Для цього спочатку слід поставити курсор у потрібне місце і, натиснувши ліву кнопку мишки, виділити рамку для вписування тексту. Для набору тексту потрібно помістити курсор усередину рамки, клацнути лівою кнопкою мишки і ввести текст. Розміри рамки з текстом можна збільшувати. Тип і розмір шрифтів вибирають, використовуючи панель атрибутів тексту. Використовуючи додаткове меню під піктограмами інструментів, можна вибирати кольоровий чи прозорий фон.

Вибір кольору. Для вибору кольору зображення необхідно встановити курсор на потрібний колір у меню кольорів у нижній частині екрана і клацнути лівою кнопкою мишки. Вибраний колір з'явиться у верхньому квадратику зліва від меню кольорів. Щоб задати колір фону, треба при виборі кольору з набору кольорів клацнути правою кнопкою мишки. Колір фону з'явиться у нижньому квадратику зліва від меню кольорів.

Змінювати кольори можна також за допомогою інструмента «Піпетка» з піктографічного меню. Для цього слід підвести курсор у вигляді піпетки до погрібного кольору на зображенні та клацнути лівою або правою кнопкою мишки — залежно від того, основний чи фоновий колір вибирається. Якщо при побудові зображень натискати праву кнопку мишки, то замість основного кольору буде використовуватись фоновий колір, а замість фонового — основний.

Коригування зображення. Можна відмінити кілька останніх кроків роботи графічного редактора, вибравши команду «Відмінити» із пункту меню «Правка». Відновити зображення після команди «Відмінити» можна за командою «Повторити».

Використання меню графічного редактора. Робота з пунктами меню редактора здійснюється так само, як і в інших програмах системи Windows. Пункт «Файл» дозволяє зберегти картинку у файлі (команда «Зберегти» або «Зберегти як...»). У діалоговому вікні, що з'явилося, треба ввести ім'я файла з розширенням, яке вказує тип графічного файла. При потребі виділений «ножицями» прямокутний фрагмент рисунка можна зберегти в окремому файлі.

Для коригування існуючого файла його потрібно завантажити до графічного редактора за допомогою команди «Відкрити...» із меню «Файл».

Команда «Атрибути...» меню «Рисунок» дозволяє задавати розмір нового рисунка. Це дає можливість потім при розміщенні рисунка в тексті враховувати його реальні розміри. У цій самій команді можна зазначити, яким буде нове зображення — кольоровим чи чорно-білим.

Для перегляду рисунка використовується команда «Переглянути рисунок» із меню «Вигляд».

ЗАПИТАННЯ І ЗАВДАННЯ

1. ДЛЯ ЧОГО використовується графічний редактор?
2. Які інструменти редактора використовують при створенні картинки?
3. Як можна зобразити лінію?
4. Як домалювати або змінити частини рисунка, які не видно на екрані?
5. Як виділити на рисунку фрагмент довільної форми?
6. Як зробити написи на рисунку?
7. Як зафарбувати замкнену область рисунка?
8. Як намалювати контур довільної форми?
9. Як зобразити вертикальний відрізок?
10. Який колір буде всередині прямокутника, якщо вибрано інструмент «зафарбований прямокутник»?
11. Як у графічному редакторі нарисувати точне коло або квадрат?
12. Як указати редактору, що відмічена вершина багатокутника є останньою при побудові?
13. Чим відрізняється клацання лівої кнопки мишки від клацання правої при виборі кольору з меню кольорів?
14. Як відмінити роботу, виконану з використанням певного інструмента графічного редактора?

15. Як відновити щойно стерте зображення?
16. Як можна переглянути в збільшеному вигляді дрібні деталі рисунка?
17. Як перемістити фрагмент рисунка на інше місце?
18. Як скопіювати фрагмент рисунка?
19. Як можна задати потрібні розміри рисунка?
20. Як побачити рисунок, що не вміщається у вікні редактора?

1.12. Текстовий редактор

Клавіатура комп'ютера дуже схожа на клавіатуру друкарської машинки. Але між цими двома пристроями є істотна відмінність: процес створення документа за допомогою комп'ютера відділений від процесу його друкування. Комп'ютер дає змогу виправляти, перекомпоновувати і вилучати текст у документі без друкування проміжних варіантів.

Програми, за допомогою яких можна створювати та змінювати текст, називаються *текстовими редакторами*. У тих випадках, коли має значення не тільки зміст тексту, а і його зовнішній вигляд, використовують більш складні текстові редактори, які називаються *текстовими процесорами*.

Майже всі текстові процесори можуть виконувати такі основні функції:

1) відображати на екрані і друкувати символи різних накреслень і розмірів, зокрема із такими особливостями, як виділення фрагментів тексту напівжирним шрифтом, підкресленням або курсивом;

2) виконувати пошук визначеної послідовності символів та її заміну. Ця властивість стане в пригоді, якщо, наприклад, потрібно замінити або виправити у великому документі якесь слово або групу слів;

3) автоматично центрувати чи вирівнювати текст по лівому або правому краю, встановлювати інтервали між рядками тощо;

4) автоматично друкувати номери сторінок у верхній чи нижній частині кожної сторінки;

5) формувати та друкувати верхній і нижній колонтитули (колонтитул — спеціальний чи допоміжний текст у верхній чи нижній частині кожної сторінки);

6) форматувати виноски й посилання. Якщо додаються нові виноски чи перекомпоновується текст, то програма автоматично переміщає виноски і змінює їхню нумерацію, коли це необхідно;

7) розташовувати текст у двох чи більше колонках;

8) виконувати пошук орфографічних помилок. Цей процес називається *перевіркою орфографії,* деякі програми навіть мають можливість автокорекції, тобто дозволяють автоматично виправляти помилки одночасно з набором тексту;

9) знаходити синоніми для виділеного слова. Ця можливість дуже корисна в тому разі, якщо в тексті занадто часто вживається те саме слово;

10) вводити спеціальні та математичні символи;

11) створювати маркеровані та нумеровані списки;

12) створювати спеціальні форми документів, наприклад для листів чи написів;

13) вставляти в документ лінії, рамки, малюнки, таблиці тощо;

14) друкувати адреси на конвертах і поштових етикетках.

Система Windows містить текстовий редактор, що дозволяє створювати текстові файли. Після запуску текстового редактора на екрані з'являється вікно з пунктами меню, зміст яких багато в чому збігається з відповідними пунктами меню графічного редактора. Вікно редактора з текстом зображене на рис. 22.

Створення файла. Для створення файла потрібно вибрати в меню «Файл» команду «Створити», а потім вибрати тип файла, який потрібно створити. Після натискання кнопки «ОК» можна починати введення тексту. При набиранні тексту не здійснюється перенесення слів і в кінці кожного рядка не треба натискати на клавішу **Enter.** Редактор переводить курсор на новий рядок автоматично. Клавіша **Enter** натискується тільки в кінці абзацу. Для збереження набраного тексту на диску треба відкрити меню «Файл» і вибрати команду «Зберегти як...». При цьому з'явиться діалогове вікно, в якому треба вказати ім'я файла, а також каталогу та диска.

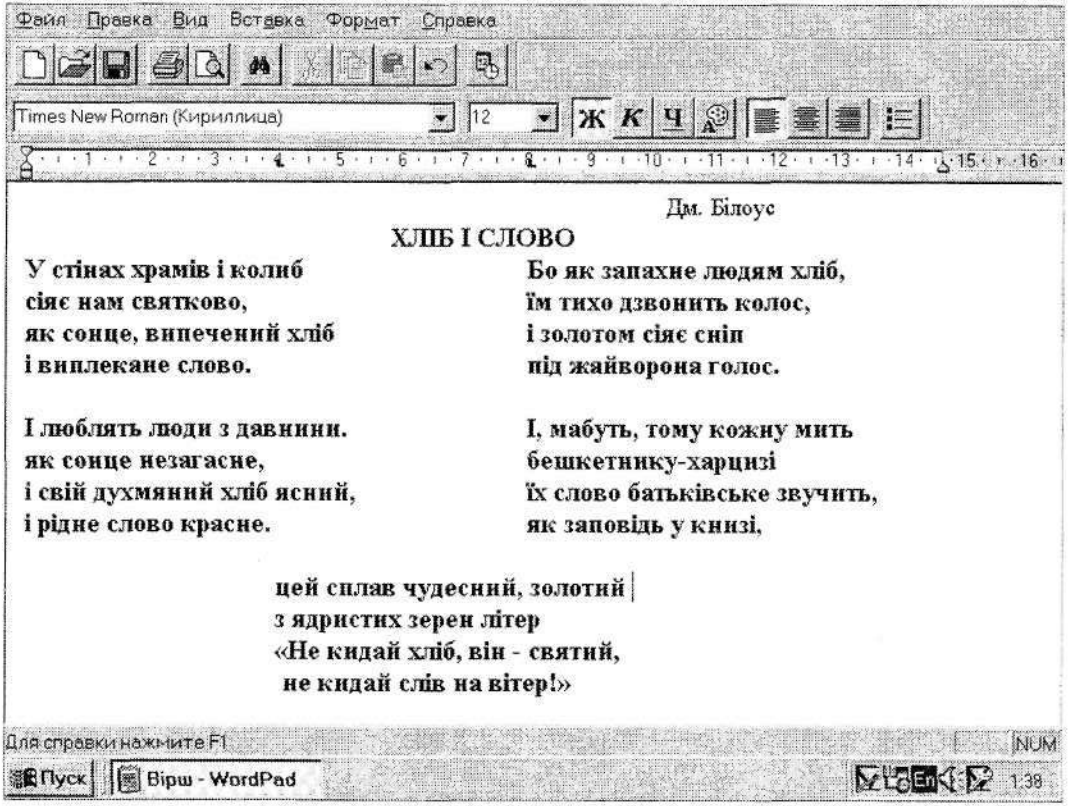

Рис. 22. Вікно текстового редактора

Використання лінійки. На рис. 22 під меню редактора розміщені панелі інструментів та лінійка. Щоб лінійка та інші панелі інструментів були на екрані, потрібно вибрати відповідні команди меню «Вигляд».

На початку лінійки розміщені два вказівники. Верхній вказівник визначає величину абзацного відступу, а нижній — встановлює виступ лівого краю тексту. На правому кінці лінійки знаходиться ще один вказівник, за допомогою якого можна встановлювати виступи тексту справа.

Форматування тексту. Форматування буває кількох видів: визначення розміру рядка для всього тексту — вказуються межі (відступи); вибір символів як за розміром, так і за накресленням (шрифт).

Панель форматування містить кнопки, що прискорюють форматування тексту, наприклад, що змінюють накреслення тексту чи стиль абзацу. Щоб відформатувати текст, потрібно спочатку його виділити, а потім натиснути відповідну кнопку на панелі форматування.

Розташування тексту на сторінці. Стандартний аркуш паперу (формат А4) має такі розміри: ширина — 21 см, довжина — 29,7 см. Поля, тобто відстань від границь тексту до країв аркуша, вказуються в команді «Параметри сторінки» меню «Файл».

Зміна шрифтів. Текст виглядає красиво і з ним зручно працювати, коли окремі його фрагменти, наприклад заголовки, набрані різними за величиною та формою літерами. Для цього в текстовому редакторі є ряд можливостей. Щоб текст був набраний відповідним шрифтом, треба його попередньо виділити. Потім треба увійти до меню «Формат», вибрати відповідний пункт «Шрифт» і клацнуги по ньому, У діалоговому вікні вибирається шрифт, його накреслення і розмір у пунктах (1/72 дюйма), демонструється зразок рядка із символів обраного шрифту. Якщо одержаний варіант шрифту задовольняє користувача, то він повинен клацнути мишкою по кнопці «OK», а якщо ні, йому слід продовжити комбінування чи скористатися кнопкою «Відміна» діалогового вікна команди.

Редагування тексту. Для внесення змін до тексту не за зовнішнім виглядом, а за змістом, використовуються команди меню «Правка». Для вилучення частини тексту її

спочатку треба виділити, а потім виконати команду «Вирізати» пункту «Правка». Фрагмент потрапляє до спеціального буфера, із якого його можна вставити в текст, починаючи з місця, яке вказується курсором. З використанням буфера також виконується команда «Копіювати» (при цьому виділений текст залишається без змін).

Для пошуку необхідного рядка тексту використовується команда «Знайти...», а для заміни одного рядка на інший — команда «Замінити...».

Об'єднання кількох файлів. Для об'єднання текстів, які знаходяться в кількох файлах, також використовується буфер. Для цього потрібно:

1) помістити у вікно редактора текст із першого файла, виконавши команду «Відкрити...» з іменем цього файла, ім'я файла з'явиться в заголовку редактора;

2) виділити весь текст або частину тексту, що переноситься до іншого файла;

3) виконати команду «Копіювати» із меню «Правка»;

4) помістити у вікно редактора текст з іншого файла, виконавши команду «Відкрити...» з іменем цього файла;

5) підвести курсор-стрілку до місця, з якого починатиметься текст, що вставляється, і клацнути лівою кнопкою мишки;

6) виконати команду «Вставити» із меню «Правка».

Створення таблиць. Текст може містити таблиці — рівні стовпці символів без рамок. Для одержання таблиці потрібно дотримуватися таких правил:

1) при набиранні рядків відокремлювати дані стовпців таблиці одним табулятором (натискати клавішу **Tab** у кінці кожного стовпця);

2) виділити частину тексту, що містить таблицю;

3) встановити межі стовпців. Для цього треба встановити курсор мишки на лінійці, де визначається межа стовпця, і клацнути лівою кнопкою, а потім клацанням по лінійці встановити потрібні позиції вирівнювання стовпців (позиції табуляції чи табулятори). При цьому на лінійці з'являються позначки у вигляді кутів, за якими по лівому краю вирівнюються стовпці. Рухаючи стрілку табуляції вздовж лінійки за допомогою мишки, можна перемістити весь стовпець таблиці на нове місце.

Табулятори можна також встановлювати або відміняти, використовуючи команду «Табуляція» із меню «Формат».

Вставляння рисунків у текст. Система Windows дозволяє здійснювати обмін даними між різними програмами за допомогою буфера обміну (спеціальної області в пам'яті комп'ютера). Так, рисунок, зроблений у графічному редакторі, можна помістити в текст, підготовлений за допомогою текстового редактора.

Слід виділити рисунок або його фрагмент у вікні графічного редактора, після чого потрібно виконати такі дії:

1) виділений фрагмент копіюється до буфера за командою «Копіювати» із меню «Правка» (або натисненням клавіш **Ctrl + C);**

2) курсор підводиться до місця вставляння рисунка і виконується команда «Вставити» із меню «Правка».

Рисунок у тексті можна переміщувати вправо і вліво. Можна також змінювати його розміри. Для цього треба підвести до рисунка курсор мишки і клацнути по ньому. Потім, встановивши курсор на границі рисунка і натиснувши ліву кнопку мишки, можна змінювати розмір рисунка.

ЗАПИТАННЯ І ЗАВДАННЯ

1. Які пункти меню текстового редактора використовуються для оформлення зовнішнього вигляду тексту?
2. Як розмістити весь текст на аркуші паперу?
3. Як розмістити заголовок у центрі рядка?
4. Як зробити текст із рівними краями як зліва, так і справа?
5. Як виділити текст?
6. Що таке шрифт?
7. Як можна повернути на попереднє місце щойно вилучений текст? Чому це можливо?
8. Як розмножити частину тексту (наприклад, свою візитну картку)?
9. Як зробити один текстовий файл із двох?
10. Як задати абзацний відступ, використовуючи лінійку?
11. Якою клавішею треба відділяти стовпці при набиранні таблиці?
12. Чи можна пересувати весь стовпець таблиці одночасно?
13. Як підготувати рисунок для вставляння в текст?
14. Створіть текстовий файл, що має вигляд таблиці з прізвищами учнів, їх адресами і телефонами.

15. Наберіть текст своєї візитної картки і відформатуйте його, виділяючи різними шрифтами прізвище, ім'я та по батькові, адресу, місце навчання.
16. Створіть бланк фірми, підготувавши в графічному редакторі емблему шириною 5 см і висотою 3 см. Емблему слід помістити у верхній частині бланка, а під нею - назву фірми, її адресу і телефон. Використайте різні шрифти для кожної частини бланка.
17. Створіть два текстових файли, один з яких містить уривок прозового твору, а другий - вірш. Об'єднайте ці файли в один.

Архівування файлів

З метою забезпечення надійного збереження інформації створюються резервні копії даних. Процес створення резервних копій даних називається *архівацією.*

Збереження великих об'ємів інформації потребує не тільки містких носіїв інформації (магнітних та оптичних дисків, магнітних стрічок тощо), а й значних матеріальних витрат. До певної міри зменшити вимоги до місткості носіїв допомагають спеціальні програми, які дозволяють стискати інформацію. Такі програми називають *архіваторами.*

До основних можливостей сучасних архіваторів належать: занесення цілих груп файлів та підкаталогів до архіву, поновлення та перевірка цілісності архівів, перегляд вмісту архівів, вилучення з них файлів, захист інформації від несанкціонованого доступу, створення багатотомних архівів та архівів, які автоматично розкриваються тощо. Сучасні архіватори дозволяють економити від 10% до 90% дискового простору.

Файлом, який міститься в архіві, можна скористатися лише після того, як його буде відновлено у початковому вигляді, тобто розархівовано.

Розархівацію виконують або ті самі архіватори, або окремі програми, які називають *розархіваторами.*

Серед користувачів персональних комп'ютерів великої популярності набули архіватори PKZIP, ARJ та RAR. Щоб скористатися одним з них, в командному рядку операційної системи слід набрати ім'я програми-архіватора, вказати

потрібні команди та параметри, ім'я архівного файла, а також, при потребі, імена файлів, з якими виконуються дії.

Архіватор PKZIP. Для створення архіву в поточному каталозі, що містить усі файли цього каталогу, потрібно виконати команду

pkzip arh1.zip або pkzip arhl

При цьому отримаємо архівний файл *arhl.zip.* Розширення файла *.zip* у команді можна не вказувати.

Якщо до архіву потрібно занести також файли, що знаходяться в підкаталогах поточного каталогу, то в команді слід вказати додаткові параметри -гр:

pkzip -rp arh2

Якщо в команді вказуються параметри, то замість рисочки (знака «мінус») перед параметром можна використовувати похилу риску («/»).

Щоб відновити файли з ZIP-архіву, потрібно скористатися програмою PKUNZIP:

pkunzip arhl

Якщо ж файли в архіві знаходяться в підкаталогах, то при розархівуванні слід вказати додатковий параметр -d:

pkunzip -d arh2

Програма-архіватор ARJ виконує як архівацію, так і розархівацію. Щоб створити архів поточного каталогу *arhl.arj,* потрібно після імені архіватора вказати літеру а. (Цій літері, що визначає команду архіватора, не повинні передувати «-» або «/».) Якщо до архіву мають увійти також і файли, що знаходяться в підкаталогах, то слід додати параметр -г:

arj a -r arhl

Для відновлення файлів з ARJ-архіву архіватору слід вказати літеру-команду х або є (при х виконується розархівація з розгортанням підкаталогів, а при є всі відновлені файли вміщуються в поточний каталог).

За допомогою архіватора ARJ створюються також багатотомні архіви, які можна розміщувати на окремих носіях.

Так, за командою

```
arj a -v1440 arh3 *.bmp
```

утворюється багатотомний архів графічних файлів типу ,bmp, частини якого можна помістити на дискети місткістю 1,44 Mb.

Ознайомитися з іншими командами та параметрами програм PKZIP, PKUNZIP та ARJ можна, набравши в командному рядку їх імена без додаткових параметрів або вказавши параметр -?.

Багато в чому подібним до ARJ є архіватор RAR. Він зручний; тим, що, крім звичайних можливостей роботи в режимі командного рядка, він має оболонку, дещо подібну до Norton Commander.

Існують Windows-версії розглянутих архіваторів, які називаються відповідно WINZIP, WINARJ та WINRAR. Після запуску такого архіватора на екрані монітора з'являється вікно, у якому можна працювати з архівами. Використовуючи меню і піктограми, які дублюють основні команди меню, можна створювати архіви в різних форматах, розархівовувати потрібні файли тощо.

При виборі конкретного типу архіватора (розархіватора) керуються двома критеріями: швидкістю його роботи та ефективністю стискування даних. При цьому для одних типів файлів кращим може бути один архіватор, а для інших файлів — інший.

ЗАПИТАННЯ І ЗАВДАННЯ

1. Що таке архівація файлів?
2. З якою метою використовують програми-архіватори?
3. Яким чином можна переглянути заархівовану інформацію?
4. Чим відрізняється архіватор RAR від архіватора WINRAR?

1,1 Комп'ютерні віруси

Важко дати загальне і всеохоплююче визначення поняття «комп'ютерний вірус». У деякому наближенні можна вважати, що комп'ютерні віруси — це програми-«паразити», які

можуть включати себе до інших програм та файлів («заражати» їх). Саме ця обставина і дає можливість вірусам розмножуватися і поширюватися. Людина, яка використовує заражену програму або файл, може й гадки не мати, що ця програма містить вірус.

Як і будь-які інші програми, віруси створюються людьми-програмістами. Мотиви для написання вірусів можуть бути найрізноманітнішими: від бажання перевірити свої сили в програмуванні до прагнення нашкодити людству. Законодавство більшості країн передбачає за створення вірусів адміністративну, цивільну і кримінальну відповідальність. Звичайно, якщо автор вірусів не афішує свою діяльність, виявити його дуже важко. Здебільшого автори вірусів залишаються анонімними. Втім, встановити особу автора вірусу та притягти його до відповідальності все-таки вдається.

Віруси, як і інші програми, можуть виконувати практично будь-які дії. Деякі віруси не приносять майже ніякої шкоди, а лише розмножуються. Інші, порівняно безпечні віруси, видають на екран відволікаючі повідомлення. Однак існують шкідливі віруси, які спричиняють «зависання» програм (припинення роботи програм) або несподівані перезавантаження комп'ютера. Нарешті, найнебезпечніші віруси можуть псувати або знищувати інформацію, яка зберігається на дисках.

Віруси класифікують, використовуючи наступні ознаки:

1) середовище, в якому знаходяться віруси;

2) спосіб зараження середовища, в якому знаходяться віруси;

3) спосіб активізації;

4) деструктивні можливості;

5) особливості алгоритму дії.

Виділяють такі основні типи вірусів, в залежності від середовища, в якому вони знаходяться: файлові, завантажувальні та мережні.

Файлові віруси приєднуються до файлів, як правило, вбудовуючи до них свій код. Якщо запустити заражений файл на виконання, вірус потрапляє до оперативної пам'яті комп'ютера і починає заражати інші файли, псувати програми та дані, спотворювати результати роботи програм.

Завантажувальні віруси розміщують інформацію для свого запуску в завантажувальних секторах дисків. Вони потрапляють до комп'ютера при завантаженні з зараженого диска чи дискети.

Мережеві віруси використовують для свого поширення засоби комп'ютерних мереж і електронної пошти.

Існують віруси, які належать одночасно до кількох груп. Так, наприклад, більшість завантажувальних вірусів можуть поширюватись як звичайні файлові віруси.

Спосіб зараження середовища, в якому мешкають віруси, тісно пов'язаний із структурою середовища, тобто залежить від типу середовища, яке заражується. Досить часто файловий вірус впроваджується в кінець файла, але може також розміщуватися на початку, середині файла. Якщо мова йде про завантажувальний вірус, то досить часто частина вірусу розміщується замість завантажувального сектора диска чи сектора системного завантажника, а решта вірусу і завантажувальний сектор розміщуються в інших секторах.

По способу активізації віруси ділять на резидентні та нерезидентні. Резидентні віруси залишають в оперативній пам'яті резиденту частину, яка під час звертання операційної системи впроваджуються до об'єктів зараження — завантажувальних секторів, файлів. Ці віруси зберігають свою активність протягом всього часу роботи комп'ютера. Нерезидентні віруси є активними у визначені моменти: під час обробки документів текстовим процесором, в момент запуску заражених програм тощо. Бувають випадки коли деякі нерезидентні віруси залишають в оперативній пам'яті невеликі резидентні програми.

По деструктивних можливостях віруси діляться на нешкідливі, безпечні, небезпечні і дуже небезпечні.

Нешкідливі віруси — зменшують обсяг пам'яті на диску в результаті свого поширення.

Безпечні віруси, крім зменшення обсягу пам'яті на диску, запроваджують графічні, звукові й інші ефекти.

Небезпечні віруси можуть привести до порушень нормальної роботи комп'ютера, наприклад до зависання або до неправильного друку документа. Дуже небезпечні віруси можуть привести до втрати програм і даних, стиранню

інформації в системних областях пам'яті і навіть приводити до виходу з ладу частин жорсткого диска. За особливостями алгоритмів розрізняють наступні віруси: супутники, черв'яки, паразитичні, студентські, невидимки, примари.

Віруси-супутники файли не змінюють, а виконуються раніше вихідної програми, а потім передають керування вихідній програмі.

Віруси-черв'яки поширюються в комп'ютерних мережах, обчислюють адреси мережних комп'ютерів і створюють там свої копії.

Паразитичні віруси можна легко виявити, оскільки вони міняють зміст дискових секторів та файлів.

Студентські віруси є найпростішими вірусами, а от віруси-невидимки досить важко виявити. Вони перехоплюють звертання операційної системи до уражених файлів і секторів дисків та підставляють незаражені ділянки диска, що заважає їх виявленню.

Віруси — примари мають зашифроване за допомогою алгоритмів шифрування-розшифрування тіло вірусу, завдяки чому дві копії одного вірусу не мають однакових ділянок коду.

Боротьба з комп'ютерними вірусами. Програми, призначені для виявлення і видалення комп'ютерних вірусів, називаються *антивірусними.* Для боротьби з комп'ютерними вірусами розроблено досить велику кількість спеціальних антивірусних програм, або антивірусів. Найбільш популярними є так звані *програми-сканери* (інша назва — вірусні детектори).

Попередній аналіз відомих вірусів дозволяє виділити послідовності кодів, характерних для кожного з них. Принцип роботи антивірусних сканерів полягає в пошуку цих кодових послідовностей у файлах, завантажувальних секторах і оперативній пам'яті. Якщо сканер знаходить таку послідовність, він повідомляє про зараження. Така перевірка дозволяє виявляти лише відомі віруси і не допомагає проти невідомих. Тому, якщо сканер повідомляє про відсутність вірусів, це ще не означає, що їх справді немає. Це зумовлює необхідність постійного оновлення програм-сканерів (нові версії деяких сканерів з'являються майже щотижня).

У сканерах використовуються також алгоритми «евристичного сканування», які виявляють фрагменти програм, поведінка яких може бути подібною до поведінки вірусів. Дуже часто це дозволяє дійти висновку про зараження новими, ще невідомими вірусами. У багатьох випадках антивірусні програми здатні «лікувати» заражені файли, вилучаючи з них вірусний код.

Серед найбільш відомих антивірусних сканерів можна назвати DrWeb, AVP, Aidstest, McAfee Virus Scan, Norton Antivirus, IBM Anti-Virus та ін.

Досить ефективними є також антивірусні *програми-ревізори,* в тому числі так звані CRC-сканери. Зараження файла вірусом призводить до зміни цього файла. Програма-ревізор контролює будь-які зміни файлів і у разі їх виявлення повідомляє про можливість вірусного зараження. Відомою програмою-ревізором є ADINF.

Не існує універсального засобу боротьби з вірусами. Але потрібно знати і виконувати хоча б основні правила антивірусного захисту, які істотно зменшують ризик зараження, а також можливі втрати від вірусів.

1. Насамперед слід робити резервні копії своїх даних. Це дозволить відновити інформацію не тільки у випадку пошкодження вірусами, а й у разі механічного псування дисків і т.п.

2. Нові файли, які заносяться до комп'ютера, повинні перевірятися антивірусною програмою, особливо це стосується програм, які будуть запускатися на виконання файлів текстових процесорів та електронних таблиць тощо.

3. Необхідно регулярно оновлювати антивірусні програми.

4. Якщо не планується завантаження саме з дискети, при ввімкненні або перезавантаженні комп'ютера ні в якому разі не можна залишати дискети в дисководі.

5. Якщо є підозра на зараження, слід якнайшвидше починати лікування (знищення вірусів у пам'яті, в завантажувальних секторах і у файлах). Рекомендується навіть завантажитися з системної дискети (звичайно, сама ця дискета гарантовано повинна не містити вірусів) і запустити антивірусну програму саме з цієї дискети.

6. Під час роботи з файлами в текстових редакторах збереження файлів у форматі RTF дозволяє запобігати їх зараженню вірусами.

ЗАПИТАННЯ І ЗАВДАННЯ

1. Що таке комп'ютерний вірус?
2. Звідки беруться віруси і яким чином вони можуть поширюватися?
3. Наведіть відому вам класифікацію вірусів.
4. До яких наслідків можуть призводити віруси?
5. Назвіть відомі вам антивірусні програми.
6. Поясніть принцип дії вірусних сканерів. Чи можуть вони виявляти та знешкоджувати невідомі віруси?
7. Чому антивірусні програми постійно оновлюються?
8. Назвіть основні принципи антивірусного захисту.

Інформаційні системи

Розв'язання практичних задач у будь-якій галузі діяльності людини потребує опрацювання великої кількості даних. Кваліфікація фахівця визначається двома показниками: як багато він знає фактів і як багато він знає правил щодо використання цих фактів. Факти являють собою дані та їх комбінації, вони можуть бути впорядковані за різними ознаками, описані, названі унікальними іменами і вміщені до комп'ютерної бази даних. Факти об'єднуються за допомогою правил, які встановлюють залежності між фактами, а також види цих залежностей. Правило може бути одержане з наукової теорії або сформульоване на основі професійного досвіду однієї людини або групи людей. Сукупність правил являє собою *базу знань,* яка може зберігатися в комп'ютері у вигляді наборів даних і програм. Бази знань і бази даних служать для одержання різних відомостей, які потім використовуються для прийняття рішень в управлінні, постановці діагнозів, навчанні та інших видах діяльності людини.

Інформаційна система являє собою разом з відповідним апаратним забезпеченням систему комп'ютерних програм, що використовуються для аналізу даних і залежностей між ними з метою отримання різних відомостей. Основною частиною інформаційної системи є база даних. *База даних* — це сукупність даних і зв'язків між ними.

Розглянемо види зв'язків між даними.

І. Зв'язок «один до одного». Найпростіший спосіб показати зв'язок між даними — це помістити дані разом. Наприклад, в одному рядку розмістити відомості про працівника: прізвище, посаду та зарплату. Усі рядки такої відомості мають одну структуру, в стовпцях знаходяться дані одного типу. Кожний рядок як елемент відомості і як дані про одного працівника є єдине ціле і може бути вміщений у файл як окремий запис. Така база даних називається *двовимірним файлом.* В одному записі файла може зберігатися велика кількість відомостей про певну людину чи який-небудь об'єкт або явище. Не завжди всі ці відомості потрібні, здебільшого потрібні всього лише деякі з них. Тому весь запис як одиницю обміну між файлом на диску і пам'яттю комп'ютера зчитують до оперативної пам'яті, а потім вибирають з нього необхідні дані. Для зручності вибору дані кожного стовпця мають свої імена і називаються *полями запису.* Кожний запис, щоб його можна було швидко знайти серед інших такого самого типу, також має унікальне ім'я — *ключ запису,* яке зберігається в першому його полі. Отже, при розробці бази даних спочатку вивчаються необхідні для зберігання відомості, зв'язки між об'єктами, визначаються об'єкти та їх властивості, значення атрибутів. Потім дані систематизуються за типами, визначаються необхідні місця даних в запису, імена одержуваних полів. Можлива структура записів представлена в таблиці 1.

При розробці бази даних для отримання відомостей про робітників підприємства виділяються об'єкти — робітники, їх властивості — професія і освіта, атрибути — посада і зарплата. Для однієї і тієї самої професії може бути кілька можливих посад, наприклад продавець, старший продавець, завідувач відділу магазину. Посада, як значення атрибута «посада», є елементом даних, який належить зберігати в базі даних. Сума зарплати, що відповідає посаді, є значенням іншого атрибута — «зарплата» і являє собою елемент даних іншого поля запису для об'єкта, що обробляється. Ключем запису, поданого в таблиці 1, може бути прізвище та ініціали робітника. Однак на практиці ключем часто служить унікальний номер запису, наприклад табельний номер працівника або шифр об'єкта.

Таблиця 1

Ім'я поля запису	Прізвище	Посада	Зарплата
Тип даного поля	Рядок символів	Рядок символів	Число
Екземпляр запису	Іваненко І.А.	Бухгалтер	250

2. Зв'язок «один до багатьох». Кожному елементу, що розглядається як об'єкт, властивість чи атрибут в інформаційній системі, може відповідати кілька інших об'єктів, властивостей або атрибутів. Подібна структура мас кілька рівнів. Кожний її елемент може бути зв'язаний з кількома іншими, що знаходяться на нижчому рівні, і тільки з одним — з вищого рівня. Така структура називається *деревовидною* або *деревом.* Деревовидну структуру утворюють каталоги, підкаталоги і вміщені в них файли (рис. 9). Таку саму структуру мають описи практично всіх підприємств, міністерств, відомств, військових частин тощо (рис. 23). Для подання дерев у пам'яті комп'ютера існують різні підходи.

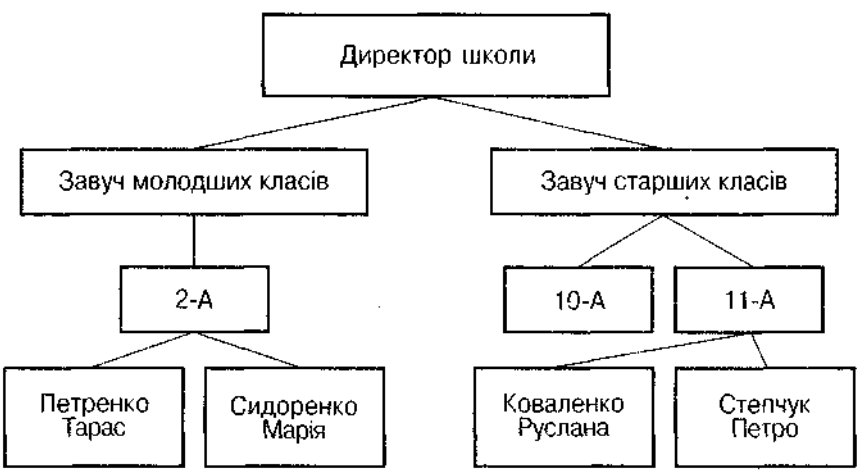

Рис. 23. Деревовидна модель

Пошук даних, об'єднаних у деревовидні структури, відбувається набагато швидше, ніж послідовний перегляд усіх даних. Бази даних з деревовидною структурою називаються *ієрархічними*. Дані, що зберігаються в них, можна переглядати по-різному: рухатися від верхньої точки дерева (кореня) вниз по гілці до кінцевої точки (листка дерева), переглядати дані одного рівня, здійснювати перехід від даних нижчого рівня до вищого. Можливості використання бази залежать від її призначення.

3. Зв'язок «багато до багатьох». Відносини між підприємствами часто мають складний характер, який відображається в мережевих структурах. *Мережа* — це багаторівнева структура, кожний елемент якої може бути пов'язаний з кількома елементами різних рівнів. Наприклад, магазин одержує товари від кількох постачальників. Той самий товар можуть постачати різні постачальники, водночас один постачальник може постачати кілька товарів. Якщо розглядати зв'язки товар — постачальник, то вони являють собою мережу. Мережева модель отримується також при розгляді зв'язків учитель — клас (рис. 24).

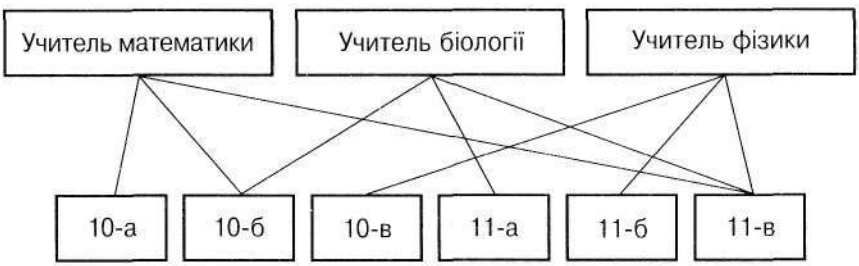

Рис. 24. Мережева модель: один учитель веде свій предмет у різних класах, в одному класі уроки ведуть різні вчителі

Реалізовувати мережеві моделі баз даних набагато складніше, ніж деревовидні. Щоб мережева база даних дозволяла відобразити всі зв'язки, вводять додаткові дані. У теорії баз даних розглядаються загальні вимоги до баз, серед яких — мінімальний надлишок даних. У мережевій моделі ця вимога практично завжди порушується, оскільки для швидкого пошуку створюються додаткові файли, які явля-

ють собою або реалізацію зв'язків «один до багатьох», або, найчастіше, «один до одного». Мережева модель використовується і при поданні знань, і при розробці інформаційних систем з гіпертекстом — коли за виділеним в тексті словом (як у меню розглянутих раніше програмних засобів) можна перейти до іншого тексту, а там знову або використати виділені слова, або виділити курсором необхідне, і перейти до нового тексту або до рисунка з такими самими виділеними елементами. Так побудовані геоінформаційні системи, що дозволяють виділяти на поданій на екрані карті географічні об'єкти (міста, гори тощо) і одержувати текстову інформацію про них або докладнішу карту.

4. Реляційні бази даних. *Реляційною* називається база даних, в якій всі дані, що доступні користувачу, мають вигляд таблиць, а всі дії з даними зводяться до дій з таблицями. Таблиці можна «розрізати» на стовпці або групи стовпців і по-різному їх «склеювати». Такий підхід дозволяє реалізувати і деревовидні, і мережеві моделі. Кожному зв'язку або послідовності зв'язків відповідає рядок таблиці. Так, для рис. 23 — чотирирівневої деревовидної структури — таблиця буде складатися з чотирьох стовпців і стількох рядків, скільки учнів у школі. У відповідному рядку для кожного учня буде вказано прізвище директора, прізвище завуча і клас. Для рис. 24 — дворівневої мережі — таблиця матиме два стовпці (якщо не вказувати окремо прізвище вчителя) і стільки рядків, скільки зв'язків між рівнями мережі, тобто 9 рядків: для кожного з шести класів треба перелічити прізвища вчителів математики, фізики і біології, які пов'язані з даним класом. Таку таблицю можна побудувати у двох варіантах: перелічити в першому стовпці прізвища вчителів, а в другому — класи (таблиця 2); в першому стовпці вказати класи, а в другому — прізвища вчителів (таблиця 3).

Запит до бази даних. Інформаційні системи поділяються на документальні і фактографічні.

Фактографічні системи відповідають на конкретні питання, видаючи відомості про об'єкти в різних комбінаціях. У запитах до фактографічної системи точно вказано, які властивості об'єктів є критерієм відбору з бази даних. Як правило, видається список об'єктів із вказаними в запиті властивостями. Так, використовуючи базу, представлену

Таблиця 2

Клас	Предмет	Прізвище
10-А	Математика	Іванов
10-Б	Математика	Іванов
10-Б	Біологія	Сидоров
10-В	Біологія	Сидоров
11-А	Фізика	Петров
11-Б	Фізика	Петров
11-В	Математика	Іванов
11-В	Біологія	Сидоров
11-В	Фізика	Петров

Таблиця 3

Прізвище	Предмет	Клас
Іванов	Математика	10-А
Іванов	Математика	10-Б
Іванов	Математика	11-В
Петров	Фізика	11-А
Петров	Фізика	11-Б
Петров	Фізика	11-В
Сидоров	Біологія	10-Б
Сидоров	Біологія	10-В
Сидоров	Біологія	11-В

таблицями 2 і 3, можна отримати такі відомості: в яких класах викладає вчитель Петров; які предмети вивчаються в 11-му класі і хто їх веде; список класів, про які є інформація в системі; список навчальних предметів; які предмети викладає кожний учитель. Оформляти такий запит можна як пункт меню, як послідовність вибору з кількох меню, наприклад як пошук потрібного файла через входи до каталогу і підкаталогів. Також може бути вказаний список конкретних питань, які можна задати системі, і користувач або вибирає питання, або вводить його як команду. Сучасні інформаційні системи дають користувачеві можливість формувати і вводити запит у різних формах. Так, у комп'ютерних енциклопедіях можна: використовувати атлас з позначеними географічними пунктами і знаходити за цими пунктами статті про відповідні країни; будувати дерево пошуку за розділами науки і техніки для швидкого знаходження матеріалу; переглядати історичні події в часі з переходом до потрібної статті через піктограму; відбирати матеріал з різними засобами представлення інформації (відео, анімації, звуку і зображень); виходити в режим автоматичного перегляду всіх рисунків і таблиць із зупинкою за вимогою користувача і переходом до статті, що відповідає зображенню; переходити до словника для коленого слова статті і ознайомлюватися з правилами граматики для цього слова та прикладами його використання.

Документальні системи застосовуються, наприклад, в бібліотеках. У таких системах для кожного документа, що в них зберігається (статті, книги, журнали), використовується набір ключових слів.

Користувач формує запит у вигляді набору ключових слів, причому може вказати, що всі слова мають бути в одному документі, або хоча б одне з них, або якогось слова не повинно бути.

Документальні системи ґрунтуються на мережевих базах даних, оскільки те саме ключове слово може міститися в кількох документах, а самі документи містять список кількох слів. Для таких громіздких систем використовується багаторівневий пошук, як в каталозі бібліотеки, коли в розділі «математика» містяться підрозділи з алгебри, геометрії, а потім — окремі розділи цих наукових дисциплін. Такі розділи є в систематичному каталозі, де книги упорядковані за тематикою. Водночас алфавітний каталог дозволяє відразу знайти книгу за прізвищем автора. В автоматизованих бібліотеках, які під'єднані до комп'ютерних мереж, може бути видана відповідь на запит, отриманий від користувача з домашнього комп'ютера. Таким чином, не виходячи з дому, можна працювати з фондами головних бібліотек світу.

ЗАПИТАННЯ І ЗАВДАННЯ

1. Чим відрізняються факти від правил?
2. Що таке база даних?
3. Назвіть види зв'язків між даними.
4. Що таке запис файла, з чого він складається? Як зберігаються дані у двовимірному файлі?
5. Яка структура називається деревовидною?
6. Чим відрізняється деревовидна структура від мережевої?
7. Наведіть приклад організації, для якої може бути створена ієрархічна база даних.
8. Побудуйте деревовидну модель свого учбового закладу. Які питання можна знайти відповіді за допомогою інформаційної системи, заснованої на цій моделі?
9. Побудуйте таблиці реляційної бази, що відповідає створеній деревовидній моделі п. 8. Від чого залежить кількість рядків в таблиці?

10. Розробіть структуру запису для даних, що містяться про кожного учня в класному журналі. Якою вона буде, якщо до цих даних приєднати відомості про батьків?
11. Який вигляд може мати запит до фактографічної інформаційної системи?
12. Як організовані дані в документальних інформаційних системах?
13. Які ключові слова можуть характеризувати зміст даного параграфа?

Електронні таблиці

При розв'язанні задач обробки даних зручно подавати дані у вигляді таблиць, які складаються з рядків і стовпців. Кожний стовпець заповнюється даними одного типу, наприклад, в один стовпець можна записати список товарів, а в другий — ціни для кожного з них. Вигляд таблиці має і «Відомість на зарплату». Деякі стовпці можуть розраховуватися за формулами з використанням даних з інших стовпців. Наприклад, при обчисленні суми виплати необхідно від зароблені суми відняти податок, і зробити це потрібно для даних усього стовпця. Часто потрібно обчислити суму чисел усього стовпця. Подібні задачі зручно розв'язувати за допомогою програм, які називаються *електронними таблицями*. Сучасні електронні таблиці дозволяють також ілюструвати дані та результати їх обробки у вигляді графічних зображень — графіків, різнотипних діаграм тощо.

Існує кілька розповсюджених програм — електронних таблиць для персональних комп'ютерів. Одна з них, яка використовується із системою Windows, називається Excel. Зручність подання даних, різноманітні можливості обчислень та графічного подання інформації в Excel дозволяють користувачеві ефективно проводити кількісну та якісну оцінку даних і допомагають у прийнятті рішень. Вигляд вікна Excel наведено на рис. 25.

Етапами опрацювання таблиці є її створення, розрахунок рядків і стовпців, побудова графіків для ілюстрації даних, що є в таблиці.

Створення таблиці. При запуску програми обробки електронних таблиць на екрані з'являється вікно із заголовком «Книга 1», рядками меню, інформаційним рядком і таб-

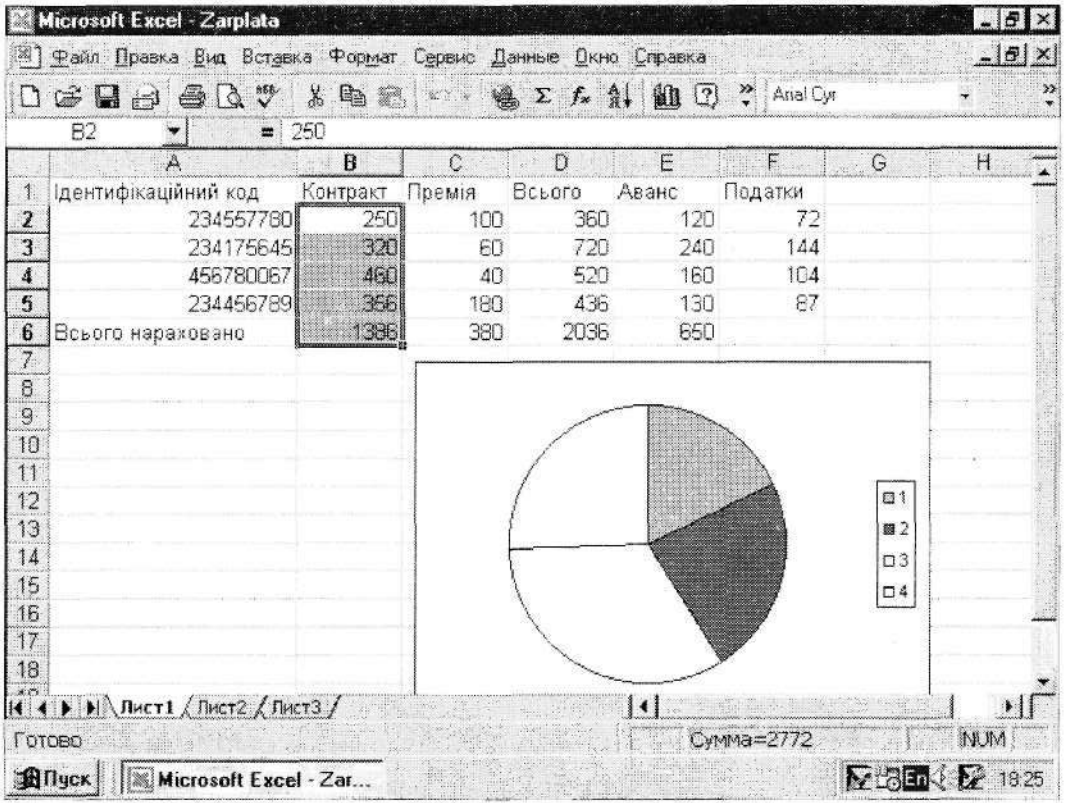

Рис. 25. Вікно табличного процесора Excel

лицею під іменем «Лист 1». Ім'я файла «Книга 1» може змінитись після виконання команди «Зберегти як ...» меню «Файл», якщо вказується інше ім'я файла. Під інформаційним рядком розміщений рядок з іменами стовпців. Імена стовпців — букви латинського алфавіту. Зліва в таблиці вказані номери рядків. На рис. 25 показано фрагмент таблиці. У деяких версіях системи Excel кількість стовпців може сягати 256, а рядків — до 16000. На перетині рядків і стовпців знаходяться комірки таблиці. Кожна комірка має своє ім'я, яке складається з імені стовпця та імені рядка. На рис. 25 виділені комірки, перша з яких має ім'я B2. Це ім'я вказується в інформаційному рядку, який знаходиться відразу над таблицею.

Для занесення інформації в комірку до неї треба підвести курсор (стрілка повинна бути всередині комірки) і клацнути лівою кнопкою мишки. Переміщення по таблиці можна здійснювати також за допомогою клавіш управління курсором. Вибрана комірка відмічається рамкою, після чого до неї можна вводити текст або число. У першому рядку таблиці, яка зображена на рис. 25, розміщені назви стовпців. При потребі ширину стовпця можна збільшити за допомогою команди «Ширина стовпця» із меню «Формат», в діалоговому вікні якої вказується потрібний розмір.

Розрахунок даних у комірках. Щоб у певній комірці отримати значення, яке залежить від даних в інших комірках, до неї треба ввести формулу, за якою здійснюється розрахунок. Введення формули починається зі знака « = » (дорівнює). В інформаційному рядку з'являється формула, в якій можуть бути використані арифметичні операції « + » — додавання, « —» — віднімання, «*» — множення і «/» — ділення, числа, імена комірок. У формулі можуть використовуватись також деякі функції, наприклад, підсумовування SUM, тригонометричні функції, квадратний корінь SQRT і багато інших.

Ілюстрація даних. Програма обробки електронних таблиць дозволяє проілюструвати дані у вигляді різних графіків, кругових і стовпчастих діаграм. Для побудови ілюстрацій необхідно передусім виділити потрібні дані. Потім курсор мишки потрібно підвести до піктограми «Майстер діаграм» і клацнути лівою кнопкою мишки. З'являється вікно, в яко-

му потрібно вибрати тип та вид діаграми, наприклад, кругову діаграму, яка складається із секторів (рис. 25). Для підтвердження слід вибрати «Далі». Потім вказуються, які будуть підписи і чи треба вводити додаткові відомості (наприклад, назва фірми з рядка, назва відповідного стовпця тощо), а також місце, де буде розміщене графічне зображення. Коли вибрані всі необхідні параметри діаграми, потрібно клацнути по кнопці «Готово».

ЗАПИТАННЯ І ЗАВДАННЯ

1. У програмі Excel введіть таблицю 4.

Таблиця 4

	A	B	C	D	E
	Об'єм продаж	Ціна	Собівартість	Прибуток	Чистий прибуток
1	14007	56	15,46	=(B2-C2)*A2	
2	5843	32	15,89		
3	12981	45	13,46		

2. Розрахуйте загальний прибуток (табл.4).
3. Побудуйте кругові діаграми споживання різних продуктів кожним членом сім'ї. Підпишіть кожний сектор діаграми.

1.17, Системи управління базами даних

При проектуванні бази даних, насамперед, вирішуються дві задачі: організація логічної структури всієї бази даних, що визначає типи зв'язків між даними, і організація структури самих даних. Логічна структура бази даних також називається її *логічним представленням* і може бути таблицею, деревом чи мережею. Крім логічного існує також фізичне представлення бази даних. *Фізичне представлення* бази — це спосіб розміщення її у пам'яті комп'ютера, як

правило, на жорсткому магнітному диску. Для розроблювача бази даних і її користувача не є істотним, у якому місці диска зберігається той чи інший запис, як саме розташовані дані в комп'ютері. Наприклад, таблиця може зберігатися повністю (кожен її рядок є окремим записом файла) чи за стовпцями (кожен стовпець — у вигляді окремого файла). Головне при цьому — забезпечити ті зв'язки, що відбиті в логічній структурі бази даних.

Програма, що дозволяє забезпечити зв'язок між фізичними даними і їхнім логічним описом, а також обробку даних, називається *системою управління базами даних* (СУБД).

Існує кілька вимог до баз даних, основними з яких є несуперечливість даних і їх безпека. База даних може використовуватися різними користувачами, однак доступ до операцій виправлення даних, занесення нових даних і їх видалення повинна мати одна людина, або обмежена кількість користувачів. Для захисту даних використовується так званий «замок таємності», який може зберігатися в довільному записі, а також може вводитися як пароль на початку роботи з базою даних.

Сучасні СУБД дозволяють швидко описати структуру даних — запис і його поле, створити таблицю (двовимірний файл) і організувати пошук і сортування даних. Однак для опису складної структури доводиться застосовувати спеціальні засоби, наприклад, представляти мережеву модель (рис. 24) у вигляді таблиць прямого й зворотного перегляду (таблиці 2 і 3).

Одною із розповсюджених СУБД на персональних комп'ютерах є система управління базами даних Access. Вона дозволяє створювати таблиці, що містять записи у вигляді рядків цієї таблиці, а також зв'язувати таблиці між собою через однойменні поля.

Перш ніж приступити до створення таблиці, необхідно визначитися із змістом і типом кожного її стовпця і дати стовпцям назви. Наприклад, формується список працівників підприємства. Відповідна таблиця буде складатися з наступних стовпців: ім'я (прізвище, ім'я, по батькові), посада, рік народження, заробітна плата. Типи даних у полях — текстовий, дата/час, числовий. Останні два поля можуть викорис-

товуватися для обчислень, тому їх краще визначити як поля спеціального виду.

Для створення бази даних за допомогою СУБД Access необхідно виконати наступні дії:

1) ввійти до СУБД Access через список програм у меню «Програми» після натискання клавіші «Пуск»;

2) у меню «Файл» вибрати пункт «Створити» (або в додатковому вікні Access відмітити пункт «Нова база даних» і натиснути Enter);

3) у вікні «Файл нової бази даних», що з'явилося, написати ім'я файла і клацнути по кнопці «Створити»;

4) у вікні «База даних» із введеним іменем бази натиснути кнопку «Створити»;

5) у вікні «Нова таблиця» вибрати пункт «Конструктор», натиснути «ОК»;

6) для кожного поля в першому стовпчику необхідно ввести його ім'я (для першого — слово *«Ім'я»,* для другого — *«Посада»,* для третього — *«Рік народження»,* для четвертого — *«Зарплата в гривнях»),* в другому — тип даних (можна вибрати із запропонованого списку, відповідно — *«Текстовий», «Дата/Час», «Числовий»}.* У третьому стовпці можна ввести коментар (наприклад, для першого поля — «Прізвище, ім'я та по батькові»). При введенні назв полів, їхніх типів і коментарів слова в лапки не треба брати;

7) при створенні формату чи запису після формування всього запису в цілому молена встановити поле первинного ключа, по якому буде зроблено пошук запису. Для цього треба встановити курсор на відповідне поле, клацнути правою кнопкою мишки й у головному меню екрана «Виправлення» виконати команду «Ключове поле». Біля імені поля з'явиться піктограма «ключик»;

8) коли всі поля запису визначені, ввійти в меню «Файл» і обрати команду «Зберегти». При цьому у вікні, що з'явилося, можна ввести назву таблиці, наприклад *«Список робітників»* (рис. 26).

Можна також виконати збереження, клацнувши на піктограмі з дискетою на головній панелі інструментів, ввести ім'я таблиці і натиснути кнопку «ОК».

Після того як задані параметри для кожного поля, можна приступити до заповнення таблиці даними — формувати

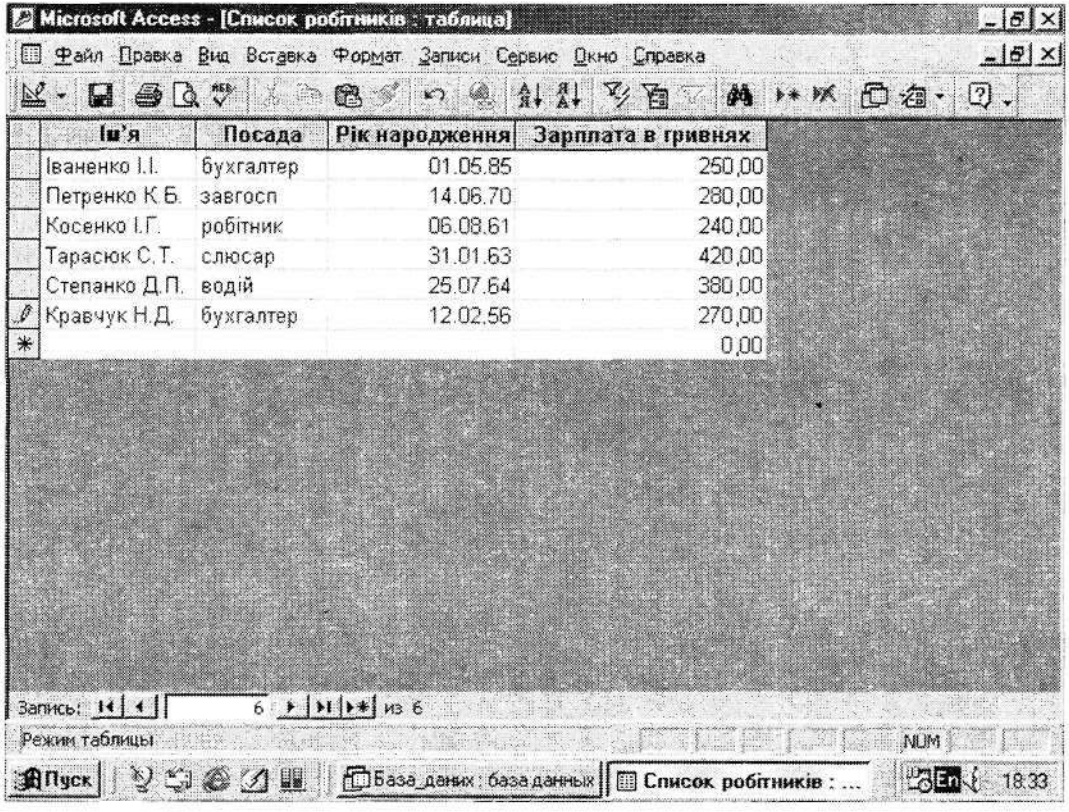

Рис. 26. Вид бази даних після заповнення

власне базу даних. Для цього слід закрити вікно з формуванням полів, натиснувши на кнопку «х» і в новому вікні з ім'ям таблиці натиснути «Відкрити». Таблиця заповнюється фактичними даними, перехід від одного стовпця до іншого здійснюється за допомогою клавіші табуляції. Коли всі дані введені, вікно введення треба закрити, при цьому дані зберігаються в базі.

При створенні бази поле *«Посада»* було відмічене як поле первинного ключа — зображенням ключика. Це означає, що вміст цього поля повинен бути унікальним, не містити повторюваних значень. Тому не можна в даному прикладі записати в базу двох «бухгалтерів». Як правило, первинним ключем є ім'я працівника або порядковий номер працівника в базі (на великих підприємствах два працівники можуть мати однакові прізвища та ініціали, тому ключем є табельний номер працівника). Для зміни первинного ключа й закріплення його за полем *«Порядковий номер»* треба запустити СУБД Access, виділити назву бази даних *«Список співробітників»* у вікні «Таблиця» і вибрати пункт «Конструктор». У структурі бази, що з'явилася, додати поле *«Порядковий номер»* і вибрати тип даних «Лічильник», а також відмітити це поле як поле первинного ключа. При закритті вікна «Конструктор» з'являється питання, чи зберегти зміни в структурі, на яке треба дати позитивну відповідь, При новому відкритті таблиці з даними бази в додатковому полі будуть проставлені порядкові номери записів. Вигляд модернізованої бази даних представлений на рис. 27.

При роботі з базою даних можна формувати різні запити. Наприклад, ми хочемо одержати список бухгалтерів підприємства, який містить імена працівників і величину їхньої заробітної плати. Для цього у вікні з базою даних вибираємо пункт «Запит» (відповідне вікно виходить на перший план) і натискаємо кнопку «Конструктор». У нижній частині вікна, що з'явилося, у графі *«Поле»* відмічаємо поле, за яким буде відбуватися відбір, а в цьому самому стовпчику внизу в графі «значення» — набираємо слово «бухгалтер» (без лапок, програма сама обмежить його лапками). Потім у першому рядку в графі *«Поле»* зазначаємо ті відомості, які хочемо побачити в результаті запиту; у другому стовпчику графи *«Поле»* вибираємо назву поля *«Ім'я»,* а в

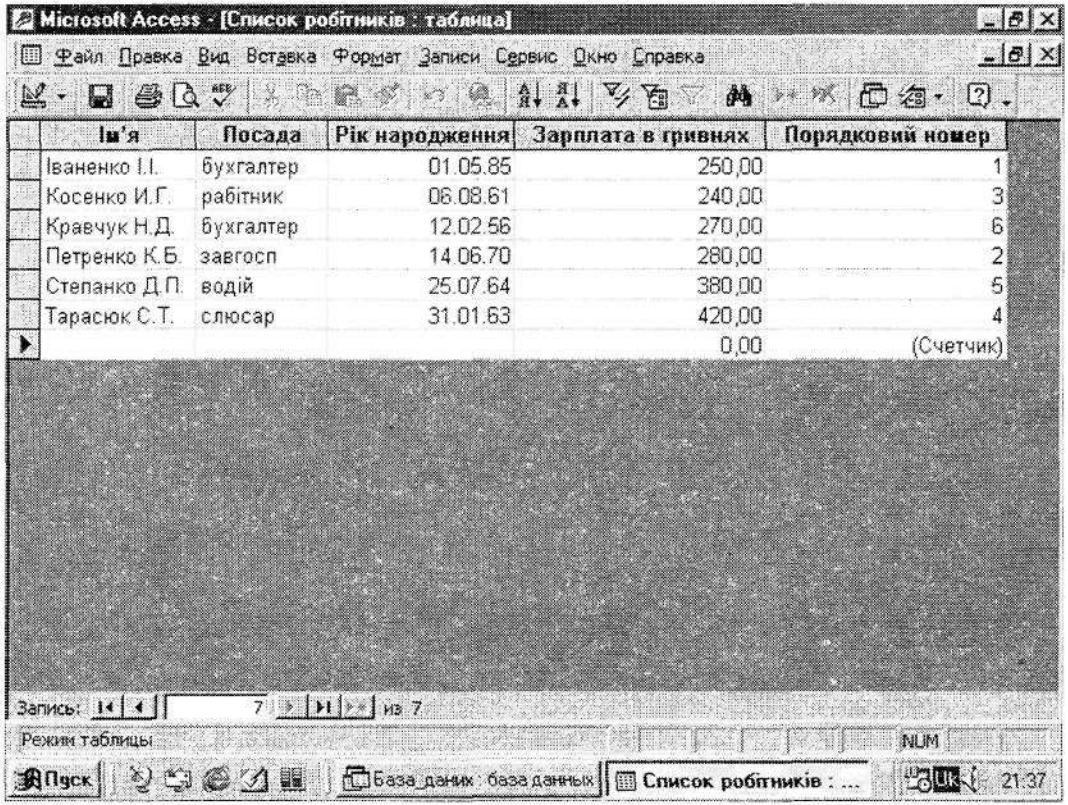

Рис. 27. Вид бази даних після зміни структури

третьому — *«Зарплата в гривнях»*. У рядку *«Виведення на екран»* відмічаємо всі три поля (у відповідних квадратах з'являться помітки). Запит сформовано, у головному меню верхньої частини екрана натискаємо на піктограму «!» (знак оклику). У новому вікні з'являється таблиця, що містить у другому стовпчику однакові слова — *«бухгалтер»,* а в інших стовпцях — імена бухгалтерів з бази даних (Кравчук Н.Д. і Іваненко І.І.), а також, величини їхньої заробітної плати. Результат виконання запиту представлено на рис. 28.

ЗАПИТАННЯ І ЗАВДАННЯ

1. Який вид може мати логічне представлення бази даних?
2. Яким чином організуються складні зв'язки між даними в базі даних?
3. Що собою являє система управління базою даних?
4. Яке логічне представлення даних реалізоване в СУБД Access? До якого типу належать бази даних, створювані за допомогою СУБД Access?
5. Як називається рядок таблиці, створюваної за допомогою СУБД Access?
6. Як називаються стовпці таблиці, створюваної за допомогою СУБД Access?
7. Що собою являє поле первинного ключа, як його задати в СУБД Access?
8. Що потрібно зробити, перш ніж приступити до заповнення таблиці даними?
9. Як додати нове поле в таблицю?
10. Як сформувати запит до бази даних?
11. Створіть базу даних учнів вашого класу, задавши такі поля, як ім'я, рік народження (дата народження), середній бал успішності, виконуване громадське доручення (назва громадської організації). Визначте тип полів, заповніть базу даними. Сформуйте наступні запити й отримайте відповідь:

- список учнів даного року народження;
- список учнів із заданим середнім балом успішності (наприклад, відмінників);
- список учнів за алфавітом;
- список членів даної громадської організації.

Доповніть базу новими даними і повторіть запити.

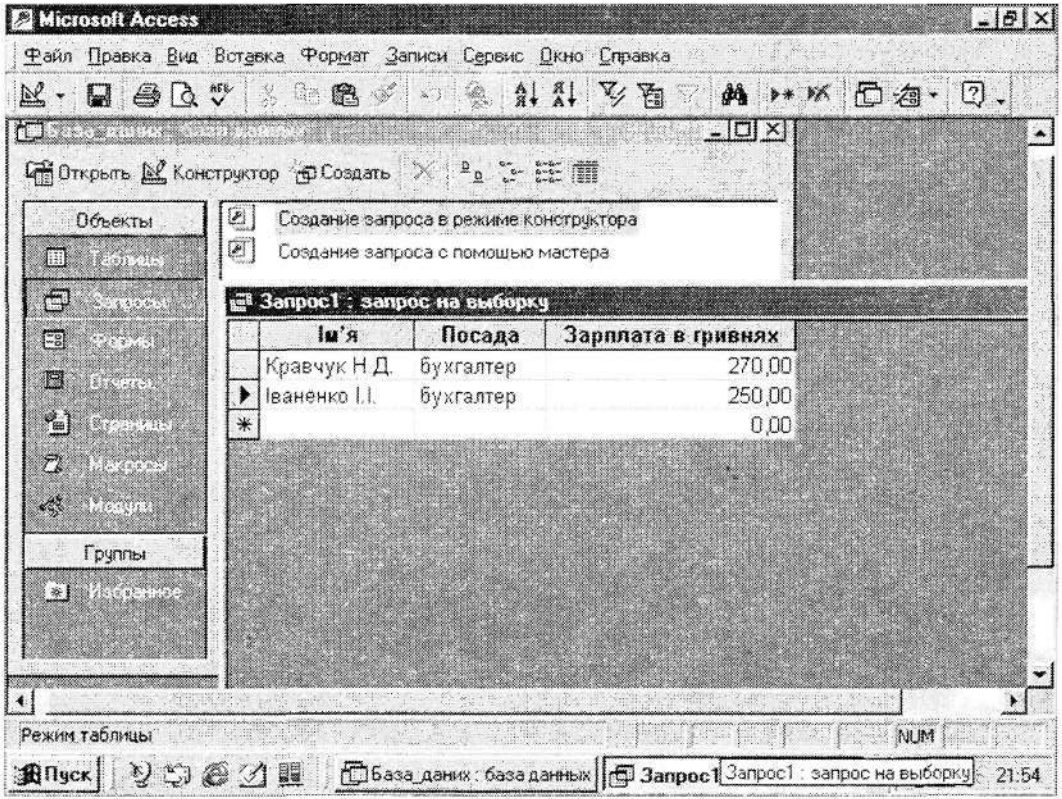

Рис. 28. Результат виконання запиту — список бухгалтерів підприємства

Інтелектуальні комп'ютерні системи

Сучасні комп'ютери можуть безпомилково виконувати мільярди арифметичних операцій за секунду, що абсолютно неможливо для людини. Але існують задачі, з якими людина справляється краще, ніж комп'ютер. Це, зокрема, прийняття рішень у складних ситуаціях, використання аналогій, смисловий переклад з однієї природної мови на іншу тощо.

З метою імітування на комп'ютері процесів мислення створюються автоматизовані системи, які називають системами штучного інтелекту. Для одержання результату в системі штучного інтелекту застосовується, як правило, механізм логічного виведення.

Системи штучного інтелекту поділяються на три класи:

1) *інтелектуальні інформаційні системи,* що дозволяють отримати відповідь на питання з певної галузі знань професійною мовою користувача;

2) *розрахунково-логічні системи,* що розв'язують складні задачі в режимі діалогу;

3) *експертні системи,* які дозволяють подавати знання в описовій формі і розв'язувати задачі з обґрунтуванням отриманого результату.

При реалізації цих систем вирішуються три основні проблеми штучного інтелекту: подання знань, використання природної мови і моделювання на комп'ютері людських міркувань. Усі названі види систем ґрунтуються на знаннях, тому подання знань є центральною проблемою систем штучного інтелекту.

Найпростішим способом подання знань є використання системи *продукційних правил (евристик),* які мають вигляд «ЯКЩО... ТО...». Частина правила «ЯКЩО» називається посиланням, а частина «ТО» — висновком або дією. Посилання може складатися з кількох фактів або умов, висновок — це факт або дія. Правила можна розуміти по-різному:

1) ЯКЩО істинні всі названі факти, ТО має місце даний факт, або

2) ЯКЩО виконуються деякі умови, ТО виконати вказану дію.

У такому вигляді можна подавати знання з різних галузей. Наприклад:

ЯКЩО промінь виходить із вершини кута і ділить його навпіл, ТО проміні, є бісектрисою;

ЯКЩО учені, їде до школи й автобус зупинився біля школи, ТО треба вийти;

ЯКЩО точки А, В і С не лежать на одній прямій, ТО фігура АВС — трикутник;

ЯКЩО фрукт м'який і синій і має одну кісточку, ТО цей фрукт — слива.

Перше, друге і останнє правила містять кілька фактів у частині ЯКЩО, з'єднаних сполучником «і».

Для отримання нових знань у системі продукційних правил або обгрунтування деякого факту використовується пряме й зворотне логічне виведення.

Пряме (індуктивне) виведення служить для прогнозу. Воно відповідає на питання «Що буде, якщо...?». Припустімо, система містить такі правила:

1) ЯКЩО придбати машину, ТО поїхати до моря;

2) ЯКЩО поїхати до моря, ТО загоряти;

3) ЯКЩО загоряти, ТО отримати сонячний удар.

При відповіді на питання «Що буде, якщо купити машину?» система відповість, що можна отримати сонячний удар. Така відповідь здивує будь-кого, тому при розробці систем із зворотним виведенням потрібно ретельно складати правила. При прямому виведенні поставлена мета (придбання машини) шукається в частині ЯКЩО правил системи. Коли факт знайдено, то розглядається частина ТО цього правила, в даному випадку — правила 1. Висновок цього правила (поїздка до моря) шукається в частині ЯКЩО іншого правила системи. Якщо такого правила немає, то як відповідь системи видається висновок першого правила (поїздка до моря). Якщо правило знайдено, то його висновок шукається знову в частині ЯКЩО інших правил. Оскільки у випадку, що розглядається, більше правил немає, кінцевим є висновок із третього правила. У даному прикладі перше правило містить малоймовірний висновок, тому результат роботи даної системи явно незадовільний. Щоб уникнути подібних висновків, в частині ТО вказують дробове число, що показує ймовірність (або правдоподібність) висновку.

Зворотне (дедуктивне) виведення обґрунтовує або пояснює наявні факти. Таке виведення використовує відомий детектив Шерлок Холмс при розкритті злочинів. Одна з перших експертних систем MYCIN, розроблена в Стенфордському університеті (США) для діагностики захворювань, використовує 500 евристичних правил. У процесі роботи програма запитує нові дані й обґрунтовує свої висновки. При зворотному виведенні наявний факт шукається в частині ТО правил. Якщо його знайдено, то факти з частини ЯКЩО цього правила використовуються для пошуку в частинах ТО нових правил. Якщо таких правил немає, то одержані факти розглядаються як причина вихідного. Якщо вони знайдені, то пошук триває й останні отримані факти видаються як першопричина, що привела до появи факту-запиту.

Якщо в розглянутому прикладі з придбанням машини систему запитати «Чому стався сонячний удар?», то відповіддю буде «Тому що придбали машину». Як обгрунтування буде видано зворотний ланцюжок міркувань; «Сонячний удар стався через те, що ви загоряли, а загоряли тому, що поїхали до моря, а поїхали до моря тому, що купили машину».

Для подання знань із будь-якої галузі наукової чи практичної діяльності людини використовуються також семантичні мережі — *мережеві структури,* що дозволяють зв'язати різні поняття, встановивши зв'язки між ними. Головне призначення семантичної мережі — правильно відобразити зміст кожного поняття. Залежно від призначення семантична мережа може містити різні зв'язки між об'єктами, а також між об'єктами та їх атрибутами. Переходячи від одного елемента семантичної мережі до іншого, можна встановити зв'язки між незв'язаними, на перший погляд, поняттями. Фрагменти семантичної мережі можуть бути подані у вигляді евристик, але в правилі тільки констатується зв'язок між фактами і не вказується тип цього зв'яжу. Фрагмент семантичної мережі для поняття «файл» подано на рис. 29. Зв'язки мають напрямки. Так, зв'язок «поле має ім'я» відображено направленою стрілкою від поняття «поле» до поняття «ім'я» і має назву «характеризується». При перегляді семантичної мережі спрямованість зв'язку має суттєве

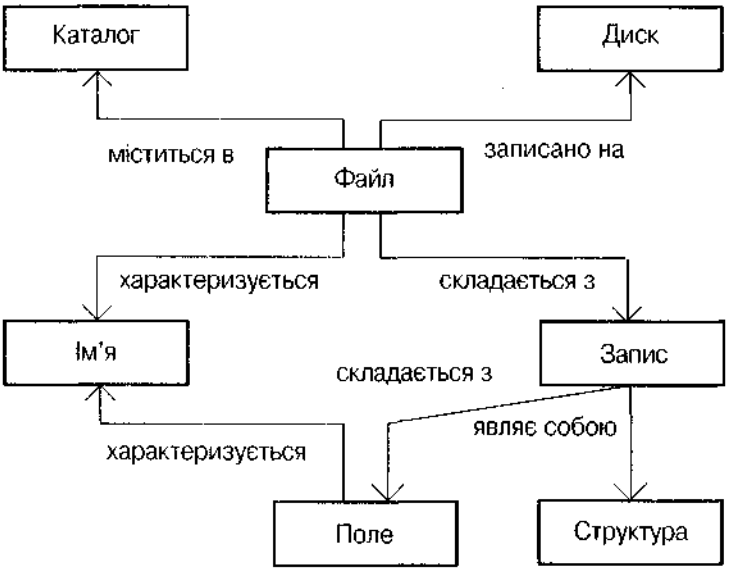

Рис. 29. Фрагмент семантичної мережі з вказаними зв'язками між поняттями

значення, оскільки, якщо не вказати напрям, то виникає неоднозначність і незрозуміло, що чим характеризується: чи поле запису своїм іменем, чи ім'я — якимось полем.

Третій спосіб подання знань — створення *фреймів* (рамок). Фрейм — це структура даних, іцо описує деякий клас об'єктів. У центр фрейму вміщуються об'єкти, а навколо нього з відповідними зв'язками — його атрибути. З'єднуючись через однотипні атрибути, фрейми також угворюють мережу. Фрейм відрізняється від семантичної мережі тим, що він може також містити процедури (звернення до програм) обробки елементів фрейму. Теорія фреймів стосується розуміння того, що ми бачимо й чуємо. Фрейм із точки зору людського мислення є одиницею подання знань. Це наче заповнена раніше рамка, деталі якої при потребі уточнюються або замінюються. Коли людина отримує нові відомості, вона зіставляє їх із наявними, порівнюючи з деякими шаблонами, які існують у пам'яті. При цьому нові відомості класифікуються і включаються до ієрархічної структури. Фрейми дозволяють зберігати відомості про один об'єкт

разом, водночас дозволяючи зв'язати їх з іншими об'єктами.

Створення систем штучного інтелекту є складною задачею, яка поки ще далека від повного вирішення.

Можна зазначити певні досягнення у комп'ютерному вирішенні таких інтелектуальних задач:

— логічне виведення та автоматизоване доведення теорем;
— спілкування мовою, наближеною до природної;
— автоматизований переклад з однієї мови на іншу;
— ігрові задачі (наприклад, існують програми, які вміють грати в шахи на рівні чемпіонів світу);
— розпізнавання образів, тобто віднесення об'єкта, який спостерігається системою, до того чи іншого класу (наприклад, діагностична система повинна прийняти рішення про те, чи є певний виріб справним чи ні; у разі несправності треба визначити її тип);
— створення роботів, які можуть орієнтуватися в зовнішньому світі і виконувати в ньому ті чи інші операції. Роботи можуть взяти на себе функції, які людині виконати не під силу, або працювати в умовах, які людина не може витримати.

Найбільшого розвитку серед інтелектуальних комп'ютерних систем набули експертні системи, які здійснюють дедуктивне логічне виведення на основі наявних знань. Існують різні визначення експертних систем, основна суть яких зводиться до такого.

Експертні системи — це інтелектуальні системи, здатні одержувати, накопичувати та коригувати знання із заданої предметної галузі, виводити нові знання, розв'язувати на основі цих знань практичні задачі та пояснювати хід їх розв'язку. В основі роботи експертних систем лежить логічне виведення нових фактів з існуючих. Типове застосування експертних систем — консультування фахівців середньої кваліфікації і неспеціалістів у тій галузі, для якої розроблені експертні системи.

Конкретна предметна галузь передбачає обмежений набір явищ та понять із певної галузі людської діяльності та обмежене коло задач, які вирішуються в цій галузі. Створено чимало експертних систем у таких галузях, як медична діагностика, юриспруденція, керування технологічними процесами і т.п.

У створенні експертних систем повинні брати участь фахівці як мінімум двох категорій: експертів та інженерів знань. Експерти — це висококваліфіковані фахівці у даній предметній галузі, знання яких потрібно передати експертній системі. Завданням інженерів знань є формалізація знань експертів і приведення їх до вигляду, придатного для занесення до бази знань.

В основі експертної системи лежить база знань про предметну область, тобто сукупність знань, доступних системі. Слід розрізняти знання двох типів:

1) явні факти, які зберігаються в пам'яті системи в явному вигляді;

2) правила, які дозволяють на основі відомих знань отримувати нові знання.

Найбільш відомою мовою програмування, призначеною для розробки експертних систем, вважається мова Пролог. Ця мова має зручні засоби для відображення фактів та правил, а в основі виконання прологівської програми лежить вбудований механізм логічного виведення.

ЗАПИТАННЯ І ЗАВДАННЯ

1. З чим пов'язана необхідність інтелектуалізації комп'ютерних технологій?
2. Які існують основні класи систем штучного інтелекту?
3. Для чого використовуються експертні системи?
4. Які проблеми вирішує штучний інтелект як науковий напрям?
5. Як здійснюється пряме логічне виведення?
6. Що дозволяє зробити і як проводиться зворотне виведення?
7. Для чого використовується семантична мережа?
8. Чим відрізняється фреймова мережа від семантичної?
9. Складіть продукційні правила, що дозволяють відрізнити птаха від літака.
10. Наведіть приклади практичних задач, які вирішуються за допомогою систем штучного інтелекту,
11. Схарактеризуйте поняття експертної системи.
12. Які ви знаєте мови для програмування експертних систем?

Комп'ютерні мережі

Для передавання від одного комп'ютера до іншого різноманітної інформації, в тому числі програм і даних, використовуються *комп'ютерні мережі*. У комп'ютерній мережі у вузлах мережевої структури знаходяться комп'ютери, з'єднані між собою за допомогою провідникових чи волоконно-оптичних ліній або бездротовими засобами зв'язку. Обмін інформацією в мережах здійснюється на основі мережевих протоколів. *Мережевий протокол* являє собою сукупність правил, які дозволяють комп'ютерам, що підключені до мережі, «розуміти» один одного та обмінюватися між собою даними. Існує багато типів мережевих протоколів, які розрізняються між собою функціональними можливостями.

Існуючі комп'ютерні мережі можна поділити на два класи: локальні та глобальні.

Локальна мережа використовується для передавання даних на невеликі відстані, наприклад в межах однієї кімнати, однієї будови чи одного підприємства. Прикладом локальної мережі може бути шкільний комп'ютерний клас, де є головний комп'ютер, за яким працює вчитель, і комп'ютери учнів. Головний комп'ютер у мережі називається *сервером*. За його допомогою здійснюється управління мережею чи її ділянкою. Сервер повинен мати достатньо велику оперативну пам'ять і місткий диск.

Глобальними називаються мережі, які об'єднують комп'ютери, розміщені на великій території. Здебільшого глобальні мережі є об'єднанням багатьох локальних мереж.

Комп'ютери зв'язуються між собою за допомогою спеціальної апаратури, куди входить мережевий адаптер. *Мережевий адаптер* — це пристрій, який забезпечує під'єднання комп'ютера до мережі.

Глобальною мережею, точніше супермережею, є мережа Інтернет, яка за останнє десятиліття стала основою світового інформаційного простору, забезпечивши недосяжні раніше інформаційні та комунікаційні можливості. Історія Інтернету бере початок від 60-х років. Перша мережа (ARPAnet) мала дещо подібні можливості, розроблялася в США як мережа спеціалізованого призначення і об'єднувала навчальні

заклади та військові організації. У ній були закладені дві основні ідеї, реалізовані в сучасній мережі Інтернет: 1) кожен вузол незалежно одержує і передає інформацію; 2) повідомлення ділиться на окремі частини (пакети), що передаються окремо і збираються разом у вузлі-одержувачі. В основі роботи мережі Інтернет лежить мережевий протокол TCP/IP, хоча до Інтернету можуть бути під'єднані і мережі, що використовують інші протоколи.

Найпоширенішим способом під'єднання комп'ютерів до Інтернету є використання телефонних ліній. Це під'єднання здійснюється за допомогою пристроїв, які називаються *модемами.* Організації та фірми, які забезпечують підключення до Інтернету і надають відповідні послуги, називаються *провайдерами.*

Адреси в Інтернеті, Кожен комп'ютер у мережі має свою адресу, яка називається *IP-адресою* і складається з чотирьох чисел, кожне з яких приймає значення від 0 до 255 і які при написанні розділяються крапками, наприклад: 194.44.192.20. Оскільки цифрові адреси важко запам'ятовувати, то для зручності IP-адресі ставиться у відповідність адреса, яка нагадує поштову адресу.

Кожній країні присвоєно своє унікальне ім'я, зареєстроване у світовій мережі: uk — Англія, jp — Японія, ru — Росія та ін. Після створення СНД колишні республіки отримали свої адреси. Ім'я України в Інтернеті — ua. Адреси утворюють деревоподібну структуру, яка забезпечує швидкий пошук абонентів. У цій деревоподібній структурі немає єдиного кореня, але кожна країна утворить своє піддерево. Так, в Україні вузлами наступного після «ua» рівня є назви міст, областей чи регіонів, наприклад **«kiev.ua», «crimea.ua», «lviv.ua»** та ін. Така система імен називається *доменною.* Кожен сервер у мережі, що належить одній чи кільком організаціям, об'єднаним у локальну мережу, також має своє ім'я. Таким чином, адреса складається з декількох частин, розділених крапками: спочатку ставиться ім'я сервера, після нього — назва міста чи регіону, а в кінці — ім'я країни, яке відповідне домену вищого рівня. Існує класифікація, що бере до уваги тип установи:

— **com** — комерційні установи (фірми, компанії і т.п.);
— **edu** — навчальні заклади;

— **gov** — урядові організації;
— **mil** — військові організації;
— **net** — організації, які надають мережні послуги;
— **int** — міжнародні організації;
— **org** — організації інших типів (часто безприбуткові).

Основні послуги, що надаються Інтернетом.

1. *Інтернет* акумулює інформаційні ресурси в світовому масштабі і забезпечує до них швидкий і ефективний доступ. Будь-який документ, розміщений в Інтернеті у будь-якій точці земної кулі, є доступним для кожного, хто має вихід до Інтернету. Існують різні системи забезпечення доступу до інформації (Gopher, WAIS і т.п.).

Домінуючою на сьогодні стала система WWW (World Wide Web, що перекладається з англійської мови як «всесвітня павутина»). В основі системи WWW лежить концепція *гіпертексту:*

Гіпертекст — це метод представлення та зберігання інформації, який забезпечує зв'язок між ключовими елементами та даними, які пов'язані з цими елементами. Досить часто ключові елементи знаходяться не тільки в різних місцях одного документа, але й у різних документах. При цьому забезпечується можливість швидкого перегляду документів за ключовими елементами.

World Wide Web з'явилась у 1991 році. Фізик Тім Бернерс-Лі працював у Європейському Центрі ядерних досліджень, який розташований в Женеві, де вивчав можливі способи дистанційного керування комп'ютерами. Йому вдалось створити систему, яка забезпечувала взаємодію комп'ютерів у мережі.

Складовими гіпертекстової системи є мова розмітки гіпертексту — HTML, універсальний метод адресації між комп'ютерами або мережами комп'ютерів — URL, і протокол обміну гіпертекстовою інформацією — HTTP.

Гіпертекстовий документ, розміщений в Інтернеті, прийнято називати *Web-сторінкою,* або просто сторінкою, якщо це не викликає непорозумінь. Широкого розповсюдження набув термін *сайт* (англ. site — ділянка, місце, місцезнаходження). Сайтом називають або окрему Web-сторінку, або сукупність взаємопов'язаних сторінок.

2. Електронна пошта (E-mail). Користувачі Інтернету (люди, які працюють з комп'ютерами, що мають доступ до Інтернету) можуть надсилати один одному повідомлення. Природно, що ці повідомлення надходять до адресата значно швидше, ніж: звичайні листи.

Кожний користувач (абонент) має власну «поштову скриньку», тобто виділений йому дисковий простір в комп'ютері, де зберігаються повідомлення, що надходять на його адресу. Абонент знайомиться з листами і відповідає па них у зручний для нього час. Повідомлення може включати не тільки текст, а й графіку та звук. Якщо однакові листи надсилаються різним абонентам, мережа надає можливість розсилання їх за всіма адресами, що вказані відправником.

Одним сервером, як правило, користується група абонентів. У такому разі ім'я кожного абонента відділене від адреси спеціальним знаком «@»:

ім'я_абонента@адреса_сервера

де ім'я абонента — це послідовність символів, яку вибрав абонент.

3. *Телеконференції* (система Usenet). Телеконференції можна порівняти з дошкою оголошень або клубом за інтересами. Телеконференції об'єднують людей, які цікавляться певною тематикою і дозволяють розміщувати свою інформацію для загального ознайомлення. Сьогодні в Інтернеті існують десятки тисяч різноманітних телеконференцій.

4. *Передавання файлів.* Для передавання файлів з одного комп'ютера до іншого в Інтернеті можуть використовуватись як засоби WWW та електронної пошти, так і спеціальний протокол, який має назву FTP.

5. *Комп'ютерні бесіди* (IRC) забезпечують живе спілкування між людьми на будь-якій відстані. На відміну від електронної пошти, при користуванні якою між надсиланням листа і отриманням відповіді на нього може пройти певний час, комп'ютерна бесіда — це письмове спілкування двох або більшої кількості співрозмовників шляхом листування, але безпосередньо, в реальному часі. Сторінки для спілкування називають *Інтернет-чатами.*

Чат (від англ. chat — дружня розмова, бесіда, балакання) — це сторінка, що спеціально створена для спілкування

в реальному часі. Для участі в діалозі потрібно вказати своє ім'я (воно потім буде знаходитись поруч з вашим повідомленням). Як тільки хто-небудь «заговорить», його репліка з'явиться у вікні вашого дисплея. Ви пишете репліку у відповідь. Спілкування відбувається безупинно, причому в звичайному режимі: всі бачать усі репліки, адресовані кожному. Нова фраза вводиться в спеціальному вікні, поруч з яким традиційно розташовуються дві кнопки: «відправити» і «обновити». Друга кнопка особливо корисна в тих випадках (досить частих), коли не працює автоматичне відновлення вікна.

Для кожного чата існує стандартний набір — поле для введення імені, вікно повідомлень і форма для уведення фрази. У залежності від розвиненості чататам можуть бути присутніми і інші, більш спеціалізовані засоби.

Програмне забезпечення для роботи в Інтернеті. існує велика кількість програм, призначених для забезпечення роботи в Інтернеті. Важливий клас складають *броузери* — програми для перегляду Web-сторінок. Сьогодні найбільш популярними є два броузери - Netscape Communicator та Microsoft Internet Explorer. Крім перегляду сторінок, ці два програмні продукти забезпечують інші необхідні можливості: електронну пошту, телеконференції тощо.

На рис. ЗО представлений типовий вигляд екрана комп'ютера при роботі з Microsoft Internet Explorer.

У верхній частині вікна броузера у невеличкому віконці після слова «Адреса» знаходиться адреса Web-сторінки http://www.lucky.net/support/index.ru.shtm, а сама сторінка відображена нижче у великому вікні. Літери Мір свідчать про те, що використовується гіпертекстовий протокол, www.lucky.net — це ім'я Web-сервера (комп'ютера, на якому розміщена Web-сторінка і який надає до неї доступ), вираз support/index.ru.shim описує папку, де знаходиться файл зображення, що з'явилось на екрані, та ім'я цього файла. Вираз support/index.ru.shtm часто називають специфікацією файла, і якщо вона не вказана, то відкривається сторінка, яка прийнята за умовчанням для даного Web-сервера. Гіпертекстові посилання у вікні броузера відображаються іншим кольором і часто підкреслюються. Підведений до них курсор мишки набуває вигляду руки.

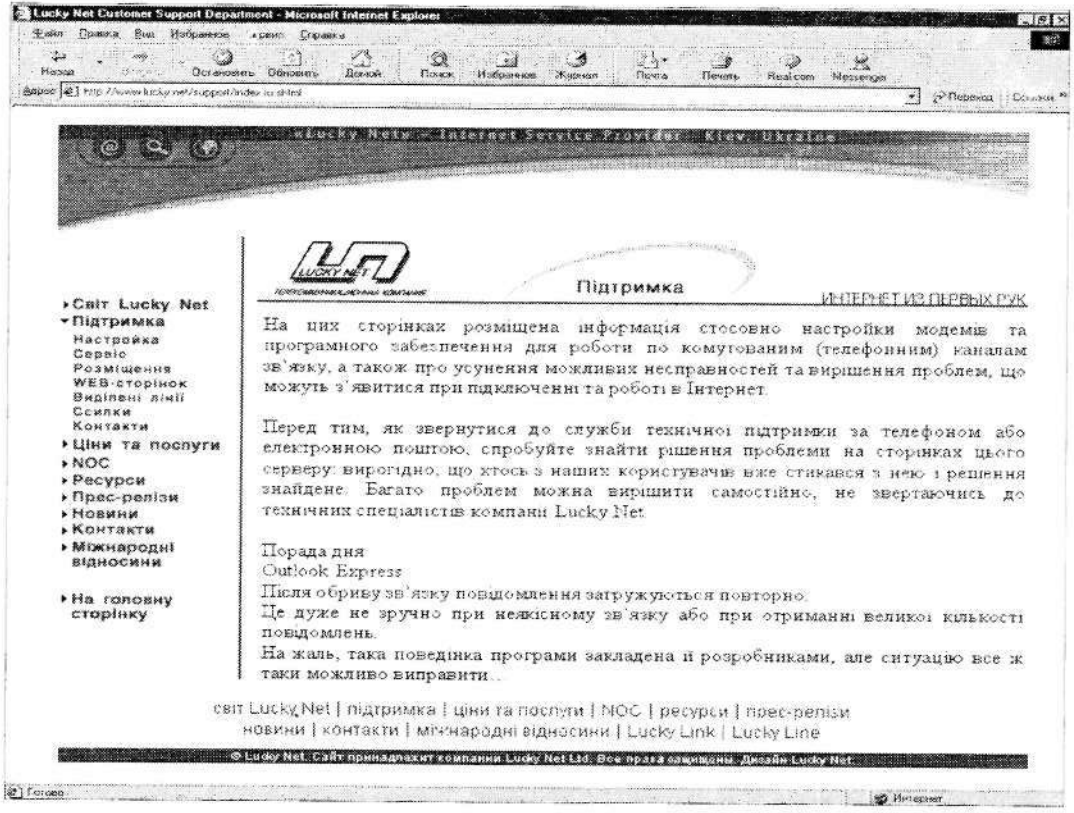

Рис. 30. Вікно Microsoft Internet Explorer

При клацанні лівою кнопкою мишки відбувається перехід до документа, на який вказує це посилання.

Щоб користувач міг завантажити Web-сторінку з Інтернету або з локального диска, потрібно встановити курсор мишки у віконці після слова «Адреса», набрати адресу потрібної сторінки і натиснути клавішу **Enter.**

Робота з поштою. Існують програмні засоби, які дозволяють вести електронну переписку. Досить поширеною програмою є Outlook Express.

Усі поштові повідомлення, що мають відношення до даного користувача, зберігаються в п'ятьох папках: *Вхідні, Вихідні, Відправлені, Вилучені* та *Чернетки.* Після зчитування із сервера пошта надходить до папки *Вхідні.* Після прочитання повідомлення залишаються в тій самій папці, але вже позначаються як прочитані. У папці *Вилучені* зберігаються ті повідомлення, що були вилучені користувачем. У папці *Вихідні* відображаються ті повідомлення, які користувач створив для відправлення. У момент з'єднання із сервером повідомлення переміщаються в папку *Відправлені.* Якщо сеанс зв'язку був невдалим, то повідомлення залишаються в папці *Вихідні.*

Користувач має можливість подивитися, які повідомлення до нього надійшли, прочитати будь-яке з них та зберегти його на диску. Можна зразу відповісти на це повідомлення, встановивши курсор мишки на пункт меню «Відповісти» і клацнути лівою кнопкою мишки (рис. 31). На екрані з'являється вікно для створення повідомлення, куди вноситься необхідна інформація. Разом з текстом, який набирається у вікні повідомлення, можна також переслати один або декілька файлів. Для створення нового повідомлення курсор мишки слід встановити на пункт меню «Створити» і клацнути лівою кнопкою мишки. При цьому з'являється вікно для створення повідомлення, в якому користувач повинен ввести поштову адресу, за якою слід надіслати це повідомлення. Бажано також вказати тему повідомлення.

Програмні засоби Netscape Communicator і Internet Explorer дозволяють користувачеві працювати з електронною поштою. Наприклад, в Microsoft Internet Explorer виклик поштової програми здійснюється за допомогою пункту «Пошта та новини» меню «Сервіс».

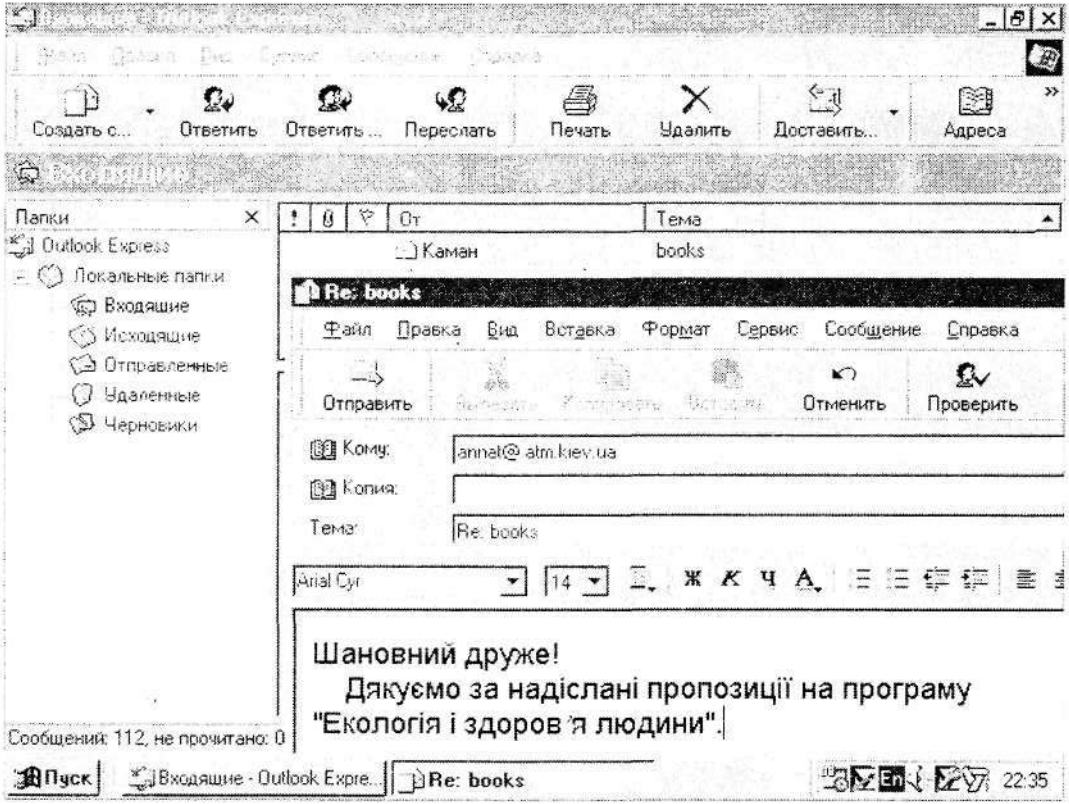

Рис. 31. Написання відповіді на лист

Пошукові системи. Зростання обсягів інформації в Інтернаті приводить до ускладнення пошуку необхідної інформації, і виникає питання, як конкретному користувачу знайти та відібрати в мережі Інтернет необхідні дані? Для того щоб отримати з Інтернету необхідну інформацію, можна скористатися допомогою пошукових систем, які здійснюють пошук інформації за вибраним критерієм.

Найбільш популярними пошуковими системами є *Yahoo!* та *Alta Vista*. Щоб зайти до Yahoo!, необхідно у віконці набрати адресу http://www.yahoo.com, а для Alta Vista — http://www.altavista.com. Існує кілька інформаційно-пошукових серверів, що охоплюють велику частину російськомовного ресурсу. Це такі сервери, як Aporl (http://www.aport.ru) — 26 тис. серверів, Yandex (http://www.yandex.ru) — 32 тис. серверів, і Rambler (http://www.rambler.ru) — 15 тис. серверів. Серед українських інформаційно-пошукових серверів можна скористатись сервером Mela (http://meta-ukraine.com) — 5 тисяч українських серверів (рис. 32), Ukrop (http://www.ukrop.com) тощо.

Щоб ефективно використовувати можливості мережних інформаційно-пошукових серверів, корисно довідатися, як вони працюють «зсередини», ознайомитися з принципами пошуку і організацією даних.

Найчастіше інформація в бази даних пошукових серверів попадає автоматично в результаті роботи програм, що називаються *Web-роботами*. Web-роботи — це такі програми, які безупинно «оглядають» ресурси WWW, переходячи з однієї сторінки на іншу, з метою збору інформації для формування бази даних про вміст Web-серверів.

Для забезпечення інтерактивного доступу користувачів до інформації, зібраної роботами, використовуються пошукові механізми — основні складові інформаційно-пошукових систем (ІПС). Ефективність і швидкодія пошукових механізмів багато в чому визначаються архітектурою баз даних, які використовуються тією чи іншою ІПС.

У пошукових системах користувач може здійснювати пошук потрібної йому інформації, набираючи ключове слово або декілька слів.

Розглянемо варіанти пошуку в сучасних пошукових системах. Усі пошукові системи забезпечують пошук хоча б

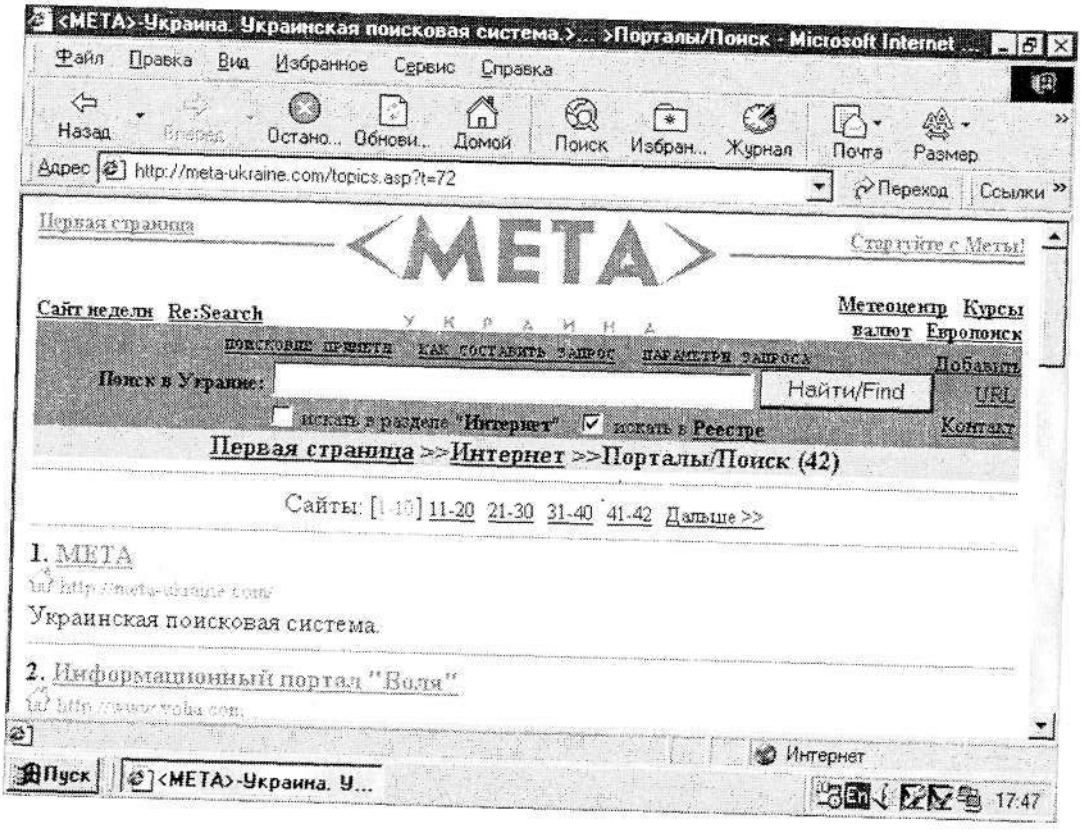

Рис. 32. Головна сторінка пошукової системи Мета

по одному слову. Засоби навігації в Інтернеті, що не забезпечують такого пошуку, називаються каталогами, колекціями посилань і т.п. У деяких відомих системах є можливість пошуку за скороченням слів (AltaVista, Rambler). В цьому випадку замість закінчення слова досить поставити символ «*». Пошук по словоформах є результатом серйозного лінгвістичного аналізу і реалізований у двох російськомовних системах — Yandex і Aport. Інколи доводиться шукати по фрагментах тексту (стійких словосполученнях), а багато систем здатні реалізовувати контекстний пошук фрази, що знаходиться в лапках. Можливість пошуку по полях найчастіше дозволяє обмежувати діапазон пошуку датами, заголовками і т. ін.

Розглянемо деякі рекомендації стосовно отримання інформації з Інтернет. Пошуковий сервер має свої власні характерні риси, тому треба уважно прочитати інструкцію до обраної пошукової системи. Слід виділити для пошуку ключові слова, що найточніше відбивають проблематику, якою ви цікавитесь. Оптимальний шлях полягає у використанні характерних для даної предметної області сполучень слів, кожне з яких має досить широку сферу застосувань. Розпочинати пошук слід з найбільш відомих і потужних пошукових серверів. Там дуже велика кількість посилань, але це дозволить скласти загальне уявлення про інформаційний стан даної предметної області, що дуже важливо для подальшої деталізації пошуку.

Можна використовувати метапошукові системи, що забезпечують пошук відразу на кількох подібних серверах. Прикладом є система Metacrawler, що розміщена за адресою http://www.metacrawler.com. Такі системи добре придатні для нескладних запитів, але й у випадку складних запитів їхнє застосування буває ефективним. Також можна застосовувати методику поетапного уточнення пошуку, починаючи з елементарних запитів у режимі простого пошуку. В міру одержання результатів можна розширювати та уточнювати запити за допомогою додаткових можливостей, використовуючи пошук за полями і т.п.

Створення Web-сторінки. У найпростішому випадку Web-сторінка являє собою гіпертекстовий документ, який зберігається у вигляді текстового файла, як правило, з роз-

ширенням *.htm* або *.html*. Для оформлення гіпертекстових документів використовується спеціальна *мова розмітки,* яка називається HTML.

Мова HTML складається із спеціальних вказівників для розмітки тексту, які називаються тегами. *Тег* являє собою інструкцію для броузера, яка визначає спосіб відображення тексту. Тег завжди починається із знака «<» («менше») і закінчується знаком «>» («більше»). Існують два типи тегів — парні і непарні. Парний тег можна порівняти з дужками в математичному виразі. Парний тег впливає на текст з того місця, де вжитий, і до того місця, де вказана ознака закінчення його дії. Цією ознакою служить той самий тег, але він починається із похилої риски «/» (англійською мовою вона називається *слеш).* Крім того, теги поділяються на категорії залежно від функцій, які вони виконують: *структурні теги, теги форматування абзаців, символів, визначення гіперпосилань, включення графіки* тощо.

Для опису структури HTML-файла використовуються такі парні теги:

<HTML> </HTML> — вказують броузеру, що далі йде HTML-файл. Ці теги обрамляють документ, тобто весь текст знаходиться між цими тегами. Далі потрібно розділити документ на дві частини, одна з яких є заголовок, а друга — текст;

<HEAD> </HEAD> — між цими тегами повинен знаходитися заголовок документа, що складається з кількох частин, основна з яких є заголовок вікна;

<TITLE> </TITLE> — теги заголовка. Заголовок розміщується між тегами HEAD;

<BODY> </BODY> — між цими тегами подається інформація, яка повинна відображатися на екрані при перегляді за допомогою броузера. Тег BODY може мати кілька параметрів, що описують колір фону вікна перегляду, малюнок у ньому, колір тексту і т.п.

Для форматування символів використовуються теги:

<B> </B> — для напівжирного шрифту;

<I> </I> — для подання тексту курсивом;

<U> </U> — для підкреслення тексту.

Крім вказаних тегів, існують теги для управління зовнішнім виглядом Web-сторінки, для включення графіки в Web-сторінку тощо.

Теги, що визначають гіперпосилання, дозволяють вказати, як ця сторінка пов'язана з іншими документами Інтернету. Для того щоб задати гіперпосилання, треба знати ім'я файла, до якого потрібно переміститися, і текст або зображення, з яким буде пов'язане це гіперпосилання. Для вставляння гіперпосилання до Web-сторінки використовується тег <A>:

<A HREF = «ім'я файла»> *слово (група слів), малюнок або інший об'єкт, за яким здійснюється гіперпосилання на файл, ім'я якого вказане в HREF* </A>

Розглянемо приклад створення найпростішої сторінки (рис. 33). Для створення цієї сторінки слід набрати в будь-якому текстовому редакторі наступний текст:

```
<HTMI_>
<HEAD>
<TITLE> MY FIRST PAGE </TITLE>
</HEAD>
<BODY BGCOLOR = WHITE>
<P> <FONT COLOR = BLACK SIZE = 6>
<CENTER > Вітання <B>всім</B>,
хто заглянув до моєї <U> <I>сторінки </Ix/U>!
</CENTER>
</FONT>
</BODY>
</HTML>
```

Файл з набраним текстом слід зберегти на диску у текстовому форматі, але з розширенням *.hlm.* Сторінку можна переслати на сервер (при наявності відповідних прав), а можна відкривати з локального диску.

Існують інструментальні засоби, які полегшують створення Web-сторінок (Composer, який входить до складу Netscape Communicator, FrontPage та in.). Сучасні текстові процесори також дозволяють зберігати документ у форматі Web-сторінки.

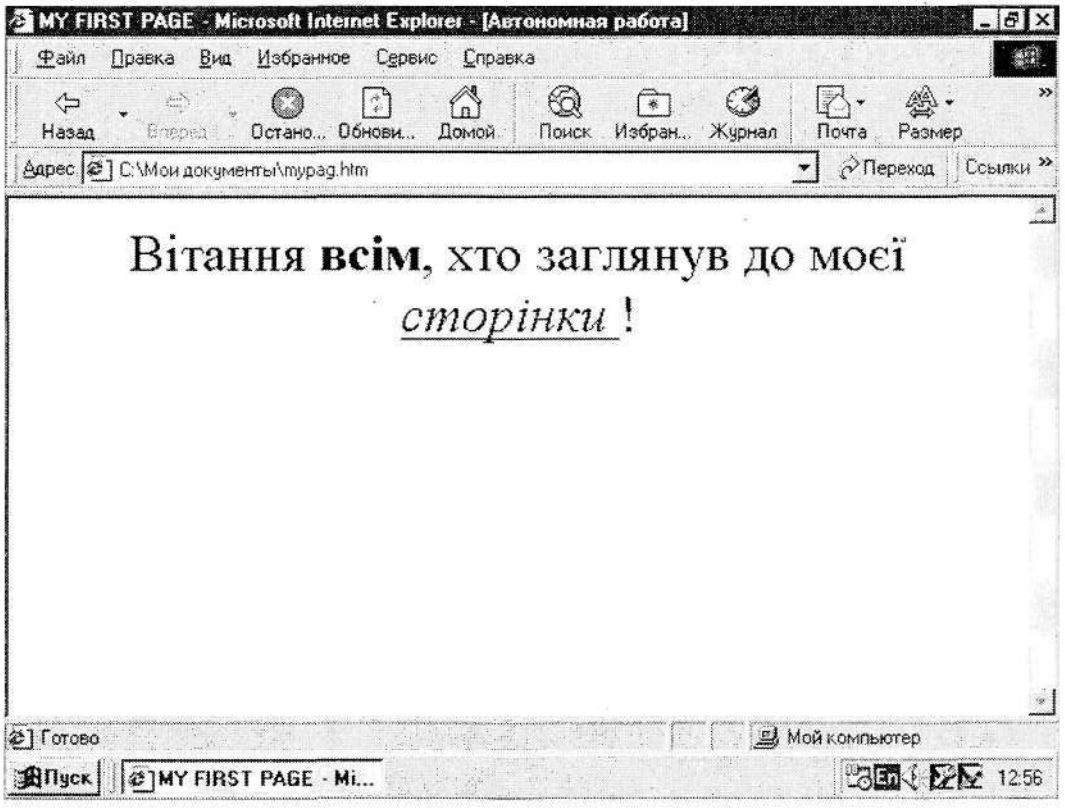

Рис. 33. Відображення Web-сторінки у вікні

ЗАПИТАННЯ І ЗАВДАННЯ

1. Що являє собою Інтернет?
2. Опишіть основні можливості використання Інтернету.
3. Охарактеризуйте поняття WWW.
4. Що таке гіпертекст?
5. Що таке Web-сторінка?
6. Що таке електронна пошта?
7. Охарактеризуйте поняття «телеконференція».
8. Як називається пристрій, за допомогою якого здійснюється під'єднання комп'ютера до Інтернету з використанням телефонних ліній?
9. Що таке провайдер?
10. Охарактеризуйте доменну систему імен.
11. Назвіть найбільш відомі домени вищого рівня.
12. Яке доменне ім'я вищого рівня присвоєне Україні?
13. Як формується адреса електронної пошти?
14. Які існують програми для роботи в Інтернеті?
15. Опишіть відомі вам засоби пошуку інформації в Інтернеті.
16. Як можна створити Web-сторінку?

Частина 2

Основи алгоритмізації і програмування

2.1. Алгоритми

Досвід практичної діяльності дозволив людині виробити спосіб розв'язування складних задач, який називається *алгоритмічним.* Цей спосіб полягає в тому, що складний процес розв'язування задачі поділяється на декілька етапів, кожному з яких відповідає дія, виконання якої не становить труднощів. Якщо виконавець процесу розв'язування певної задачі (людина, комп'ютер, робот тощо) відомий, то всі дії, що відповідають етапам — складовим цього процесу, можна подати як конкретні команди. Виконавець обов'язково повинен їх правильно розуміти (у прямому чи переносному смислі) та вміти виконувати. У такому разі кажучі., що розв'язування задачі зводиться до виконання певного алгоритму.

Алгоритм — це скінченна послідовність вказівок (команд), формальне виконання яких дозволяє за обмежений час отримати розв'язок задачі.

Інакше кажучи, алгоритм — це певна інструкція для виконавця, яка може бути задана різними способами — словами, формулами, послідовністю обчислювальних операцій чи логічних дій тощо. Але не кожна інструкція може бути алгоритмом. Алгоритм повинен відповідати певним вимогам, мати такі властивості:

1. Масовість. Алгоритм має бути придатним для багатьох задач, що належать до певного класу.

2. **Однозначність (детермінованість).** Ця властивість означає, що кожна команда не повинна допускати двоякого тлумачення. Кожний крок алгоритму повинен бути точно визначеним.

3. **Дискретність.** Процес, який визначається алгоритмом, повинен мати дискретний (перервний) характер, тобто

являти собою послідовність окремих завершених кроків — команд або дій.

4. **Результативність,** Результативність означає, що кожна дія повинна приводити до цілком певного результату.

5. **Формальність.** Будь-який виконавець, здатний сприймати та виконувати вказівки алгоритму (навіть не розуміючи їх змісту), діючи за алгоритмом, може отримати розв'язок поставленої задачі.

6. **Скінченність.** Діючи за алгоритмом, виконавець одержує розв'язок задачі за скінченну кількість кроків.

Досконалим виконавцем алгоритмі!! обробки інформації є комп'ютер, робота якого здійснюється під управлінням програми. *Програма* — це реалізація алгоритму, команди якої «зрозумілі» комп'ютеру і можуть бути ним виконані. Кожний тип комп'ютера має свій власний набір операцій, із яких можна складати програми. Це так званий набір машинних команд комп'ютера, що визначає обмеження на розробку програм. Тому створення програм являє собою творчий процес, коли потрібно «навчити» комп'ютер розв'язувати різноманітні, як правило складні задачі, використовуючи обмежений набір простих операцій (команд). Разом з тим набір операцій кожного із сучасних комп'ютерів є алгоритмічно повним. Тобто, з цих простих комп'ютерних операцій, як із цеглинок, можна складати алгоритми для розв'язування задач будь-якої складності.

Алгоритм описується засобами мови, зрозумілої виконавцю. Для людини — це природна мова. Комп'ютер виконує операції з багаторозрядними числами у двійковій системі числення, тобто мовою комп'ютера є послідовність кодів із нулів і одиниць. Використання людиною такої комп'ютерної мови для складання програм дуже неефективне. Записи програм, отримані таким чином, виявляються дуже докладними та громіздкими, процес розробки програм — програмування — складним і трудомістким. Тому для зручності при створенні програм розроблені спеціальні мови — так звані *мови програмування.* Мова програмування дозволяє записувати команди в такій формі, щоб їх можна було замінити на машинні коди. Це перетворення автоматично здійснюється за допомогою спеціальних програм-перекладачів, які називаються *трансляторами.* Транслятори є складовою части-

ною системного програмного забезпечення комп'ютера. У сучасних комп'ютерах використовуються мови програмування, які за своїми властивостями наближаються до природної мови людей.

При розробці й поданні алгоритмів можуть бути застосовані різні способи їх запису в текстовій та графічній формі. Широкого розповсюдження набув найбільш наочний спосіб зображення алгоритмів у вигляді *графічних схем* (схем алгоритмів).

Схема алгоритму складається з елементів двох типів. Перший тип — це графічні фігури (прямокутники, ромби тощо), кожна з яких відображає один з етапів процесу розв'язування задачі і містить у собі текст відповідної команди. При побудові схем алгоритмів здебільшого використовуються такі графічні позначення: овали — для початку й кінця алгоритму, паралелограми — для введення та виведення даних, прямокутники — для обчислень, ромби — для перевірки

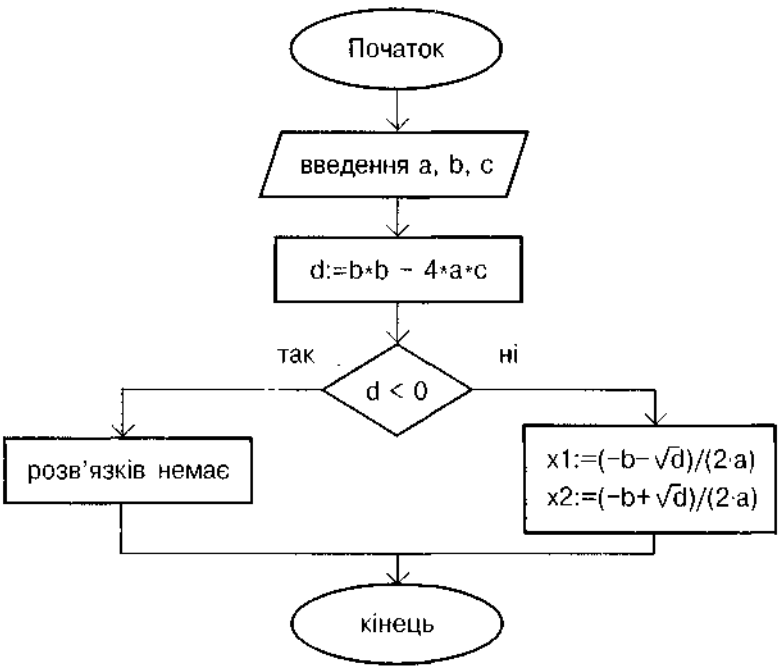

Рис. 34. Приклад схеми алгоритму

умов. У прямокутниках зображується також будь-яка результативна команда по перетворенню даних або ситуації. Другий тип елементів — це лінії зі стрілками, що вказують послідовність (порядок) виконання етапів.

У цілому такий спосіб запису алгоритмів можна розглядати як певну алгоритмічну мову — систему позначень і правил для однотипного запису алгоритмів та їх виконання.

Схема алгоритму ілюструється на рис. 34 на прикладі процесу розв'язування квадратного рівняння.

ЗАПИТАННЯ І ЗАВДАННЯ

1. Що таке алгоритм? Назвіть і поясніть властивості алгоритмів.
2. Зобразіть схему алгоритму ділення відрізка пополам за допомогою циркуля й лінійки.
3. Зобразіть схему алгоритму знаходження кількості додатних парних чисел, сума яких дорівнює 46.
4. Зобразіть у вигляді схеми алгоритми обробки текстів у текстовому редакторі.

2 Мова програмування Паскаль

Одна з найпопулярніших мов програмування — це мова Паскаль, яку створив у 1968 р. швейцарський вчений Ніклаус Вірт. Вона дозволяє записувати команди, завдяки яким комп'ютер може розв'язувати математичні задачі, обробляти тексти, будувати зображення на екрані дисплея.

Як кожна мова, Паскаль має свій алфавіт. До нього входять латинські літери, цифри від 0 до 9, спеціальні знаки (+, —, круглі, квадратні і фігурні дужки, крапка, кома та ін.), а також службові слова мови (begin, end, for, while тощо). У текстах програм службові слова виділяються жирним шрифтом. Для зручності до текстів програм вносяться пояснення (коментарі), які в Паскалі записуються у фігурних дужках.

Одним із основних етапів розроблення програми для розв'язування задачі є присвоєння даним, які використовуються в цій задачі, імен. Ім'я в Паскалі — це слово, яке починається з літери і містить літери, цифри та знаки

підкреслення. Як імена не можна використовувати службові слова мови Паскалі».

Кожне ім'я відповідає певній ділянці пам'яті, куди записуються значення даних. Оскільки на одну й ту саму ділянку можна записати різні значення, то ім'я також називають змінною або ім'ям змінної. Ділянка — поняття умовне: це послідовність різної кількості байтів пам'яті для різних даних. Для кожної змінної треба вказати її тип, щоб вказані транслятору (програмі, яка перекладає з мови програмування на мову машинних команд), скільки місця в пам'яті потрібно виділити для цієї змінної.

У мові Паскаль розрізняють цілі і дійсні числа. їм відповідають змінні цілого та дійсного типів. Для цілих чисел в пам'яті комп'ютера відводиться два байти, а для дійсних — шість.

Для додатних чисел знак « + » можна не писати. При записі дійсних чисел ціла частина відокремлюється від дробової не комою, а крапкою. Дійсні числа можуть записуватись у двох формах: із фіксованою крапкою, наприклад 7.23, 897.5, —0.11, та в експоненціальній (з плаваючою крапкою), наприклад 1.354E+ 12. Літера E означає множення на степінь 10. Таким чином, запис — 4.9876543234E —02 означає те саме, що й —0,049876543234. Незалежно від форми запису дійсні числа зберігаються в пам'яті машини у формі з плаваючою крапкою.

Для роботи з числами використовуються шість операцій: « + » — додавання, «-» — віднімання, «/» — ділення, «*» — множення, «div» — ділення націло і «mod» — знаходження залишку від ділення. Останні дві операції застосовуються тільки до цілих чисел.

З імен змінних, чисел, знаків арифметичних операцій і функцій складаються арифметичні вирази. Щоб указати порядок дій, використовують тільки круглі дужки; їх може бути кілька, але кількість відкритих дужок повинна дорівнювати кількості закритих.

До мови Паскаль вбудовані засоби обчислення значень основних математичних функцій. Запис мовою Паскаль синуса, косинуса та натурального логарифма збігаються із загальноприйнятими: sin(x), cos(x), ln(x). Піднесення аргументу до квадрата позначається sqr(x), добування квадрат-

ного кореня sqrt(x), а модуль — abs(x). Аргументом функції може бути арифметичний вираз.

Приклад арифметичного виразу:

математичний запис: $\dfrac{x}{1+\dfrac{x^2}{5+x}}$;

запис на Паскалі: x / (1 + sqr(x) / (5 + x • sqr(x))

Алгоритм перетворення даних на Паскалі складається з *операторів та підпрограм,* що є головними структурними елементами програм. Кожний оператор перетворюється транслятором у послідовність машинних команд. Підпрограма має структуру, аналогічну до структури програми і використовується для опису послідовності дій, виконання яких повторюється. Такі підпрограми називають *процедурами* або *функціями.* Процедури та функції поділяються на бібліотечні та ті, що написані користувачем. Прикладом бібліотечних процедур є процедури введення, виведення, роботи з графікою, математичні та інші.

Оператор присвоювання призначений для надання змінній нового значення. Загальний вигляд оператора присвоювання:

ім'я змінної := арифметичний вираз;

Знак «: = » читається «присвоїти» (надати значення). У кінці запису оператора Паскаля ставиться крапка з комою.

При виконанні оператора присвоювання обчислюється значення арифметичного виразу, що стоїть у правій частині. При цьому з ділянок оперативної пам'яті (з відповідними іменами) зчитуються відповідні значення і над ними виконуються вказані дії. Одержаний результат записується до ділянки пам'яті, ім'я якої вказано зліва від знака присвоювання.

Приклади оператора присвоювання:

```
x := 3.14;     {змінній x присвоїти значення 3.14}
a := b + c;    {з ділянок b і c зчитуються заздалегідь
                записані туди дані, обчислюється сума,
                результат записується до ділянки a}
i := i + 1;    {значення змінної і збільшується на
                одиницю}
```

Для типів змінної ліворуч і арифметичного виразу праворуч від знака присвоювання існують обмеження:

1) якщо змінна ліворуч — дійсного типу, то арифметичний вираз може бути як цілого, так і дійсного типу, тобто містити або цілі змінні й допустимі для них операції, або дійсні, або і ті, й ті (тоді вираз перетворюється на вираз дійсного типу);

2) якщо змінна ліворуч — цілого типу, то арифметичний вираз може бути тільки цілого типу.

Це означає, що можна, наприклад, дійсній змінній присвоїти ціле значення. У пам'яті комп'ютера воно буде перетворено на дійсний тип. У фігурних дужках поряд з оператором подається коментар його дій.

ЗАПИТАННЯ І ЗАВДАННЯ

1. Хто створив мову програмування Паскаль?
2. Які символи входять до складу алфавіту Паскаля?
3. Для чого в Паскалі розрізняються цілі й дійсні числа?
4. Що таке арифметичний вираз і з чого він може складатися?
5. Як працює оператор присвоювання?
6. Які обмеження накладає Паскаль на типи даних при присвоюванні?
7. Що таке транслятор?
8. Запишіть у вигляді арифметичного виразу квадратний тричлен.

2.3 Структура опису програм на мові Паскаль. Введення і виведення даних

Програма на Паскалі починається із заголовка, далі розміщується описова частина, в якій визначаються дані, що використовуються в програмі, а після цього — тіло програми або програмний блок (блок операторів), що містить оператори для перетворення даних.

Загальний вигляд програми:

```
program    ім'я програми;
uses       {список програмних модулів, що
           використовуються};
```

```
label   {список міток};
const   {список сталих величин - констант};
type    {описи нестандартних типів даних};
var     {описи змінних, що використовуються в
         програмі};
begin   {позначення початку програмного блоку}
         {програма - послідовність операторів}
end.    { кінець програми}
```

Ім'я програми складається не більше як з 8 символів. Воно починається з літери і містить латинські літери, цифри й знаки підкреслення. В імені програми не використовуються зарезервовані слова, такі як begin, end і т. д. Програма починається зі слова **program** і закінчується словом **end** із крапкою. Оператори, заголовок програми, описи типів та змінних закінчуються крапкою з комою.

До описової частини програми входять розділи міток **label,** констант **const,** нестандартних типів даних **type** і змінних **var.** Основним із них є розділ змінних **var.** У ньому вказуються імена змінних, що використовуються в програмі, та їх тип. Для числових даних використовуються основні описувачі типів **integer** (цілий) і **real** (дійсний). Наприклад, якщо у програмі використовуються дві цілочислові змінні *i, J* та одна дійсна x, то тоді розділ змінних може мати вигляд

```
var i,j: integer; x: real;
```

Імена змінних одного типу пишуться через кому, потім після двокрапки вказується їх тип. Опис кожного типу закінчується крапкою з комою. Коли при перекладі на мову машинних кодів транслятор зустрічає опис змінної, він відводить для цієї змінної ділянку пам'яті і ставить у відповідність до імені змінної адресу першого байта ділянки.

Програмний блок містить послідовність команд для реалізації алгоритму розв'язування задачі.

Для введення даних у комп'ютер і виведення їх використовуються відповідно процедури введення і виведення. Процедура введення поміщає значення змінної, яке вводиться, у відповідну ділянку пам'яті. Процедура введення має вигляд

read (список імен);

При виконанні процедури read (читати) програма зупиняється і чекає доти, доки користувач набере на клавіатурі число і натисне Enter. Якщо список введення містить кілька імен, то для кожного треба ввести своє значення. Числа, що вводяться, відокремлюються пропуском чи комою або після кожного натискується Enter. Наприклад, процедура

read (i,j);

потребує введення двох цілих чисел. Після виконання цієї процедури курсор на екрані дисплея знаходиться в кінці останнього числа і не переходить на новий рядок. Щоб після введення даних курсор на екрані дисплея переходив на новий рядок, треба використовувати процедуру

readln (список імен);

Для виведення результаті її роботи програми на екран дисплея служить процедура

write (список виведення);

До списку виведення заносять імена змінних або вирази, відокремлені між собою комами. До списку виведення можуть також входити взяті в апострофи тексти. Наприклад,

write ('x= ', x);

При виконанні цієї процедури па екрані буде надруковано текст, що міститься між апострофами, і значення змінної x. Значення буде виведене у формі дійсного числа з плаваючою крапкою. Щоб число було виведене у формі з фіксованою крапкою, після імені змінної треба вказати два цілих числа, відокремивши їх від імені змінної та одне від одного двокрапками. Перше з цих чисел вказує, скільки позицій займає число (включаючи десяткову крапку і знак числа). Друге число вказує кількість цифр дробової частини числа. Наприклад, для друкування числа —46.17 як значення змінної x процедуру друкування слід подати у вигляді

write ('x- ', x:6:2);

На екран буде виведено: x= —46.17

Якщо після друкування треба перевести курсор на новий рядок, використовується процедура

writeln (список виведення);

Для переведення курсору па новий рядок використовується процедура виведення без параметрів

writeln;

Розглянемо приклад. Нехай потрібно обчислити суму, добуток і різницю двох даних чисел. Для кожного з чисел треба придумати ім'я змінної і вказати її тип. Потім ввести ці числа у відповідні ділянки пам'яті і, використовуючи можливість включення до процедури виведення арифметичних виразів, надрукувати результати:

```
program E1;
var a,b: real;
begin
  write('введіть два числа через пропуск, потім
        натисніть  Enter');
  readln(a,b);
  writeln('a+b =', a+b, 'a*b =', a*b, 'a-b = ', a-b)
end.
```

Службові слова виділені жирним шрифтом, на письмі вони підкреслюються. При наборі тексту програми на клавіатурі службові слова ніяк не виділяються, їх розрізняє транслятор. Службові слова не можна використовувати як імена. Перша процедура програмного блоку виводить на екран підказку для користувача, що він повинен зробити. При введенні даних рекомендується робити подібні підказки. Слід зауважити, що перед службовим словом end крапку з комою можна не ставити.

При розв'язуванні задач імена присвоюються не тільки вхідним даним, а й результатам і проміжним значенням. Оскільки в розглянутому прикладі треба одержати три результати, введемо для них імена *x, y, z*. У програмі цим змінним будуть присвоєні значення суми, добутку та різниці двох чисел, що вводяться.

Програма має вигляд:

```
program E2;
var a,b,x,y,z: real;
begin
  write ('введіть два числа через пропуск, потім
         натисніть Enter');
  readln(a,b);
  x:= a + b;
  y:= a * b;
  z:= a - b;
  writeln('a + b =', x, ' a*b =', y, ' a - b =', z)
end.
```

ЗАПИТАННЯ І ЗАВДАННЯ

1. Для чого потрібно описувати дані в програмі?
2. Як описати змінні одного типу, наприклад дійсного?
3. Яка процедура використовується для введення даних? Як вона працює?
4. Куди потрапляють введені з клавіатури числа при роботі процедури введення?
5. Як перевести курсор на новий рядок після введення даних?
6. Як вивести результати роботи програми на екран дисплея?
7. Як зробити, щоб виведення даних здійснювалось з нового рядка?
8. Напишіть програму обчислення середнього арифметичного двох чисел.
9. Напишіть програму обчислення відстані між двома точками площини.
10. Напишіть програму обчислення площі трикутника за формулою Герона.
11. Напишіть програму обчислення площі бічної поверхні куба.
12. Напишіть програму обчислення площі та гіпотенузи прямокутного трикутника, якщо відомі його катети.
13. Напишіть програму обчислення суми модулів трьох дійсних чисел.
14. Напишіть програму обчислення площі круга, якщо відома довжина кола.
15. Напишіть програму обчислення площі рівностороннього трикутника.
16. Напишіть програму піднесення числа до четвертого степеня за дві операції.
17. Напишіть програму піднесення числа до сьомого степеня за чотири операції.
18. Напишіть програму визначення моменту зустрічі двох автомобілів,

якщо відома відстань між пунктами, звідки вони вийшли назустріч один одному одночасно, та їх швидкості.

19. Напишіть програму обчислення суми членів арифметичної прогресії, якщо відомо її початковий член і різниця, а також кількість її членів. Вказівка: при роботі на комп'ютері вкажіть різні формати виведення чисел з фіксованою крапкою, виконайте програму для різних даних кілька разів.

2.4. Програмування мовою Turbo Pascal

Turbo Pascal —діалект мови Pascal. Turbo Pascal розроблено для комп'ютерів типу IBM PC фірмою Borland (США). Одна з перших версій цього продукту з'явилася на міжнародному ринку ПЕОМ у 1985 році. Існує кілька версій цієї мови програмування, що містить транслятор, редактор, різні сервісні функції для роботи з файлами, а також бібліотеки, які дозволяють будувати зображення, використовувати в програмі засоби операційної системи MS-DOS тощо.

Створення програми. Каталог, що містить систему Turbo Pascal, називається TP; після цих літер вказується версія системи, наприклад 5. У цьому каталозі треба знайти файл з іменем turbo.exe, підвести до нього курсор і натиснути **Enter.** При запуску системи з'являється вікно редактора текстів програм (цей редактор можна використовувати і як редактор текстів) (рис. 35).

Щоб увійти до меню, яке розташоване зверху екрана, використовується клавіша **F10.** Переміщення курсора вздовж рядка меню здійснюється клавішами управління курсором. Якщо екран порожній, можна відразу набирати текст програми. При набиранні програм рекомендується робити відступи, як зроблено в цій книзі. Це полегшує читання текстів програм і пошук помилок.

Якщо на екрані після запуску системи знаходиться непотрібна програма, треба ввійти в пункт меню «File» і виконати команду «New». Екран очищається, і зверху з'являється ім'я програмного файла noname.pas (програма без імені). Набір кожного рядка програми завершується натисненням клавіші **Enter.** Курсор можна переміщувати по тексту за

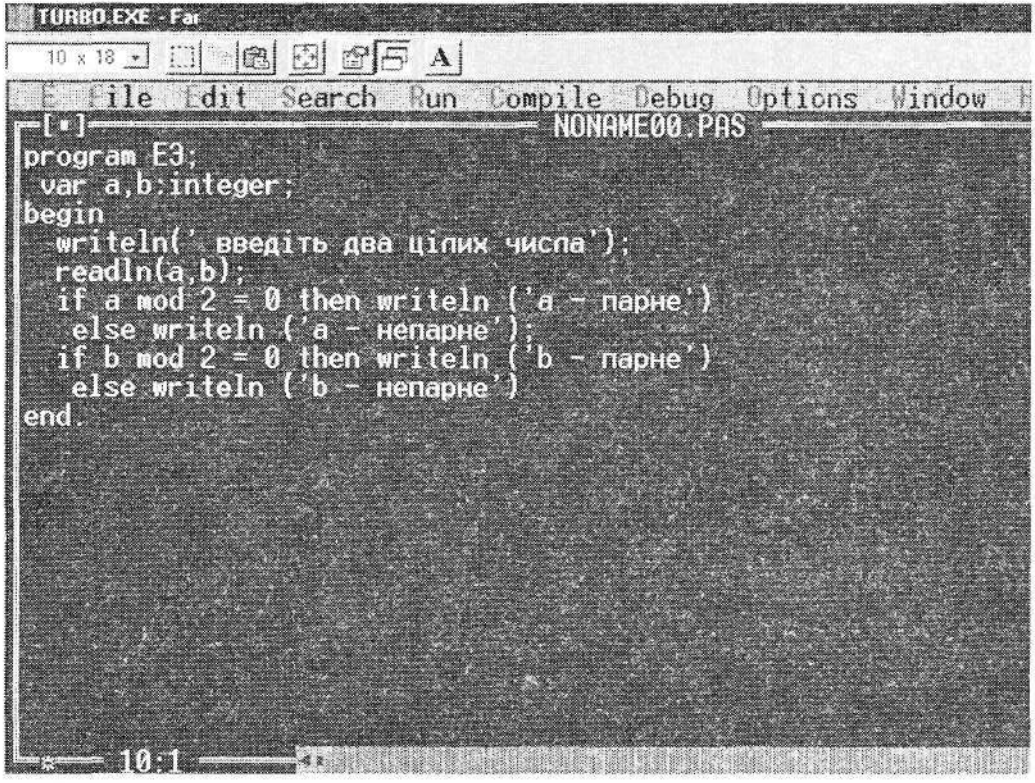

Рис. 35. Вікно редактора текстів програм

допомогою клавіш управління курсором. На початок рядка можна перейти за допомогою клавіші **Home,** а в кінець рядка за допомогою клавіші **End.**

Стерти непотрібний рядок можна натисненням комбінації клавіш **Ctrl** + Y, вставити — натисненням **Enter** (курсор при цьому повинен знаходитись у кінці рядка, після якого здійснюється вставляння). Якщо відбувся випадковий розрив рядка (натисненням клавіші **Enter** у середині рядка), то треба підвести курсор до кінця верхнього рядка і натиснути **Delete.**

Запуск програми. Для виконання програми треба ввійти до меню і в пункті «Run» виконати команду «Run». Система запускає спочатку транслятор, який перекладає програму з Паскаля на мову машинних кодів і шукає синтаксичні помилки в програмі. Якщо вони є, то програма не виконується і відбувається повернення до редактора. Над текстом програми з'являється червоне вікно з повідомленням про тип помилки. Після натиснення клавіші Esc вікно зникає, курсор встановлюється в рядок з помилкою. Для отримання докладної інформації про помилку треба натиснути **Ctrl** + **Fl.**

Щоб побачити результати, що повинні бути виведені на екран, потрібно в пункті меню «Windows» (вікна) вибрати підпункт «Output» (вихідне), в результаті чого відкриється вікно виведення. Розмір цього вікна, так само як і інших вікон у системі Turbo Pascal, можна змінювати.

Коли всі помилки виправлено, програма починає виконуватись. Якщо в системі вже є програма з ім'ям noname.pas, то з'являється вікно, де про це повідомляється. Якщо ім'я залишається без змін, натискується **Enter,** в результаті чого з'являється ще одне вікно, в якому запитується, чи буде програма з таким іменем записана поверх тієї, що вже є. Якщо користувач із цим погоджується, він натискує Y (yes — так). Якщо ім'я програми потрібно змінити, нове ім'я треба ввести в першому вікні, яке з'явиться, стерши noname.pas і записавши нове ім'я. Після цього програма починає виконуватись.

Програму можна модифікувати і виконувати скільки завгодно разів. Щоб помістити у вікно редактора програму, яка знаходиться на диску, треба виконати команду «Load» із пункту меню «File» (або натиснути F3). При цьому з'яв-

ляється вікно, в якому набирається ім'я файла. Якщо замість цього натиснути Enter, то з'явиться список файлів з розширенням *.pas,* із якого можна вибрати потрібний файл.

ЗАПИТАННЯ І ЗАВДАННЯ

1. Що таке Turbo Pascal?
2. Яка послідовність дій при наборі програми на комп'ютері?
3. Як здійснити запуск програми на виконання?

Умовний оператор

ДЛЯ запису алгоритмів використовуються три типи команд: присвоювання, розгалуження і повторення. Команді розгалуження в Паскалі відповідає *умовний оператор.*

Умовний оператор може мати одну з двох структур, поданих на рис. 36 і 37. На рис. 36 показано неповну форму умовного оператора, коли дія здійснюється тільки тоді, коли виконується записана в ромбі умова. У разі невиконання умови відбувається перехід до наступного оператора (вихід із структури). На рис. 37 зображено повну форму умовного

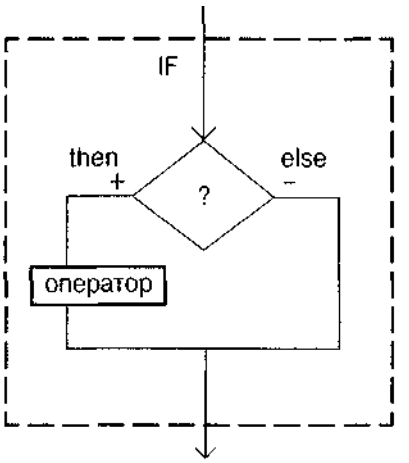

Рис. 36. Неповна форма умовного оператора

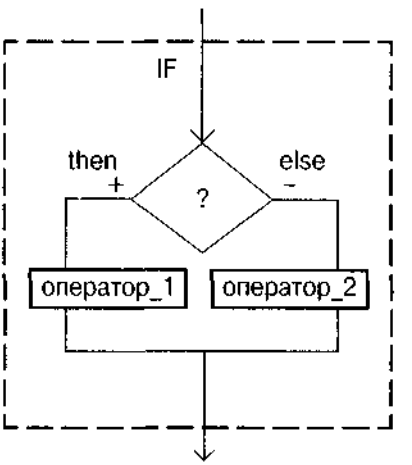

Рис. 37. Повна форма умовного оператора

оператора: при виконанні умови (вихід « + ») виконується одна дія, при невиконанні (вихід « — ») — інша дія. Кожна структура має один вхід і один вихід. Програму рекомендується будувати з послідовних, логічно завершених блоків. При цьому не допускається передача управління між блоками інакше як через вихід з одного блока та через вхід до іншого. У побудованій таким чином програмі буде менше помилок при розробці і її легше перевіряти на правильність виконання.

Неповний умовний оператор має вигляд

```
if умова then оператор;
```

Повний умовний оператор записується у вигляді

```
if умова then оператор_1 else оператор_2;
```

Якщо після слів then і else треба записати не один оператор, а кілька, тоді ці оператори беруться в так звані операторні дужки: відкриваюча дужка — begin, і закриваюча — end:

```
begin
  {оператори}
end;
```

Перед словом else крапка з комою не ставиться. Кожну пару begin...end записують в одному стовпці: так легше перевірити наявність для кожної відкриваючої операторної дужки відповідної закриваючої.

Приклади умовного оператора:

```
if a < b then y := x;
if x < 0 then x := -x;  {зміна знака змінної x}
if a + b < c then
begin
  z := x; {обмін значеннями змінних x і y }
  x :=y;
  y :=z
end;
```

В умовному операторі може бути інший умовний оператор. Наприклад:

```
if sqr(x) + sqr(y) > 1 then
  if x > y then z := 0
  else z := 1;
```

При такій формі запису, яка використовує зсув праворуч для кожної внутрішньої дії, легко зрозуміти, до якого з двох слів if відноситься слово **else.** Якщо цей оператор записати в один рядок, то відповідь буде неоднозначною. Транслятор працює таким чином. Зустрівши складну конструкцію з вкладених умовних операторів, він аналізує її з кінця, приписуючи останнє знайдене **else** першому зустрінутому при перегляді справа наліво if.

Розглянемо приклад програми з використанням умовного оператора. Нехай для двох цілих чисел треба визначити, якими вони є — парними чи непарними. Для перевірки парності використовуємо умову: залишок від ділення парного числа на два дорівнює нулю. Програма матиме вигляд:

```
program E3;
var a,b: integer;
begin
  writelnf введіть два цілих числа');
  readln(a.b);
  if a mod 2 = 0 then writeln (a - парне')
  else writeln ('a - непарне');
  if b mod 2 = 0 then writeln ('b - парне')
  else writeln ('b - непарне')
end.
```

Логічні вирази. Подана на рис. 34 схема алгоритму розв'язування квадратного рівняння містить перевірку умови $d < 0$. Дві величини d і 0 зв'язані відношенням < (менше) і утворюють *логічний вираз.* Якщо умова виконується, то кажуть, що відповідний вираз істинний; якщо умова не виконується, то вираз хибний. Для побудови складених умов у мові Паскаль існують логічні операції **and** (і), or (або) і **not** (ні). Позначивши істинне значення 1 і хибне 0, побудуємо таблиці істинності для цих логічних операцій (табл. 5 — 7). Розглянемо приклади побудови складених логічних виразів.

1. Нехай потрібно визначити, чи належить точка з координатою x відрізку [а; *Б].* Якщо записати цю умову подвійною

Таблиця 5

X	Y	X and Y
1	1	1
1	0	0
0	1	0
0	0	0

Таблиця 6

X	Y	X or Y
1	1	1
1	0	1
0	1	1
0	0	0

Таблиця 7

X	not X
1	0
0	1

нерівністю, то ця нерівність читається так: *x* менше або дорівнює *Ь* і більше або дорівнює *a (a<x<b).* Відношення «менше або дорівнює» у Паскалі записується двома знаками.

Аналогічно записується і «більше або дорівнює». Однак у Паскалі не можна записувати подвійну нерівність. Використовуючи логічну операцію «і», запишемо:

```
(x >= a) and (x <= Ь)
```

Вирази, між якими стоїть логічна операція, беруться в круглі дужки.

2. Нехай є прямокутний отвір із сторонами *a* і Ь і цеглина з ребрами *x, y, z.* Потрібно скласти умову входження цеглини в отвір (рис. 38).

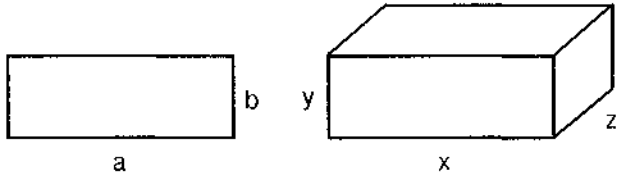

Рис. 38. «Отвір» і «цеглина»

Цеглина пройде в прямокутний отвір, якщо буде виконана така складна умова:

```
(x <= a) and (y <= b) or
(y <= a) and (x <= b) or
(x <= a) and (z <= b) or
```

```
(z <= a) and (x <= b) or
(y <= a) and (z <= b) or
(z<=a) and (y <= b)
```

Для трьох граней одержано шість умов тому, що кожну грань можна повернути на 90° і перевірити для кожної грані два випадки.

3. Нехай потрібно визначити, чи належить точка з координатами *x,y* трикутнику *ABC* (рис. 39). Сторони трикутника є відрізками прямих, кожна з яких ділить площину на дві частини. Для кожної прямої визначимо півплощину, в якій знаходиться трикутник. Півплощина, яка знаходиться вище осі *x,* визначається нерівністю y>0. Півплощина, яка знаходиться справа від прямої і з'єднує точки (— 1,0) та (0,2), задається нерівністю *y— 2x —* 2<0. Півплощина, яка знаходиться зліва від прямої і з'єднує точки (1,0) та (0,2), задається нерівністю *y+2x* — 2<0. Отже умова належності точки *(x,y)* фігурі має такий вигляд:

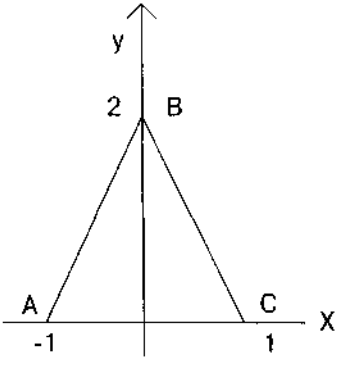

Рис. 39

```
(y > 0) and (y - 2*y - 2 < 0) and (y + 2*x - 2 < 0)
```

4. Наведемо приклад програми визначення існування трикутника із сторонами a, Ь і c. Умова існування трикутника відома з геометрії: сума двох будь-яких сторін повинна бути більша третьої. Отже, для всіх сторін умова «сума двох більше третьої» повинна виконуватись. Програма матиме вигляд:

```
program E4;
var a,b,c: real;
begin
  writeln('введіть довжини трьох сторін трикутника ');
  readln(a,b,c);
  write('трикутник із сторонами ', a,Ь,c);
  if (a + b > c) and (b + c > a) and (c + a > b)
```

```
  then writeln('існує')
  else writeln('Не існує')
end.
```

ЗАПИТАННЯ І ЗАВДАННЯ

1. Як транслятор аналізує вкладені умовні оператори?
2. Як виконується неповний умовний оператор?
3. Як перевірити, чи є ціле число непарним?
4. Як виконуються логічні операції «і», «або», «ні»?
5. Напишіть програми на Паскалі для розв'язування таких задач:
   а) Дано три числа a, b і c. З'ясуйте, чи має місце $a<b<c$. Відповідь подати у вигляді тексту: «так» чи «ні».
   б) Дано додатні числа a, *b,* c, x. З'ясуйте, чи пройде цеглина з ребрами a, *b, c* у квадратний отвір із стороною x.
6. З'ясуйте, чи належать числа a і b проміжку (-1;1).
7. Присвоїти z значення більшого з чисел x і y у тому разі, якщо $x < 0$, і меншого, якщо $x > 0$.
8. Дано три дійсних числа. Вибрати ті з них, які належать відрізку [1;3].
9. Присвоїти змінній a значення найбільшого з трьох заданих чисел.
10. Дано два числа. Вивести перше з них, якщо воно більше другого, і обидва числа, якщо це не так.
11. Перевірити, чи є серед трьох заданих чисел рівні.
12. Дано два дійсних числа. Менше з них замінити півсумою цих чисел, а більше - їх добутком.
13. Обчисліть найменше з трьох заданих чисел.
14. Знайти розв'язок рівняння $ax + b = 0$, якщо він існує.
15. Дано два дійсних числа x та y. Якщо число x менше нуля, то z присвоїти значення більшого з двох чисел x і y, в противному разі - z присвоїти значення півсуми цих чисел.
16. Дано три дійсних числа. Знайти найбільші значення їх попарних сум і добутків.
17. Дано дійсні числа a, *b* і c. Подвоїти ці числа, якщо вони впорядковані за зростанням.

2.6. Організація циклів

Циклом у програмуванні називається повторення одних і тих самих дій. Цикл повинен закінчуватись за якої-небудь умови. Перевіряти цю умову можна на початку кожного по-

вторення, і в цьому разі цикл називається «доки». При перевірці умови в кінці кожного повторення цикл називається «до». Схеми цих двох типів циклів наведені на рис. 40.

У циклі «доки» спочатку перевіряється умова і якщо вона виконується, тобто логічний вираз істинний, то виконується оператор і знову перевіряється умова. Записана в циклі «доки» умова є *умовою продовження циклу*. Як тільки вона перестане виконуватись, цикл завершиться. На рис. 40 вихід із ромба « + » (або «так») означає виконання умови повторення циклу, «—» (або «ні») — невиконання. Цикл «доки» не виконається жодного разу, якщо умова при вході в структуру відразу виявиться хибною. Таким чином, цикл «доки» містить умову продовження повторення, а цикл «до» — *умову закінчення повторення*. Обидві структури мають один вхід і один вихід. Однак цикл «до» завжди виконується хоча б один раз, тому що умова перевіряється після виконання операторів, що входять до циклу. Це ускладнює перевірку правильності програми, тому краще використовувати цикл «доки». Оператор у циклі може бути простим або складеним, тобто являти собою групу операторів, обмежених операторними дужками begin...end. Його називають тілом циклу. Цикли можна організовувати, використовуючи різні засоби мови Паскаль.

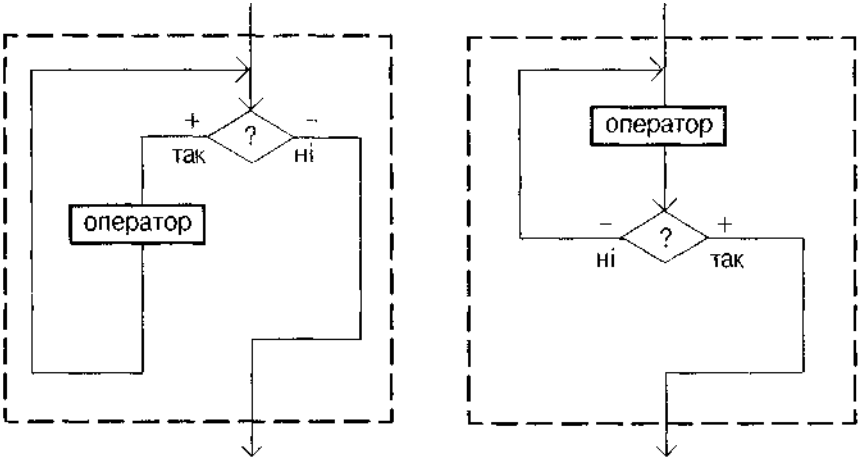

Рис. 40. Циклічні структури

Оператор безумовного переходу. Цей оператор дозволяє перейти без перевірки умови або на один з попередніх операторів, або на один із наступних, тобто змінити порядок виконання команд. Перед оператором, на який здійснюється безумовний перехід, потрібно поставити мітку — ціле число, що складається не більше ніж із чотирьох цифр. Від оператора мітка відділяється двокрапкою.

Перехід до поміченого оператора здійснюється за допомогою *оператора безумовного переходу,* який має вигляд

```
goto n;
```

де *n* — мітка. Мітки повинні бути визначені в описовій частині програми в розділі **label.**

Організація циклів за допомогою операторів умовного і безумовного переходів. Нехай потрібно знайти найбільший спільний дільник двох натуральних чисел *A* і *B.* Скористаємось алгоритмом Евкліда: будемо зменшувати кожний раз більше з чисел на величину меншого доти, доки обидва значення не стануть рівними. У таблиці 8 подано алгоритм Евкліда.

Таблиця 8

Вхідні дані	Перший крок	Другий крок	Третій крок	НСД(A,B)=5
A = 25	A = 10	A = 10	A = 5	
B = 15	B = 15	B = 5	B = 5	

Програма матиме вигляд:

```
program E5;
label 1,2;
var a,b: integer;
begin
  write('введіть два натуральних числа');
  readln(a.b);
1:if a = b then goto 2;
  if a > b then a := a - b
  else b := b - a;
  goto 1;
2:writeln ('НСД =', a)
end.
```

Оператор циклу «доки». Як видно з попереднього прикладу, циклічний процес можна організувати без використання спеціальних операторів. Однак при складанні досить складних програм використовувати оператор безумовного переходу не рекомендується, оскільки можна швидко заплутатись при перевірці програми. Оператор циклу «доки» має вигляд:

while умова do оператор;

і виконується таким чином: оператор (тіло циклу) повторюється доти, доки виконується умова (істинний логічний вираз). Оператор може бути простим або складеним, вміщеним в операторні дужки **begin...end.**

Для алгоритму Евкліда програма набуде вигляду:

```
program E6;
var a,b: integer;
begin
  write('введіть два натуральних числа');
  readln(a,b);
  while a<>b do
  if a>b then a := a-b
  else b := b-a;
  writeln ('НСД=', a)
end.
```

Оператор циклу «до». Перевірка умови в циклі «до» здійснюється після виконання оператора. Якщо умова в циклі «доки» є умовою продовження повторень, то умова в циклі «до» — умовою виходу з циклу, його завершенням. Тому для тієї самої задачі ці умови протилежні.

Загальний вигляд оператора:

repeat оператор until умова;

Між словами **repeat** (повторити) і **until** (до того часу, поки) можна записати будь-яку кількість операторів без використання операторних дужок. Перед словом **until** не ставиться крапка з комою.

Програма знаходження найбільшого спільного дільника набуде такого вигляду:

```
program E7;
var a,b: integer;
begin
  write('введіть два натуральних числа');
  readln(a,b);
  repeat
  if a > b then a := a - b;
  if b > a then b := b - a
  until a = b;
  writeln('НСА=', a)
end.
```

Оператори циклів «перелік». При виконанні програм знаходження найбільшого спільного дільника кількість повторень неоднакова для різних даних. Коли відоме число повторень, зручно використовувати цикл «перелік». У мові Паскаль є два оператори для організації циклів! — *прямий* і *зворотний* «перелік». Прямий перелік іде від відомого меншого числа до відомого більшого, на кожному кроці додається одиниця (наприклад, від 120 до 140: 121, 122,: 123, ..., 139, 140). Опис оператора прямого переліку має вигляд

for i := n1 to n2 do оператор;

(читається «для *і* починаючи з n1 до л2 виконати оператор»).

Змінна *і* називається *змінною циклу,* яка при прямому переліку завжди змінюється від меншого значення до більшого. Треба звернути увагу, що n1 повинно бути не більше ніж л2 (л1<л2). При $i = n\backslash$ цикл виконується перший раз. Потім до значення змінної *і* додається одиниця і здійснюється перевірка, чи не стало отримане значення більшим ніж л2. Якщо i+1<n2, то оператор виконується, а якщо i+ 1>л2, то відбувається вихід із циклу і виконується оператор програми, який слідує за оператором циклу. Оскільки оператор циклу **for** сам змінює значення змінної циклу, то її не можна змінювати іншим способом, наприклад присвоюванням їй якого-небудь значення в тілі циклу (вона не повинна з'явитися зліва від знака : =).

Оператор у циклі може бути простим або складеним, взятим в операторні дужки.

Розглянемо приклади використання операторів циклу.

1. Обчислення добутку *a* . Відомо, що для того щоб отримати цілий степінь *n* числа *a,* потрібно помножити його самого на себе *n* разів. Цей добуток при виконанні програми буде зберігатися в ділянці пам'яті з іменем p. Щоразу, при черговому циклі, з цієї ділянки буде зчитуватись попередній результат, помножуватись на основу степеня *a* і знову записуватись у ділянку p. Основний оператор у тілі циклу повторюється n разів і має вигляд:

```
p := p*a;
```

При першому виконанні циклу в ділянці p повинно знаходитись число, що не впливає на множення, тобто перед циклом у цю ділянку треба записати одиницю.

Програма має вигляд:

```
program E8;
var a,p: real;  i,n: integer;
begin
  write('введіть a - основу степеня, a = ');
  readln (a);
  write('введіть  ціле n - показник степеня, n = ');
  readln(n);
  p:=1;
  for i:=1 to n do
    p :=p*a;
  writeln('p = ', p)
end.
```

У таблиці 9 відображено протокол виконання програми при піднесенні числа 2 до п'ятого степеня: $a = 2$, $n — 5$.

Таблиця 9

i		1	2	3	4	5
p	1	2	4	8	16	32

Протокол (таблиця), заповнений вручну, використовується для перевірки всіх етапів роботи програми — *тестування.* Для перевірки роботи програм, алгоритми яких реалізують методи розв'язування складних задач, використ-

товують контрольні приклади (тести). Програма виконується комп'ютером і звіряються всі проміжні дані, отримані при обчисленні, і кінцеві результати. Для розуміння роботи програми і виконання окремих операторів потрібно заповнювати подібні протоколи.

2. Обчислення *p = n!* (*n* факторіал).

За означенням $n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$. Використовуючи попередню програму, обчислимо p як добуток чисел від *1* до *n,* тобто p кожний раз множиться не на одне й те саме число, а на значення змінної циклу. Програма матиме такий вигляд:

```
program E9;
var p,i,n: integer;
begin
  write('введіть ціле n=');
  readln(n);
  p:=1;
  for i:=1 to n do
    p:=p*i;
  writeln(n, '!=', p)
end.
```

3. Складання таблиці значень функції y=sin *x*. Нехай треба скласти таблицю значень функції *y=sin x* на відрізку [0; 3.14] з кроком 0.1. Щоб не визначати кількість повторень обчислень, можна скористатися циклом «доки». Використовуючи виведення дійсних чисел із фіксованою крапкою, визначимо, що кількість цифр після крапки в дробовій частині значення функції дорівнюватиме п'яти. Тоді подання всього числа, враховуючи область значень синуса, займе сім позицій (числа додатні, тому додається позиція для десяткової крапки і цілої частини числа).

Програма матиме вигляд

```
program E10;
var x,y: real;
begin
  x:=0;
  writeln('x':10,'sin x':10);
  while x <= 3.14 do
    begin
```

```
    y := sin(x);
    writeln(x:10, '   ', y:7:5);
    x := x + 0.1
  end
end.
```

При кожному виконанні циклу спочатку перевіряється умова *x<=3.14,* потім обчислюється значення функції, друкується аргумент *x* (для нього відведено десять позицій, із них одна — для цифри дробової частини) і, через три пропуски — значення функції, і нарешті, для наступного кроку циклу обчислюється нове значення аргументу *(x* збільшиться на 0.1). Цикл «доки» дозволяє змінювати змінну циклу як завгодно, збільшуючи її або зменшуючи на будь-яке значення.

4. Додавання чисел. При додаванні кількох чисел необхідно накопичувати результат у певній ділянці пам'яті, щоразу зчитуючи з цієї ділянки попереднє значення суми і додаючи до нього наступний доданок. Нехай відомо, що будуть додаватися n чисел. У такому випадку треба n разів виконати дію

s := s + a;

де *a* — наступне число, що вводиться з клавіатури. Для першого виконання цього *оператора присвоювання* потрібно з ділянки з іменем *s* взяти таке число, яке не впливало б на результат додавання. Отже, перед початком циклу слід присвоїти змінній s значення нуль.

Програма має вигляд

```
program E11;
var a,s: real; i,n: integer;
begin
  write ('введіть кількість доданків n=');
  readln(n);
  s := 0;
  for i:= 1 to n do
    begin
      write(i,'- е число =');
      readln(a);
      s := s + a
    end;
```

```
  writeln('сума s =',s)
end.
```

Якщо кількість чисел невідома, то можна задати число-обмежувач, наприклад нуль. У такому разі використовується цикл while або repeat:

```
s := 0; readln (a);
while a <> 0 do
  begin  s := s + a;
   readln(a)
  end;
```

```
s := 0;
repeat
  readln (a);
  s := s + a
until a = 0;
```

Оператор циклу зворотного переліку виконується аналогічно до опера гора циклу прямого переліку, але змінна циклу не збільшується на одиницю з кожним кроком, а зменшується. Цей оператор має вигляд

```
for i := n2 downto n1 do оператор;
```

Для цього оператора повинна також виконуватись умова n2>n1.

При використанні в програмі операторів циклу необхідно дотримувати таких правил:

1) всередині циклу може знаходитись інший цикл, але цикли повинні мати різні змінні і внутрішній цикл повинен повністю знаходитись у тілі зовнішнього циклу;
2) не можна передавати управління всередину циклу, обминаючи заголовок (це означає, що якщо всередині циклу є мітка, то оператор goto переходу до неї також повинен знаходитись всередині цього циклу);
3) якщо необхідно обійти групу операторів у тілі циклу і продовжити цикл, тобто виконати його наступний крок, то треба передати управління на кінець циклу;
4) можна достроково вийти з циклу, використовуючи або оператор goto, або змінивши параметр умови в операторах while чи repeat так, щоб цикл більше не виконувався.

ЗАПИТАННЯ І ЗАВДАННЯ

1. Нехай тіло циклу в програмі Е7 таке саме, як в програмі Е6. Як працюватиме програма, якщо ввести два однакових числа a і b?

2. Скільки разів буде виконуватися оператор циклу repeat, якщо умова після слова until істинна при входженні в цикл?
3. Поясніть, яка різниця між умовами, записаними після слів while та repeat для тієї самої задачі?
4. Напишіть програми обчислення сум:.
   а) для сорока доданків виду *n-i,* де *i*=1, 2, 3, ..., 40, а n- дане число;
   б) для *n* доданків виду *x+i,* де x - задане число, а і змінюється від 1 до *n*;
   в) для ста доданків, що мають вигляд дробу (i+1)/(i+2);
   г) для *n* доданків виду $(i+1)^2$, /=1, 2, ..., *n*;
   д) для *n* доданків sin *x* + sin x^2 + sin x^3 + ... + sin x ;
   е) для *n* доданків sin x + sin 2x + sin 3x + ... + sin *n*x;
   є) кубів *n* перших натуральних чисел.
5. Для різних цілих чисел, що вводяться з клавіатури, знайти суму додатних непарних.
6. Напишіть програми обчислення добутків:
   а) a(a+1)(a+2) ... (a+*n*-1);
   б) a(a-*n*)(a-2*n*) ... (a-*n* *n*);
   в) (x-1)(x-2)(x-3) ... (x-*n*);
   г) 2 4 6 ... (2*n*);
   д) (1+ sin 0.1)(1 + sin 0.2) ... (1 + sin 10);
   є) усіх чисел від 1 до 100 кратних 3, але не кратних 6;
   є) *n* співмножників вигляду $(x+i)^2$.
7. Для додатного числа *A* знайти серед чисел 1, 1 + 1/2, 1 + 1/3, ... перше, більше ніж *A.*
8. Ввівши числа з клавіатури без обмеження їх кількості (кінець введення - число нуль), знайти суму додатних і добуток від'ємних чисел.

2.7. Комп'ютерні обчислення

Числа, з якими доводиться мати справу в житті, бувають двох типів. Одні точно задають значення величини, інші — наближено. Перші називають *точними,* другі — *наближеними.* Досить часто використовують наближене число замість точного і цього достатньо. Інколи буває так, що є потреба в точному значенні величини (наприклад, визначення кількості токсичної речовини, при якій починається руйнація клітин в організмі), але немає можливості його отримати.

Якщо говорять про відстань між сусідніми селами, то цю відстань вказують у кілометрах. Вимірювати цю відстань можна також у метрах або навіть і в сантиметрах. Тобто відстань між селами може бути вказана з точністю до кілометра, метра або сантиметра. В усіх цих випадках числа, які характеризують відстань, є наближеними. Результат обчислень із використанням наближених чисел є наближеним числом.

Нескінченний неперіодичний десятковий дріб можна обчислити тільки наближено, із скінченною кількістю десяткових знаків, або, як прийнято говорити, — із певною точністю. Точність вказує на кількість десяткових знаків, які задовольняють користувача. Якщо число є результатом обчислень на комп'ютері, то кількість десяткових знаків обмежується розміром пам'яті, яка виділяється для цього числа.

Дійсні числа, що використовуються в мові Паскаль, займають шість байтів у пам'яті комп'ютера. Тому точність подання дійсних чисел обмежена розміром даної пам'яті і не перевищує 11 десяткових знаків.

Комп'ютери дозволяють обчислювати значення таких чисел з різним ступенем точності, але ця точність не може бути вищою ніж комп'ютерна — не можна отримати більшу кількість цифр у числі, ніж уміщується у відведеній для числа пам'яті.

Розглянемо приклади обчислень, які виконуються на комп'ютері, із заданою точністю.

1. Обчислення кореня квадратного з даного числа. Точному значенню кореня квадратного з числа A відповідає точка на числовій прямій. Візьмемо деяке наближене значення цього числа ліворуч від даної точки: нехай це буде x_0 (рис. 41).

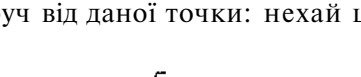

Рис. 41. Вибір наближеного значення кореня

Тоді значення A/x_0 на числовій прямій відповідає точці, яка знаходиться праворуч від точного значення кореня.

За перше наближення ірраціонального числа молена взяти півсуму чисел x_0 і A/x_0:

$$x_1 = 1/2\ (x_0 + A/x_0).$$

Продовжуючи міркування, дістанемо послідовність значень $x_1, x_2, \ldots, x_n$. Загальний вигляд формули, за якою проводять обчислення, такий:

$$x_{n+1} = 1/2\ (x_n + A/x_n).$$

Обчислювальний процес, при якому кожне наступне значення визначається з використанням одного або кількох попередніх значень, називається *ітераційним.* Розглянутий алгоритм, за допомогою якого можна одержати наближене значення кореня квадратного з даного числа, був відомий ще грецькому математику Герону (близько I ст. н.е.), і називається *алгоритмом Герона.*

Можна узагальнити наведену вище формулу для кореня m-го степеня:

$$x_{n+1} = 1/m\ ((m-1)x_n + A/x_n^{m-1}).$$

Як же визначити, коли буде досягнута потрібна точність обчислень? Для ітераційних процесів одна з ознак досягнення заданої точності може бути така: модуль різниці двох сусідніх значень не повинен перевищувати задану *похибку* є (читається «епсілон»):

$$|x_{n+1} - x_n| \le \varepsilon.$$

Оскільки в мові Паскаль немає грецької літери є, позначимо похибку обчислень через *eps.* У програмі Е12 використовуються всього дві змінні: *x* і *x2,* але обчислення можуть повторюватися досить велику кількість разів (залежно від заданої точності). При побудові послідовності уточнених значень невідомо, скільки її членів буде одержано при заданому значенні похибки.

Отже, невідомо, який об'єм пам'яті потрібен для зберігання всіх цих даних. Використовуючи той факт, що на кожному кроці обчислень необхідно знати значення тільки одного, попереднього, члена одержуваної послідовності, будемо щоразу нове значення записувати в ділянку з іменем x1, а попереднє брати з ділянки з іменем x0. Щоб на кожному кроці обчислень використовувалось щойно одержане значення, будемо заносити *x0* щоразу до ділянки x1:

```
x1 := x0;
```

Тоді, незалежно від числа повторень і отриманих при цьому членів послідовності, у програмі можна обмежитись двома змінними. За початкове значення кореня можна взяти будь-яке число (крім нуля), наприклад одиницю.

Програма наближеного обчислення кореня квадратного з числа *A* із заданою точністю *eps* може бути такою:

```
program E12;
var A,eps,x0,x1: real;
begin
  write('введіть додатне число A =');
  readln (A);
  write('введіть точність обчислень');
  readln(eps);
  x0 := 1;
  x1 := 0.5*(x0 + A / x0);
  while abs(x1 – x0) > eps do
    begin
      x0 := x1;
      x1 := 0.5*(x0 + A / x0)
    end;
  writeln('корінь квадратний з числа ', A, '=', x1)
end.
```

2. Обчислення значень тригонометричних функцій. Відомо, що багато функцій, у тому числі й тригонометричних, можна подавати у вигляді нескінченної суми. Для того щоб обчислити значення такої суми із заданою точністю, потрібно взяти певну кількість доданків. Кількість доданків залежить від заданої точності обчислень.

Наприклад, функцію sin *x* можна подати у вигляді

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots + (-1)^{n-1}\frac{x^{2n-1}}{(2n-1)!} + \ldots, \quad n = 1, 2, 3, \ldots$$

Це представлення зручно використовувати для обчислення значень синуса при $0 \le x \le \pi/4$.

Для обчислення одного за іншим доданків розглянутої суми, що включають степені і факторіали, використовують-

ся залежності між послідовними значеннями. Це дозволяє на кожному кроці одержувати наступний доданок, знаючи попередній, для чого досить попередній доданок помножити на деяке число.

Для отримання залежності між двома сусідніми доданками запишемо два послідовних доданки в загальному вигляді і поділимо наступний на попередній. У результаті отримаємо:

$$a_n = (-1)^{n-1}\frac{x^{2n-1}}{(2n-1)!};\quad a_{n+1} = (-1)^{n+1}\frac{x^{2n+1}}{(2n+1)!} = -a_n\frac{x^2}{2n(2n+1)}.$$

Обчислення припиняються, коли модуль різниці двох послідовних сум буде меншим або дорівнюватиме заданій похибці. Різниця двох послідовних сум дорівнює наступному доданку, тому умова закінчення обчислень запишеться так: $|a_{n+2}| < \varepsilon$.

Використаємо в програмі наближеного обчислення значення sin *x* одну змінну *a* для обчислення кожного доданка. У програмі зручно ввести змінну я2, яка на кожному кроці буде збільшуватися на 2. При цьому програма матиме вигляд:

```
program E13;
var x,a,s,eps: real;  n2: integer;
begin
  write("x=");
  readln(x);
  write("eps = ");
  readln(eps);
  a := x;
  s := x;
  n2 := 1;
  repeat
    a := -a * sqr(x)/((n2+1)*(n2+2));
    s := s + a;
    n2 := n2 + 2
  until abs(a) < eps;
  writeln("sin ", x, "= ", s)
end.
```

3. Уточнення кореня рівняння методом ділення відрізка навпіл. Нехай задано рівняння

$$x^2 - \cos x + 1/2 = 0.$$

У шкільному курсі математики розглядаються методи відшукання точних розв'язків деяких типів рівнянь: квадратних, тригонометричних, показникових тощо. Якщо ж алгебраїчне або трансцендентне рівняння досить складне (як у розглянутому випадку), то в більшості випадків знайти його точний розв'язок неможливо. Тому важливого значення набувають методи наближеного обчислення коренів рівняння. Одним із таких методів є метод ділення відрізка навпіл. Розглянемо рівняння

$$f(x) = 0,$$

де функція *f[x)* визначена і неперервна на деякому інтервалі $a<x<b$.

Будемо виходити з припущення, що рівняння $f(x) = 0$ має лише ізольовані корені, тобто для кожного кореня рівняння існує деякий відрізок на осі *Ox,* який не містить інших коренів цього рівняння.

Наближене обчислення ізольованих дійсних коренів рівняння складається з двох етапів:

1) відокремлення коренів, тобто встановлення невеликих (по можливості менших) проміжків, кожний з яких містить тільки один корінь рівняння;

2) уточнення наближених коренів, тобто проведення обчислень до досягнення наперед заданої точності.

Якщо функція. *f(x)* набуває нульового значення, то це означає, що або вона перетинає вісь *Ox* у деякій точці, а знак функції змінюється з мінуса на плюс (рис. 42), або з плюса на мінус, або дотикається осі *Ox* у цій точці. Якщо кратних коренів немає, тобто рівняння має лише ізольовані корені, тоді графік функції $y = f(x)$ не може дотикатися осі *Ox.*

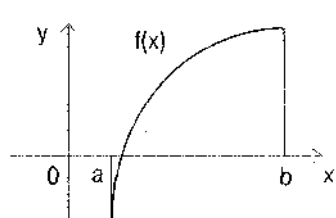

Рис. 42

Для відокремлення коренів можна використати графік функції *f(x)*. Графік можна побудувати досить наближено, щоб за ним можна було визначити кінці інтервалу, який містить корінь. Потім треба переконатися, що на даному інтервалі функція дійсно змінює знак, тобто слід обчислити *f*(*a*) і *f*(*b*).

Для уточнення кореня обчислимо значення функції *f(x)* у середній точці відрізка:

$$c := (a + b)/2;$$

Якщо в точці с функція *f(x)* має такий самий знак, як і в точці a, перенесемо точку a вправо в точку *c* (при цьому змінна *a* набуде значення с, тобто *a: = c)*. Якщо в точці *c* функція має такий самий знак, як і в точці b, перенесемо точку b вліво в точку *c (b: = c)*. Після перенесення точки новий відрізок ділиться навпіл і знову визначається знак функції в його середній точці.

Обчислюючи значення кореня за описаним методом, переносячи кінці відрізка в нові точки і тим самим поступово звужуючи відрізок, отримуємо послідовні наближення до точного значення шуканого кореня рівняння — координати точки перетину графіка функції *f(x)* з віссю абсцис. Обчислення продовжуються доти, доки не буде досягнута задана точність.

Запишемо програму відшукання наближеного розв'язку рівняння *f(x)* = 0 за методом ділення відрізка навпіл для конкретної функції

$$f(x) = x^2 - \cos x + 1/2 \quad \text{на відрізку } [0; 1]:$$

```
program E14;
var a,b,c,eps: real;
begin
  a := 0;  b := 1;  eps := 0.0001;
  while abs(b–a) > eps do
    begin c := ( a + b ) / 2;
      if (a*a–cos(a)+0.5)*(c*c–cos(c)+0.5)>0
      then a := c
      else b := c
    end;
```

```
writeln("корінь рівняння f(x)=0 на відрізку ", a, ",", b,
        " дорівнює ", (a+b)/2)
end.
```

ЗАПИТАННЯ І ЗАВДАННЯ

1. Поясніть ідею методу обчислення наближеного значення кореня квадратного з числа *A,* використовуючи алгоритм Герона.
2. Напишіть програму обчислення кореня п'ятого степеня з числа *A.*
3. Як визначити кількість ітерацій при обчисленні sin x у програмі Е13?
4. Поясніть, чому достатньо двох величин для обчислення будь-якої кількості членів послідовності в програмі Е12?
5. Чому точність обчислень на комп'ютері не може бути довільною?
6. Чи можна вибрати відрізок для уточнення кореня, якщо на кінцях відрізка функція має однакові знаки?
7. Чи можна сказати, що відрізок [-5; +5] для функції $f(x)=3x^2-5x-3$ містить ізольовані корені?
8. Напишіть програму обчислення кореня рівняння для функції $f(x)=0$ із програми Е14 на відрізку [0; 1] методом ділення відрізка навпіл 20 разів. Визначіть, із якою точністю буде обчислений корінь рівняння.
9. Використовуючи програму Е12, обчисліть корені квадратні з чисел 1.45, 1.678, 0.2, 131 з точністю 1Е-6.
10. Використовуючи програму Е13, обчислити значення sin x при $x = \pi/8$ із точністю 1Е-5. Число π задати як константу 3.141592.

2.8. Масиви

У розглянутих раніше прикладах програм проводилась обробка одиничних даних — значень простих змінних. При розв'язуванні практичних задач часто доводиться аналізувати не окремі значення, а деякі сукупності таких значень (наприклад при табличному заданні функцій). Такі дані об'єднуються в різні структури даних, найпростішими з яких є масиви. *Масив* — це впорядкований іменований набір із фіксованої кількості однотипних даних. Доступ до будь-якого елементу масиву здійснюється за його номером. У масиви можна об'єднати результати експериментів, списки прізвищ співробітників, різні складні структури даних. На-

приклад, список прізвищ учнів у класному журналі 10-А є масивом. У масиві дані розрізняються за своїми порядковими номерами (індексами). Якщо кожний елемент масиву визначається за допомогою одного номера, то такий масив називається *одновимірним,* якщо за двома — то *двовимірним.* Двовимірний масив — це таблиця з рядків і стовпців. У таблицях перший номер вказує на рядок, а другий — на положення елемента в рядку. Усі рядки таблиці мають однакову довжину.

Одновимірний масив може бути набором чисел, сукупністю символьних даних чи елементів іншої природи (навіть масив масивів). Так само, як і в послідовності, в одновимірному масиві можна вказати елемент з конкретним номером, наприклад a_5, або записати загальний вигляд елемента, використовуючи як індекс змінну *i,* вказуючи діапазон її зміни:

$$a[i],\ i=1, 2, \ldots, n.$$

Задачі, в яких використовуються масиви, можуть мати різне формулювання. Наприклад, задача починається зі слів «Дано *n* чисел...» і далі вказується, що треба зробити з цими числами. При написанні програми мовою Паскаль для розв'язання подібної задачі слід виконати такі дії:

1) визначити, які числа задані: цілі чи дійсні; якщо про це конкретно не сказано, то краще вважати їх дійсними;

2) назвати весь масив одним ім'ям (це ім'я буде використовуватись для кожного елемента, тільки до імені масиву додаватиметься номер елемента — індекс);

3) описати масив у розділі змінних var, тим самим відводячи місце в пам'яті для масиву;

4) ввести дані до пам'яті.

В описі масиву використовують спеціальне слово array (масив), після якого у квадратних дужках через дві крапки вказують діапазон змінювання номерів елементів, далі слово of (із) і тип даних масиву. Коли транслятор зустрічає опис масиву, він відводить для нього стільки послідовних ділянок пам'яті, скільки вказано в квадратних дужках, і такої довжини, яка відповідає типу даних у масиві. Здебільшого номери елементів змінюються від 1 до заданого числа л, яке є останнім значенням (верхньою межею) номера елемента ма-

сиву. Значення n можна задати в розділі констант **(const).** Приклад опису масиву:

```
const n = 10;
var a: array [1..n] of real;
```

Цей опис означає, що для масиву *а* буде відведено десять ділянок оперативної пам'яті по шість байтів кожна. Мовою Паскаль імена цих ділянок записуються так:

a[1], a[2], a[3], a[4], a[5], a[6].

В описі після імені масиву *а* ставиться двокрапка, після якої вказується тип даного — масив. Якщо в програмі кілька масивів одного розміру і типу, то, як і для простих змінних, їх імена можна дати через кому, а потім, після двокрапки, вказати опис масиву.

Для введення даних у пам'ять необхідно організувати цикл. Введення значень елементів описаного масиву *а* може мати вигляд:

```
for i := 1 to n do read(a[i]);
```

Значення, що вводяться, набираються на клавіатурі і відокремлюються одне від одного пропусками, після чого натискається **Enter.** Введення можна прокоментувати і вводити кожний елемент в окремому рядку після підказки:

```
for i := 1 to n do
begin
  write('a [', i, '] = ');
  readln(a[i])
end;
```

При обробці масивів розв'язування багатьох задач базується на простіших задачах: обчислення суми (добутку) елементів масиву; знаходження найбільшого (найменшого) елемента; упорядкування елементів за зростанням або спаданням. Розглянемо ці базові задачі.

1. Обчислення суми елементів масиву. Знаходження суми елементів масиву нічим не відрізняється, по суті, від додавання значень простих змінних (програма Е11), Розв'язування задачі поділяється на три основні етапи:

1) введення даних;
2) обчислення суми;
3) друкування результатів.

Програма матиме вигляд:

```
program E15;
const n = 7;
var a: array [1..n] of real; s: real; i: integer;
begin
  write ('введіть елементи масиву – ');
  write (n,' дійсних чисел через пропуск');
  for i := 1 to n do
    read(a[i]);
  s := 0;
  for i :=1 to n do
    s := s + a [i];
  writeln;
  writeln(' сума елементів масиву s = ',s)
end.
```

Результати виконання цієї програми наведено в таблиці **10.**

Таблиця 10

Вхідні дані: 3, –2, 7, 9, –1, 6, 1							
i	1	2	3	4	5	6	7
a[i]	3	–2	7	9	–1	6	1
S	3	1	8	17	16	22	23

2. Знаходження найбільшого елемента масиву. Для відшукання найбільшого числа серед сукупності чисел потрібно послідовно переглядати і порівнювати між собою числа, записані в пам'яті. Уявімо, що кожне число написане на окремій картці і картки складені купкою. Перше число запам'ятаємо відразу як найбільше і перевернемо картку. Далі візьмемо наступну картку. Тепер у нашому розпорядженні є два числа: одне бачимо, друге — пам'ятаємо. Порівнюючи їх між собою, запам'ятаємо більше число, тобто якщо число, запам'ятоване раніше, більше, то запам'ятову-

вати нове число не треба, а необхідно дивитись наступну картку. Якщо друге число більше першого, перше далі пам'ятати не треба і запам'ятовуємо лише друге. Таким чином, на кожному етапі порівняння пам'ятатимемо більше з переглянутих чисел і наприкінці знайдемо найбільший елемент, тобто розв'яжемо поставлену задачу. Записавши цей алгоритм дій у вигляді відповідного набору операторів, дістанемо програму знаходження найбільшого значення. Проміжні значення шуканої величини і остаточний результат містить змінна max. Програма має вигляд:

```
program E16;
const n=7;
var a: array [1..n] of integer; max,i: integer;
begin
  for i := 1 to n do
    begin
      write('a[', i,']= ');
      readln(a[i]);
    end;
  max := a[1];
  for i := 2 to n do
    if max < a[i]
    then max := a[i];
  writeln('найбільший елемент масиву max =', max)
end.
```

3. Упорядкування масиву за зростанням. Упорядкування масивів за будь-якою ознакою називається також *сортуванням.* Існують різні методи сортування. Вони розрізняються переважно швидкістю отримання результату. Розглянемо один із них — *метод «бульбашки».* Нехай є послідовність чисел a_1, a_2, ..., a_n, яку необхідно впорядкувати за зростанням. Зафіксуємо перший елемент і будемо послідовно порівнювати його з елементами, що знаходяться справа від нього. Якщо якийсь з елементів справа виявиться меншим, ніж зафіксований, поміняємо місцями цей елемент із зафіксованим і продовжимо порівняння вже нового елемента, який знаходиться на першому місці, із тими елементами, що залишилися правіше. Якщо знову знайдеться елемент, менший зафіксованого, повторимо переставляння. У результаті

першого перегляду послідовності на першому місці буде найменший з усіх елементів, тобто він як «найлегший» наче «спливає» вгору. Звідси й назва методу — метод «бульбашки». Далі зафіксуємо другий елемент і повторимо процедуру, виконуючи при потребі переставляння елементів і т.д. З'ясувавши ідею розв'язування, зупинимося на двох питаннях: як фіксувати елементи і як здійснювати перестановку двох елементів. Для того щоб при перебиранні елементів, які знаходяться правіше від того, що перевіряється, не змінювався індекс останнього, необхідно, щоб індекси зафіксованого елемента і тих, що стоять правіше від нього, були різні: наприклад *i* та *j*. Значення індексу *i* змінюється від 1 до $n - 1$. Значення індексу j завжди більше *i* (воно змінюється від $i + 1$ до n). Для кожного значення *i* індекс *j* повинен послідовно набути всіх допустимих значень. Отже, конструкція програми, що відображає повний перебір усіх елементів та їх упорядкування за зростанням, являє собою подвійний цикл. При перестановці двох елементів використовується третя змінна. Обмін значеннями в пам'яті двох змінних *a* і b виглядає так:

$$1)\ c := a;\quad 2)\ a := b;\quad 3)\ b := c.$$

Програма сортування методом «бульбашки» має вигляд:

```
program E17;
const n = 7;
var a: array [1..n] of real; i,j: integer; c: real;
begin
  for i := 1 to n do
    begin
      write('a[', i, ']= ');
      readln(a[i])
    end;
  for i := 1 to n-1 do
    for j := i+1 to n do
      if a[i] > a[j] then
        begin
          c := a[i];
          a[i] := a[j];
          a[j] := c
        end;
```

```
    writeln('упорядкований за зростанням масив');
  for i := 1 to n do
    writeln(a[i])
end.
```

4. Пошук елемента в масиві. Одна з важливих не обчислювальних задач — пошук даного елемента серед елементів масиву. Такий пошук називається також *пошуком за ключем.* На практиці пошук здійснюється в упорядкованому масиві, алгоритми пошуку бувають різні, Розглянемо приклад, де здійснимо пошук елемента масиву шляхом суцільного перебору. Якщо елемент знайдено, надрукуємо його номер; якщо ні, то видаємо відповідне повідомлення.

Суттєвим є те, який з однакових елементів масиву, що збігається з даним, нас цікавить: той, що трапився першим при пошуку чи останнім. У даному прикладі будемо шукати перший. Пошук здійснюється в циклі: як тільки елемент знайдено, треба припинити пошук інших елементів (вийти з циклу). Для дострокового виходу з циклу **for** використовуємо оператор **goto.** Якщо достроковий вихід не відбудеться, то елемента, що дорівнює даному, в масиві немає. У такому разі повідомлення про відсушість елемента має бути згаданим відразу після закінчення пошуку. Для цього треба використовувати ще один оператор **goto** і ще одну мітку в програмі. Програма пошуку даного елемента в масиві може мати вигляд:

```
program E18;
label 1,2;
const n = 7;
var a: array [1.. n] of real; x: real; i: integer;
begin
  writeln('введіть елементи масиву');
  for i := 1 to n do
    read(a[i]);
  writeln;
  write('введіть число для пошуку в масиві:  x =');
  readln(x);
  for i := 1 to n do
    if a[i] = x then goto 1;
  writeln('такого числа в масиві немає');
```

```
  goto 2;
1:writeln('номер елемента масиву, що дорівнює
  даному ', i)
2:end.
```

Якщо шукати не перший елемент, що дорівнює заданому, а останній, то зручно використати цикл зворотного переліку:

```
for i := n downto 1 do
```

ЗАПИТАННЯ І ЗАВДАННЯ

1. Як описується масив мовою Паскаль?
2. Для чого потрібний опис масиву?
3. Що треба зробити, щоб почати розв'язувати на комп'ютері задачу, формулювання якої починається зі слів: «Дано n чисел...»?
4. За якими ознаками дані об'єднуються в масиви?
5. Чи можна в прикладі програми Е15 обмежитися одним оператором циклу?
6. Що треба змінити в програмі Е16, щоб здійснився пошук не найбільшого, а найменшого елемента масиву?
7. Які зміни в програму Е16 треба внести, щоб одночасно із значенням найбільшого числа визначався його порядковий номер?
8. Поясніть роботу подвійного циклу в програмі Е17.
9. Змініть програму Е18 так, щоб замість циклу «перелік» при пошуку елемента використовувався цикл «доки». Застосуйте змінну-прапорець, яка до циклу мала б нульове значення, а у випадку знаходження необхідного елемента змінила б значення на 1. Як при цьому обійтись без операторів goto?
10. Як у заданій послідовності цілих чисел визначити кількість і суму елементів, кратних 10?
11. Дано n чисел. Як знайти суму чисел, більших ніж задане число a?
12. Як у заданому масиві замінити найбільший елемент нулем?
13. Як знайти суму квадратів невід'ємних елементів і кількість додатних чисел у заданому цілочисловому одновимірному масиві?
14. Як у послідовності n чисел поміняти місцями перший і найменший елементи?
15. Дано n чисел. Як замінити всі від'ємні числа їх модулями?
16. Як обчислити середнє арифметичне найбільшого і найменшого з n чисел?

2.9, Алгоритми обробки таблиць

Двовимірний масив або прямокутна таблиця *B* з n рядків і *m* стовпців у загальному вигляді записується так:

$$\begin{matrix} b_{11} & b_{12} & \dots & b_{1m} \\ b_{21} & b_{22} & \dots & b_{2m} \\ \dots & \dots & \dots & \dots \\ b_{n1} & b_{n2} & & b_{nm} \end{matrix}$$

У мові Паскаль імена елементів двовимірного масиву записуються так: вказується ім'я масиву, після якого в квадратних дужках записуються індекси, відокремлені комою, наприклад b[1,1], Ь[1,2], ..., *b[1,m],* b[2,l], b[2,2], ... , *b[2,m],* ..., b[n,m]. Перший із цих індексів вказує номер рядка в таблиці, другий — номер елемента в рядку. У пам'яті комп'ютера елементи двовимірного масиву розташовані один за одним: після елементів першого рядка йдуть елементи другого рядка таблиці і т.д.

Якщо число рядків таблиці дорівнює числу стовпців, то така таблиця називається *квадратною.* Головна діагональ квадратної таблиці проходить із лівого верхнього кута до правого нижнього.

Розглянемо деякі задачі.

1. Обчислення суми елементів головної діагоналі квадратної таблиці. Для розв'язання цієї задачі треба виконати такі дії:

1) ввести таблицю до пам'яті;
2) знайти суму елементів головної діагоналі;
3) надрукувати результат.

Опис таблиці, як і опис одновимірного масиву, використовується для виділення пам'яті. В описі таблиці вказуються діапазони зміни для двох номерів — номера рядків і номера стовпців:

```
const n = 3;
  var b: array [1..n, 1..n] of real; i,j: integer;
```

При обробці масивів у розділі змінних опису програми з'являються імена індексів елементів: для одновимірного

масиву — однієї, для двовимірного — двох цілочислових змінних.

При обчисленні суми елементів діагоналі слід звернути увагу на імена елементів, що додаються: обидва індекси мають однакові значення, тобто в загальному вигляді ім'я елемента діагоналі — b*[i, i]*. Це означає, що сукупність діагональних елементів можна розглядати як одновимірний масив і використовувати один цикл для обчислень.

Програма має вигляд:

```
program E19;
const n = 3;
var b: array[1..n, 1..n] of real;  i,j: integer;  s : real;
begin
  writeln('введіть значення елементів таблиці
           порядково');
  writeln('у кінці кожного рядка натискайте Enter');
  for i := 1 to n do
    begin
      for j := 1 to n do
        read(b[i,j]);
      writeln;
    end;
  s:= 0;
  for i := 1 to n do
    s := s + b[i,i];
  writeln(' сума елементів діагоналі таблиці S =', s)
end.
```

2. Знаходження найбільших елементів кожного рядка таблиці. Кожний рядок таблиці 11 можна розглядати як одновимірний масив і використовувати ідею знаходження найбільшого значення в програмі Е16. Знайдені значення вміщатимемо в одновимірний масив у таблиці 11.

У наступній програмі для кожного рядка таблиці змінна *a[i]* відіграє таку саму роль, як змінна max у програмі Е16. Для кожного рядка (її задає змінна і) елемент *a[i]* набуває значення першого елемента рядка. Потім виконання внутрішнього циклу за змінною *j* дозволяє переглянути всі елементи даного рядка і, якщо серед них трапиться елемент, значення якого більше, ніж запам'ятоване у *a[i],* то значен-

ня цього елемента присвоюється змінній *a[i].* Щоб надрукувати результати роботи програми — масив *a* — використовується цикл. Коментар, взятий у фігурні лужки, дозволяє при читанні програми виділити її окремі частини. Програма може мати вигляд:

Таблиця 11

Масив результатів		Задана таблиця		
a[1]	6	5	6	1
a[2]	15	4	12	15
a[3]	2	2	-3	0

```
program E20;
const n - 3;
var b: array [1..n, 1..n] of integer;  i,j: integer;
    a: array [1 ..n] of integer;
begin
  writeln('введіть значення елементів таблиці
           порядково');
  writeln('y кінці кожного рядка натискуйте Enter');
  for i := 1 to n do
    begin
      for j := 1 to n do
        read(b[i,j]);
      writeln
    end;
{побудова масиву найбільших значень елементів
 рядків таблиці}
  for i := 1 to n do
    begin
      a[i] :=b[i,1];
      for j := 2 to n do
        if a[i] < b[i,j]
        then a[i] := b[i,j];
    end;
  writeln('найбільші числа рядків таблиці');
  for i := 1 to n do
    writeln(a[i])
end.
```

3. Знаходження сум елементів стовпців таблиці. При обробці таблиць можна здійснювати операції як над рядками, так і над стовпцями. При знаходженні сум елементів стовпців можна використовувати алгоритм прикладу Е15.

Введемо змінну 5 для обчислення суми, а потім для кожного стовпця запишемо отриманий результат в масив а, тобто присвоїмо змінній a[j], де j' — поточний номер стовпців таблиці. Програма може мати вигляд:

```
program E21;
const n = 3;
var b: array [1..n, 1.. n] of integer;  S, i, j: integer;
    a: array [1..n] of integer;
begin
  writeln('введіть значення елементів таблиці
           порядково');
  writeln('y кінці рядка натискайте Enter');
  for i := 1 to n do
    begin
      for j := 1 to n do
        read( b[i,j] );
      writeln
    end;
{побудова масиву сум  елементів стовпців таблиці}
  for j := 1 to n do
    begin
      S:=0;
      for i := 1 to n do
        S:=S + b[i,j];
      a[j] := S
    end;
  writeln(суми  елементів
стовпців  таблиці');
  for i := 1 to n do
    writeln(a[i])
end.
```

Таблиця 12

Таблиця		
5	6	1
4	12	15
2	-3	0
Масив результатів		
a[1]	a[2]	a[3]
11	15	16

В таблиці 12 наводяться значення елементів таблиці в програмі E21 та результати роботи програми.

4. Перестановка рядків таблиці. У прямокутній таблиці b із *n* рядків і *m* стовпців треба поміняти місцями два рядки. При розв'язуванні цієї задачі можна використовувати ал-

горитм обміну значеннями двох змінних із програми сортування (програма Е17).

Для цього достатньо організувати цикл за номером стовпця *j*, використовуючи проміжну змінну, міняти місцями кожну пару елементів, які стоять в одному стовпці в заданих рядках. При заданих номерах рядків *K* і *L* програма може мати вигляд:

```
program E22;
const n = 3; m = 4;
var b: array [1..n, 1..m] of real;  c: real;  i,j,K,L: integer;
begin
  write ('введіть номери рядків таблиці, що міняються
         місцями');
  readln (K,L);
{введення таблиці}
  for i := 1 to n do
    begin
      writeln(i, '–ий рядок таблиці');
      for j := 1 to m do
        read(b[i,j]);
      writeln
    end;
{перестановка рядків}
  for j := 1 to m do
    begin
      c := b[K,j];
      b[K,j] := b[L,j];
      b[L,j] := c
    end;
{друкування результатів}
  writeln;
  writeln('таблиця з переставленими рядками');
  for i := 1 to n do
    begin
      for j := 1 to m do
        write (b[i,j]);
      writeln
    end
end.
```

ЗАПИТАННЯ І ЗАВДАННЯ

1. Як у квадратній таблиці, що не містить від'ємних елементів, знайти корінь квадратний з добутку діагональних елементів?
2. Як знайти найбільший елемент квадратної таблиці?
3. Як знайти найменший елемент квадратної таблиці і замінити його нулем?
4. Як у прямокутній таблиці замінити всі елементи їх квадратами?
5. Як у цілочисловій прямокутній таблиці збільшити на 0.5 всі від'ємні елементи?
6. Як у квадратній таблиці знайти найбільший і найменший елемент діагоналі?
7. Як переставити місцями перший і останній рядки прямокутної таблиці?
8. Знайти добуток елементів рядків прямокутної таблиці.

2.10. Оператор варіанта

Умовний оператор дозволяє здійснювати розгалуження програми тільки за двома напрямами, один із яких відповідає виконанню перевіреної умови, а другий — невиконанню. Якщо необхідно здійснити ряд дій, які залежать від інших умов, то треба записувати або вкладені умовні оператори, або кілька таких операторів підряд. В такій ситуації зручно використовувати оператор варіанта. Дана структура називається також *перемикачем* і виконується так: вхід у структуру містить обчислене або раніше отримане значення змінної (індексу варіанта). Це значення може збігатися з міткою, яка стоїть перед оператором на одній з гілок перемикача. У такому разі виконується оператор із цією міткою і обробка структури закінчується. Оператор може бути простим або складеним, обмеженим операторними дужками begin ... end;. Якщо значення індексу варіанта не збіглося ні з однією із міток, то виконується оператор з номером n+1 із рядка else. Якщо оператор варіанта містить рядок else, то це — *повна форма оператора;* якщо такого рядка немає, то таку форму оператора варіанта називають *скороченою.* Мітки оператора варіанта можуть бути константами будь-якого типу, їх тип повинен збігатися з типом змінної

індексу варіанта. Індекс варіанта може бути як ім'ям змінної, так і виразом відповідного типу.

У мові Паскаль оператор варіанта має вигляд:

```
case індекс варіанта of
  мітка 1: оператор 1;
  мітка 2: оператор 2;
    . . .
  мітка n: оператор n;
else оператор n + 1
end;
```

Наведемо приклад програми, яка містить оператор варіанта. Поширеною задачею з розділу молекулярної фізики є задача, пов'язана з обчисленням кількості молекул в одиниці об'єму в тілі із заданою масою і в тілі з відомим об'ємом. Для розв'язування таких задач можна побудувати програму.

Задано молярну масу речовини *M,* густину даної речовини *P,* масу *R* або об'єм даного тіла V. Треба знайти число молекул *K:*

1) в одиниці маси речовини;
2) в тілі з заданою масою;
3) в одиниці об'єму речовини;
4) в тілі із заданим об'ємом.

Для розв'язування задачі скористаємось формулою

$$K = N_A / M,$$

де $N_A = 6.022 \cdot 10^{23}$ г/моль — число Авогадро.

На основі цієї формули отримаємо розрахункові формули для програми:

1) $K=N_A$ /M; 2) $K=N_A R/M$; 3) $K=N_A PV/M$; 4) $K = N_A P/M$.

Програма може мати вигляд:

```
program E23;
const NA = 6.022E23;
var N: integer;  M,R,P,V,K: real;
begin
  writeln('Якщо відоме число Авогадро, густина P');
  writeln('цієї речовини і її молярна маса M,');
  writeln('то можна знайти кількість молекул:');
```

```
    writeln('1. В одиниці маси речовини');
    writeln('2. В тілі масою R');
    writeln('3. В одиниці об'єму');
    writeln('4. В тілі об'ємом V');
    write('Введіть номер задачі, що розв'язується ');
    readln(N);
    write('Введіть початкові дані : M=');
    readln(M);
    case N of
      1:K := NA/M;
      2:begin
          write('R= ');
          readln(R);
          K := NA * R / M
        end;
      3:begin
          write('густина речовини P=');
          readln(P);
          write(' V= ');
          readln(V);
          K := NA * P * V / M
        end;
      4:begin
          write('густина речовини P=');
          readln(P);
          K := NA * P / V
        end;
    end;
    writeln('кількість молекул K =', K)
end.
```

ЗАПИТАННЯ І ЗАВДАННЯ

1. Для чого використовується оператор варіанта?
2. Чим відрізняється повна форма оператора варіанта від скороченої форми?
3. Навести приклади алгоритмів, які потребують використання операторів варіанта.

Підпрограми

При розробці програми іноді з'являються повторювані групи дій або виникає необхідність поділити програму на функціональні модулі, зробити її структуру ієрархічною. Для цього у всіх мовах програмування існують засоби організації підпрограм. При розв'язанні складної задачі рекомендується спочатку алгоритм, а потім і програму розроблювати «згори донизу», від більш загального плану до докладного. У такому вигляді головна програма відповідає узагальненому плану розв'язання задачі, а її команди — виклику відповідної підзадачі, реалізованої у вигляді *підпрограми.* Ідучи таким шляхом створення програми, можна окремі функції, що найчастіше використовуються, реалізувати спочатку у вигляді невеликих підпрограм, перевірити їх на контрольних прикладах і, переконавшись у правильності, включати їх до складу основної програми. Це зручніше робити, маючи в розпорядженні мову, яка дозволяє повністю виділити підпрограму з тексту основної програми у вигляді окремого модуля. Використання підпрограм дозволяє розробляти програму частинами, доручати реалізацію великих проектів окремим групам розробників.

У Паскалі підпрограма є частиною основної програми, її опис подається між розділом var головної програми та її програмним блоком (першим begin). Підпрограм може бути кілька, їх описи розміщуються у довільному порядку один за одним. Опис підпрограми можна порівняти із записуваною в математиці формулою «у загальному вигляді», в яку при розрахунках підставляються конкретні значення. Як і формулу, підпрограму можна використовувати для різних даних, які передаються з програми, що її викликає. Тому параметри, які використовуються в описі підпрограми, називають *формальними,* а параметри, над значеннями яких виконуються операції, вказані в підпрограмі, називають *фактичними.*

Підпрограма — це спеціально оформлена реалізація алгоритму, яку можна використовувати багато разів при розв'язуванні задач,

Підпрограми мають структуру, аналогічну до структури головної програми. Вони починаються з заголовка із спе-

ціальним словом — ознакою підпрограми, далі вказується ім'я підпрограми і, при потребі, список формальних параметрів. Далі розміщуються всі розділи описів, які є в головній програмі: мітки, константи, типи і змінні. У цих розділах описуються дані, які використовуються тільки всередині підпрограми. Такі дані називаються *локальними*. У підпрограмі можуть використовуватися також змінні, описані у викликаючій програмі. Такі змінні називаються *глобальними*. Вони можуть використовуватися не лише програмами, які викликають, а й підпрограмами.

При роботі з підпрограмою завжди виділяється два етапи: опис підпрограми, тобто запис алгоритму розв'язання задачі в спеціальній формі, і виклик підпрограми — передача їй даних на обробку з викликаючої програми та отримання результатів.

Розглянемо способи організації підпрограм у Паскалі.

Підпрограми-процедури. Процедура — це підпрограма, яка має будь-яку кількість вхідних і вихідних даних. Процедура може бути описана без параметрів і з параметрами. Параметри в заголовку процедури використовуються для обміну інформацією між процедурою і програмою, яка її викликає. Вони визначають дані, які передаються на обробку до процедури, і дані, які отримуються у вигляді результатів.

Процедури без параметрів. Опис процедури має такий вигляд:

```
procedure ім'я;
{опис локальних змінних}
begin
  {оператори}
end;
```

Процедура без параметрів може реалізовувати будь-який алгоритм. Усі змінні, з якими проводять дії оператори процедури, визначаються у викликаючій програмі, їм присвоюються потрібні для виконання процедури значення. Для виклику процедури без параметрів просто вказується її ім'я.

Розглянемо приклад обчислення найменшого спільного кратного двох натуральних чисел НСК(Х,У), яке можна знайти, використовуючи найбільший спільний дільник

(НСД) цих чисел, за формулою

$$НСК(X, Y) = X \cdot Y / НСД(X, Y).$$

При складанні програми оформимо як процедуру без параметрів програму Е7 — обчислення НСД за алгоритмом Евкліда. Результат роботи процедури заносимо до ділянки з іменем *M*. Змінна *M* описується як глобальний параметр, вона використовується і у викликаючій програмі, і в процедурі. У програмі будемо обчислювати НСК кількох чисел, заносячи їх до масиву *C*. Цей масив формується в розділі констант головної програми. Якщо дані визначаються в розділі констант, то вони не потребують додаткового опису в розділі змінних (var). Змінна *X* спочатку набуває значення першого числа, а потім їй присвоюється значення, отримане в результаті виконання підпрограми — НСК двох перших чисел. Значення змінної *Y* є друге число з кожної пари, для якої обчислюється найменше спільне кратне. Таким чином, на кожному наступному кроці циклу обчислюється НСК двох чисел, перше з яких *X* містить результат попереднього кроку. Наочно цей процес подано в таблиці 13.

Таблиця 13

X := НСК (X, Y)	36	108	216	216	1080
Y := C [i]	54	72	18	15	
M := НСК (X, Y)	18	36	18	3	

Програма може мати вигляд:

```
program E24;
const c: array[1 .. 5] of integer = ( 36,54,72,18,15 );
var x,y,i,m: integer;
procedure NSD; {заголовок процедури}
var a,b: integer; {опис локальних змінних}
begin
  a := x; b := y; {збереження початкових даних }
  while a <> b do
  if a > b then a := a – b
  else b := b – a;
```

```
    m := a {результат роботи процедури присвоюється
            глобальній змінній}
end; {кінець процедури}
begin { початок головної програми }
  x := c[1];
  for i := 2 to 5 do
    begin
      y := c [i];
       NSD; {виклик процедури без параметрів}
       x := x * y div m {div – ділення націло
                         для цілочислових даних}
    end;
  writeln('НСК = ', x)
end.
```

Процедури з параметрами. Для передавання даних до процедури і отримання з неї результатів використовуються формальні і фактичні параметри.

Формальні параметри — це умовні позначення в описі процедури; вони описуються в її заголовку. При виклику процедури після її імені в дужках слід вказати список *фактичних параметрів,* які конкретизують значення, над якими будуть виконуватись операції в тілі процедури. Послідовність, кількість і тип формальних і фактичних параметрів повинні збігатися. Формальні параметри, описані в заголовку процедури, більше ніде не описуються. їх опис схожий на опис даних у розділі змінних і може також містити слово **var.** Слово **var** у заголовку процедури ставиться перед тими параметрами, імена яких відповідають початковим даним.

Фактичні параметри, які відповідають формальним, перед якими стоїть слово **var,** можуть бути тільки іменами змінних. Перед іменами формальних параметрів, які є вхідними даними процедури, слово **var** можна не вказувати. Якщо перед формальним параметром у заголовку процедури немає слова **var,** то йому може відповідати формальний параметр, який має вигляд виразу відповідного типу. Якщо для вхідних даних процедури при описі формальних параметрів вказано слово **var,** то їм також відповідають фактичні параметри — імена змінних.

Наприклад, процедура NSD (знаходження найбільшого спільного дільника) із параметрами може мати заголовок:

```
procedure NSD (a,b: integer; var k: integer);
```

Виклик цієї процедури:

```
NSD (x,y,m);
```

або

```
NSD (36,54,m);
```

Змінні a, b, k у заголовку процедури NSD — це формальні параметри, які заміняються при виконанні процедури на конкретні значення змінних *x* і *y* або числа 36 і 54. У заголовку процедури NSD описані формальні параметри: *a* і *b* — вхідні дані, для яких знаходиться найбільший спільний дільник; k — результат роботи процедури. При виклику процедури змінна *a* набуде значення змінної *x,* а змінна b — значення змінної y. Результат виконання процедури надійде до ділянки пам'яті з іменем *m.*

Програма при використанні процедури з параметрами може мати вигляд:

```
program E25;
const c: array [ 1..5 ] of integer = ( 36,54,72,18,15 );
var x,y,i,m: integer;
procedure NSD (a,b: integer; var k: integer); {заголовок
                                              процедури}
begin
  while a <> b do
    if a > b then a := a – b
    else b := b – a;
  k := a {значення змінної k – результат роботи
          процедури }
end; {кінець процедури}
begin {початок головної програми}
  x := c[1];
  for i := 2 to 5 do
    begin
      y := c [i];
      NSD (x, y, m); {виклик процедури з фактичними
                      параметрами}
```

```
        x := x * y div m
      end;
    writeln('НСК =', x)
end.
```

Розглянемо ще один приклад використання процедури з параметрами. Знайдемо за допомогою процедури середнє арифметичне, найбільший і найменший елементи масиву.

Програма запишеться у вигляді:

```
program E26;
const n = 10;
type R = array [1..n] of real;
var Y: R; A,B,C: real; i: integer;
procedure Stat ( X: R; var S,min,max: real);
begin
  S := 0; min := x[1]; max := x[1];
  for i := 1 to n do
    begin
      S := S + x[i];
      if x[i] < min then min := x[i];
      if x[i] > max then max := x[i]
    end;
  S := S/n
end;
begin {головна програма}
  for i := 1 to n do
    read ( Y[i] );
  Stat (Y, A, B, C); {виклик процедури}
  writeln;
  write(' середнє =', A, ' найменше =', B);
  writeln(' найбільше = ', C)
end.
```

У програмі E26 з'явився новий розділ описів — розділ типів даних type, в якому можна описати новий тип даних через уже відомі типи. Тип даних *R* — це масиви з n дійсних чисел, *R* — ім'я типу. Далі цей тип дозволяє скоротити описи, він використовується в головній програмі при опису вихідного масиву *Y* і в заголовку процедури при опису формального параметра — масиву *X*.

Головна програма складається з трьох основних частин:
1) введення даних — масиву *Y;*
2) виклику процедури Stat з фактичними параметрами — масивом *Y* і отримуваними результатами, які надходять відповідно до ділянок *A* (середнє значення), *B* (найменше) і C (найбільше);
3) друкування результатів роботи програми.

Підпрограми-функції. Підпрограма, яка має єдиний результат, може бути оформлена як функція. Опис її має вигляд:

```
function ім'я_функції (вхідні дані): тип_результату;
{ опис локальних змінних }
begin
  { оператори }
  ім'я_функції := результат;
end;
```

Після опису формальних параметрів, які є аргументами функції, в заголовку вказується тип результату, тобто тип значення самої функції. Цей опис відноситься до імені функції, якому необхідно присвоїти значення результату роботи підпрограми. Як і процедура, функція може містити всі чотири розділи опису локальних змінних — label, const, type і var. Ім'я функції не можна використовувати для проміжних обчислень.

Функція викликається за допомогою вказівника. *Вказівник* — це ім'я функції, після якого в круглих дужках перелічені фактичні параметри — аргументи функції. Вказівник має вигляд:

ім'я функції (список фактичних параметрів)

Вказівник може з'явитися у виразі відповідного типу, в умовах операторів **if** , **while** і **repeat** після слова **until,** а також в процедурі друку **write.** Прикладами виклику функцій є звернення до арифметичних функцій, наприклад write (sin (x));

Розглянемо третій варіант програми обчислень найменшого спільного кратного. Оскільки найбільший спільний дільник двох натуральних чисел — єдине число, то підпрограму для його обчислення можна оформити як функцію.

Програма матиме вигляд:

```
program E27;
const c: array [ 1..5 ] of integer = ( 36,54,72,18,15 );
var x,y,i: integer;
function NSD (a,b: integer): integer; {заголовок функції}
begin
  while a <> b do
    if a > b then a := a – b
    else b := b – a;
  NSD := a {результат роботи функції присвоюється її
            імені}
end; {кінець опису функції}
begin {початок головної програми}
  x := c[1];
  for i := 2 to 5 do
    begin
      y := c[i];    x := x * y div NSD (x, y)
    end;
  writeln('НСД=',x)
end. {кінець головної програми}
```

Одна підпрограма може викликати іншу. Розглянемо програму уточнення кореня рівняння методом ділення відрізка навпіл (приклад Е14), яка включає дві підпрограми. Перша підпрограма — процедура, яка реалізує метод ділення відрізка навпіл, друга — підпрограма-функція, яка обчислює значення функції у потрібних точках. Програма при цьому матиме вигляд:

```
program E28;
var a1,b1,e,y: real; { a1, b1 – кінці відрізка,
                       e – точність, y – результат
                       обчислень}
function f (x: real): real; {заголовок функції}
begin
  f := x * x – cos(x) + 0.5;
end; {кінець опису функції}
procedure precise(a,b, eps: real; var c: real);
{процедура уточнення кореня методом ділення відрізка
 навпіл}
```

```
begin
  while abs(b - a) > eps do
    begin
      c := (a+b)/2;
      if f(a)*f(c) >0
      then a := c
      else b := c
    end;
  c := (a + b)/2
end; {кінець процедури}
begin {початок головної програми}
  write ('введіть значення кінців відрізка');
  readln (a1, b1);
  write ('введіть точність');
  readln (e);
  precise(a1, b1, e, y);
  writeln('Корінь f(x) на відрізку ', a1, ',', b1, ' дорівнює ', y)
end. {кінець головної програми}
```

ЗАПИТАННЯ І ЗАВДАННЯ

1. Що таке підпрограма і для чого вона використовується?
2. Поясніть призначення локальних і глобальних змінних.
3. Що таке формальні і фактичні параметри?
4. До чого відноситься опис типу в кінці заголовка підпрограми-функції?
5. Чим відрізняється виклик функції від виклику процедури?
6. Як задати значення елементів масиву без використання процедури введення?
7. Оформити програми попередніх параграфів, крім розглянутих у даному, із використанням процедур.

2.12. Рекурсія

Якщо поставити два дзеркала одне проти одного і між ними помістити предмет, то вийде нескінченна кількість зображень, кожне з яких містить само себе. Кожне з цих зображень можна розглядати як рекурсивний об'єкт, тобто

такий, який частково складається або визначається за допомогою самого себе. Рекурсивні об'єкти мають такі властивості: простоту побудови; несхожість кінцевого результату з початковими даними; внутрішню самоподібність.

У математиці трапляються рекурсивні означення, які дозволяють описати об'єкти за допомогою самих себе. Одним з таких означень є, наприклад, означення натурального числа:

1) одиниця є натуральне число;

2) наступне за натуральним ціле число є натуральним числом.

Означення, яке задає деякий об'єкт у термінах більш простого випадку цього ж об'єкта, називається рекурсивним. Як і цикл, рекурсивне означення містить повторення, але це повторення є неявним. *Рекурсія* — це спосіб опису функцій або процесів із використанням самих себе.

Процес може бути описано деяким алгоритмом, що називається у даному разі рекурсивним. У таких алгоритмах виділяється два етапи виконання:

1)«занурення» алгоритму в себе, тобто використання визначення «в інший бік», доки не буде знайдене початкове визначення, яке не є рекурсивним;

2) послідовне виконання операцій від початкового визначення до визначення з введеним в алгоритм значенням.

Розглянемо приклади рекурсивних алгоритмів, які часто оформляються у вигляді процедур і функцій.

1. Найпоширенішим прикладом рекурсії є визначення факторіала (не рекурсивне обчислення факторіала наведено в прикладі Е9):

$$\begin{aligned} &(a)\quad 1! = 1, \\ &(b)\quad n>1 \ : \ n! = n(n-1)! \end{aligned}$$

На основі цього визначення можна записати програму обчислення факторіала, яка використовує рекурсивну функцію:

```
program E29;
var n,y: integer;
function F (x: integer): integer; {опис рекурсивної
                                   функції}
```

```
begin
  if x = 1
   then F := 1 {виклик для початкового визначення}
   else F := x * F(x-1); {виклик для попереднього
                          визначення}
end; {кінець опису функції}
begin {початок головної програми}
  write ('введіть натуральне число не більше 14 n=');
  readln (n);
  Y := F(n); {виклик функції у головній програмі}
  writeln(n,'!=' ,Y)
end. {кінець головної програми}
```

Виконаємо програму Е29 для n = 4. Рекурсивна функція буде працювати таким чином (при виклику функції значення n буде присвоєно змінній *x).* Спочатку здійснюється «занурення», працює оператор гілки else умовного оператора:

1-й крок: $x = 4$, $x - 1 = 3$,

виконується проміжне обчислення 4! = 4*3!

2-й крок: $x - 3$, $x - 1 = 2$,

виконується проміжне обчислення 3! = 3*2!

3-й крок: $x = 2$, $x - 1 = 1$,

виконується проміжне обчислення 2! = 2*1!

4-й крок (останній): 1! = 1

за початковим визначенням, працює оператор F: = 1 гілки then умовного оператора.

Наступний етап виконання рекурсивного алгоритму — побудова «прямого» визначення, від початкового до отримання результату з вихідними для алгоритму даними (числом 4). При цьому здійснюється підстановка попередніх обчислень (більш пізніх кроків) в більш ранні:

5-й крок: 2! =2*1 =2

6-й крок: 3! =3*2 = 6

7-й крок: 4! = 4*6 = 24 — отримано результат, який повертається до головної програми і присвоюється змінній Y.

2. Обчислення степеня з натуральним показником можна визначити рекурсивно:

(a) $x_0 = 1$

(b) $\kappa > 0$: $x^{\kappa} = x\, x^{\kappa}$ '

Цьому визначенню відповідає рекурсивна функція power(k,x). Програма має вигляд:

```
program E30;
var y: real;  n: integer;
function power (k: integer; x: real): real; {опис
                        рекурсивної функції}
begin
  if k = 0
  then power := 1    { початкове визначення }
  else power := x * power(k-1, x);{рекурсивне
                                  визначення}
end; { кінець опису функції }
begin { початок головної програми }
  write(' основа степеня x =');
  readln(y);
  write('показник степеня k =');
  readln(n);
  writeln(y, 'в  степені ', n, '=', power (n,y))
{ виклик функції і друкування результату }
end. {кінець головної програми}
```

3. Розглянемо обчислення чисел Фібоначчі. Італійський математик Фібоначчі придумав послідовність натуральних чисел: 1, 1, 2, 3, 5, 8, 11, ... Перші два члени послідовності дорівнюють одиниці, а кожний наступний, починаючи з третього, дорівнює сумі двох попередніх. Для чисел Фібоначчі справедливе співвідношення:

$$F_k = F_{k-1} + F_{k-2}$$

Рекурсивна процедура отримання значення n-го числа Фібоначчі має вигляд:

```
function Fib (n: integer): integer;
begin
  if k < 3
  then Fib := 1
  else Fib := Fib ( n – 1 ) + Fib ( n – 2 )
end;
```

Для чисел Фібоначчі використовується таке рекурсивне визначення:

(a) $n = 1$, n = 2 : fib(n) = 1
(b) n > 2 : fib(n) = fib(n-2) + fib(n-l)

Для того щоб визначити fib (6), застосовуючи дане рекурсивне визначення, зробимо такі обчислення:

```
fib(6)= fib(4) + fib(5) = fib(2) + fib(3) + fib(5) =
      = 1 + fib(3) + fib (5) =
      = 1 + fib(l) + fib(2) + fib(5) =
      - 1 + 1 + 1 + fib(5) =
      = 3 + fib(3) + fib (4) =
      = 3 + fib(1) + fib (2) + fib(4) =
      = 3 + 1 + 1 + fib(4) =
      = 5 + fib (2) + fib(3) =
      = 5 + 1 4 - fib(l) + fib(2) =
      = 6 + 1 + 1 = 8
```

Кількість дій у даних обчисленнях при використанні рекурсивного визначення чисел Фібоначчі різко збільшується, тому що це визначення посилається саме на себе двічі. При обчисленні факторіала кілвкість дій при виконанні програми з рекурсивною функцією і прикладу Е9 однакова.

4. Рекурсивні алгоритми можуть бути оформлені і у вигляді процедур. Прикладом такої процедури є процедура розв'язування задачі про *ханойські вежі* (рис. 43).

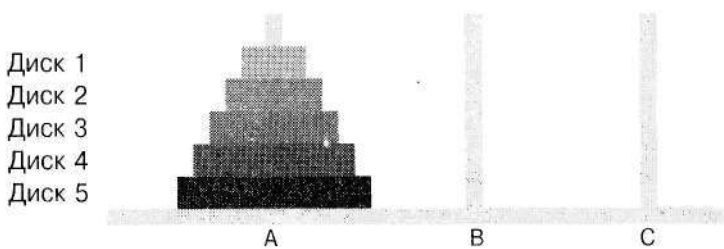

Рис. 43. Ханойські вежі

Ця задача пов'язана з легендою про те, що в одному із східних храмів знаходиться бронзова плита з трьома діамантовими стержнями. На один із них Бог при створенні світу нанизав 64 диски з чистого золота так, як показано на рис. 43. Жреці повинні переносити диски з одного стержня на інший, дотримуючись таких правил:

1) диски можна переносити тільки по одному;

2) не можна класти більший диск на менший.

Згідно з легендою, коли всі диски будуть перенесені з одного стержня на інший, настане кінець світу.

Розв'язування цієї задачі реалізовано у вигляді рекурсивного алгоритму, який являє собою інструкцію щодо переміщення дисків. Сформулюємо задачу, присвоївши імена стержням (A, *B, C)* і номери дискам (від 1 до n). Треба перенести диски із стержня *A* на стержень *C,* використовуючи *B* як допоміжний і дотримуючись наведених вище правил перенесення дисків.

Опис алгоритму природною мовою має вигляд:

1. Якщо n = 0, слід зупинитись.
2. Перемістити верхні n — 1 дисків із стержня A на стержень *B,* використовуючи стержень *C* як допоміжний.
3. Перемістити диск, що залишився, зі стержня *A* на *C.*
4. Перемістити n— 1 дисків із стержня *B* на стержень C, використовуючи стержень *A* як допоміжний.

У процедурі з'являється новий тип даних — char, значення якого — один символ, який береться в апострофи.

Програма матиме вигляд:

```
program E31;
var k: integer;
procedure Hanoy (n: integer;  One, Two, Three: char);
begin
  if n > 0 then
  begin
    Hanoy (n-1, One, Three, Two);
    writeln('перемістити диск', n, ' із стержня',
                One,' на стержень ', Three);
    Hanoy (n-1, Two, One, Three)
  end;
end;
begin
  write('введіть кількість дисків');
  readln(k);
  Hanoy (k, 'A', 'B', 'C')
end.
```

Результат роботи програми для n = 3 — це інструкція із семи пунктів (для *n — 4* — інструкція з 15 пунктів):

1) перемістити диск 1 із стержня А на стержень С;
2) перемістити диск 2 із стержня А на стержень В;
3) перемістити диск 1 із стержня С на стержень В;
4) перемістити диск 3 із стержня А на стержень С;
5) перемістити диск 1 із стержня В на стержень А;
6) перемістити диск 2 із стержня В на стержень С;
7) перемістити диск 1 із стержня А на стержень С.

ЗАПИТАННЯ І ЗАВДАННЯ

1. Що таке рекурсивний об'єкт і які його властивості?
2. Наведіть приклади рекурсивного визначення в математиці.
3. Що таке рекурсія?
4. Як виконується рекурсивний алгоритм?
5. Поясніть виконання рекурсивної функції обчислення степеня з натуральним показником.
6. Напишіть головну програму для обчислення n-го числа Фібоначчі.
7. Чому використовувати рекурсивний алгоритм обчислення n-го числа Фібоначчі невигідно?
8. Напишіть нерекурсивну програму обчислення n-го числа Фібоначчі, використовуючи три змінні (див. п. 2.7).
9. Визначіть рекурсивно множення як додавання і ділення як віднімання і оформіть алгоритми у вигляді рекурсивних функцій з викликом із головних програм.

Обробка рядків у Паскалі

У пам'яті комп'ютера можуть зберігатися числа й символи. Кожний символ займає один байт пам'яті. Для елементів даних, значенням яких є одиночний символ, використовується опис char. Символи можуть об'єднуватися в масиви. Кожному елементу масиву, як і для числових даних, відповідає порядковий номер, а. ім'я елемента складається з імені всього масиву і його номера. Значення символьного даного — це символ, взятий в апострофи, наприклад A', '?', '5'.

Знак апострофа представляється при записі двома апострофами.

Приклади описів:

```
var a: array [1 ..50] of char;  x,y: char;
```

Масив *a* може складатись з 50 символів, йому відводиться при трансляції програми 50 байтів пам'яті. Елементи масиву: a[1], a[2], ..., a[50]. Змінні *x* і *y* — прості, їх значення — одиночні символи. Для введення символьного масиву слід використовувати такий цикл:

```
for i := 1  to  n  do read(a[i]);
```

При введенні такого масиву можна набрати рядок із *n* символів і натиснути Enter.

В описі можна задати таблицю символів і для її введення використати подвійний цикл:

```
const n = 10; m = 15;
var b: array [1..n,  1..m] of char; i,j: integer;
begin
  for i := 1 to n do
    begin
      for j := 1 to m do
        read(b[i,j]);
      writeln
    end
end.
```

У прикладі розглядається таблиця b з 10 рядків по 15 символів кожна. При її введенні необхідно набирати рядки по 15 символів і натискувати Enter. Незручність такого способу введення полягає в тому, що всі рядки повинні мати 15 символів, тобто слова, що набираються, не повинні складатися більше ніж з 15 літер, а для коротких слів треба додавати пробіли.

При роботі із символьними масивами використовуються такі самі алгоритми, як і при роботі з числовими. Наприклад, слово, задане як масив символів, треба записати у зворотному порядку, тобто справа наліво. При розробці алгоритму можна використати таку постановку задачі: даний

числовий масив переписати так, щоб останній елемент встав на перше місце, передостанній — на друге і т.д., а перший — на останнє, тобто необхідно з масиву $a_1, a_2, \ldots, a_n$ одержати $a, a_n, \ldots, a_1$, який буде знаходитись у масиві b (рис. 44).

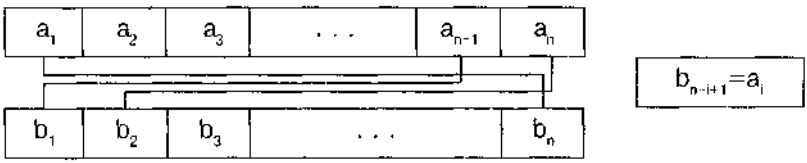

Рис. 44. Переміщення елементів з масиву *a* в масив *b* і формула перерахунку індексів

На рис. 44 рамкою обведена формула перерахунку індексу: коли в масиві *a* номери перераховуються в прямому порядку, тобто поточний індекс елемента масиву змінюється від 1 до n, то в масиві b індекси повинні змінюватись від n до 1. Таку зміну індексів і забезпечує наведена формула для індексів масиву Ь. Програма, яка виконує переставляння елементів масиву у зворотному порядку, для символьних даних називається *програмою обернення слова.* Вона має вигляд:

```
program E32;
const n = 15;
var a,b: array[1..n] of char;  i: integer;
begin
  for i := 1 to n do
    begin
      read(a[i]);
      b[n-i+1] :=a[i]
    end;
  writeln;
  for i := 1 to n do
  writeln(b[i])
end.
```

Рядок — це взята в апострофи послідовність будь-яких символів, записаних підряд. Довжина рядка, що обробляєть-

ся, не повинна перевищувати 255 символів (обмежувальні апострофи не враховуються). Це пов'язано з тим, що в нульовому байті рядка зберігається значення довжини цього рядка, тобто кількість символів у ньому, а найбільше ціле число, яке може бути записано в байті, — це 255. Якщо треба опрацювати текст, довжина якого понад 255 символів, то слід використовувати масив рядків.

Опис рядка має вигляд:

```
var x: string[20];
```

Рядок *x* повинен бути не більший, ніж 20 символів, а якщо він менший, то займатиме в пам'яті стільки байтів, скільки знаків він містить (плюс 1 байт для зберігання довжини рядка). Тому при введенні рядка немає необхідності доповнювати його до вказаної в описі довжини.

Для опрацювання рядка використовуються спеціальні операції і підпрограми. Такі операції дозволяють працювати з рядками як з окремими об'єктами, а підпрограми застосовуються переважно для опрацювання окремих символів або частин рядка.

Операції над рядками — це об'єднання, порівняння і присвоювання.

Об'єднання рядків. Ця операція дозволяє з'єднати два рядки в один, приєднавши початок другого рядка до кінця першого. Об'єднання позначається знаком « + ». Наприклад:

```
var x,y,z: string[10];
begin
  x := 'тепло';
  y := 'хід';
  z := x + y;
  writeln(z)
end.
```

Змінним *x* та *y* присвоюються значення рядків, а змінній z — результат об'єднання цих рядків в один: «теплохід». При друкуванні рядка буде виведено зміст ділянки пам'яті, яка називається *z*. Очевидно, що операція об'єднання рядків некомутативна, тобто для неї $a + b \quad b + a$, тому при використанні об'єднання необхідно передбачати, з якого боку до даного рядка приєднується інший: зліва чи справа.

Як і для арифметичних операцій, для даної операції над рядками існує нейтральний елемент, який не впливає на її результат. Це рядок нульової довжини (порожній рядок), який позначається двома апострофами, що стоять поряд (''). Такий рядок можна приєднати до будь-якого рядка зліва або справа, від чого рядок не зміниться.

Порівняння рядків. Для порівняння рядків використовуються операції, які мають такі самі позначення, як і операції відношення для чисел, але вони мають дещо інший зміст. Якщо рядки порівнювати на «дорівнює» (=), то рівність означає посимвольний збіг рядків. Відповідно «не дорівнює» (<>) означає, що рядки не збігаються. При застосуванні решти операторів (<, < = , > = , >) відношення рядків заміняється відношенням перших двох відповідних один одному, але не рівних символів (символи порівнюються за їх порядком у таблиці символів). Якщо така пара символів не знайдена, а рядки збігаються до останнього символу рядка, який має меншу довжину, то більш короткий рядок вважається меншим. Використовуючи ці оператори відношення можна впорядковувати слова за алфавітом (як у словнику).

Присвоювання. Оператор присвоювання при обробці рядків має вигляд:

ім'я_рядкової_змінної := рядковий вираз;

Ім'я рядкової змінної може бути просте або з індексом (елементом масиву рядків). Якщо в результаті виконання всіх операцій рядкового виразу дістанемо рядок, довжина якого перевищує довжину в описі змінної, що вказана зліва від знака присвоювання, то одержаний рядок скорочується справа до допустимої довжини:

```
var x: string[6];
begin
  x := 'коли' + 'барбарисовий';
  writeln(x)
end.
```

У результаті роботи цієї програми буде надруковано слово «колиба», бо допустима довжина рядка x — 6 символів, і значення виразу, одержане після виконання операції об'єд-

нання рядків — «колибарбарисовий» скоротиться до «колиба», інші символи будуть відкинуті.

Довжина рядка. Функція довжини рядка видає кількість символів у рядку:

length (рядковий вираз)

Як приклад використання цієї процедури розглянемо програму, яка визначає довжину двох рядків та їх об'єднання:

```
program E33;
var x,y: string[20];  k,l,n: integer;
begin
  writeln ('введіть два рядки');
  readln(x);  readln(y);
  k := length(x); l := length(y);  n := length(x+y);
  writeln('довжина першого рядка' :25,
          'довжина другого рядка' :25);
  writeln(k:25, 1:25);
  writeln(x+y, 'довжина рядка ', n)
end.
```

У програмі Е33 використовується виведення результату з форматуванням. Перший раз формат (:25) вказано після рядка, що виводиться на екран («довжина першого рядка»). Це означає, що для виведення даного рядка відводиться по 25 позицій. Оскільки текст, що виводиться, коротший (20 символів), то він доповниться на початку пропусками, тобто буде розміщений справа у відведеному йому полі. Аналогічно розміщуються у відведених для них місцях цілі числа, — довжини рядків. Результат роботи наведеної програми матиме такий вигляд:

```
довжина першого рядка    довжина другого рядка
                    7                       10
```

За допомогою форматування можна розміщувати дані, що виводяться, в стовпцях, будувати на екрані дисплея таблиці.

Копіювання рядка або його частини. Функція копіювання називається також «вирізанням». Вона дозволяє скопіювати одну ділянку пам'яті в іншу.

Для копіювання необхідно вказати рядковий вираз, із значення якого виділяється частина, а також початковий номер символу й кількість символів частини, що копіюється:

copy (рядковий вираз, початковий номер символу, кількість символів)

Наприклад, результатом роботи функції

copy ('інформатика',**3,5)**

буде слово «форма».

Застосуємо дану функцію для розробки іншої версії програми обернення слова. Оброблятимемо слово, вилучаючи з нього літери і приєднуючи до результату зліва. Змінній y, яка містить результат, спочатку присвоюється значення порожнього рядка. Значення змінної циклу змінюється від 1 (першого символу слова) до довжини рядка, що вводиться (номера останнього символу слова).

Програма має вигляд

```
program E34;
var x,y: string[10];  i: integer;
begin
  write('введіть слово');
  readln(x);
  y := ''; { присвоєння результату початкового
           значення - порожнього слова}
  for i := 1 to length( x ) do
    y := copy(x,i,1) + y; {приєднання літери, що
                          копіюється, зліва}
  writeln;
  writeln(y)
end.
```

Пошук підрядка в рядку. Функція пошуку визначає, з якої позиції (номера символу) один рядок (підрядок) міститься в іншому (даному рядку). Опис функції мовою Паскаль має вигляд

pos (підрядок, вихідний рядок)

Якщо входження підрядка в рядок має місце, то результатом роботи функції буде номер символу у вихідному рядку, з якого починається входження підрядка. Якщо входження немає, то результат — нуль. Аргументи функції можуть бути рядковими виразами.

Вставляння рядка в рядок. В один рядок можна вставити інший рядок, вказавши номер символу, починаючи з якого здійснюється вставляння. Опис мовою Паскаль процедури вставлення має вигляд

```
insert (рядок_що_вставляється, вихідний рядок,
        цілочисловий вираз);
```

Вхідні дані процедури — рядок, що вставляється, рядок, в який здійснюється вставляння, і цілочисловий вираз, який задає позицію, із якої слід починати вставлення. Рядки також можуть бути задані рядковими виразами. Результат роботи процедури поміщається на місце, відведене для вихідного рядка, і рядок при цьому «розширюється». Якщо довжина рядка, який вставляється разом із довжиною вихідного рядка, перевищує допустиму довжину вихідного рядка, то результат вставляння скорочується справа до допустимої довжини.

Вилучення частини рядка. Частину рядка можна вилучити, і рядок при цьому «стискується». Для вилучення необхідно вказати рядок (у вигляді рядкового виразу), номер початкового символу частини рядка, що вилучається, і кількість символів, які потрібно вилучити. Опис мовою Паскаль процедури вилучення має вигляд

```
delete (рядок, початковий номер,
        кількість символів);
```

Розглянемо приклад заміни літери в слові. Перетворимо слово «форма» на слово «фірма». Програма матиме вигляд:

```
program E35;
var x: string[10];
begin
  x := 'форма';
  insert('i', x,2); { вставка літери 'i', маємо слово
                      'фіорма'}
```

```
  delete( x, 3, 1); { вилучення третьої літери -'o'}
  writeln(x)
end.
```

Приклад програми послівного перекладу з англійської мови. Припустімо, треба побудувати програму-перекладач, яка б, не враховуючи правил граматики, просто перекладала кожне слово речення, що вводиться. Оскільки програма демонстраційна, то використовуються невеликі словники — всього по 10 слів. Однак у разі розширення словників програму можна використовувати і для реального послівного перекладу. Ідея виділення слів із речення, що вводиться, ґрунтується на тому, що слова відокремлюються, як правило, пропусками, і вирізається слово між двома пропусками. Далі при звертанні до словника відшукується збіг виділеного слова зі словом у словнику. При знаходженні такого збігу зі словника перекладів друкується слово з таким самим індексом елемента, як і у словнику англійських слів. Така ідея пошуку може бути використана і в інших таблицях, коли, наприклад, за назвою хімічного елемента відшукується його маса.

Розглянемо етапи виконання програми на прикладі перекладу речення «I like a cat». Для виділення слів із речення використовуватимемо два вказівники. Перший з них — змінна *m,* її значення завжди 1, оскільки вона вказує на перший символ слова, що вирізається (копіюється) із речення. Другий вказівник — значення змінної *к,* яке завжди вказує позицію пропуску за виділеним словом і є його номером. Для першого слова речення перше значення змінної *к* — це 2. Функція пошуку підрядка в рядку працює таким чином, що пошук входження завжди здійснюється з першої позиції вихідного рядка. Якщо не змінювати вихідний рядок, то програма буде завжди знаходити перший пропуск. Тому після кожного виділення з рядка слова рядок необхідно «зрізувати», залишаючи тільки неопрацьовану частину рядка. Таке «зрізування» проводиться за допомогою функції

```
a := copy (a, k, n-к+1);
```

де $n - к + 1$ — довжина частини рядка, що залишилась після виділення чергового слова. Так, після виділення першого

слова з речення (слова «I»), яке буде присвоєно змінній x, і скорочення рядка в рядок *a* набуде вигляду «like a cat». При пошуку в словнику англійських слів (масив e) значення змінної *x* порівнюється з кожним словом словника. Якщо відбувся збіг, то друкується слово з словника українських слів (масив r) з таким самим порядковим номером, як і в англійському словнику. Якщо весь словник переглянуто, але слово не знайдено, то друкується повідомлення «слова в словнику немає». Програма при обробці визначених і невизначених артиклів пропускає їх і переходить до обробки наступного слова речення.

Програма має вигляд:

```
program E36;
label  1,2,3;
const p = 10; { кількість слів у словнику }
var e,r: array [1..p] of string[10];  x: string[10];
     i,k,m,n : integer;  a,b: string;
begin
{ формування словників }
  writeln ('введіть ', p, ' англійських слів');
  for i := 1 to p do
    readln (e[i]);
  writeln ('введіть українські слова');
  for i := 1 to p do
    readln(r[i]);
{ введення слів і виділення слів із речення }
  writeln ('введіть речення');
  readln(b);
  a := b; { збереження вихідного речення }
  m := 1; n := length(a); {n - довжина речення}
1:k := pos (' ', a);  { пошук пропуску }
  if k = 0 then k := length(a) + 1; { виставлення пропуску
                                      після останнього
                                      слова}
  x := copy(a, m, k - m); {вирізання слова довжиною
                           к-m  символів}
  if ( x = 'a') or ( x = 'the') then goto 3; {обробка
                                              артиклів}
{пошук слова в словнику}
```

```
    for i := 1 to p do
      if e [i] = x then goto 2;
    writeln('слова в словнику немає');
    goto 3;
  2:writeln(r[i]); {виведення слова з українського
                    словника}
  3:k:= k + 1; {перехід до наступного слова в реченні}
    a := copy(a, к, n - к + 1);
    if a<> '' then goto 1
  end.
```

ЗАПИТАННЯ І ЗАВДАННЯ

1. Чим відрізняється символьний тип даних від рядкового?
2. Як використовуючи символьний масив, визначити, скільки слів у тексті «I like a cat»?
3. Як використовуючи символьний масив, порахувати, скільки літер «а» у слові «програма»?
4. Як використовуючи засоби обробки рядків, виправити слово «вилисипидисти»?
5. Як використовуючи ідею програми Е34 обернення слова, подвоїти кожну букву в слові «урок»?
6. Як використовуючи програму обернення слова Е34, визначити, чи є дане слово паліндромом, тобто таким словом, яке однаково читається в обох напрямках (наприклад, «потоп», «кок» і т.д.)?
7. Дано рядок із кількома комами. Як знайти текст між першою і другою комами? Розв'яжіть задачу з використанням масиву символів і рядка символів.

Комп'ютерна графіка

Засоби мови Паскаль дозволяють будувати зображення на екрані дисплея. Для цього використовується спеціальна бібліотека підпрограм, яка називається Graph. До неї входять графічні процедури і функції для побудови різних за формою фігур і ліній, а також засоби організації графічного режиму. Ці засоби призначені для аналізу можливостей використовуваного дисплея, розподілу поля екрана на різну кількість дрібних квадратиків, кожний з яких вважається

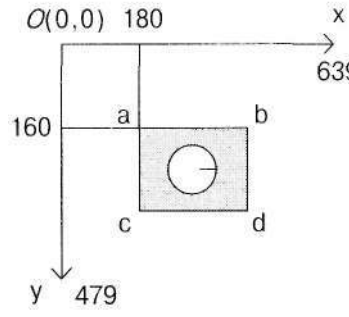

Рис. 45

окремою точкою зображення і називається пікселем. Для дисплея типу VGA кількість точок може бути 640x480, а кількість використовуваних кольорів — 16. Складніші засоби мови Паскаль дозволяють збільшити кількість кольорів до 256. Колір, як і координати екранної точки, задається цілим числом. Початок координат екрана знаходиться в лівому верхньому куті (точка (0,0)), вісь *Ox* направлена вправо, *Oy* — вниз (рис. 45). Цю особливість необхідно враховувати при побудові зображень, особливо графіків функцій і діаграм.

При побудові зображень також використовуються так звані графічні примітиви — точка, відрізок, дуга.

Точка. Кожне зображення можна будувати по точках, вказуючи для кожної точки її координати й колір. Для видачі точки на екран використовується процедура

PutPixel(x, y, номер кольору);

де *x, y* — координати точки. За допомогою цієї процедури можна побудувати графік функції.

Побудова графіка функції. Щоб побудувати графік функції *y=f(x),* необхідно спочатку вказати відрізок, на якому задана функція, і відобразити цей відрізок на вісь *Ox* екрана дисплея. Точці *x* відрізка [a, b] з області задання функції повинна відповідати точка x, екрана дисплея, а точці y=/(x) (значенню функції) — точка y, екрана (рис. 46). Таким чином, відбувається масштабування графіка, що дозволяє відображати на екран будь-який відрізок області задання функції. Тут необхідно враховувати роздільну здатність екрана і добирати масштаби так, щоб побачити

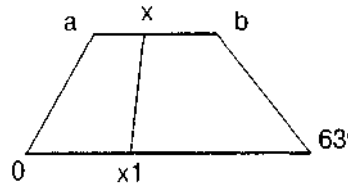

Рис.46. Встановлення відповідності між точками відрізка [*a*, *b*] і віссю *Ox* екрана дисплея

поведінку функції. Координати точок на екрані, на відміну від точок геометричної прямої, мають дискретні значення. Точки осі *Ox* нумеруються від 0 до 639, тобто $x_i=0,$ 1, 2, 3, ..., 639.

З пропорції

$$(x-a)/(b-a)=x/640$$

дістанемо розрахункову формулу для обчислення аргументу функції, що відповідає заданій точці екрана:

$$x = a + x.\ (b - a)/x\max ,$$

де x =640. Для одержаного значення аргументу обчислюється значення функції:

$$y = f(x).$$

Тепер необхідно помножити одержане значення функції на коефіцієнт масштабування, який показує, скільки пікселів (фізичних точок екрана) міститься у вибраній одиниці масштабу:

$$k = x \quad /(b-a).$$

Використовуючи цей масштабний коефіцієнт, знаходимо значення координати у на екрані:

Тепер слід врахувати особливість розташування осі *Oy* на екрані і «перевернути» графік з урахуванням традиційного положення осей координат, а також змістити графік так, щоб його зручно було аналізувати, наприклад,

$$\mathbf{y, = y_{max} /2\text{-}У,}$$

(у цьому випадку нуль осі *Oy* буде знаходитися на середині екрана). З одержаними координатами x_1 і y_1 точка виводиться на екран. Оскільки реальні координати графіка функції, із якими вона обчислюється, — дійсні числа, а координати точок екрана — цілі, то при обчисленні y_1 необхідно округлювати отриманий результат до найближчого цілого числа. Округлення виконується за допомогою функції trunc, аргументом якої є вираз дійсного типу:

```
y1 := trunc(y * xmax / (b - a));
```

Програма побудови графічних зображень повинна починатися з оператора підключення бібліотеки графічних процедур і спеціальної бібліотеки, яка дозволяє використовувати функції операційної системи (бібліотеки Crt). Цей оператор розміщується відразу за заголовком програми і має вигляд:

```
uses Graph, Crt;
```

Сама програма складається з описів процедур побудови зображень, опису функції, для якої будується графік, і опису спеціальної процедури установки графічного режиму. Головна програма при цьому містить оператори виклику цих процедур і оператор

```
repeat until keypressed; {виконати порожній цикл до
                          натиснення будь-якої клавіші}
```

який затримує зображення на екрані до натискання будь-якої клавіші (інакше після побудови останньої точки зображення воно зникає).

Програма побудови графіка функції y=sin(x) на відрізку [— ;] має такий вигляд:

```
program E37;
uses Graph, Crt;
procedure Init;
var gr,gm: integer;
begin
  gr := 0;  { автоматичне розпізнавання типу дисплея }
  InitGraph( gr,gm, ' '); {файл egavga.bgi знаходиться
                           в одному каталозі з turbo.exe}
  if GraphResult <> grOk
  then Halt (1);  {перевірка правильності
                   встановлення графічного режиму}
end;
function f (x: real): real; { обчислення функції f(x)}
begin
  f := sin(x);
end;
procedure grafic;
var xmax,ymax,x1,y1: integer; x,y,a,b:real;
```

```
begin
  xmax := 640; ymax := 480; a := -pi; b := pi;
  for x1 := 1 to xmax do
    begin
      x := a + x1 * (b - a) / xmax;
      y := f(x); { виклик підпрограми-функції}
      y1 := trunc(y * xmax / ( b - a));
      y1 := ymax div 2 - y1;
       PutPixel(x1, y1, 2) { виведення точки
                             зеленим кольором }
    end
end;
begin {головна програма}
  Init; { виклик процедури установки графічного
          режиму }
  Grafic; { виклик процедури побудови графіка функції}
  repeat until keypressed
end.
```

При побудові складних рисунків, які виводяться поточково, можна використовувати прийом мозаїки: намалювати малюнок по клітинках (рис.47) і заповнити таблицю цілих чисел: кожне число відповідає номеру кольору відповідної клітинки, а координати клітинки — індексам елементів таблиці 14.

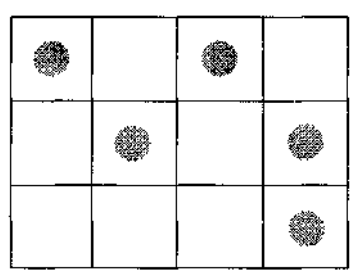

Рис. 47. Рисунок по клітинках

Таблиця 14

0	15	0	15
15	0	15	0
15	15	15	0

Задання кольору. Деякі графічні процедури, такі, як побудова точки, вміщують параметр кольору, але для багатьох інших використовується поточний колір, який визначений останнім процедурою задання кольору, що має вигляд

SetColor (колір);

Наводимо таблицю кольорів (таблиця 15), де кожному кольору відповідає певний номер.

Таблиця 15

Чорний	Синій	Зелений	Бірюзовий
0	1 1	2 2	3
Червоний	Малиновий	Коричневий	Сірий-с
4	5	6	7
Сірий-т	Голубий	Зелений-с	Бірюзовий-с
8	9	10	11
Червоний-с	Малиновий-с	Жовтий	Білий
12 ^	13	14	15

Відрізок. Для побудови відрізка необхідно вказати координати його кінців. Відрізок зображується поточним кольором. Процедура побудови відрізка має вигляд

```
Line(x1, y1, x2, y2);
```

При побудові зображення комп'ютер ніби малює променем. Промінь зупиняється в останній точці зображення, координати якої запам'ятовуються. Можна побудувати відрізок, вказавши або зміщення від цієї точки вздовж осей *Ox* і *Oy,* або координати кінця відрізка (початок там, де зупинився промінь). Зміщення вказується у вигляді цілого числа: додатне означає збільшення останньої координати на величину зміщення, від'ємне — відповідне зменшення координати. Зміщення зручно використовувати для побудови зображення у відносних координатах, коли достатньо вказати координати першої точки, а координати решти точок обчислюються. В такий спосіб можна переміщувати картинку на екрані.

Процедура побудови відрізка з вказуванням зміщення має вигляд

```
LineRel(dx, dy);
```

Для побудови відрізка з вказуванням останньої його точки використовується така процедура:

```
LineTo (x, y);
```

Цю процедуру зручно використовувати для побудови графіка функції, яка або різко зростає, або різко спадає. У такому разі зображення, побудоване по точках, має розрив, оскільки для двох сусідніх аргументів два значення функції знаходяться далеко одне від одного. Перша точка графіка виводиться за допомогою процедури **PutPixel,** а всі інші точки, координати яких обчислюються й в циклі, будуються з використанням процедури **LineTo.**

Промінь, який будує зображення, може переміщуватись, не залишаючи сліду. Це можна зробити, використовуючи процедури, аналогічні двом попереднім:

```
MoveRel (dx, dy);
MoveTo (x, y);
```

Аналогічно до відрізка будується прямокутник: для його побудови необхідно вказати координати кінців будь-якої діагоналі. Прямокутник може бути побудований у вигляді контуру або зафарбованої фігури. Для контурного зображення прямокутника використовується процедура

```
Rectangle (x1, y1, x2, y2);
```

а для зафарбованого —

```
Bar(x1,y1,x2, y2);
```

Побудова стовпчикової діаграми. На рис. 48 подана стовпчикова діаграма падіння успішності учня по чвертях навчального року. У першій чверті успішність становила 75%, у другій — 50% , у третій — 25%. Побудуємо таку діаграму за допомогою програми, розмістивши найвищий стовпець у центральній частині екрана (його довжина — 480:3 = = 160 пікселів) і розрахувавши

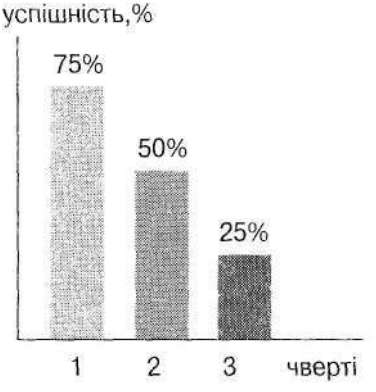

Рис. 48. Стовпчикова діаграма успішності учнів по чвертях

решту пропорційно їх значенням відносно цього розміру. За шириною вся діаграма займає також 1/3 частину екрана, тобто початкове значення координати *x* дорівнює 210 пікселів, ширина стовпців по 35 і відстань між ними також 35 пікселів. Для обчислення висоти другого стовпця складемо пропорцію 75:50 = 160:x. Звідси $x = 106$ (з округленням до цілого). Висота другого стовпця 53 пікселі. Оскільки нижні основи прямокутників знаходяться на одному рівні, на якому $y=320$, то інші координати (вздовж осі *Oy)* обчислюються відніманням від 320 висоти відповідного стовпця.

Розміщення тексту на зображенні здійснюється за допомогою процедури

OutTextXY (x, y, 'текст');

де *x, y* — координати, починаючи з яких буде виведено текст.

Програма побудови стовпчикової діаграми має таку саму структуру, як і програма E36:

```
program E38;
uses Graph, Crt;
procedure Init;
var gr,gm: integer;
begin
  InitGraph( gr,gm, ' '); {файл egavga.bgi знаходиться
                           в одному каталозі з turbo.exe}
  if GraphResult <> grOk
  then Halt (1)  {перевірка правильності
                  встановлення графічного режиму}
end;
procedure Diagram 1;
begin
  SetColor(14); Bar(210, 160, 245, 320);
  SetColor(10); Bar(210 + 2*35, 320 - 106, 210 + 3*35,320);
  SetColor(2);  Bar(210 + 4*35, 320 - 53, 210 + 5*35, 320);
  OutTextXY(210, 320 + 3O, 'Успішність по чвертях')
end;
begin {головна програма}
  Init; {виклик процедури встановлення графічного режиму}
  Diagram 1;
```

```
  repeat until keypressed
end.
```

Побудова дуги кола. Для побудови дуги кола потрібно вказати центр відповідного кола (x, y), початковий кут, кінцевий кут і радіус. Кути відраховуються проти годинникової стрілки, їх величина вказується в градусах (від 0 до 360). Відлік кута проводиться від направленого вправо по горизонталі радіуса. Процедура побудови дуги має вигляд

Arc (x, y, початковий кут, кінцевий_кут, радіус);

Для зафарбованого сектора використовується процедура із такими самими параметрами:

PieSlice (x, y, початковішій_кут, кінцевий кут, радіус);

Якщо вказати значення кутів відповідно від 0 до 360, то одержимо коло або зафарбований поточним кольором круг. Коло можна також зобразити за процедурою

Circle (x, y, r);

Розглянемо приклад побудови з кіл «рогу достатку». Центри кіл переміщуються по колу, що є траєкторією точки, одна координата якої задається косинусом, а друга — синусом того самого кута *x/к*. При *к = 20* маємо замкнене коло, при *к* = 10 — два кола, при *к = 30* — півколо.

Програма матиме вигляд

```
program E39;
uses Graph, Crt;
procedure Init;
var gr,gm: integer;
begin
  gr := 0;  { автоматичне розпізнавання типу дисплея }
  InitGraph ( gr.gm, ' '); {файл egavga.bgi знаходиться
                            в одному каталозі з turbo.exe}
  if GraphResult <> grOk
  then Hait(1)  {перевірка правильності
                 встановлення графічного режиму}
end;
procedure Rog;
var xmax, ymax,x1,y1,x,y,r: integer;
```

```
begin
  xmax := 640; ymax := 480;
  forxl := 1 to 125 do
    begin
      x := trunc(xmax /2 + ymax/2 * cos( x1/20 ));
      y := trunc(ymax /2 + xmax/4 * sin( x1/20 ));
      r := trunc(ymax /4 - x1  );
      Circle (x, y, r)
    end
end;
begin {головна програма}
  Init; {виклик процедури встановлення графічного
        режиму}
  Rog;
  repeat until keypressed
end.
```

Побудова кругової діаграми. Кругові діаграми являють собою круги, поділені на сектори різної площі. Площа кожного сектора відповідає деякій частині круга, як правило, у відсотковому відношенні, а за одиницю (ціле) береться площа всього круга. Замість площі сектора можна обчислити його кутовий розмір. Побудуємо кругову діаграму успішності учня, що відповідає діаграмі рис. 49.

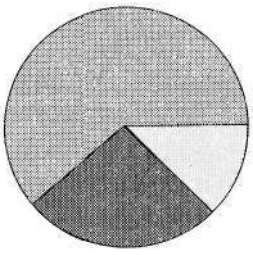

Рис. 49. Кругова діаграма успішності

Програма може мати вигляд:

```
program E40;
uses Graph, Crt;
procedure init;
var gr.gm: integer;
begin
  gr := 0;  { автоматичне розпізнавання типу дисплея }
  InitGraph (gr, gm, ' '); {файл egavga.bgi знаходиться в
                           одному каталозі з turbo.exe}
  if GraphResult <> grOk
```

```
    then Halt (1)  {перевірка правильності
                    встановлення графічного режиму}
end;
procedure Diagram2;
var xmax,ymax,x1,y1,x,y,r:  integer;
begin
  xmax := 640; ymax := 480;
  x := 320;  y := 240;  r := 200; {координати центра
                                   та радіус діаграми}
  x1  := trunc( 75 * 36/15 ); {обчислення довжини першої
                               дуги}
  SetColor (14);  PieSlice ( x, y, 0, x1, r);
  y1 :=trunc(50 * 36/15) + x1;
  SetColor (10);
  PieSlice( x, y, x1 + 1, y1, r); {друга дуга + перша дуга}
  SetColor (2);
  PieSlice( x, y, y1 +1, 360, r) {замикання кола}
end;
begin {головна програма}
  Init; {виклик процедури встановлення графічного
       режиму}
  Diagram2;
  repeat until keypressed
end.
```

ЗАПИТАННЯ І ЗАВДАННЯ

1. Як розташовані осі координат на екрані дисплея?
2. Що таке графічні примітиви?
3. Як зафарбувати один піксел екрана?
4. Як відобразити відрізок [a, b] на всю ширину екрана?
5. Скільки пікселів буде містити одиниця вимірювання на відображеному на екрані відрізку [a, b]?
6. Як розмістити графік на екрані з врахуванням направленості вісі *Oy* в протилежному від традиційного напрямку?
7. Якого типу дані відповідають екранним координатам?
8. В якій бібліотеці мови Паскаль містяться графічні підпрограми і як підключити цю бібліотеку при виконанні програми?
9. Як затримати зображення на екрані після повної його побудови?
10. Як задати колір для побудови ліній, прямокутників і дуг?
11. Як побудувати лінійну діаграму?

12. За допомогою якої процедури можна побудувати відрізок, використовуючи тільки одну пару координат?
13. Як перемістити промінь, що малює, по екрану так, щоб не було сліду?
14. Як побудувати графік функції, яка різко зростає або різко спадає?
15. За допомогою яких процедур можна побудувати контурний і зафарбований прямокутники?
16. Як підписати малюнок?
17. Як побудувати дугу кола? Як за допомогою процедури побудови дуги побудувати коло?
18. Як зобразити зафарбований сектор?
19. Для чого використовується функція trunc?
20. Що таке кругова діаграма?
21. У програму побудови графіка функції додати виведення на екран координатних осей і позначок одиниць вимірювання на них.
22. За допомогою програми побудови графіка функції побудувати графік $\log_2 x$ на різних частинах області визначення цієї функції.
23. Програму побудови стовпчикової діаграми доповнити так, щоб на зображенні були як на рис. 48, всі підписи, і в кожному секторі помістити прямокутник білого кольору з необхідним текстом.
24. Як побудувати концентричні кола, зафарбувавши кожне іншим кольором?
25. Як побудувати ялинку з n трикутників, визначивши її попередньо рекурсивно і використавши в програмі рекурсивну графічну процедуру?
26. Як використовуючи ідею дитячої гри «мозаїка», побудувати зображення різнокольорового метелика, зберігши його попередньо у вигляді таблиці цілих чисел (значень кольорів) і виводячи потім цю таблицю за точками на екран, застосовуючи подвійний цикл?

Записи

Раніше запис був визначений як сукупність байтів на носієві, обмежена з двох боків спеціальними позначками. Таке визначення дає уявлення про запис як про одиницю обміну між зовнішньою й оперативною пам'яттю комп'ютера. Однак сам запис може бути складною структурою, яка містить різні дані. Запис може, наприклад, відповідати рядку відомості заробітної плати, в якій вказані прізвище і кілька чисел, або рядку класного журналу, де також стоять

прізвища й оцінки. Отже, запис є складною конструкцією. Тому в мові Паскаль й інших програмних системах слово «запис» має подвійний зміст: це і складна структура, і одиниця даних на носієві (наприклад, дискові).

Запис — це сукупність різнорідних даних, що описуються та обробляються як єдине ціле.

Дані, із яких складається запис, називаються його полями. Поля можуть бути як простими даними, так і складеними, наприклад масивами або записами. За допомогою записів зручно описувати властивості об'єктів, зберігаючи їх разом. Із записів складаються бази даних, що включають описи кількох об'єктів.

Опис запису складається з ключового слова **record,** після якого вказуються імена полів і тип кожного поля. Тип поля відділяється від імені двокрапкою. Опис запису закінчується словом **end** і крапкою з комою.

Записи описуються в розділі типів даних **type.** У цьому розділі вказується ім'я класу об'єктів (ім'я типу) і опис цього класу. Для кожного об'єкта класу є своє ім'я в розділі змінних **var** з описом даного типу. Це ім'я використовується далі в програмі.

Приклад 1. Об'єкт — фізичне тіло з параметрами *a* (довжина), b (ширина), c (висота). Його опис може мати вигляд а) коли для кожного поля вказано тип даного, або б) коли однотипні поля, що йдуть підряд, описані разом:

```
а) type z = record      б) type z = record
     a: integer;              a, b, c: integer
     b: integer;           end;
     c: integer            var x: z;
   end;
  var x: z;
```

Приклад 2. Об'єкт — товар, який характеризується назвою і ціною:

```
type tovar = record
  sign: string[ 20];   price: real
end;
```

Приклад 3. Об'єкт — дата народження: день, місяць, рік. День можна вказати як діапазон значень — такий тип

даних називається інтервальним. Він уже траплявся при описі масивів, де вказані інтервали номерів елементів. Цей тип можна використовувати для цілочислових і символьних даних в описах, а також як позначення оператора варіанта. Даними інтервального типу можна задавати значення констант у розділі **const:**

```
type
  date of birth = record
  day: 1 ..31;
  month: string[10];
  year: integer
end;
var date: date_of_birth;
```

Записи можна об'єднувати в масиви. Масив записів може бути описаний у розділі **type** або **var.** Для прикладу 2 розділ змінних може мати вигляд

```
var x: array [1..100] of tovar; y: tovar;
```

x — масив записів, до кожного елемента якого використовується звичне звернення, наприклад, *x[i]; y* — проста змінна.

Для звернення до поля запису використовується складене ім'я, яке складається з двох імен, розділених крапкою. Перше з цих імен — ім'я змінної типу запис з розділу **var,** друге — ім'я поля цього запису з розділу **type.** Так, для товарів імена полів у програмі мають вигляд:

```
y.sign, y.price, x[1].sign, x[i].price
```

Приклад 4. Припустімо, потрібно описати відомості про робітника підприємства: прізвище, посаду, дату народження й зарплату (таблиця 16).

Таблиця 16

Зміст відомостей	Прізвище	Посада	Дата народження	Зарплата
Ім'я поля запису	name	position	date	salary
Тип даних поля	string [20]	string [10]	date_of_birth	real

Кожному полю запису потрібно спочатку присвоїти ім'я, потім визначити, який тип найзручніший для обробки цих даних. Описувані відомості включають структуру типу запис у вигляді поля дати народження. її також потрібно уточнити і описати в розділі типів раніше, ніж запис, що стосується робітника.

В описі даного запису використовується тип «день народження» *(date_of_birth)* з прикладу 3. Загальний опис запису має вигляд:

```
type
date_of_birth = record
  day   : 1 ..31;
   month : string[10];
  year   : integer
end;
worker = record
  name     : string[ 20 ];
  position  : string[ 10 ];
  date      : date_of_birth;
  salary    : real
end;
var x: array [ 1 ..7] of worker; w : worker;
```

Поле запису *date* містить запис із трьох полів. При формуванні імені поля цього внутрішнього запису необхідно використовувати потрійне ім'я: ім'я змінної розділу var, ім'я поля запису *worker* та ім'я поля запису *date_of_birth.*

Наприклад, для змінної *w* звернення до місяця народження робітника в програмі матиме вигляд

```
w.date. month
```

У програмах введення й виведення записів виконується окремими полями, але можна присвоїти одному запису значення іншого, при цьому проходить копіювання ділянки пам'яті:

```
x[1] := w;
```

Оператор приєднання. Для обробки запису використовується оператор **with,** який дозволяє вказати один раз ім'я запису з розділу змінних, а потім, у всій області дії оператора, вказувати тільки імена полів цього запису з розділу **type.**

Після слова **with** можна написати кілька імен полів із розділу змінних.

Оператор приєднання має вигляд:

with список **імен** записів **do** оператор;

Оператор може бути простим або складеним, обмеженим операторними дужками. Список імен записів може складатись з одного імені.

Приклад 5. Дано масив записів, який містить відомості про робітників підприємства (див. приклад 4). Надрукуйте:

1) список прізвищ бухгалтерів;

2) список прізвищ робітників віком від ЗО до 50 років;

3) середню заробітну плату на підприємстві.

Як і в інших задачах, де сказано, що «дано n чисел» або що-небудь інше, вихідні дані необхідно описати і ввести в комп'ютер. Для записів цієї задачі визначаються, як і в прикладі 4, їх структура, імена й типи полів. Потім визначається ім'я масиву записів, який використовується в програмі, і допоміжні змінні, а також імена результатів.

Програма розв'язування задачі має вигляд

```
program E41;
const n = 3;
type date of birth = record
  day: 1 ..31;
  month: 1..12;
  year: integer
end;
worker = record
name: string[20];
position: string[10];
date: date_of_birth;
salary: real
end;
var x: array [ 1..n ] of worker; i,j,g:integer;
    S: real; p: string[10];
begin
{формування масиву записів}
  for i := 1 to n do
    with x [i] do
```

```
      begin
        writeln(' відомості про ', i, 'робітника');
        writeln('прізвище, ініціали');
        writeln('посада');
        readln(position);
        writeln('число, місяць і рік народження');
        readln(date.day, date.month, date, year);
        writeln(date.day, date.month, date.year);
        writeln( 'зарплата');
        readln(salary)
      end;
{розв'язування задачі 1 - друкування списку прізвищ
 бухгалтерів}
  p := 'бухгалтер';
  j := 0; {лічильник рядків списку - кількість бухгалтерів}
  for i := 1 to n do
    with x[i] do
      if p = position
      then
        begin
          j:=j+1;
          writeln(j, '.', name); {після номера в списку
                                  друкується крапка та ім'я}
        end;
{розв'язування задачі 2 - список прізвищ робітників
 віком від 30 до 50 років}
  writeln('список прізвищ робітників віком від 30 до 50
           років ');
  j:=0;
  write('введіть поточний рік у вигляді чотиризначного
         числа ');
  readln(g);
  for i := 1 to n do
    with x[i] do
      begin
        if ( g - date.year < 50 ) and ( g - date.year > 30 )
        then
          begin
            j:=j+1;
            writeln(j, '.', name)
```

```
            end
        end;
    {розв'язування задачі - обчислення середньої заробіт-
      ної плати}
      S :=0;
      for i := 1 to n do
         S := S + x[i].salary;
      writeln('середня зарплата =', S/n:10:2)
    end.
```

ЗАПИТАННЯ І ЗАВДАННЯ

1. Як розуміти слово «запис»?
2. Як звернутися в програмі до поля запису?
3. Що необхідно зробити, щоб описати в програмі об'єкт, характеристиками якого є різнотипні дані?
4. Яке ім'я даного, що містить відомості про те, якого числа народився робітник підприємства?
5. Поясніть, що означають імена: x[2].position, x[5].date.month, w.name.
6. Для чого використовується оператор with?
7. Опишіть об'єкт «учень 11 класу», використовуючи запис.
8. Створіть базу даних для свого класу і з її допомогою отримайте відомості:
   а) про найстаршого учня в класі;
   б) про наймолодшого учня;
   в) про оцінки кожного учня з усіх предметів;
   г) про відмінників;
   д) про найбільш відстаючого учня;
   є) про середній бал для кожного учня за оцінками останньої чверті;
   ж) про середню успішність класу з усіх предметів.

2.16, Файли у мові Паскаль

Файл у Паскалі складається з однотипних даних. З даними файла можна проводити дві операції: запис або читання. У Паскалі проводиться обробка послідовних файлів, в яких дані записуються або зчитуються одне за одним. Запис можна читати, пропускаючи попередні, якщо відомий його порядковий номер у файлі. Для того щоб із дани-

ми файла проводити дії, файл потрібно відкрити для відповідної операції.

Розглянемо етапи, які необхідно виконати для кожної операції при роботі з файлом.

Операція запису. Запис у файл означає введення до нього нових даних. Файл розміщується на носіях в зовнішній пам'яті комп'ютера. Дані для занесення у файл формуються в оперативній пам'яті як значення деякої змінної. Операцією запису у файл це дане копіюється з оперативної пам'яті до зовнішньої. Отже, форма подання даного, його тип і структура повинні бути однакові і для записів файла, і для змінної, з якої це дане копіюється.

1. Опис файла. Опис файла може бути поданий в розділі типів або в розділі змінних. Нехай файл f складається з цілих чисел. Його опис має вигляд:

```
var f: file of integer;  a: integer;
```

де *a* — компонента файла, даного того самого типу, що й записи файла. Тип даних файла вказується після слова of в описі — це може бути числовий або символьний тип, масив або запис. Складений тип запису файла необхідно попередньо описати в розділі **type.**

2. Встановлення відповідності між логічним і фізичним іменами файла. *Логічне ім'я* — це ім'я змінної з розділу **var.** За цим іменем до файла звертаються в програмі. *Фізичне* ім'я — це ім'я, під яким файл записаний на диску. Процедура встановлення відповідності між іменами файлів має такий вигляд:

```
assign (логічне ім'я файла, фізичне ім'я);
```

ДЛЯ наведеного опису ця процедура мас вигляд:

```
assign (f, 'F.DAT');
```

Фізичне ім'я взято в апострофи. Воно з'явиться в змісті диска в тому самому каталозі, в якому знаходиться файл *turbo.exe.*

3. Відкриття файла для операції «запис». Ця дія виконується процедурою

```
rewrite (f);
```

При відкритті файла для занесення до нього даних на диску з'являються два спеціальні записи: початок файла, який містить фізичне ім'я, і ознака кінця файла. Кожне відкриття файла для запису означає створення файла. Якщо для операції «запис» відкрити файл з уже існуючими даними, то всі дані файла зникнуть. Тому відкривати для запису можна тільки файли з новими іменами. При занесенні даних до файла вони будуть розміщуватись у кінці файла.

Файл може містити довільну кількість даних. Обмеження розміру файла в програмі ніяк не оговорюється. В оперативній пам'яті досить однієї ділянки, яка має формат із записів файла, а на диску файл може бути такого розміру, скільки є вільного простору в момент його створення.

Запис даних у файл здійснюється за допомогою процедури

write (f, a);

Приклад 1. Нехай потрібно створити файл з 10 цілих чисел.

Програма має вигляд

```
program E42;
var f: file of integer;  a,i: integer;
begin
  assign(f,  'F.DAT');
  rewrite(f);
  writeln('введіть  10 цілих чисел,  після кожного');
  writeln('натискайте  Enter');
  for i := 1 to 10 do
      begin
        readin (a);
       write (f,a)
      end
end.
```

Операція читання. Для читання даних із файла його необхідно описати, встановити відповідність між логічним і фізичним іменами, відкрити для читання і зчитувати дані. Перші два кроки — опис і встановлення відповідності імен — такі самі, як і для операції запису. Якщо з файлом вико-

нуються різні операції, то перед виконанням наступної операції його необхідно закрити процедурою

close (f);

Відкриття файла для читання проводиться за процедурою

reset (f);

Для читання даних із файла використовується процедура

read (f, a);

Після створення файла і кількох перетворень може бути невідомою кількість його записів. Тому при читанні даних із файла зручно використовувати спеціальну функцію, яка контролює ознаку кінця файла. Ця функція набуває значення «істинно», якщо трапляється ознака кінця файла, і «хибно», якщо прочитано інший запис. Оскільки при відкритті файла для читання вже зчитується перший його запис, який містить ім'я файла, то можна поставити контроль ознаки кінця файла, навіть не зчитавши жодного запису процедурою **read(/,** а);

Аналіз ознаки кінця файла виконується за допомогою функції

eof (f)

Оскільки кількість записів у файлі невідома, то використовувати при читанні даних файла цикл «перелік» не можна, використовується цикл «доки». Його заголовок

while not eof (f) do

треба розуміти так: поки не зустрілась ознака кінця файла, виконувати цикл.

Приклад 2. Дано файл цілих чисел. Порахувати кількість додатних, від'ємних і нульових елементів у файлі. Використаємо у даному прикладі програму створення файла цілих чисел Е40:

```
program E43;
const k = 5;
var f: file of integer;  a,i,n,p,z: integer;
```

```
begin
  assign(f, 'F.DAT');
  rewrite(f); {створення файла}
  writeln('введіть ', к,' цілих чисел, після кожного
           натискайте Enter');
  for i := 1 to k do
    begin
      readln(a);
      write(f,a)
     end;
  close(f); {закриття файла для операції запису}
{ розв'язування задачі - підрахунок різних елементів}
  n := 0; p := 0; z := 0;   { n - від'ємні, p - додатні, z -нулі}
  reset(f);
  while not eof(f) do
    begin
      read(f,a);
      if a = 0then z :=z +1;
       if a < 0 then n := n + 1;
      if a > 0 then p := p + 1;
    end;
  write('n = ',n, ' z = ', z, '  p = ',p);
end.
```

Приклад 3. Нехай потрібно розширити даний файл, додавши до нього нові дані. Як відомо, файл із даними не можна відкривати для запису, тому для розв'язування подібних задач необхідно використовувати допоміжний файл.

Розв'язування задачі розширення файла складається з таких етапів:

1) відкрити даний файл /для читання, а допоміжний g — для запису;
2) читати дане з вихідного файла /і відразу записувати його у файл g;
3) після закінчення переписування даних закрити файл f;
4) вводити нові дані з клавіатури і записувати їх у файл g, додаючи до вже існуючих там даних файла f;
5) закрити файл g;
6) відкрити файл /для запису, а файл g — для читання;
7) читати дані з файла g і записувати їх у файл f.

Таким чином, у файл /до наявних там даних додаються нові. Якщо потрібно вставити нові дані в середину файла, то треба в другому пункті розв'язування контролювати зчитані дані і, дійшовши до місця вставляння, припинити читання, записати потрібні дані до файла g, а потім дописати туди решту з вихідного файла (пункти 3 і 4), далі виконати пункти 5, 6, 7.

Програма доповнення даними файлів (використовується готовий файл f із прикладу Е40) може мати вигляд:

```
program  E44;
var f,g:file of integer;   a: integer;
begin
   assign(f, 'F.DAT');
   assign(g, 'G.DAT');
{ перезапис даних з вихідного файла в допоміжний}
  reset(f);
  rewrite(g);
  while not eof(f) do
    begin
      read(f,a);
      write (g,a)
    end;
  close(f);
{ додавання даних до допоміжного файла}
  readln (a);
   while a <> 0 do { ознака закінчення введення нових
                     даних  - нуль}
    begin
      write (g,a);
      readln (a)
    end;
  close(g);
  reset(g);
  rewrite (f);
{ перезапис даних назад у вихідний файл}
   writeln('модифікований файл');
  while not eof(g) do
    begin
      read(g,a);
```

```
        write(f,a);
        writeln(a)
      end
  end.
```

ЗАПИТАННЯ І ЗАВДАННЯ

1. Які операції можна проводити з даними файла?
2. Чому другий параметр а процедури write(f, a) і read(f, a) повинен бути такого самого типу, як і дані файла?
3. Що таке фізичне ім'я файла, чим воно відрізняється від логічного імені?
4. Як відкрити файл для запису?
5. Що буде, якщо раніше створений файл із даними відкрити для запису?
6. Чим обмежена кількість даних у файлі?
7. Чим відрізняється файл від масиву?
8. Як прочитати дані з файла, не знаючи кількості цих даних?
9. Як додати дані до існуючого файла?
10. Як для файла цілих чисел знайти найбільший елемент даних?
11. Як для файла цілих чисел переписати додатні числа в один додатковий файл, а від'ємні - в інший?
12. Як для задачі 8 попереднього параграфа створити файл із записами про учнів класу і виконати всі пункти завдання, використовуючи дані файла?

Поняття про візуальну мову програмування - Delphi

Паскаль — це мова процедурного програмування. Розробка процедурних програм знаходила найбільше застосування при створення підпрограм, що виконують великий обсяг обчислень. Однак процедурні програми звичайно мали надто складні меню і використовували командні рядки, тісно пов'язані з функціями прикладних програм. З метою розширення можливостей мови Паскаль на програмування складних об'єктів була запропонована мова програмування Delphi.

Delphi — це візуальний компілятор, що працює в середовищі Windows на базі мови Pascal, а якщо точніше, то на базі мови Object Pascal. З появою Delphi відпала необхідність програмувати стандартні елементи управління Windows, такі як рядок редагування, кнопки і діалогові вікна, усе це вже є у вигляді готових компонентів шаблонів.

Якщо для описання зручного інтерфейсу користувача на мові Паскаль потрібен додатковий час, то у Delphi на створення інтерфейсу користувача іде порівняно мало часу, тому що багато елементів інтерфейсу вже представлені у вигляді готових до використання компонентів. І для їх використання досить за допомогою мишки перетягнути їх на форму програми.

Delphi є мовою програмування реакцій на події. Подія — це відгук на одну чи більше зовнішніх впливів. Суть керованого подіями програмування — це відстеження таких подій, що вимагають реагування програми. У процесі функціонування Windows виникає дуже багато подій, але тільки деякі з них можуть мати відношення до нашої програми і вимагають реагування.

Delphi поряд із процедурним програмуванням підтримує й об'єктно-орієнтоване програмування. Основні концепції якого — інкапсуляція, поліморфізм і наслідування.

Інкапсуляція — це механізм, що поєднує дані та програму, що маніпулює цими даними, а також захищає і те, і інше від зовнішнього втручання чи неправильного використання. В об'єктно-орієнтованому програмуванні програма і дані можуть бути об'єднані разом; у цьому випадку говорять, що створюється так званий «чорний ящик». Коли програма і дані з'єднуються таким чином, то створюється об'єкт.

Поліморфізм — це властивість, що дозволяє те саме ім'я використовувати для розв'язування двох чи більше схожих, але технічно різних задач. Метою поліморфізму в об'єктно-орієнтованому програмуванні є використання одного імені для задавання загальних для класу дій. Виконання кожної конкретної дії буде визначатися типом даних.

Наслідування — це процес, за допомогою якого один об'єкт може здобувати властивості іншого. Точніше, об'єкт може наслідувати основні властивості основного об'єкта і додавати до них риси, характерні тільки для нього.

У Delphi, на відміну від інших середовищ, об'єктами можуть бути будь-які елементи. Наприклад кнопки, рядки, поля даних — у загальному випадку елементи, які можна створити в процесі виконання програми.

З використанням візуальних засобів програмування можна безпосередньо працювати з об'єктами, відразу спостерігаючи за ними на екрані. При розміщенні об'єкта у вікні форми (рис. 50) у візуальному середовищі основні параметри об'єкта (розмір, положення, властивості і т.д.) зберігаються в програмі, що виконується. При переміщенні об'єкта на форму відповідна програма записується у вихідний файл. Програму досить часто називають *програмним кодом* або *кодом*.

Для забезпечення можливості реагувати на зовнішні події об'єктам приписують властивості. *Властивості* — це атрибути, що складають індивідуальність об'єкта і допомагають описати його. Властивості включають, крім іншого, такі елементи як колір, висоту, ширину і положення об'єкта. Властивість може впливати на зовнішній вигляд об'єкта так само, як і інші не візуальні елементи об'єкта можуть впливати на його поводження.

Для об'єкта властивості — це те саме, що і локальні змінні для процедури. Локальні змінні описуються в процедурі і використовуються нею в процесі роботи. Точно таке ж відношення до об'єктів мають властивості — це атрибути об'єкта, що описують його особливості.

Для роботи з базами даних запропонований потужний об'єкт Borland Database Engine (BDE). З його допомогою здійснюється доступ до основних форматів баз даних (Oracle, Informix, Dbase, Paradox). При цьому відпадає потреба глибокого знання внутрішнього представлення даних у конкретній СУБД. Просто кількома клацаннями мишки у вікно форми виводяться дані, які знаходяться в таблиці.

Також уведена підтримка SQL (Structured Query Language — структурована мова запитів). З її допомогою можна проводити вибірку даних по заданих параметрах, вводити і видаляти дані з таблиці, створювати і видаляти файли даних. У зв'язку з цим Delphi можна застосовувати для створення повноцінних програмних засобів для роботи з базами даних.

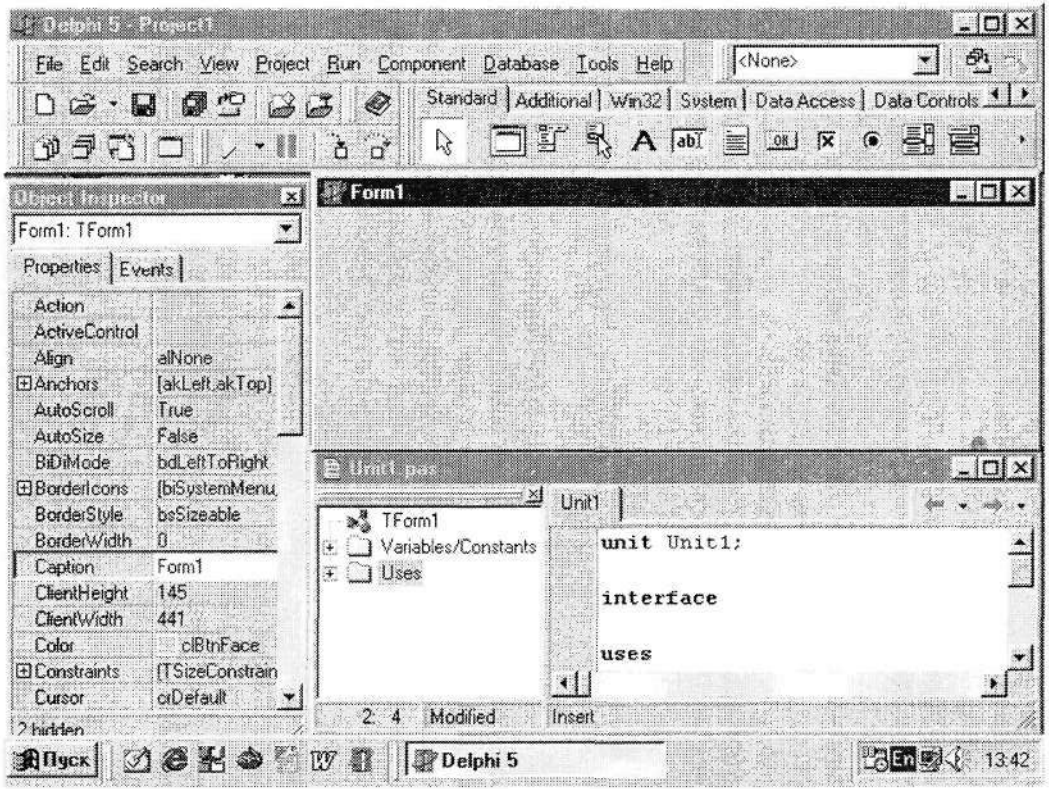

Рис. 50. Середовище програмування Delphi

Після запуску середовища програмування Delphi на екрані відображаються чотири вікна: вікно головного меню з панелями інструментів (стандартна, виведення визначеного вікна на екран, налагодження, візуальних компонентів); вікно інспектора об'єктів (Object Inspector), редактор коду і вікно форми (рис. 50).

У головному меню маємо вказівки:

File — команди для файлових операцій з файлами проекту;

Edit — команди для редагування коду програми у вікні редактора коду;

Search — для контексного пошуку в тексті програми й автоматичної заміни одного тексту іншим;

View — для виведення на екран (на передній план) якогось з вікон середовища програмування, а також файлу або форми;

Project — для під'єднання готового модуля до проекту або від'єднання модуля від проекту;

Run — засіб запуску програми і її налагоджень;

Component — для включення в палітру компонентів нових компонентів і створення власних нових компонентів;

Database — засіб для доступу до баз даних;

Tools — засіб для настроювання середовища програмування, а також виклику допоміжних програм, таких як Database;

Desktop — для створення файлів даних;

Image Editor —для створення власних рисунків;

Help — довідник стосовно Delphi, допоміжних програм, а також основних функцій Windows API (Application Programs Interface — програмний інтерфейс прикладних програм).

У палітрі компонентів розміщені всі встановлені на даний момент візуальні компоненти Delphi. Для того, щоб використовувати якийсь з компонентів у своїй формі, необхідно натисканням лівої клавіші мишки вибрати необхідний компонент і помістити його на вікно форми. Необхідні зміни у вихідний код програми вводяться автоматично.

Перейшовши у вікно інспектора об'єктів, вибираємо за допомогою мишки потрібне вікно з лівого боку на екрані або

натисканням клавіші F11 на клавіатурі. У вибраному вікні за допомогою команди Properties (властивості) можна змінити параметри поточного (виділеного у вікні форми) компонента. Наприклад розмір і розміщення компонента на формі, а якщо є напис на компоненті, то зробити його видимим чи невидимим, доступним чи недоступним тощо. Команда Events (події) допомагає, згідно вибраної події, описати порядок дій програми при виникненні даної події. Якщо для даної події нічого не прописано, то виконуються дії за замовчуванням або нічого не відбувається. Якщо вибрати необхідну подію і двічі клацнути лівою кнопкою мишки, на передньому плані з'являється вікно редактора коду, куди вже автоматично занесена назва процедури та операторні дужки. А функції і процедури описуються так само як і в мові Паскаль.

ЗАПИТАННЯ І ЗАВДАННЯ

1. Назвіть основні концепції об'єктно-орієнтованого програмування.
2. Опишіть можливості середовища програмування Delphi.
3. Які вікна відображаються на екрані дисплея комп'ютера після запуску середовища програмування Delphi?
4. За допомогою якої команди виконується редагування коду програми?