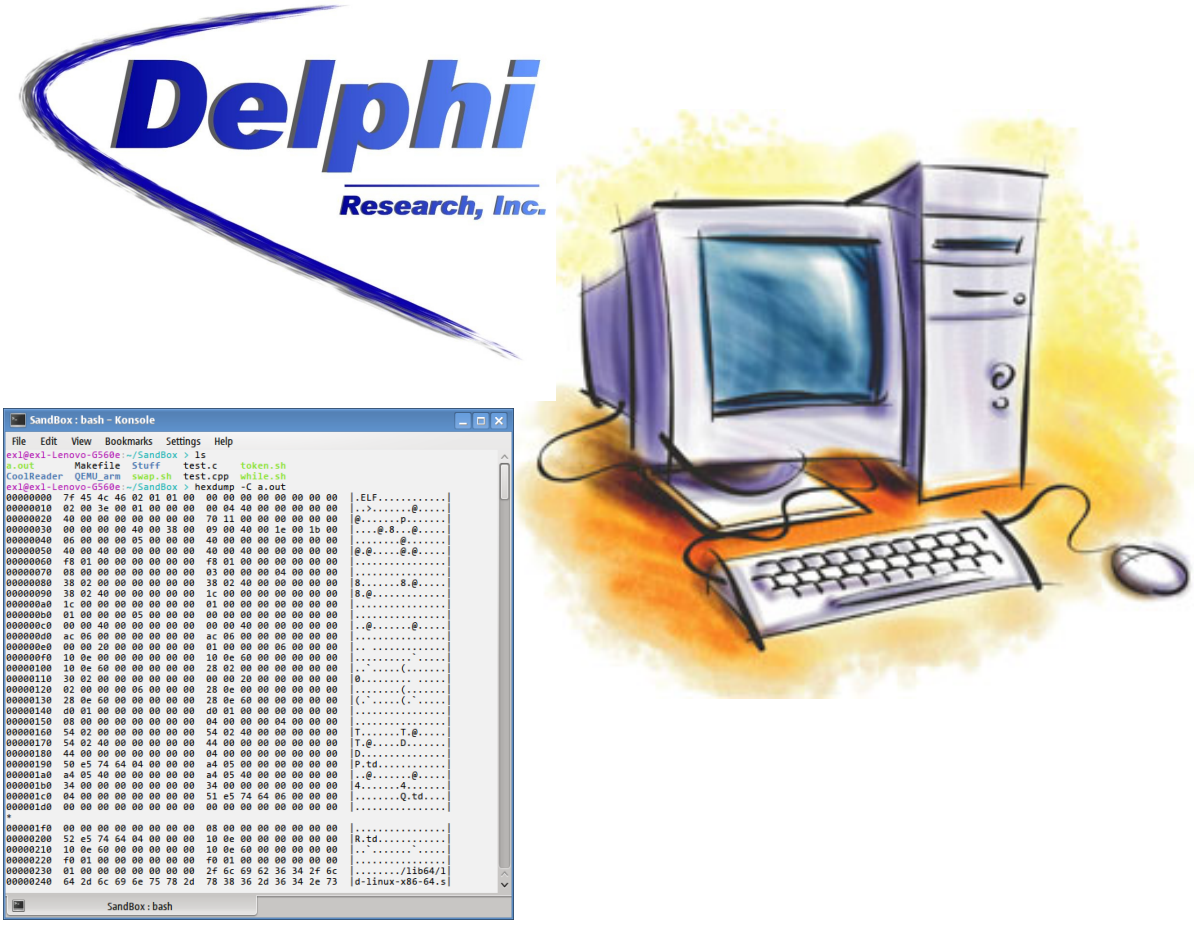
С. В. Костішин, Д. Х. Штофель, Р. С. Белзецький

ІНФОРМАТИКА

Частина 1



Міністерство освіти і науки України
Вінницький національний технічний університет

С. В. Костішин, Д. Х. Штофель, Р. С. Белзецький

І Н Ф О Р М А Т И К А
Частина 1

Лабораторний практикум

Вінниця
ВНТУ
2018

УДК 004.42 (075)
К72

Рекомендовано до друку Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України (протокол № 9 від 19.01.2017 р.)

Рецензенти:
Р. О. Ткачук, доктор технічних наук, професор
В. М. Кичак, доктор технічних наук, професор
О. В. Осадчук, доктор технічних наук, професор

Костішин, С. В.

К72 Інформатика. Частина 1 : лабораторний практикум / С. В. Костішин, Д. Х. Штофель, Р. С. Белзецький. – Вінниця : ВНТУ, 2018. – 113 с.

В лабораторному практикумі з дисципліни «Інформатика» наведено теоретичні відомості та запропоновано практичні рекомендації із формування навичок користування персональним комп'ютером та програмним забезпеченням, а також інформацію щодо створення програмних засобів в середовищі програмування Delphi. Видання розроблено відповідно до плану кафедри біомедичної інженерії та програми дисципліни «Інформатика» для виконання лабораторних робіт студентами спеціальності «Біомедична інженерія» (освітньо-кваліфікаційний рівень – бакалавр). Лабораторний практикум може бути використаний при підготовці студентів інших спеціальностей.

УДК 004.42 (075)

ЗМІСТ

ВСТУП

Сучасний світ і тенденції розвитку інформаційних технологій дозволяють говорити про глобальну інтеграцію комп'ютерної техніки в життя людини. Але сама по собі комп'ютерна техніка не більше, ніж проста купа металу й пластику. Інтелектуальності та швидкодії їй надає саме програмна складова інформаційних систем. Програмне забезпечення дозволяє вирішувати найрізноманітніші задачі, які постають перед інженером-програмістом. Таким чином, часто доводиться мати справу з задачами, що вимагають певної комп'ютерної підготовки та ґрунтовних навичок і знань по написанню та вирішенню базових задач.

Лабораторний практикум з дисципліни «Інформатика» розроблений для студентів І курсу бакалаврату спеціальності 163 – «Біомедична інженерія», але може стати у пригоді студентами інших спеціальностей.

Основна мета лабораторного практикуму – закріпити знання, одержані на лекційних заняттях та в результаті самостійної роботи. Студенти опановують засоби опису даних та процедурні елементи мови Delphi, набувають навичок складання програм та алгоритмізації задач, освоюють прийоми налагодження програмного забезпечення для одержання необхідних результатів.

Мова програмування Delphi є елементом середовища Borland Delphi, до складу якого входять компілятор, редактор коду та інші засоби, що прискорюють процес проектування, написання, налагодження програмного забезпечення. Крім того, можливості мови Delphi значно розширюються розробками сторонніх компонентів.

Процес виконання кожної лабораторної роботи складається з розробки алгоритму розв'язування задачі на ПК; складання структурних блок-схем алгоритмів; їх програмної реалізації; тестування та налагодження розробленої програми та складання інструкції щодо її експлуатації; аналізу результатів розрахунків; підготовки та оформлення звіту.

Звіт виконується на папері формату А4 і повинен мати наскрізну нумерацію (титульний аркуш входить до наскрізної нумерації, але не нумерується). Структура звіту:

1) титульний аркуш (обов'язково містить номер лабораторної роботи, тему, прізвище та групу виконавця, прізвище та посаду викладача);
2) мета роботи;
3) індивідуальне завдання (у випадку його наявності);
4) алгоритм роботи програми (у випадку його наявності);
5) текст (лістинг) програми (у випадку його наявності);
6) результати виконання поставленого завдання (скріншоти вікна роботи програми або відповідні розрахунки);
7) висновки за підсумками виконання лабораторної роботи.

Базове програмне забезпечення – MS Office (Word, Excel, PowerPoint, Access), Borland Delphi 7.0. Мова програмування – Delphi.

ЛАБОРАТОРНА РОБОТА № 1
ЗНАЙОМСТВО З АРХІТЕКТУРОЮ ПЕРСОНАЛЬНОГО КОМП'ЮТЕРА

Мета: вивчити склад персонального комп'ютера та його логічну структуру. Отримати навички підключення основних складових частин системного блока та периферійних засобів.

1.1 Теоретичні відомості

Комп'ютер (computer) – це пристрій, призначений для виконання заданої, чітко визначеної послідовності дій, метою яких є обробка деякої інформації.

Комп'ютер складається з апаратної частини та програмного забезпечення. *Апаратна частина комп'ютера* (hardware) – це набір технічних пристроїв, які ієрархічно взаємопов'язані між собою та виконують задані функції. *Програмне забезпечення* (software) – це набір програм, що керують діями апаратної частини.

Програма (program) – це сукупність чітких, однозначних, зрозумілих комп'ютеру команд (інструкцій), яка визначає послідовність операцій для досягнення розв'язку конкретної задачі.

До апаратної частини комп'ютера відносяться: системний блок, монітор, клавіатура та інші периферійні пристрої.

Системний блок зі своїм апаратним вмістом є центральним елементом персонального комп'ютера, який зберігає та обробляє інформацію. До складу системного блока входять:

1. Материнська плата (МП);
2. Центральний процесорний пристрій (ЦПП);
3. Вінчестер (жорсткий диск);
4. Відеокарта;
5. Блок живлення;
6. Дисководи;
7. Плати оперативної пам'яті;
8. Плати-розширювачі для отримання додаткових функціональних можливостей.

Найпростіша класифікація системних блоків виконується за їх зовнішнім виглядом (за формою корпусу). Корпуси персональних комп'ютерів випускаються в горизонтальному (desktop) та вертикальному (tower) виконаннях. Корпуси, що мають вертикальне виконання, розрізняють по габаритах: повнорозмірний (big tower), средньорозмірний (midi tower) і малорозмірний (mini tower). Серед корпусів, що мають горизонтальне виконання, виділяють плоскі й особливо плоскі (slim).

Крім форми, для корпусу існує важливий параметр, який називається форм-фактор. Він регламентує вимоги, що висуваються до пристроїв, які

розміщуються в системному блоці. На сьогоднішній день, в основному, використовуються корпуси двох форм-факторів: AT і ATX. Форм-фактор корпусу має бути обов'язково узгодженим з форм-фактором материнської (системної) плати комп'ютера.

Основним елементом системного блока є *материнська плата* (mainboard), до якої підключаються решта складових системного блока, а також зовнішні периферійні прилади. Для підключення зовнішніх приладів і засобів материнська плата має панель роз'ємів (рис. 1.1).

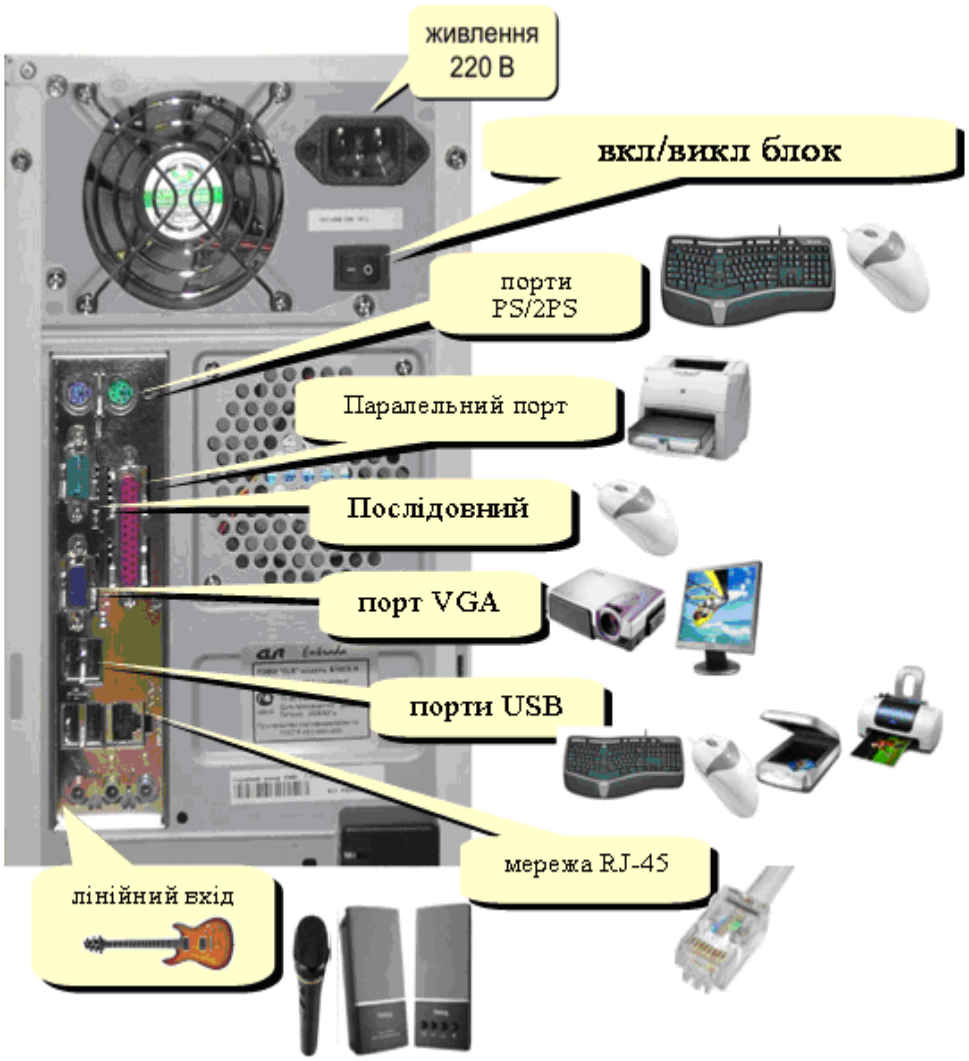


Рисунок 1.1 – Задня панель системного блока

Підключення відповідного пристрою відбувається за допомогою спеціалізованого штекера (рис. 1.2).

Центральний процесорний пристрій (central processing unit, CPU, ЦПП) – це основний елемент комп'ютера, призначений для безпосереднього виконання дій над інформацією та керування роботою всього персонального комп'ютера. Центральний процесор – це «мозок» комп'ютера, він розпізнає та виконує команди, які користувач ставить перед комп'ютером, читає та записує інформацію в пам'ять, передає

команди іншим частинам комп'ютера. Від того, наскільки потужний процесор, багато в чому залежить продуктивність комп'ютера.

Рисунок 1.2 – Основні типи штекерів для підключення периферійних засобів

Внутрішня пам'ять ПК складається з оперативного запам'ятовувального пристрою, постійного запам'ятовувального пристрою, кеш-пам'яті ЦПП (буферної пам'яті).

Оперативний запам'ятовувальний пристрій (RAM, Random Access Memory, ОЗП) – це оперативна пам'ять (пам'ять з вільним доступом), яка конструктивно виконана на окремих друкованих платах з набором мікросхем. Вона складається з певної кількості комірок пам'яті, кожна з яких має свою власну адресу. Оперативна пам'ять призначена, в основному, для зберігання програм та їхніх даних протягом часу роботи комп'ютера. ОЗП може бути обсягом 1, 4, 8, 16, 32, 64, 128, 256, 512, 1024 і більше мегабайт (обсяг має бути кратним 2^n). Сімейство ОЗП персонального комп'ютера поділяється на два основних типи: статична пам'ять (SRAM – Static RAM) і динамічна пам'ять (DRAM – Dynamic RAM).

Кеш-пам'ять (cache) – це швидкодійна буферна пам'ять між процесором і основною пам'яттю. Кеш-пам'ять призначена для часткової компенсації різниці в швидкості процесора та основної оперативної пам'яті – там розміщують дані, які використовуються найчастіше.

Постійний запам'ятовувальний пристрій (Read Only Memory, ROM, пам'ять тільки для читання, ПЗП) складається з декількох основних модулів. *BIOS* (Basic Input/Output System) – основна система введення-виведення, яка записана в ПЗП (звідси назва ROM BIOS). Вона є набором програм перевірки та обслуговування апаратури комп'ютера і виконує роль посередника між операційною системою та периферійною

апаратурою. *Енергонезалежна пам'ять* (CMOS-пам'ять, Complementary Metal-Oxid-Semicondactor) зберігає інформацію про різні параметри конфігурації комп'ютера, наприклад, кількість і тип дискових накопичувачів, тип відеоадаптера, наявність співпроцесора та деякі інші дані. Мікросхема CMOS-пам'яті також містить звичайний електронний годинник. Для того, щоб при відключенні живлення комп'ютера вміст CMOS-пам'яті не зникав і годинник продовжував відраховувати час, мікросхема CMOS-пам'яті живиться від спеціального акумулятора.

Запам'ятовувальний пристрій (зовнішня пам'ять**,** *жорсткий диск* (hard drive) – частина комп'ютера, де зберігається інформація, з якою користувач безпосередньо працює. На відміну від оперативної пам'яті, дані на жорсткому диску зберігаються постійно, не зникають після вимикання комп'ютера або від'єднання вінчестера від материнської плати (дані можуть видалятися або змінюватися користувачем або програмами). Порівняно з ОЗП, швидкість доступу до даних на жорсткому диску в сотні разів нижча.

Відеокарта (videocard, графічний процесор, GPU) – плата, що здійснює електронну обробку графічної інформації, встановлюється в спеціальний роз'єм материнської плати. На деяких материнських платах є вмонтовані (інтегровані) графічні процесори. Порівняно з відеокартами, продуктивність таких процесорів значно нижча, але їхніх можливостей цілком достатньо для вирішення нескладних завдань (робота з текстом, перегляд сторінок Інтернету, перегляд простих форматів відео тощо).

На материнській платі також розміщується набір мікросхем системної логіки. Більшість наборів мікросхем системної логіки мають дворівневу архітектуру та складаються з двох блоків: *північного та південного мостів* (North Bridge та South Bridge). Набір мікросхем підтримує частоту системної шини, частоту інтерфейсу відеокарти та частоту інтерфейсів периферійних пристроїв (наприклад, 100 МГц, 66 МГЦ і 33 МГЦ, відповідно).

Основним блоком набору мікросхем системної логіки є північний міст, в нього включений інтерфейс між процесором і рештою системної плати. Північний міст містить контролери кеш-пам'яті й оперативної пам'яті, інтерфейс між швидкодійною шиною процесора, шиною прискореного графічного порту AGP, шиною оперативної пам'яті та південним мостом.

Південний міст – компонент у наборі мікросхем системної логіки з нижчою швидкодією; він завжди знаходився на окремій мікросхемі. Південний міст підключається до шини PCI додаткових плат розширення (33 МГц). Крім того, зазвичай він містить дві схеми, що реалізовують інтерфейс контролера жорсткого диска IDE і інтерфейс USB периферійних пристроїв, а також схеми, що реалізовують функції пам'яті CMOS і годинника.

1.1.1 Слоти розширення. Види шин архітектури ПК

З іншими пристроями, і, насамперед, з оперативною пам'яттю, процесор зв'язаний групами провідників, які називаються шинами. Основних шин три:

- шина даних;
- адресна шина;
- командна шина.

Адресна шина. Дані, які передаються по цій шині, трактуються як адреси комірок оперативної пам'яті. Саме з цієї шини процесор зчитує адреси команд, які необхідно виконати, а також адреси даних, якими оперують ці команди. Сучасні процесори мають 32-розрядну адресну шину. Це означає, що вона складається з 32-х паралельних провідників.

Шина даних. По цій шині відбувається копіювання даних з оперативної пам'яті в регістри процесора в обидва напрямки. У ПК на базі процесорів Intel Pentium шина даних 64-розрядна. Це означає, що за один такт роботи процесора на обробку надходить одразу 8 байт даних.

Командна шина. По цій шині з оперативної пам'яті надходять команди, які виконуються процесором. Команди подані у вигляді байтів. Прості команди вкладаються в один байт, але є такі команди, які займають два, три і більше байтів. Більшість сучасних процесорів мають 32-розрядну командну шину, хоча існують й процесори з 64-розрядною командною шиною.

Шини на материнській платі використовуються не лише для зв'язку з процесором. Решта внутрішніх пристроїв материнської плати, а також пристрої, що підключаються до неї ззовні, також взаємодіють між собою за допомогою шин. Від архітектури цих елементів значно залежить продуктивність ПК загалом.

Розглянемо коротко *основні шинні інтерфейси* (interfaces) – засоби сполучення і узгодження материнських плат з іншими пристроями:

PCI (Peripherial Component Interconnect) – стандарт підключення зовнішніх пристроїв, що був введений в ПК на базі процесора Pentium. За своєю суттю, це також інтерфейс локальної шини з роз'ємами для під'єднання зовнішніх пристроїв. Цей інтерфейс підтримує частоту шини до 66 МГц і забезпечує швидкодію до 264 Мбайт/с незалежно від кількості під'єднаних пристроїв. Важливим нововведенням цього стандарту була підтримка механізму plug-and-play, суть якого полягає в тому, що після фізичного підключення зовнішнього пристрою до роз'єму шини PCI відбувається автоматичне конфігурування, налаштування цього пристрою.

FSB (Front Side Bus) – спеціальна шина, що використовується, починаючи з процесора Pentium Pro, для організації зв'язку з оперативною пам'яттю. Ця шина працює на частотах 100–133 МГц і має пропускну здатність до 800 Мбайт/с. Частота шини FSB є її основним параметром, саме вона зазначається в специфікації материнської плати. За шиною PCI залишається лише функція підключення нових зовнішніх пристроїв.

AGP (Advanced Graphic Port) – спеціальний шинний інтерфейс для підключення відеоадаптерів. Цей інтерфейс був розроблений через те, що параметри шини PCI не відповідали вимогам відеоадаптерів до швидкодії. Частота цієї шини – 33 або 66 МГц, пропускна здатність до 1066 Мбайт/с.

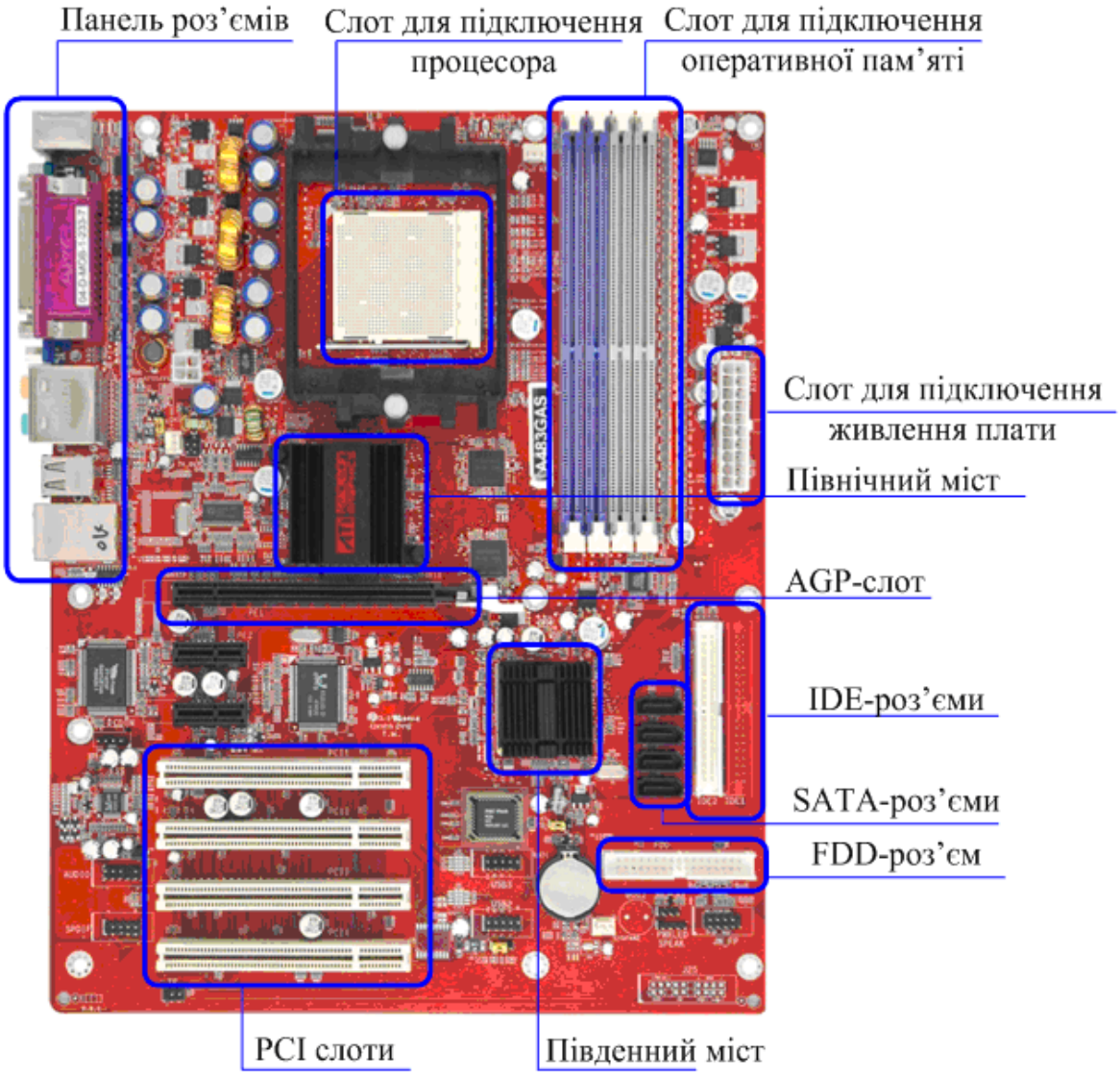


Рисунок 1.3 – Материнська плата персонального комп'ютера

USB (Universal Serial Bus) – стандарт універсальної послідовної шини, що визначає принципово новий спосіб взаємодії комп'ютера з периферійним обладнанням. Він дозволяє підключати до 256-ти різних пристроїв з послідовним інтерфейсом, причому пристрої можуть під'єднуватися ланцюжком (послідовно один за одним). Продуктивність шини USB відносно невелика та становить 1,55 Мбіт/с. Серед переваг цього стандарту слід відзначити можливість підключати та відключати пристрої в «гарячому режимі» (тобто без перезавантаження комп'ютера), а також можливість об'єднання декількох комп'ютерів в просту мережу без використання спеціального апаратного та програмного забезпечення.

1.1.2 Зовнішня периферія

До комп'ютера можуть під'єднуватися багато інших пристроїв з метою вирішення широкого спектра задач (аудіосистеми, приводи компакт-дисків, вебкамери, мікрофони, TV-тюнери, мережеві карти, принтери, сканери і т. д.). Ці пристрої не є обов'язковими елементами.

Монітор (monitor) – пристрій візуального подання даних. Це не єдиний, але головний пристрій виведення інформації. Його основними параметрами є: розмір і крок пікселів екрана, максимальна частота регенерації зображення, клас захисту тощо.

Монітор відображає сформоване зображення на екрані. За принципом дії їх поділяють на такі типи:

- на основі електронно-променевої трубки;
- на основі рідких кристалів.

Розмір монітора вимірюється між протилежними кутами екрана по діагоналі. Одиниця виміру – дюйми. Стандартні розміри: 14"; 15"; 17"; 19"; 20"; 21" тощо.

Частота регенерації (відновлення) зображення показує, скільки разів протягом секунди монітор може повністю змінити зображення екрана (тому її також називають частотою кадрів). Цей параметр залежить не лише від монітора, але і від властивостей та налаштувань відеоадаптера, хоча граничні можливості визначає монітор.

Увага!

1. Перед підключенням пристроїв до системного блока необхідно переконатися, що всі пристрої, включно із системним блоком, вимкнені з мережі електропостачання.
2. Пристрої, що знаходяться зовні системного блока, підключаються до нього за допомогою кабелів у відповідні «рідні» роз'єми на задній стороні системного блока. Підключення пристрою до «чужого» роз'єму може призвести до виходу з ладу пристрою або комп'ютера.

1.2 Хід роботи

1. З'ясувати функціональне призначення материнської плати, процесора, пам'яті, системи живлення та системи охолодження персонального комп'ютера, користуючись теоретичними відомостями.
2. Визначити основні технічні характеристики та параметри материнських плат, процесорів, пам'яті, систем живлення та систем охолодження ПК, а також дослідити їх маркування, яке використовується для подальшої класифікації.
3. Ознайомитися з конструктивними елементами материнської плати та з'ясувати їх функціональне призначення.
4. Ознайомитись з технічними характеристиками персонального комп'ютера.

5. За допомогою викладача або лаборанта встановити процесор (з системою охолодження) та ОЗП на системну плату.

6. За допомогою викладача або лаборанта встановити системну плату в корпус комп'ютера, під'єднати блок живлення.

7. Ознайомитися з системою живлення та підключити її до ПК.

8. Увімкнути комп'ютер та налаштувати його систему належним чином (за необхідності).

9. Вимкнути комп'ютер.

1.3 Контрольні питання.

1. З яких елементів складається ПК?
2. Що входить до складу системного блока?
3. Яке призначення центрального процесорного пристрою?
4. Для чого необхідна оперативна пам'ять?
5. Які роз'єми для зовнішніх периферійних пристроїв Ви знаєте?
6. Які роз'єми містяться на материнській платі персонального комп'ютера?
7. Назвіть типи ЦПП.
8. Які існують види внутрішньої пам'яті та які їхні характеристики?
9. Що таке відеосистема, її призначення та види?
10. Для чого служить порт IDE? Яким чином виконується підключення пристроїв до цього порту?

ЛАБОРАТОРНА РОБОТА № 2
РОБОТА З ОПЕРАЦІЙНОЮ СИСТЕМОЮ MS WINDOWS

Мета: вивчити особливості роботи в графічній операційній системі Windows. Ознайомитись з основними поняттями та термінологією, що використовується в цій операційній системі, одержати навички роботи в файловій системі за допомогою штатних засобів операційної системи.

2.1 Теоретичні відомості

2.1.1 Основні об'єкти та прийоми керування Windows

Сучасний Windows, як і будь-яка операційна система, управляє роботою персонального комп'ютера. Windows має зручний графічний інтерфейс користувача. На відміну від застарілої операційної системи DOS з текстовим інтерфейсом, Windows не потребує знання команд операційної системи та точного введення їх з клавіатури. Переважна більшість операцій керування роботою персонального комп'ютера виконуються маніпулятором «миша» над графічними об'єктами Windows або короткими комбінаціями клавіш («гарячими клавішами») на клавіатурі.

Інтерфейс користувача (user interface) – це сукупність методів та засобів взаємодії людини з апаратним та програмним забезпеченням комп'ютера.

Стартовий екран Windows являє собою системний об'єкт, що називається робочим столом.

Робочий стіл (desk) – це графічне середовище, на якому відображаються об'єкти та елементи керування Windows. На робочому столі можна бачити значки (піктограми), ярлики та панель завдань (основний елемент керування). Після встановлення Windows на робочому столі присутні щонайменше три значки: «Мій комп'ютер» («Мой компьютер»), «Мережеве оточення» («Сетевое окружение»), «Кошик» («Корзина»). На робочому столі можуть бути розташовані й інші значки. Його можна використати як тимчасове сховище файлів, але після завершення роботи в навчальній аудиторії вони мають бути або вилучені, або переміщені у власні папки.

Значки – це графічні зображення об'єктів, які дозволяють керувати ними. Іншими словами, це графічне зображення об'єкта в згорнутому вигляді, що відповідає папці, програмі, документу, мережевому пристрою або комп'ютеру. Значки, як правило, мають мітки – написи, які розташовуються під ними. За замовчуванням один клік лівою кнопкою миші по значку дозволяє виділити його, а подвійний – відкрити (запустити) програму, яка відповідає за роботу з даним файлом.

Ярлик (label) – вказівник на об'єкт. Ярлик – це спеціальний *файл* (file), у якому міститься посилання на відповідний об'єкт (інформація про місце розташування об'єкта на жорсткому диску). Подвійний клік миші по

ярлику дозволяє запустити (відкрити) об'єкт, на який він посилається. При видаленні ярлика сам об'єкт не стирається, на відміну від значка. Переваги ярликів в тому, що вони забезпечують швидкий доступ до об'єктів з будь-якої папки, не витрачаючи на це часу та пам'яті. Відрізнити ярлик від значка можна по стрілці в лівому нижньому куті піктограми.

Панель завдань (taskbar) є інструментом для переходу між відкритими папками або програмами. В лівій частині панелі завдань розташована кнопка «Пуск», в правій – панель індикації. На самій панелі зображаються всі відкриті на цей момент об'єкти. Кнопка «Пуск» відкриває «Головне меню». З його допомогою можна запустити всі програми, які зареєстровані в операційній системі, одержати доступ до всіх засобів налаштування операційної системи, до пошукової та довідкової систем тощо.

Центральним поняттям Windows є вікно.

Вікно (window) – структурний і керівний елемент інтерфейсу користувача, що являє собою обмежену рамкою прямокутну область екрана, у якій може відображатися програма, документ або повідомлення.

На рис. 2.1 показано робочий стіл Windows з відкритим «Головним меню», вікном текстового редактора Word, значками та ярликами, а також згорнутою на панелі завдань програмою Total Commander.

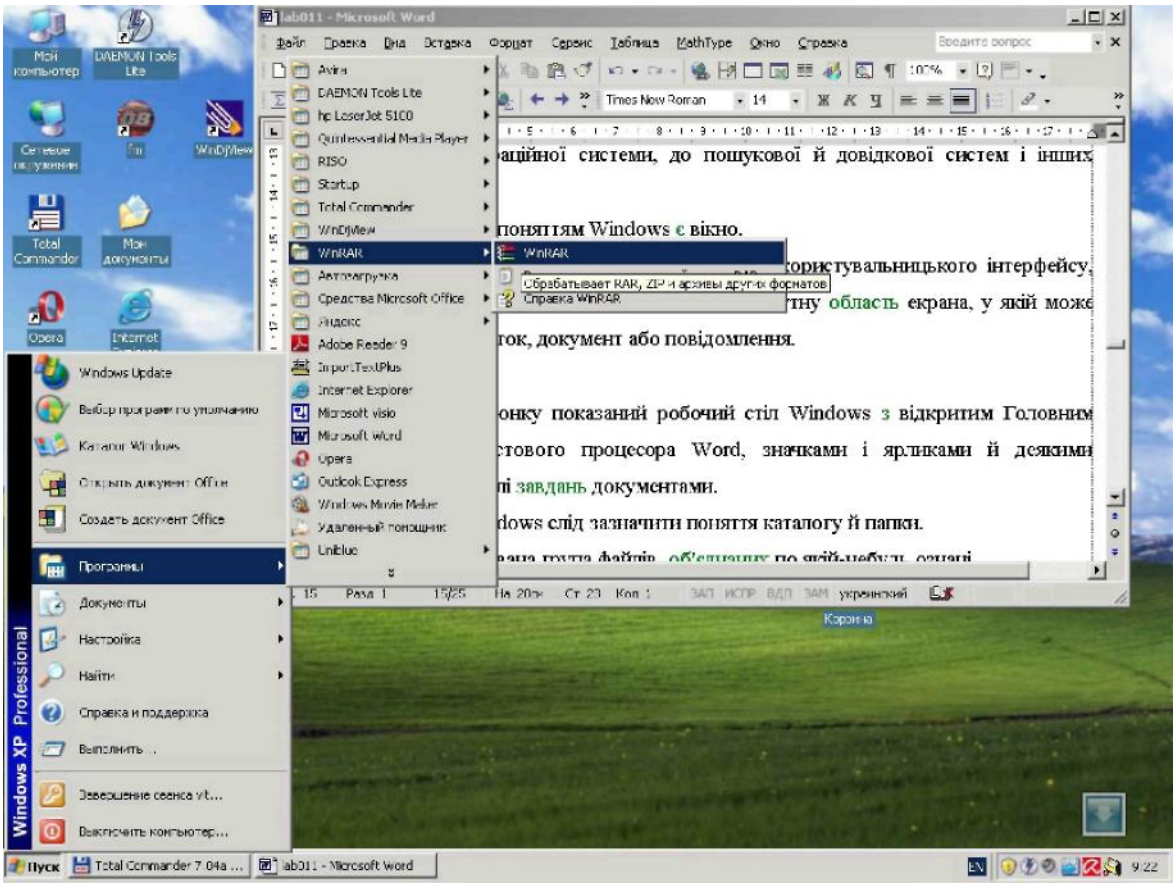


Рисунок 2.1 – Робочий стіл Windows

З інших важливих термінів Windows слід відмітити поняття каталогу та папки. *Каталог* (directory) – група файлів, об'єднаних за деякою ознакою.

Папка (тека, folder) – поняття, що використовується в Windows замість поняття каталог у більш ранніх операційних системах. Поняття папка має розширене тлумачення, оскільки поряд зі звичайними каталогами до папок належать такі об'єкти як «Мій комп'ютер», «Провідник», «Принтер», «Модем» та ін.

2.1.2 Структура вікна папки

Типове вікно папки показано на рис. 2.2.

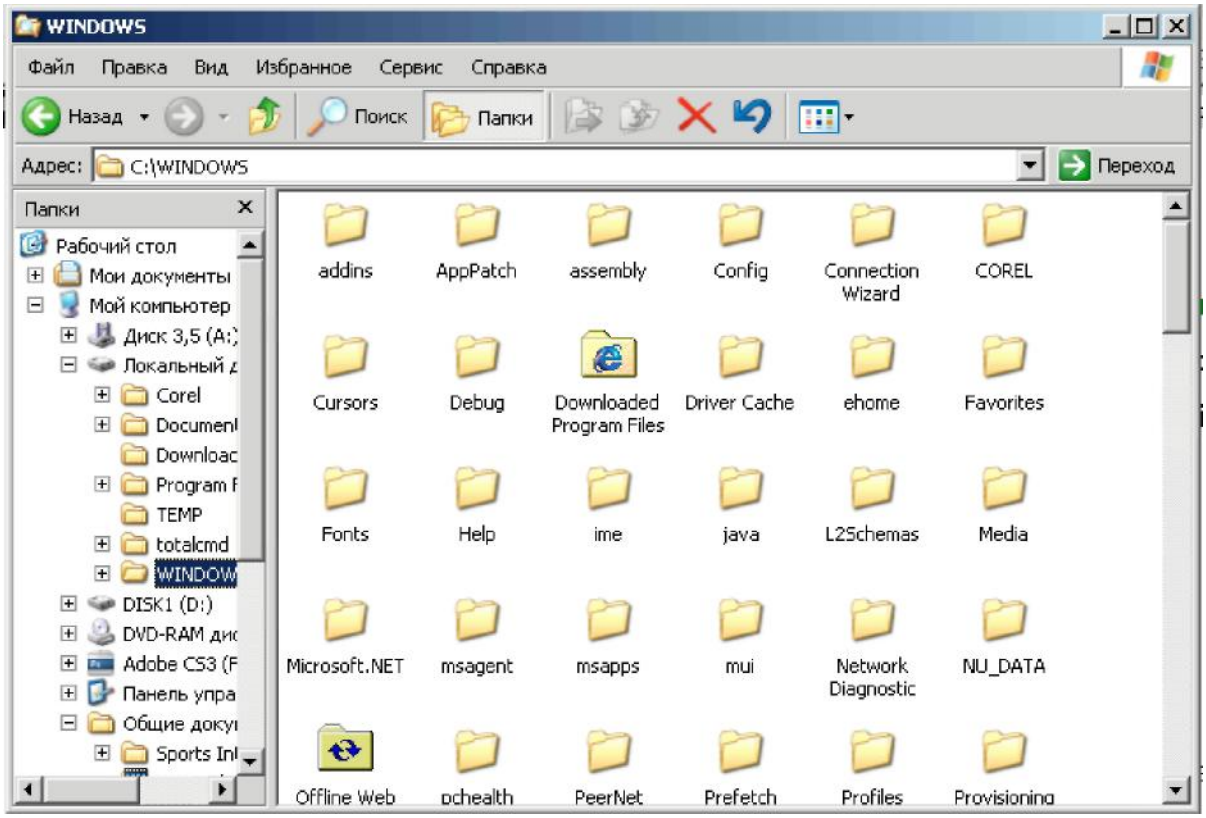


Рисунок 2.2 – Папка «Explorer»

Вікно містить такі обов'язкові елементи.

- Рядок заголовка – у ньому відображається назва папки. Використовується для «перетаскування» вікна по поверхні робочого стола.
- Системний значок. Відкриває службове меню, за допомогою якого можна управляти розміром і розташуванням вікна.
- Кнопки керування розміром: розгортальна, згортальна, закривальна.
- Рядок меню (низхідне меню). Гарантовано надає доступ до всіх команд цього вікна.
- Панель інструментів. Містить командні кнопки для виконання найбільш використовуваних операцій. Користувач може сам налаштувати цю панель, розміщуючи на ній необхідні кнопки.
- Рядок адреси. В ньому зазначається шлях доступу до поточної папки.

Дозволяє швидко перейти до інших розділів файлової структури.

- Робоча область. Відображає значки об'єктів, що зберігаються в папці, спосіб відображення користувач може налаштовувати самостійно.
- Смуги прокручування – дозволяють переглядати вміст вікна в горизонтальному або вертикальному напрямках, якщо інформація не вміщається у вікні.
- Рядок стану. Виводить додаткову інформацію про об'єкти у вікні.

2.1.3 Файлова система персонального комп'ютера

Файлова система (file system) забезпечує зберігання та доступ до файлів на диску. Принцип організації файлової системи – табличний. Поверхня диска розглядається як тривимірна матриця, розмірами якої є номери поверхні, циліндра та сектора. Під циліндром розуміють сукупність всіх рівновіддалених від осі обертання доріжок, що належать різним поверхням. Дані про те, у якому місці записаний той чи інший файл, зберігаються в системній області диска в спеціальній таблиці розміщення файлів (FAT-таблиця). FAT-таблиця зберігається в двох екземплярах, ідентичність яких контролюється операційною системою.

ОС MS-DOS, OS/2, Windows-95/NT реалізують 16-розрядні поля в FAT-таблицях. Така система називалася FAT-16. Вона дозволяє розмістити не більше 65536 записів про місце розташування одиниць зберігання даних. Найменшою одиницею зберігання даних є *сектор* (sector). Розмір сектора дорівнює 512 байтам. Групи секторів умовно поєднують в *кластер* (cluster), який є найменшою одиницею адресації даних. Розмір кластера залежить від ємності диска: в FAT-16 для дисків від 1 до 2 Гбайт 1 кластер займає 64 сектори або 32 Кбайта. Це нераціонально, оскільки навіть маленький файл займає 1 кластер. У більших файлів, що займають кілька кластерів, в кінці послідовності утворюється незаповнений кластер. Тому втрати ємності для дисків у системі FAT-16 можуть бути дуже великі. З дисками понад 2,1 Гбайт FAT-16 взагалі не працює.

В Windows 98 і пізніших версіях реалізована нова файлова система – FAT-32 з 32-розрядними полями в таблиці розміщення файлів. Вона забезпечує маленький розмір кластера для дисків великої ємності. Наприклад, для диска до 8 Гбайт 1 кластер займає 8 секторів (4 Кбайти).

Файл – це іменована послідовність байтів довільної довжини. До появи Windows 95 загальноприйнятою схемою найменування файлів була схема 8.3 (коротке ім'я) – 8 символів для імені файла, 3 символи – для розширення його імені. Недолік коротких імен – їхня низька змістовність. Починаючи з Windows 95 введено поняття довгого імені (до 256 символів). Воно може містити будь-які символи, крім дев'яти спеціальних: «\», «/», «:», «*», «?», «"», «<», «>» та «|».

Розширенням імені вважаються всі символи після останньої крапки в імені файла. У сучасних операційних системах розширення імені містить важливу інформацію про тип файла. Типи файлів реєструються та

пов’язують файл з програмою, що його відкриває. Наприклад файл «MyText.doc» буде відкритий текстовим редактором Word, оскільки розширення «.doc» зазвичай пов’язують саме із цією програмою. Якщо файл не пов’язаний з жодною з програм, то на його значку зображений прапорець – логотип MS Windows, відповідальну програму в цьому випадку користувач може вказати сам, вибравши її з наданого системою списку.

Логічно структура файлів організована за ієрархічним принципом: папки більш низьких рівнів вкладаються в папки більш високих рівнів. Верхнім рівнем вкладеності є кореневий каталог логічного диска.

Атрибути файлів (attribute) – це параметри, що визначають деякі властивості файлів. Для одержання доступу до атрибутів файла необхідно клацнути правою кнопкою миші по його значку та вибрати меню «Властивості» («Свойства»). Основних атрибутів 4: «Лише для читання» («Только для чтения»), «Прихований» («Скрытый»), «Системний» («Системный»), «Архівний» («Архивный»). Атрибут «Лише для читання» означає, що файл не призначений для внесення змін. Атрибут «Прихований» означає те, що даний файл не відображатиметься на екрані при проведенні операцій з файлами. Атрибутом «Системний» позначаються найбільш важливі файли ОС (як правило, вони також мають атрибут «Прихований»). Атрибут «Архівний» пов’язаний з резервним копіюванням файлів.

2.1.4 Операції з файлами та папками

Копіювання й переміщення.

1-й спосіб. Розмістити на робочому столі два вікна: джерело та приймач копіювання. Виділити у вікні-джерелі необхідні значки. Кілька значків виділяються при натиснутій клавіші «Ctrl». Перетягнути мишею виділені значки у вікно-приймач. При одночасно натиснутій клавіші «Ctrl» відбувається копіювання, без неї – переміщення елементів (за умови, що папки перебувають на одному диску).

2-й спосіб. Виділити елементи, які необхідно скопіювати. Вибрати пункт контекстного меню «Копіювати» («Копировать») або «Вирізати» («Вырезать»). При виборі «Вирізати» відбудеться переміщення. Відкрити папку-приймач. Вибрати пункт меню «Вставити» («Вставить»).

Видалення файлів і папок.

Видалення файлів виконується шляхом вибору елементів і натисканням клавіші «Delete». При цьому відмічені елементи переміщуються в спеціальну папку – «Кошик». При очищенні «кошика» відбувається знищення файлів. Існує ще більш надійний спосіб видалення даних, при якому операція стирання файлів проводиться спеціальними службовими програмами, в результаті чого ті кластери, в яких містилися файли, заповнюються випадковими даними.

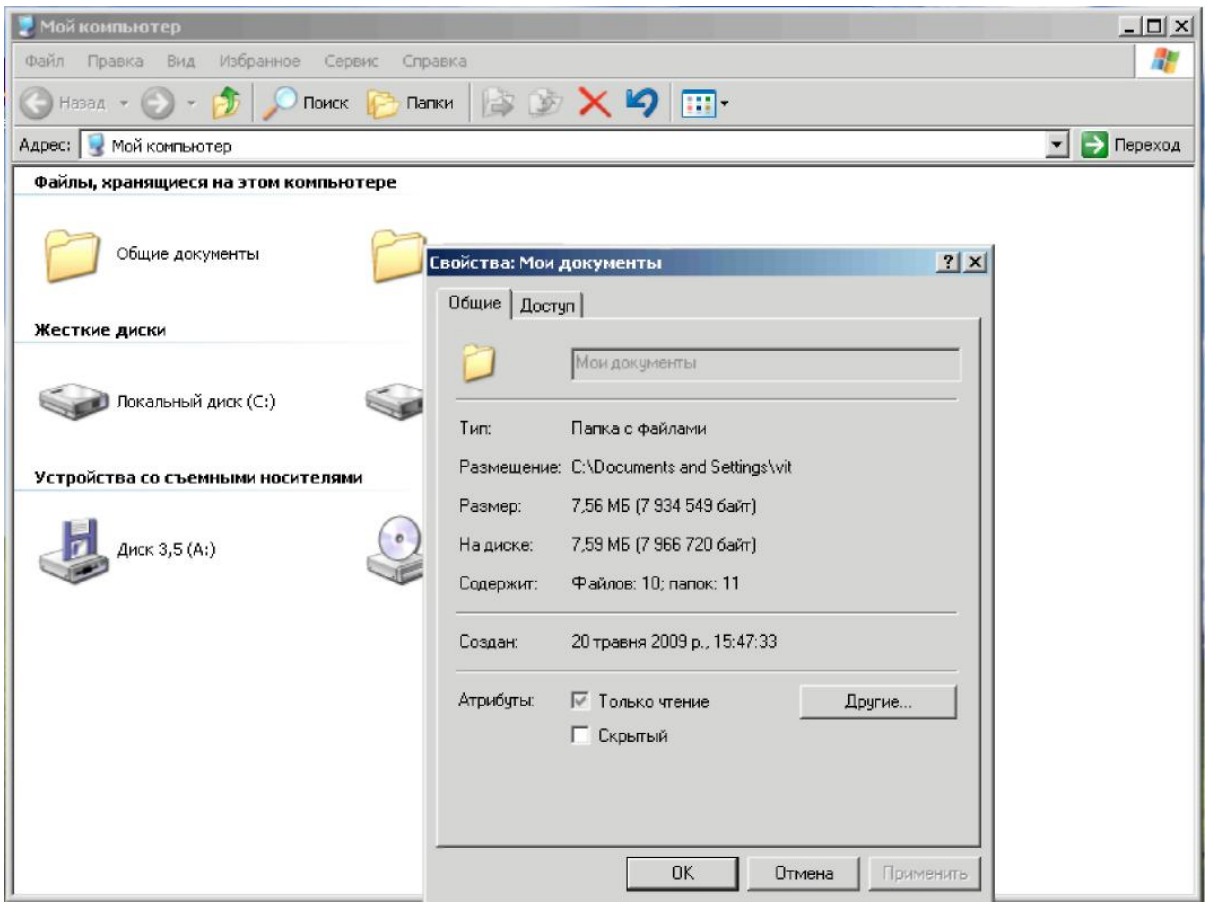


Рисунок 2.3 – Властивості папки

Групові операції з файлами.

Якщо потрібно виконати операцію копіювання або видалення з більшою кількістю файлів одночасно, то виділяти їх утримуючи «Ctrl» не дуже зручно. Можна виділити цілу групу значків, клацнувши по першому з послідовності і при натиснутій клавіші «Shift» – по останньому. Однак, у цьому випадку потрібно певним чином впорядкувати значки. Для цього необхідно відкрити папку з файлами та звернутися до меню «Сортування» або «Групування», натиснувши праву кнопку миші на області вікна. Існує 4 способи впорядкування значків у папці: за іменем, за типом, за розміром, за датою. Наприклад, необхідно скопіювати всі файли з розширенням «.txt». У цьому випадку варто впорядкувати значки за типом, після чого всі файли типу «.txt» будуть згруповані разом, далі потрібно використати клавішу «Shift» для їхнього виділення. Аналогічний прийом застосовується для виділення «старих» файлів (впорядкування за датою), «маленьких» (впорядкування за розміром) тощо.

Якщо у вікні не відображається повна інформація про файли (розширення, обсяг і дата створення), необхідно звернутися до меню вікна папки «Вид–Таблиця» («Вид–Таблица»).

Перейменування файлів і папок.

Перейменування файла або папки виконується або через контекстне меню «Перейменувати» («Переименовать»), яке викликається кліком

правої кнопки миші на відповідному значку, або кліком на імені попередньо виділеного значка.

Увага! Видалення та перейменування неможливі, якщо зазначений файл в цей момент відкритий якою-небудь програмою.

Пошук файлів за маскою.

Для виконання пошуку необхідно обрати диск (чи папку) за допомогою спеціальної опції – «Область пошуку» або зайти у відповідну папку і запустити пошук у ній. Після вибору області пошуку необхідно вказати точне ім'я файла або його маску. Також можна поєднувати ім'я файла та маску.

Маски формуються за допомогою спеціальних символів «*» та «?». Зірочка «*» означає будь-яку кількість пропущених символів, знак питання «?» – що пропущено рівно один символ.

Приклад масок: «*.*» – пошук всіх файлів; «*.exe» – пошук всіх файлів з розширенням «.exe»; «dialer.exe» – пошук файлів з конкретним ім'ям «dialer.exe»; «dialer.exe, *.dll, trojan*.sys» – пошук файлів з ім'ям «dialer.exe» та файлів з будь-яким ім'ям і розширенням «dll» та файлів з ім'ям «trojan<будь-які символи>.sys»; «*.exe, *.dll, *.sys, *.ocx» – пошук файлів з вказаними розширеннями; «bmi-1?» - пошук файлів з ім'ям, що складається з шести символів, при чому п'ять символів чітко задані, а шостий символ – будь-який.

2.1.5 Робота з буфером обміну

Операційна система Windows створює та обслуговує спеціальну область пам'яті, яка називається буфер обміну. Буфер обміну служить для обміну даними між програмами Windows. Описаний вище другий спосіб копіювання припускає використання саме буфера обміну. Для операцій, що пов'язані з буфером обміну, використовуються пункти «Копіювати», «Вирізати» та «Вставити». Перші два поміщають в буфер обміну об'єкт, останній – копіює його з буфера обміну. Якщо об'єкт (частина тексту, рисунок тощо) не виділений, то перші два пункти будуть неактивні. Якщо буфер обміну порожній, то не буде активний і третій пункт.

Операції з буфером обміну виконуються дуже часто, тому на панелі інструментів вікна містяться кнопки швидкого доступу до нього.

Найшвидший спосіб роботи з буфером обміну – використання комбінацій клавіш: «Ctrl+C» – копіювати; «Ctrl+X» – вирізати; «Ctrl+V» – вставити.

2.2 Хід роботи

1. За допомогою ОС Windows створити файли «Лабораторна робота № 3.doc», «Лабораторна робота № 3.xls» на робочому столі та в папці: «1РА_14→Прізвище→Лабораторна робота № 2».
2. Перемістити архіви в кореневу папку диска С.

3. За допомогою ОС знайти створені файли, за маскою, за ключовими словами та за першими літерами імен файлів.

4. Змінити атрибут файла «Лабораторна робота № 3.doc» на атрибут «Лише для читання».

5. За допомогою буферу обміну скопіювати дані з «Лабораторна робота № 3.xls» в «Лабораторна робота № 3.doc».

2.2.1 Студент повинен:

вивчити основні елементи робочого стола Windows, способи навігації по файловій системі комп'ютера, способи запуску програм, операції управління вікнами згідно з метою лабораторної роботи. Знати визначення понять: файл, папка, значок, ярлик, буфер обміну. Вивчити властивості зазначених об'єктів і методи керування ними: створення, копіювання, переміщення, перейменування, видалення файлів і папок.

2.2.2 Приклад виконання лабораторної роботи

1. Створюємо папку (рис. 2.4).

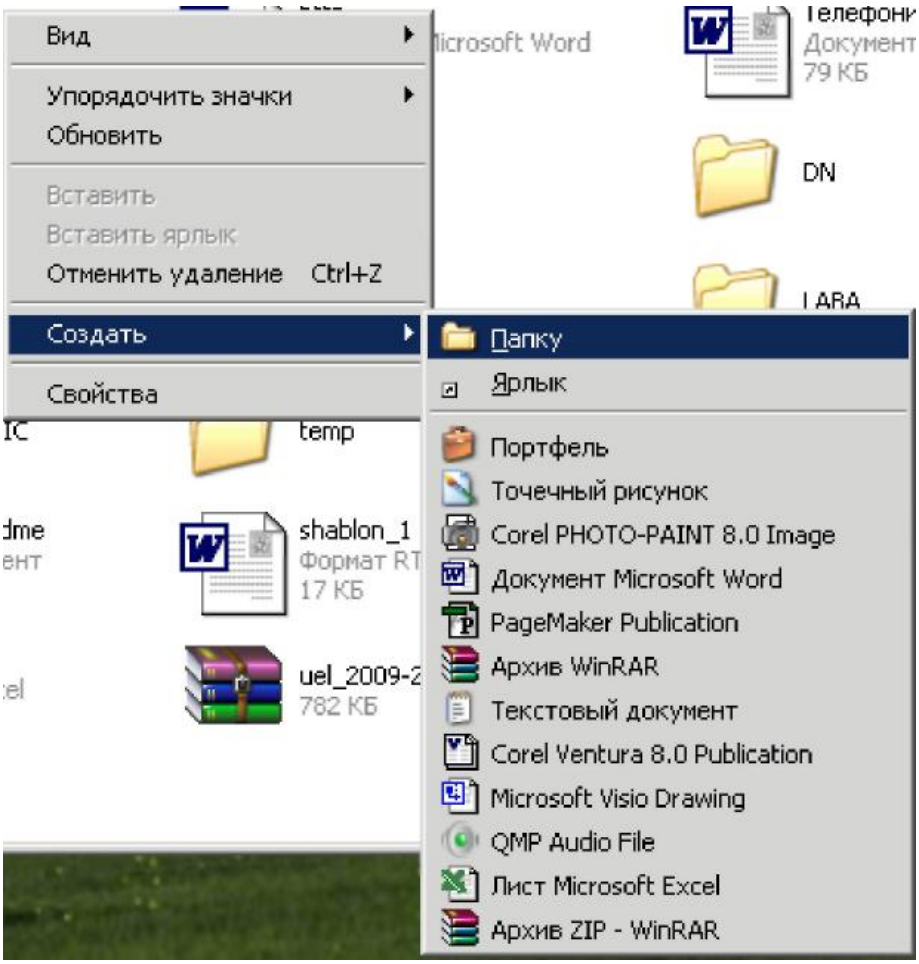


Рисунок 2.4 – Контекстне меню

2. Створюємо файл (рис. 2.5).

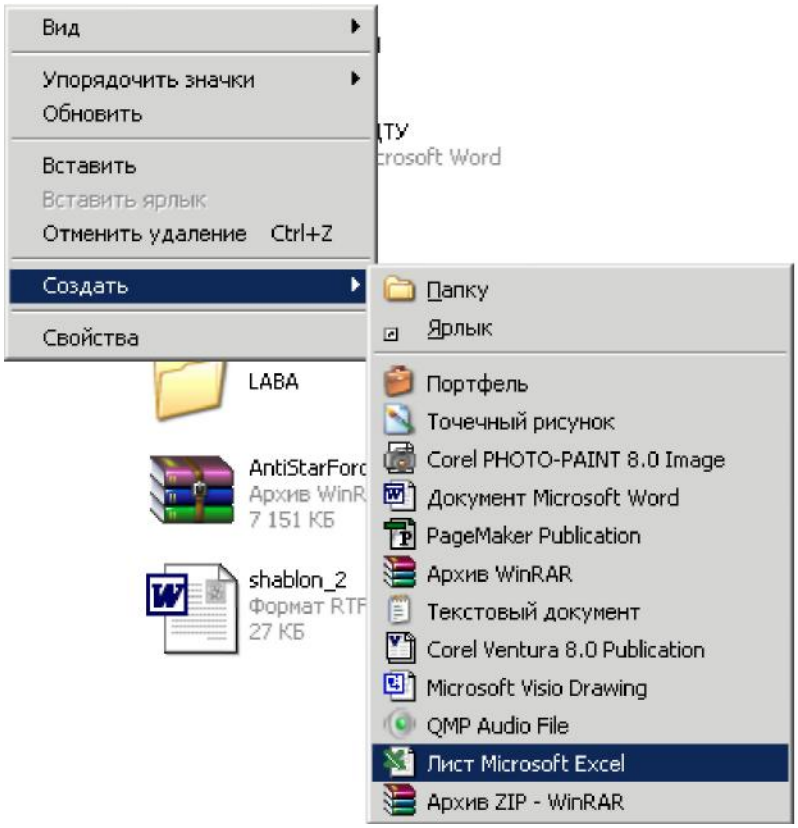


Рисунок 2.5 – Створення файла за допомогою контекстного меню

3. Виставляємо атрибути «Прихований» («Скрытый») (рис. 2.6).

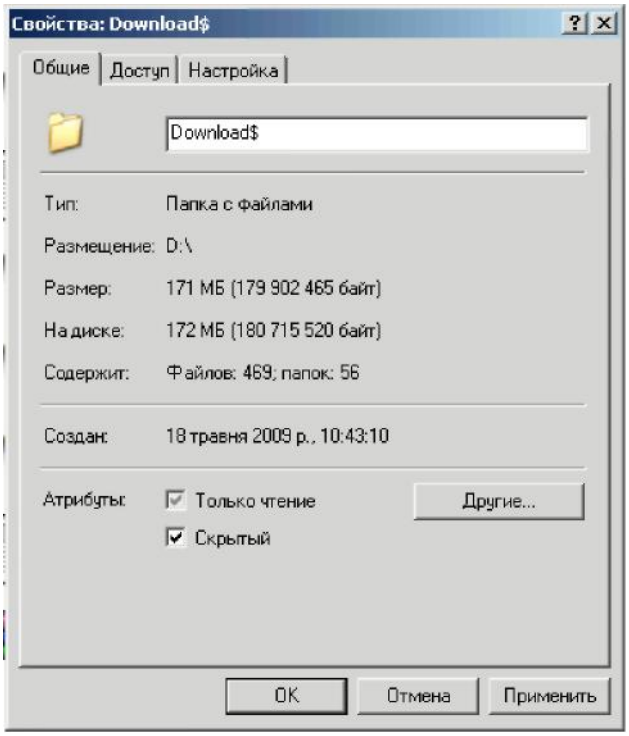


Рисунок 2.6 – Властивості файла

4. Проводимо пошук створених файлів (рис. 2.7).

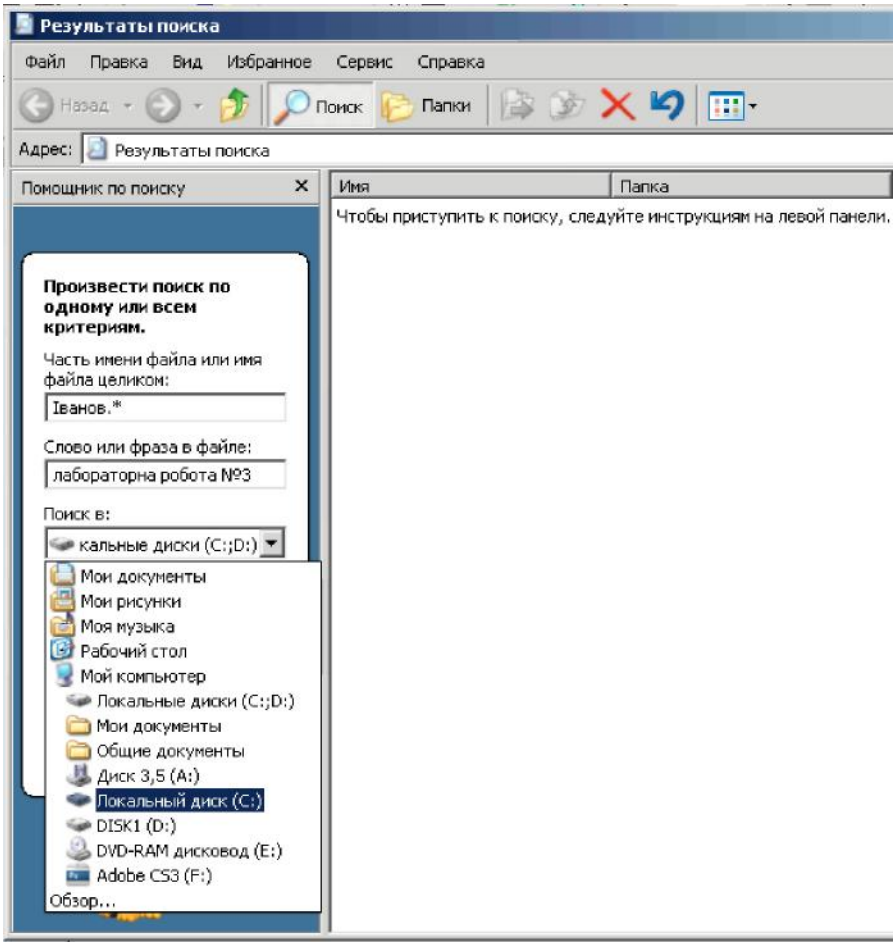


Рисунок 2.7 – Пошук файлів

2.3 Контрольні питання

1. Що таке інтерфейс користувача Windows?
2. Перерахуйте основні елементи інтерфейсу користувача. Яке їхнє призначення?
3. Що таке об’єкт Windows «Робочий стіл»?
4. Що таке панель завдань? Для чого вона призначена?
5. Що таке значок і яке його призначення?
6. Що таке ярлик, яке його призначення? У чому його відмінність від значка?
7. Які операції з відповідним об’єктом дозволяють виконувати значок і ярлик? Як вони виконуються?
8. Що таке об’єкт «Мій комп’ютер»? Які його можливості?
9. Яке призначення кнопки «Пуск»?
10. Як отримати доступ до «Головного меню» Windows? Які можливості надає «Головне меню»?
11. Яка структура вікна папки? Яке призначення елементів цього вікна?
12. Яке призначення файлової системи?

13. Який принцип організації файлової системи?

14. Що таке таблиця розміщення файлів (FAT)? Для чого вона призначена?

15. Охарактеризуйте способи реалізації FAT.

16. Що таке сектор?

17. Що таке кластер? Від чого залежить його розмір? Порівняйте 16-розрядну й 32-розрядну FAT.

18. Що називається файлом?

19. Як записується коротке ім’я файла? Який недолік такого запису?

20. За якими правилами записується довге ім’я файла?

21. Що таке розширення імені файла? Як воно використовується системою?

22. Опишіть логічну структуру файла.

23. Які властивості задають атрибути файлів? Як одержати до них доступ?

24. Наведіть основні операції з файлами та папками, опишіть способи їхньої реалізації.

25. Що називається буфером обміну? Для чого він використовується?

26. Як одержати доступ до буфера обміну? Які команди меню призначені для роботи з буфером обміну? Опишіть їх.

27. Який найшвидший спосіб роботи з буфером обміну?

ЛАБОРАТОРНА РОБОТА № 3
РОБОТА З ТЕКСТОВИМ РЕДАКТОРОМ, ЕЛЕКТРОННИМИ ТАБЛИЦЯМИ. СТВОРЕННЯ МУЛЬТИМЕДІЙНОЇ ПРЕЗЕНТАЦІЇ

Мета: отримати знання та практичні навички роботи з електронними таблицями та навчитися готувати матеріал для презентації за допомогою засобів пакета програм MS Office.

3.1 Теоретичні відомості

Електронні таблиці (ET, spreadsheets) – це програми, призначені для автоматизованої обробки даних, які подано у вигляді таблиць. В цьому випадку обробка охоплює:

- проведення різноманітних обчислень з використанням потужного апарату функцій і формул;
- дослідження впливу різних факторів на набір даних;
- отримання вибірок даних, що задовольняють певні критерії;
- побудова графіків і діаграм;
- статистичний аналіз даних.

3.1.1 Структура документа Excel

Одним з найпопулярніших засобів управління електронними таблицями є програма Microsoft Excel.

Кожний документ, що створюється у середовищі Excel, є набором таблиць – *робочою книгою* (working book), яка складається з одного або кількох робочих аркушів. Кожний робочий аркуш має власну назву та є окремою електронною таблицею.

Стовпці позначаються латинськими літерами: A, B, C, ... Якщо літер не вистачає, використовують дволітерні позначення AA, AB і т. д. Максимальна кількість стовпців в таблиці – 256. Рядки нумеруються цілими числами. Максимальна кількість рядків, яку може мати таблиця, – 65536.

Комірки розміщуються на перетині стовпців і рядків. Адреса комірки формується як об'єднання номерів стовпця та рядка без пропуску між ними. Отже, «A1», «CZ31» і «HP65000» – допустимі адреси комірок.

Одна з комірок на робочому листі завжди є активною. В більшості операцій використовується саме вона. Активна комірка обведена широкою рамкою, а її адреса (і вміст) відображаються в рядку формул над аркушем. Крім того, виділяються заголовок стовпця та номер рядка, на перетині яких знаходиться активна комірка.

3.1.2 Робота з робочими аркушами

За замовчуванням книга в Excel складається з трьох аркушів з назвами «Лист 1», «Лист 2», «Лист 3». Кожен аркуш має свій ярлик в нижньому

лівому кутку робочої області. Щоб перейти з одного листа на інший, потрібно клацнути кнопкою миші на відповідному ярлику.

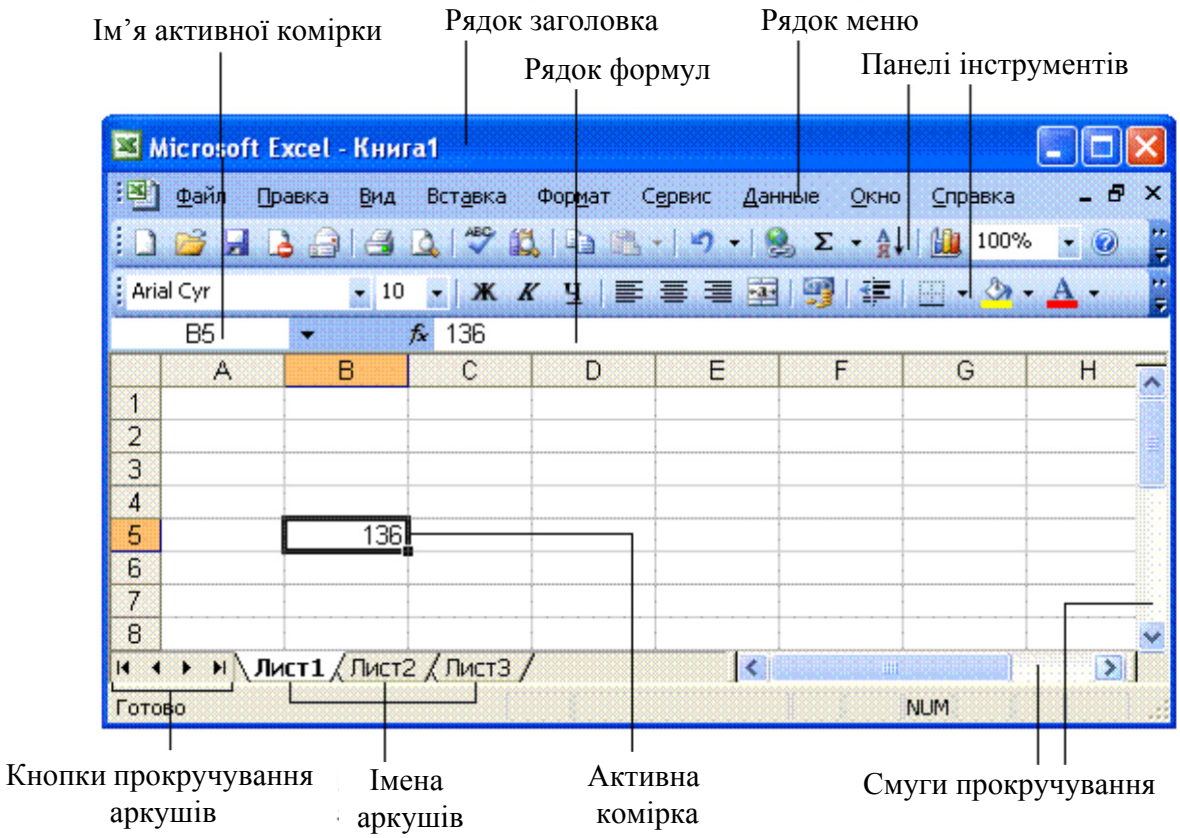


Рисунок 3.1 – Вікно MS Excel

Аркуш можна в будь-який момент перейменувати – для цього слід двічі клацнути на його ярлику, ввести нове ім’я та натиснути клавішу «Enter». Крім того, для перейменування поточного аркуша можна виконати команду «Формат – Лист – Переименовать», ввести нове ім’я та натиснути клавішу «Enter».

Щоб створити додаткові аркуші, слід вибрати робочий аркуш, перед яким повинен бути створений новий, і задати команду «Вставка – Лист». Видалити поточний аркуш можна командою «Правка – Удалить лист».

Змінити порядок послідовності робочих аркушів можна перетягуванням їхніх ярликів за допомогою миші. Якщо під час перетягування утримувати натиснутою клавішу «Ctrl», в книзі з’явиться копія аркуша.

Іноді (як правило, при роботі з великими електронними таблицями) буває зручно надавати ярликам листів різного кольору. Це можна зробити за допомогою команди «Формат – Лист – Цвет ярлыка».

Розглянуті операції з робочими листами можна також виконувати, використовуючи контекстне меню, яке зображено на рисунку 3.2.

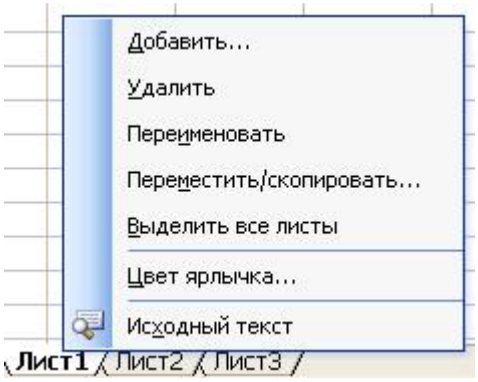


Рисунок 3.2 – Робота з листами

3.1.3 Введення інформації в таблицю. Редагування даних

Дані в програмі Excel завжди вносяться в поточну комірку. Перш ніж розпочати введення, потрібну комірку необхідно виділити. Це можна зробити, скориставшись клавішами управління курсором або клацнувши на потрібній комірці, навівши на неї курсор миші. Використовуються й деякі комбінації клавіш для швидкого переміщення аркушем:

- «Ctrl» + стрілка курсора – до найближчої непорожньої комірки у вибраному напрямку;
- «Enter» – на одну комірку вниз;
- «Tab» – на одну комірку вправо;
- «Shift» + «Enter» – на одну комірку вверх;
- «Shift» + «Tab» – на одну комірку вліво;
- «Home» – до стовпця А поточного рядка;
- «Page Up» – на одну екранну сторінку вгору;
- «Page Down» – на одну екранну сторінку вниз;
- «Alt» + «Page Up» – на одну екранну сторінку вліво;
- «Alt» + «Page Down» – на одну екранну сторінку вправо;
- «Ctrl» + «Home» – до комірки А1;
- «Ctrl» + «End» – до комірки останнього рядка і останнього стовпця, в яких містяться дані.

Натиснення клавіш з літерами, цифрами або розділовими знаками автоматично розпочинає введення даних в комірку. Закінчити введення даних можна натисненням клавіші «Enter». При введенні даних в заповнену комірку попередній її вміст втрачається. Якщо потрібно тільки відредагувати вміст комірки, а не вводити його заново, слід двічі клацнути на ній кнопкою миші. Це приведе до появи в комірці текстового курсора, який можна використовувати для редагування. Крім того, можна виділити комірку та натиснути клавішу «F2»; це також дозволить виконувати редагування безпосередньо в комірці або в рядку формул.

Дату в комірку зручно вводити за допомогою додаткової цифрової клавіатури, причому розділювачем груп даних слугує знак „/”. Для введення десяткових дробів використовується знак «.» («,») на додатковій цифровій клавіатурі. На алфавітно-цифровому полі крапка – це роздільник

дати, а кома – десяткового дробу. Для введення поточної дати використовують комбінацію клавіш «Ctrl» + «;», а поточного часу – комбінацію «Ctrl» + «Shift» + «;». Можна поєднати в одній комірці дату та час.

3.1.4 Форматування та зміна розмірів комірок

Правильна організація комірок допомагає презентувати важливу інформацію найкращим чином. Форматування комірки не змінює внутрішнього подання даних, а лише впливає на те, який вигляд вони будуть мати на екрані або в надрукованому документі.

Змінити формат даних в окремій комірці (чи діапазоні комірок) можна за допомогою команди «Форматирование» на панелі інструментів. Її елементи відомі по аналогічній панелі текстового редактора Word. Крім того, група кнопок на цій панелі дозволяє вибрати один із декількох заздалегідь підготовлених форматів для запису чисел. До них належать кнопки «Денежный формат», «Процентный формат», «Формат с разделителями», «Увеличить разрядность», «Уменьшить разрядность».

Щоб довільно відформатувати комірки певного діапазону, його слід виділити і дати команду «Формат – Ячейки». Відкривається діалогове вікно «Формат ячеек», яке містить шість закладок, що дозволяють задати різні параметри форматування.

Закладка «Число» використовується для зміни формату числових значень. Потрібний формат («Общий», «Числовой», «Денежный», «Финансовый», «Дата», «Время», «Процентный», «Дробный», «Экспоненциальный», «Текстовый», «Дополнительный», «Все форматы») вибирається зі списку «Числовые форматы». При цьому Excel відображає приклади форматування в тих випадках, коли здійснюється вибір з декількох варіантів.

Закладка «Выравнивание» дозволяє задати метод вирівнювання та кут нахилу запису, а також встановити переноси слів у комірках.

Закладка «Шрифт» визначає тип шрифту, стиль написання, його розмір та колір.

Закладка «Граница» дозволяє задати стиль рамки на зовнішніх і внутрішніх межах діапазону. Є можливість вибору типу рамки і її кольору.

Закладка «Вид» призначена для вибору кольору фону комірки.

Змінити ширину та висоту комірок можна методом перетягування меж між заголовками стовпців чи номерами рядків. Якщо потрібне більш точне значення ширини чи висоти, то потрібно вибрати команду «Формат – Столбец» (або «Формат – Строка») і вибрати потрібний варіант з меню, яке відкриється при цьому. Зокрема, команда «Автоподбор ширины» («Автоподбор высоты») дозволяє вибрати ширину стовпця (висоту рядка) таким чином, щоб в нього повністю вміщувались дані найбільш заповненої комірки.

Можна виконати комплексне форматування діапазону комірок, скориставшись командою «Формат – Автоформат». Команда відкриває діалогове вікно «Автоформат», де зі списку заздалегідь визначених типів оформлення таблиць слід вибрати найдоцільніший. Користувач не може встановити свої власні формати, але він може внести зміни у вибраний формат, натиснувши кнопку «Параметры».

3.1.5 Збереження та відкриття електронних книг

Для збереження електронної книги потрібно виконати таку послідовність дій:

1. Обрати команду «Файл – Сохранить как».
2. В діалоговому вікні вказати ім’я документа та папку для збереження цього документа.
3. Натиснути кнопку «Сохранить»**.**

Для відкриття електронної книги потрібно виконати таку послідовність дій:

1. Обрати команду «Файл – Открыть».
2. В діалоговому вікні вибрати папку та ім’я потрібного документа.
3. Натиснути кнопку «Открыть»**.**

3.1.6 Використання стандартних функцій

Обчислення, які дозволяє виконувати Excel, не обмежуються найпростішими арифметичними операціями. Програма дозволяє використовувати велику кількість стандартних функцій і здатна виконувати складні обчислення.

Будь-яка функція – це інтегрована формула для виконання операцій над вказаними даними. Вся множина стандартних функцій поділяється на декілька категорій: математичні, статистичні, фінансові, логічні, текстові та інші.

В круглих дужках (одразу за назвою функції) записуються її аргументи. Аргументами можуть бути числа, адреси комірок, адреси діапазонів або їхні списки. Елементи списку записують через символ «;». Наприклад, формула =СУММ(A1;B6:C8;20) додає числа з комірки A1, діапазону B6:C8 і числову константу 20.

Крім того, можуть використовуватись комбінації функції. Іншими словами, вони можуть бути вкладеними одна в одну таким чином, що результат однієї функції буде аргументом для другої (зрозуміло, що в цьому випадку результат першої функції має відповідати аргументу другої функції за типом). Наприклад, формула =СРЗНАЧ(5;КОРЕНЬ(9)) обчислює квадратний корінь з 9 і знаходить середнє арифметичне двох аргументів: числа 5 і знайденого кореня. Результатом є число 4.

При використанні деяких функцій аргумент не вказується, але дужки після назви функції потрібно вводити обов’язково. Прикладами таких

функцій є «ПИ» (повертає значення числа π, округленого до 15 знаків) або «СЕГОДНЯ» (повертає поточну дату).

Excel містить більше 400 вбудованих функцій, тому вводити в формулу назви функцій і значення вхідних параметрів безпосередньо з клавіатури не завжди зручно. В програмі є спеціальний засіб для роботи з функціями – «Мастер функций».

Для роботи з майстром функцій слід скористатися командою «Вставка – Функция», натиснути кнопку «Вставка функции» на панелі інструментів «Стандартные» або в рядку формул. При цьому відкриється діалогове вікно «Мастер функций – шаг 1 из 2», в якому потрібно вибрати категорію функцій. При виборі категорії в полі «Выберите функцию» виводиться список функцій даної категорії. При виборі конкретної функції зі списку виводиться короткий опис цієї функції (рис. 3.3).

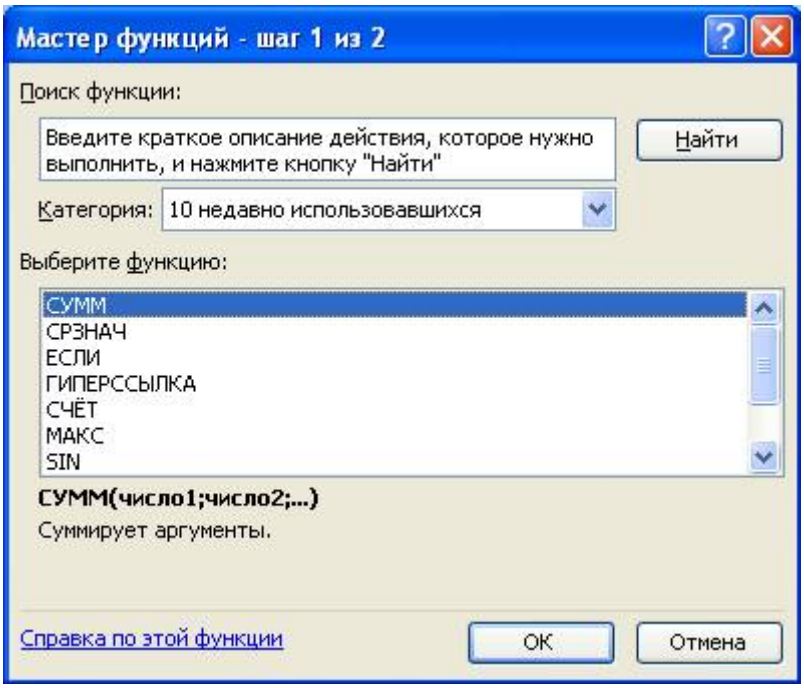


Рисунок 3.3 – Використання майстра функцій

Після вибору функції потрібно натиснути кнопку «ОК», у результаті чого відкриється діалогове вікно «Аргументы функции». У верхній частині вікна розміщуються поля для введення аргументів, а в нижній частині – довідкова інформація. Тут вказується загальний опис та призначення функції, а також відомості про аргументи. Якщо аргумент виділений напівжирним шрифтом, значить, він є обов'язковим, якщо звичайним шрифтом – то його можна пропустити. У полі «Значение» виводиться значення функції при вказаних аргументах (рис. 3.4).

Натиснення кнопки «ОК» завершує введення формули**.** Обчислений результат з'являється в активній комірці.

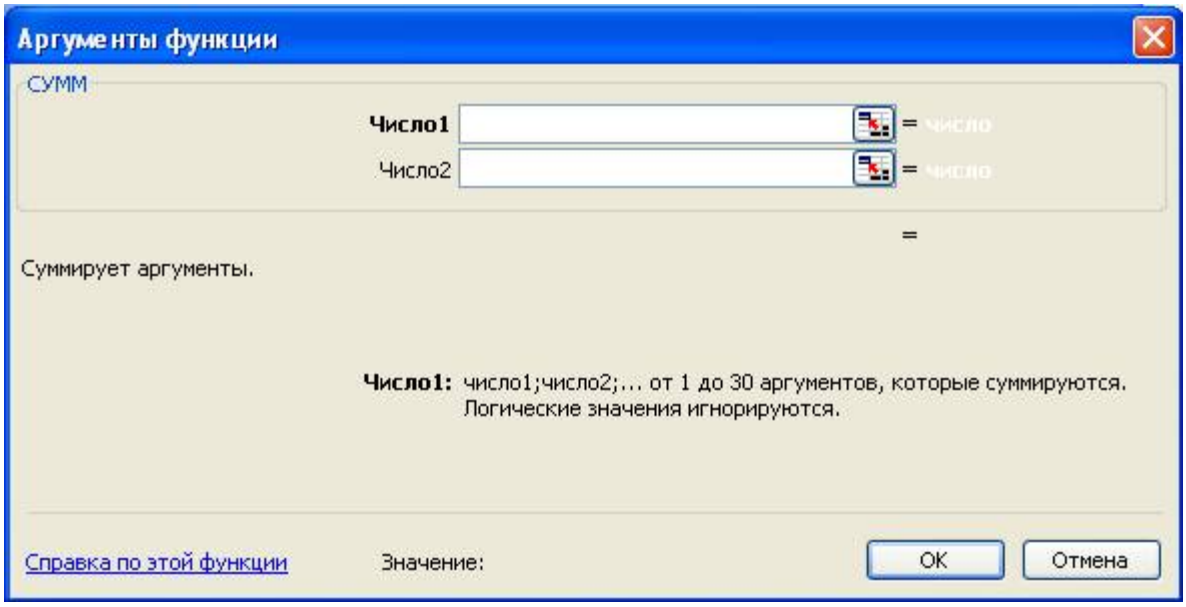


Рисунок 3.4 – Вибір аргументів функції

Оскільки функція «СУММ» використовується дуже часто, на панелі інструментів «Стандартная» знаходиться кнопка «Автосумма», яка прискорює доступ до цієї функції. Для її використання необхідно:

1) виділити комірку, в яку потрібно помістити функцію «СУММ» (при додаванні стовпця чисел слід виділити комірку, яка знаходиться безпосередньо під останнім числом у стовпці, при додаванні чисел рядка – комірку, яка знаходиться праворуч від останнього числа у рядку);

2) натиснути на кнопці «Автосумма». Excel вставляє функцію «СУММ» в рядку формул і (за можливості) автоматично виділяє діапазон сусідніх комірок як аргумент функції;

3) якщо Excel виділить саме той діапазон, суму в якому потрібно обчислити, натискання на клавішу Enter завершить введення функції та приведе до появи обчисленого результату у відповідній комірці. Якщо діапазон було вибрано неправильно, необхідно виділити новий за допомогою миші, після чого натиснути клавішу Enter.

Основні математичні функції:

- СУММ(число1;число2;…) – повертає суму аргументів;
- КОРЕНЬ(число) – повертає значення квадратного кореня;
- SIN(число) – повертає синус числа;
- COS(число) – повертає косинус числа;
- СТЕПЕНЬ(число;степень) – повертає число – результат піднесення до степеня;
- ABS(число) – повертає модуль числа;
- EXP(число) – повертає експоненту числа;
- LN(число) – повертає натуральний логарифм числа;
- LOG(число) – повертає десятковий логарифм числа;
- ОКРУГЛ(число; точність) – повертає округлене значення числа з заданою кількість знаків після коми;

- СРЗНАЧ(число1;число2;…) – повертає середнє арифметичне своїх аргументів;

- МИН(число1;число2;…) – повертає найменше значення зі списку аргументів;

- МАКС(число1;число2;…) – повертає найбільше значення зі списку аргументів.

3.1.7 Створення презентацій в MS PowerPoint

Презентація – це демонстрація інформаційних ресурсів з метою більш ефективного засвоєння матеріалу, що демонструється, за допомогою сучасних мультимедійних систем.

Види навчальних презентацій:

1. Конспект уроку. Обов'язкова наявність основних візуальних складових традиційного уроку: тема, мета, план, ключові поняття, висновки, домашнє завдання тощо. Ілюстративний ряд відіграє в даному випадку допоміжну та незначну роль. Досить часто подібні уроки за своїм змістом (а іноді, й ілюстративною частиною) орієнтовані на базовий підручник. Дану форму презентації, як правило, частіше називають традиційним мультимедійним супроводом уроку (заняття).

2. Слайд-шоу. Повна відсутність тексту та переведення акценту на яскраві, великі зображення або колажі (може демонструватися для створення певного емоційного настрою). Як правило, супроводжується музичними фрагментами.

3. Лише текст. Варіант, протилежний за своєю суттю слайд-шоу. Дидактичний ефект досягається за рахунок зміни типів використовуваних шрифтів, розміру шрифту та колірної гами. Також активно використовуються різні варіанти підкреслень.

4. Анімовані схеми. У цьому варіанті презентації особливий акцент зроблено на різних графіках і схемах. Образотворчий ряд – мінімальний. Основна сфера застосування – заняття повторювально-узагальнювального характеру.

5 Опорні сигнали. В цьому випадку передбачається інтерактивна робота в офісних програмах і в графічних редакторах.

6. Таблиці. Варіант презентації, рекомендований при проведенні уроків закріплення, систематизації навчального матеріалу. Це можуть бути тематичні, синхронні, хронологічні та інші види таблиць. Використовуючи як фоновий рисунок зображення, що характеризує тему уроку, доповідач послідовно виводить на слайд незаповнену або частково заповнену таблицю, а слухачі завершують роботу. Заповнення таблиці відбувається після відповідного обговорення.

7. Аналіз картини (схеми). Презентація, змістом якої є картина, плакат, зображення (наприклад, обговорення художньої картини або електричної схеми). На уроці доповідач виділяє та обговорює зі слухачами фрагменти цього зображення, акцентуючи увагу на деталі, організовує роботу навколо

змісту та характеристики цього зображення, історії створення, принципу роботи тощо.

8. Тренажер. Закріплення умінь, навичок учнів за допомогою виконання поставлених завдань з можливістю виправлення помилкових кроків.

9. Тестування. Варіант, який також може бути рекомендований під час проведення повторювально-узагальнювального уроку. За допомогою відповідних гіперпосилань відповідь учнів супроводжується певною реакцією програми.

10. Робочий зошит. Екранний варіант робочого зошита на друкованій основі. Заповнення зошита відбувається після відповідного обговорення в класі або самостійно вдома.

Мультимедійна презентація може відігравати роль підготовленої методичної розробки із заданою навчальною метою та завданнями; орієнтована на конкретні результати, в ній демонстраційний матеріал носить лише ілюстративний характер, або ж слухачі можуть бути активними учасниками процесу, що виконують інтерактивні завдання, втручаються в процеси, активно взаємодіють з комп'ютером.

Презентація складається з набору слайдів навчального характеру, які можуть бути зведені у три групи:

- титульний слайд;
- слайди з інформацією;
- завершальний слайд.

Розробка слайду презентації здійснюється у такій послідовності:

- вибір фону слайда;
- вставка заголовка слайда;
- вставка тексту;
- вставка ілюстрацій;
- вставка формул;
- вставка кнопок керування;
- анімація об’єктів слайда.

3.1.8 Оформлення слайдів

Коли глядач дивиться на слайд, насамперед він повинен бачити текстову інформацію, ілюстрації, а вже потім фон слайда. Так формулюється основне правило вибору фона: колір фону слайда має бути витриманим у спокійних, стриманих тонах. Невдалі кольори фону: яскраво-червоний, малиновий, жовтий, темно-синій, темно-зелений і т. д. Від яскравих кольорів око людини швидко втомлюється, а на темному фоні важко розрізнити необхідну інформацію (рис. 3.5).

Рисунок 3.5 – Невдалі фони для слайдів

Фон слайда не рекомендується вибирати у вигляді картини, на такому фоні текстова інформація не виділяється, оскільки її важко розрізняти. Фон слайдів з навчальною інформацією не потрібно перенасичувати прикрасами. На таких слайдах допускається фон у вигляді аркуша зошита, фон з лінією, яка відділяє заголовок від основного тексту тощо (рис. 3.6).

Рисунок 3.6 – Естетичне виконання слайдів

Слайди, які містять навчальну інформацію, доцільно виконувати в єдиній кольоровій гамі. Для вибору фону використовуються основні та допоміжні кольори, заливка, шаблони оформлення (рис. 3.7).

Для вибору фону слайдів використовуються такі команди:

- «Формат – Фон» для отримання однотипних кольорів і способів заливки: градієнтна, текстура і т. д.;

- «Формат – Применить шаблон оформления».

Заголовок теми, розділів, підрозділів має велике значення в презентації. На новому слайді спостерігач спочатку розрізняє заголовок і приймає рішення, чи продовжувати йому подальший перегляд презентації. Тому заголовок має привертати увагу.

Як правило, для оформлення заголовка використовують два методи:

- заголовок оформлюється за допомогою одного із шрифтів, для цього використовуються інструменти «Шрифт» і «Размер», які розташовуються на верхній панелі інструментів;

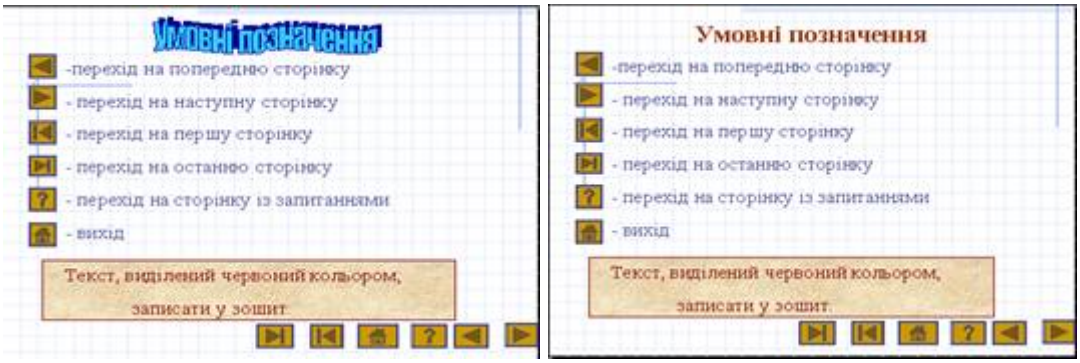


Рисунок 3.7 – Естетичне виконання слайдів

- заголовок може бути оформлений як об’єкт Word Art, з допомогою команд: «Вставка – Рисунок – Объект Word Art».

При визначенні вставки тексту розглядаються такі питання:

- обсяг тексту;
- розташування тексту.

Найпоширенішою помилкою при компонуванні слайда є велика кількість тексту та обумовлений цим дрібний шрифт. При читанні такого тексту швидко втомлюються очі та зникає бажання продовжувати знайомство з презентацією. Оптимальний обсяг тексту – не більше 15 рядків, крім заголовка.

При розміщенні тексту на слайді слід дотримуватися таких рекомендацій:

- блок тексту має виділятися;
- найбільш важливі частини тексту (визначення), доцільно виділяти рамкою або кольором;
- виділення різноманітних змістових частин тексту виконується за допомогою абзацу чи пустого рядка;
- при розміщенні на слайді рисунка та тексту часто використовується обтікання тексту навколо рисунка;
- для вставки тексту виконуються команди: «Вставка – Надпись».

Ілюстрації вставляються в текст для надання слайду більш привабливого вигляду та для покращення розуміння змісту тексту.

Ілюстрації мають відповідати темі продемонстрованого на слайді матеріалу. Не потрібно «перевантажувати» текст великою кількістю ілюстрацій.

Ілюстрації вставляються в текст таким чином:

- картинки із колекції Clip Art вставляються шляхом виконання команд: «Вставка – Рисунки – Картинки»;
- ілюстрації, які виконані в графічному редакторі Paint, наприклад, графік параболи, вставляються в слайд через буфер обміну шляхом виконання команди «Правка – Вставить»;
- ілюстрації, які зберігаються у вигляді файла, наприклад, після сканування фотографій, вставляються в слайд після виконання команди:

«Вставка – Рисунки – Из файла».

Як елементи керування в презентації використовуються кнопки різноманітної конфігурації, а також текст. Кнопки керування стандартної конфігурації встановлюються за допомогою команди: «Показ слайдов – Управляющие кнопки – ... – Настройка действия».

При використанні тексту як кнопки керування послідовність його створення така:

- вставляємо текст;
- поміщаємо текст у розмірну рамку;
- правою клавішею натискаємо на тексті;
- вибираємо команду «Настройка действия».

Зміну слайдів доцільно здійснювати за допомогою таких операцій: «Наплыв вверх», «Открытие вверх», «Появление сверху», «Растворение», «Случайный». Для цього необхідно виконати команду: «Показ слайдов – Смена слайдов».

Створення анімації об'єктів слайдів здійснюється шляхом виконання команд: «Показ слайдов – Настройка анимации».

Анімацію доцільно розробляти з врахуванням таких рекомендацій:

- об'єкти, що рухаються, не можуть накладатися на об'єкти, виведені на екран раніше;
- слід надавати перевагу використанню руху з середньою швидкістю, оскільки повільне переміщення втомлює, а надто швидке – не дозволяє розрізняти деталі переміщення;
- при організації виведення тексту, його можна виводити блоками, рядками, словами; виведення тексту літерами можна застосовувати лише для короткого тексту, наприклад, заголовка.
- слід надавати превагу виведенню об'єктів слайда в такій послідовності: заголовок, ілюстрація, текст; винятком є побудова ілюстрацій за елементами, в цьому випадку текст, який пояснює елемент, може з'являтися раніше самого елемента;
- анімаційні перетворення часто супроводжуються звуковими ефектами, які вибираються після налаштування анімації (команди: «Показ слайдов – Настройка анимации»); звукові ефекти не повинні бути різкими, голосними, найбільш використовуваними є ефекти: аплодисменти, ефект камери, дзвіночки тощо.

З метою посилення емоційного впливу презентація доповнюється мовним супроводом. Для озвучення слайдів в Power Point використовується команда: «Показ слайдов – Звукозапись».

Увага!

Будь-яка презентація є демонстраційним матеріалом, який призначений для інформування певного кола людей. Після завершення семінару його тема може викликати дискусію, в процесі якої, можливо, буде необхідно звернутися до самої презентації. Це буде зробити легше в тому випадку, коли кожна сторінка презентації буде пронумерована.

3.2 Хід роботи

3.2.1 Завдання № 1

1. Підготувати товарний чек, в якому зафіксована купівля декількох найменувань (восьми–десяти) товарів. Вхідні дані: назва товару, ціна одиниці товару, кількість товару, сума за кожен вид товару, сума до сплати.

2. Річки України. Створити табличку за зразком (табл. 3.1). У виділених комірках вказати формулу для обчислень.

Таблиця 3.1 – Зразок завдання № 1

Річки України					
	Площа басейну	частка пл. басейну	Загальна довжина	Довжина в Україні	Част. довж. в Україні
Дністер	72,1	4,50%	1362	1001	73,49%
Прут	27,5	1,72%	910	285	31,32%
Тиса	153,2	9,57%	966	285	29,50%
Дніпро	503,5	31,45%	2285	1205	52,74%
Сейм	27,5	1,72%	717	232	32,36%
Дунай	817	51,04%	2850	164	5,75%
…					
…					
Всього	1600,8		9090	3172	34,90%

3. Клієнт відкрив рахунок у банку на деяку суму під 12% річних. Яка сума буде на його рахунку через 10 років? Відобразити щорічні зміни на рахунку протягом 10 років. Відобразити зміни на рахунку у вигляді такої таблиці (табл. 3.2).

Таблиця 3.2 – Зразок завдання № 1

Нарахування відсотків протягом 10 років			
Рік	Сума	Приріст	Всього
…	…	…	…

3. Курс грошових одиниць. Створити табличку за зразком (табл. 3.3). У виділених комірках вказати формулу для обчислень.

3.2.2 Завдання № 2

1. Створити таблицю за зразком (табл. 3.4). У комірках, виділених сірим кольором, провести відповідні обчислення.

Таблиця 3.3 – Зразок завдання № 1

	Курс долара:		28,53	0,83	58,52
	Країна	**Ціна в доларах**	**Ціна в гривнях**	**Ціна в євро**	**Ціна в рублях**
1	Англія	600			
2	Болгарія	250			
3	Бразилія	1100			
4	Канада	980			
5	Німеччина	570			
6	Франція	610			

Таблиця 3.4 – Зразок завдання № 2

Назва озера	Площа плеса (тис. кв. км)	Максимальна глибина (м)	Висота плеса над рівнем моря (м)
Байкал	31,5	1520	456
Танганьїка	34	1470	773
Вікторія	68	80	1134
Гурон	59,6	288	177
Аральське	51,1	61	53
Мічиган	58	281	177
Сумарна площа			
Площа найбільшого озера			
Глибина наймілкішого озера			
Середня висота плеса озер над рівнем моря			
Середня висота плеса озер над рівнем моря (округлена до двох знаків після коми)			

2. Дано кількість опадів впродовж 10 днів. Знайти середню, максимальну та мінімальну кількість опадів за день, абсолютне відхилення для кожного дня та загальну кількість опадів.

3.2.3 Завдання № 3

Розробити презентацію на тему «Історія розвитку комп’ютерної техніки», використовуючи всі засоби створення презентацій, розглянутих у цій лабораторній роботі.

3.3. Контрольні питання

1. Перерахуйте галузі застосування електронних таблиць.
2. Як називається документ в програмі Excel? Яка його структура?
3. Як формується адреса комірки?
4. Електронна таблиця Excel може мати до 65536 рядків. Як Ви вважаєте, звідки взято це число?
5. Які дії можна виконувати з робочими аркушами? Як проводяться ці дії?
6. Які комбінації клавіш використовуються для швидкого переміщення робочим аркушем?
7. Як зберегти на зовнішніх носіях створену електронну книгу?
8. Як відкрити раніше створену електронну книгу?
9. Яке призначення клавіші F2 в Excel?
10. Наведіть способи зміни розмірів комірок.
11. Який тип адресації Excel використовує за замовчуванням?
12. Яке призначення має майстер функцій і як ним користуватись?
13. Що таке презентація та слайд?
14. Які є основні режими відображення слайдів?
15. Особливості процедури зміни слайдів в презентації.
16. Формування роздаткового матеріалу.
17. Застосування анімаційних ефектів в Microsoft PowerPoint.
18. Який порядок створення фотоальбому?
19. Яке призначення засобу перегляду в Microsoft PowerPoint?

ЛАБОРАТОРНА РОБОТА № 4
СИСТЕМИ ЧИСЛЕННЯ. ПЕРЕВЕДЕННЯ ЧИСЕЛ З ОДНІЄЇ СИСТЕМИ ЧИСЛЕННЯ В ІНШУ

Мета: навчитись подавати числа в двійковій, вісімковій, шістнадцятковій і двійково-десятковій *системах числення* (СЧ, numerical systems) та навчитись переводити числа із однієї системи числення в іншу.

4.1 Теоретичні відомості

Таблиця 4.1 – Подання чисел в різних системах числення

Десяткова	Двійкова	Двійково-десяткова	Вісімкова	Шістнадцяткова
0	0	0000	0	0
1	1	0001	1	1
2	10	0010	2	2
3	11	0011	3	3
4	100	0100	4	4
5	101	0101	5	5
6	110	0110	6	6
7	111	0111	7	7
8	1000	1000	1 0	8
9	1001	1001	1 1	9
1 0	1010	0001 0000	1 2	A
1 1	1011	0001 0001	1 3	B
1 2	1100	0001 0010	1 4	C
1 3	1101	0001 0011	1 5	D
1 4	1110	0001 0100	1 6	E
1 5	1111	0001 0101	1 7	F
1 6	10000	0001 0110	2 0	1 0
1 7	10001	0001 0111	2 1	1 1
…	…	…	…	…
3 2	100000	0011 0010	4 0	2 0

Переведення числа з десяткової СЧ у двійкову СЧ.

$$999{,}35_{10} \rightarrow ?_2$$

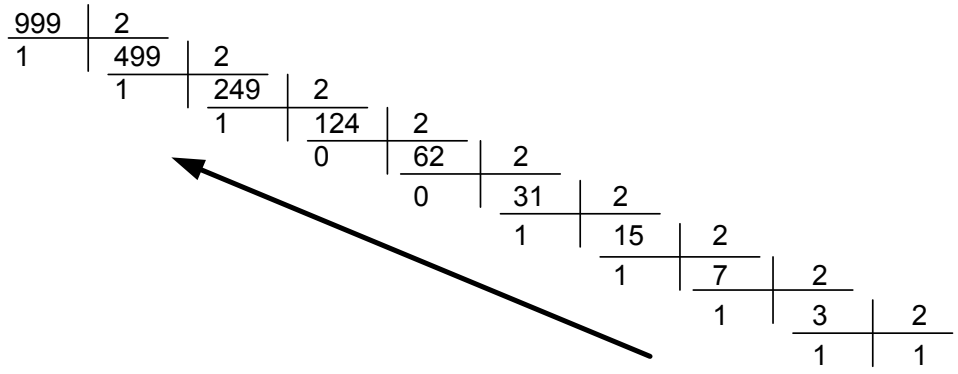
Для переведення цілої частини: число, що записане в системі числення з основою *n*, послідовно ділити на основу нової системи числення, відділяючи остачі. Останні, що записані в зворотному порядку, будуть утворювати число в новій системі числення.

Для переведення дробової частини: послідовно дробову частину множити на основу нової системи числення, виділяючи цілі частини, які будуть утворювати запис дробової частини числа в новій системі числення.

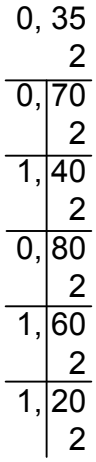
Приклади: $999{,}35_{10} = 1111100111{,}01011_2$

Рішення:

- для цілої частини:



- для дробової частини:



Обернене переведення здійснюється обчисленням суми добутків цифр двійкового числа на двійку (основу СЧ) у степені, що визначається розрядом цифри, починаючи з нуля.

Переведемо число 483 із десяткової у двійкову СЧ.

Отже, скориставшись наведеним правилом, одержимо $483_{10} \rightarrow 111100011_2$. А для перевірки результат потрібно записати у такому вигляді:

Ваговий коефіцієнт	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Число	1	1	1	1	0	0	0	1	1

Отже, $1 \cdot 256+1 \cdot 128+1 \cdot 64+1 \cdot 32+0 \cdot 16+0 \cdot 8+0 \cdot 4+1 \cdot 2+1 \cdot 1 = 483_{10}$

Переведемо число 483 (в десятковій СЧ) у вісімкову СЧ.

$483_{10} \rightarrow ?_8$

Розв’язання. Розбиваємо двійкове число на тріади справа наліво та записуємо кожну тріаду відповідною вісімковою цифрою (див. табл. 4.1):

$483_{10} \rightarrow 111100011_2 \rightarrow 111.100.011_2 \rightarrow 743_8$

Перевірка: $743_8 \rightarrow 7 \cdot 8^2 + 4 \cdot 8^1 + 3 \cdot 8^0 = 7 \cdot 64 + 4 \cdot 8 + 3 = 483_{10}$

Переведення числа 483 (в десятковій СЧ) у шістнадцяткову СЧ.

$483_{10} \rightarrow ?_{16}$

Розв'язання. Розбиваємо двійкове число на тетради (четвірки цифр) справа наліво та кожну тетраду зображаємо відповідною шістнадцятковою цифрою (див. табл. 4.1):

$483_{10} \rightarrow 111100011_2 \rightarrow 0001.1110.0011_2 \rightarrow 1E3_{16}$

Перевірка:

$1E3_{16} \rightarrow 1 \cdot 16^2 + 14 \cdot 161 + 3 \cdot 160 = 256 + 224 + 3 = 483_{10}$

Перевести число 483 (в десятковій СЧ) у двійково-десяткову СЧ.

$483_{10} \rightarrow ?_{2\text{-}10}$

Розв'язання. Кожну десяткову цифру зображаємо відповідною двійковою тетрадою (див. табл. 4.1):

4 8 3

0100 1000 0011

Відповідь. $483_{10} \rightarrow 10010000011_{2\text{-}10}$.

4.2 Хід роботи

Вибрати завдання із таблиці 4.3 відповідно до порядкового номера студента в журналі групи. Кожне число з тих, які наведено в таблиці індивідуальних завдань, записати в заданих системах числення. Значення вагових коефіцієнтів семи розрядів числа в різних системах числення подано в таблиці 4.2.

Таблиця 4.2 – Значення вагових коефіцієнтів семи розрядів числ

Номер розряду	Системи числення		
	Двійкова	Вісімкова	Шістнадцяткова
0	1	1	1
1	2	8	16
2	4	64	256
3	8	512	4096
4	16	4096	65536
5	32	32768	1048576
6	64	262144	16777216
7	128	2097152	268435456
8	256	16777216	4294967296
9	512	134217728	68719476736
10	1024	1073741824	1099511627776

Таблиця 4.3 – Індивідуальні завдання

Варіант	Завдання		
1	$11001100_2 \to_{10}$ $11010101_2 \to_{10}$ $101001000_2 \to_{10}$	$10011010_2 \to_8$ $E1_{16} \to_{10}$ $147_8 \to_{10}$	$46_{10} \to_2$ $E1_{16} \to_2$ $DA_{16} \to_8$
2	$10111101_2 \to_{10}$ $11000001_2 \to_{10}$ $101101010_2 \to_{10}$	$10000010_2 \to_8$ $A1_{16} \to_{10}$ $127_8 \to_{10}$	$52_{10} \to_2$ $A1_{16} \to_2$ $FB_{16} \to_8$
3	$10000010_2 \to_{10}$ $10010001_2 \to_{10}$ $100001111_2 \to_{10}$	$10011010_2 \to_8$ $A3_{16} \to_{10}$ $141_8 \to_{10}$	$46_{10} \to_2$ $A3_{16} \to_2$ $FA_{16} \to_8$
4	$11001111_2 \to_{10}$ $10000101_2 \to_{10}$ $101111000_2 \to_{10}$	$11100010_2 \to_8$ $A5_{16} \to_{10}$ $122_8 \to_{10}$	$59_{10} \to_2$ $A5_{16} \to_2$ $CC_{16} \to_8$
5	$10000000_2 \to_{10}$ $10011111_2 \to_{10}$ $101001000_2 \to_{10}$	$10000111_2 \to_8$ $A7_{16} \to_{10}$ $133_8 \to_{10}$	$51_{10} \to_2$ $A7_{16} \to_2$ $BA_{16} \to_8$
6	$11111110_2 \to_{10}$ $10000111_2 \to_{10}$ $101001110_2 \to_{10}$	$10101010_2 \to_8$ $A9_{16} \to_{10}$ $130_8 \to_{10}$	$50_{10} \to_2$ $A9_{16} \to_2$ $FB_{16} \to_8$
7	$11011110_2 \to_{10}$ $10010001_2 \to_{10}$ $101111000_2 \to_{10}$	$11000011_2 \to_8$ $C1_{16} \to_{10}$ $140_8 \to_{10}$	$49_{10} \to_2$ $C1_{16} \to_2$ $DA_{16} \to_8$
8	$11111111_2 \to_{10}$ $10011101_2 \to_{10}$ $101001000_2 \to_{10}$	$10011011_2 \to_8$ $C3_{16} \to_{10}$ $141_8 \to_{10}$	$54_{10} \to_2$ $C3_{16} \to_2$ $AF_{16} \to_8$
9	$11110100_2 \to_{10}$ $10000001_2 \to_{10}$ $101111000_2 \to_{10}$	$11111110_2 \to_8$ $C5_{16} \to_{10}$ $138_8 \to_{10}$	$56_{10} \to_2$ $C5_{16} \to_2$ $AB_{16} \to_8$
10	$10011101_2 \to_{10}$ $11100111_2 \to_{10}$ $101000011_2 \to_{10}$	$10111010_2 \to_8$ $C7_{16} \to_{10}$ $142_8 \to_{10}$	$57_{10} \to_2$ $C7_{16} \to_2$ $AD_{16} \to_8$

Продовження таблиці 4.3

11	$10010101_2 \rightarrow_{10}$ $10011111_2 \rightarrow_{10}$ $101111111_2 \rightarrow_{10}$	$11111111_2 \rightarrow_8$ $C9_{16} \rightarrow_{10}$ $138_8 \rightarrow_{10}$	$47_{10} \rightarrow_2$ $C9_{16} \rightarrow_2$ $AC_{16} \rightarrow_8$
12	$10000101_2 \rightarrow_{10}$ $11001100_2 \rightarrow_{10}$ $101101010_2 \rightarrow_{10}$	$10011110_2 \rightarrow_8$ $B1_{16} \rightarrow_{10}$ $143_8 \rightarrow_{10}$	$55_{10} \rightarrow_2$ $B1_{16} \rightarrow_2$ $AA_{16} \rightarrow_8$
13	$10111101_2 \rightarrow_{10}$ $11000001_2 \rightarrow_{10}$ $101001001_2 \rightarrow_{10}$	$11110011_2 \rightarrow_8$ $B3_{16} \rightarrow_{10}$ $145_8 \rightarrow_{10}$	$53_{10} \rightarrow_2$ $B3_{16} \rightarrow_2$ $BB_{16} \rightarrow_8$
14	$10000111_2 \rightarrow_{10}$ $11010101_2 \rightarrow_{10}$ $100001111_2 \rightarrow_{10}$	$10100110_2 \rightarrow_8$ $B5_{16} \rightarrow_{10}$ $129_8 \rightarrow_{10}$	$48_{10} \rightarrow_2$ $B5_{16} \rightarrow_2$ $BF_{16} \rightarrow_8$
15	$10000001_2 \rightarrow_{10}$ $10101010_2 \rightarrow_{10}$ $101011111_2 \rightarrow_{10}$	$10001111_2 \rightarrow_8$ $B7_{16} \rightarrow_{10}$ $156_8 \rightarrow_{10}$	$60_{10} \rightarrow_2$ $B7_{16} \rightarrow_2$ $BB_{16} \rightarrow_8$
16	$11111110_2 \rightarrow_{10}$ $11001111_2 \rightarrow_{10}$ $101001000_2 \rightarrow_{10}$	$10011011_2 \rightarrow_8$ $B9_{16} \rightarrow_{10}$ $132_8 \rightarrow_{10}$	$55_{10} \rightarrow_2$ $B9_{16} \rightarrow_2$ $BD_{16} \rightarrow_8$
17	$10000001_2 \rightarrow_{10}$ $11001000_2 \rightarrow_{10}$ $101001110_2 \rightarrow_{10}$	$11001011_2 \rightarrow_8$ $D1_{16} \rightarrow_{10}$ $137_8 \rightarrow_{10}$	$54_{10} \rightarrow_2$ $D1_{16} \rightarrow_2$ $CA_{16} \rightarrow_8$
18	$10000001_2 \rightarrow_{10}$ $11110100_2 \rightarrow_{10}$ $101111000_2 \rightarrow_{10}$	$10011010_2 \rightarrow_8$ $D5_{16} \rightarrow_{10}$ $111_8 \rightarrow_{10}$	$56_{10} \rightarrow_2$ $D5_{16} \rightarrow_2$ $CB_{16} \rightarrow_8$
19	$10000001_2 \rightarrow_{10}$ $10000001_2 \rightarrow_{10}$ $111111111_2 \rightarrow_{10}$	$10001111_2 \rightarrow_8$ $D9_{16} \rightarrow_{10}$ $138_8 \rightarrow_{10}$	$57_{10} \rightarrow_2$ $D9_{16} \rightarrow_2$ $CF_{16} \rightarrow_8$
20	$11110100_2 \rightarrow_{10}$ $10010001_2 \rightarrow_{10}$ $101111000_2 \rightarrow_{10}$	$10111010_2 \rightarrow_8$ $F1_{16} \rightarrow_{10}$ $145_8 \rightarrow_{10}$	$51_{10} \rightarrow_2$ $F1_{16} \rightarrow_2$ $CE_{16} \rightarrow_8$
21	$10000010_2 \rightarrow_{10}$ $10111101_2 \rightarrow_{10}$ $110110100_2 \rightarrow_{10}$	$10000011_2 \rightarrow_8$ $F5_{16} \rightarrow_{10}$ $140_8 \rightarrow_{10}$	$53_{10} \rightarrow_2$ $F5_{16} \rightarrow_2$ $EE_{16} \rightarrow_8$

Продовження таблиці 4.3

22	$10000100_2 \rightarrow_{10}$ $10011111_2 \rightarrow_{10}$ $101001000_2 \rightarrow_{10}$	$10011110_2 \rightarrow_8$ $F3_{16} \rightarrow_{10}$ $150_8 \rightarrow_{10}$	$49_{10} \rightarrow_2$ $F3_{16} \rightarrow_2$ $EA_{16} \rightarrow_8$
23	$11011001_2 \rightarrow_{10}$ $10000001_2 \rightarrow_{10}$ $101111000_2 \rightarrow_{10}$	$11010110_2 \rightarrow_8$ $F7_{16} \rightarrow_{10}$ $154_8 \rightarrow_{10}$	$56_{10} \rightarrow_2$ $F7_{16} \rightarrow_2$ $EC_{16} \rightarrow_8$
24	$10000001_2 \rightarrow_{10}$ $11001100_2 \rightarrow_{10}$ $101111111_2 \rightarrow_{10}$	$10000111_2 \rightarrow_8$ $E3_{16} \rightarrow_{10}$ $135_8 \rightarrow_{10}$	$59_{10} \rightarrow_2$ $E3_{16} \rightarrow_2$ $DF_{16} \rightarrow_8$
25	$10000010_2 \rightarrow_{10}$ $10111101_2 \rightarrow_{10}$ $100001111_2 \rightarrow_{10}$	$10011110_2 \rightarrow_8$ $E8_{16} \rightarrow_{10}$ $133_8 \rightarrow_{10}$	$52_{10} \rightarrow_2$ $E8_{16} \rightarrow_2$ $DB_{16} \rightarrow_8$
26	$11001100_2 \rightarrow_{10}$ $11010101_2 \rightarrow_{10}$ $101001000_2 \rightarrow_{10}$	$10011010_2 \rightarrow_8$ $E1_{16} \rightarrow_{10}$ $147_8 \rightarrow_{10}$	$46_{10} \rightarrow_2$ $E1_{16} \rightarrow_2$ $DA_{16} \rightarrow_8$

4.3 Контрольні питання

1. Що таке система числення?
2. Які типи систем числення Ви знаєте?
3. Що таке основа позиційної системи числення?
4. У чому полягає проблема вибору системи числення для подання чисел у пам’яті комп’ютера?
5. Яка система числення використовується для подання чисел у пам’яті комп’ютера? Чому?
6. Яким чином здійснюється переведення чисел, якщо основа нової системи числення дорівнює деякому степеню старої системи числення?
7. За яким правилом переводяться числа з десяткової системи числення?
8. За яким правилом переводяться числа в десяткову систему числення?

ЛАБОРАТОРНА РОБОТА № 5
ЗНАЙОМСТВО З СИСТЕМАМИ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ. ВИВЧЕННЯ СЕРЕДОВИЩА DELPHI ТА РОБОТА В НЬОМУ. ПРОГРАМУВАННЯ ЛІНІЙНИХ АЛГОРИТМІВ

Мета: набути навичок роботи в середовищі Delphi. Студент повинен навчитись працювати в середовищі Delphi та програмувати лінійні алгоритми.

5.1 Теоретичні відомості

5.1.1 Загальна організація програми в Delphi

Програма, яка створюється в середовищі програмування Delphi, основана на модульному принципі. Головна програма складається з конструкції оголошення списку модулів, що використовується компонентами, та декількох операторів, що створюють об'єкти для всіх форм і програм.

Всі об'єкти компонентів розміщуються в спеціальних об'єктах – *формах* (forms). Для кожної форми, яка спроектована в програмі, Delphi автоматично створює окремий модуль, в який користувач може ввести власний код, створюючи обробники різних подій. Саме в цих модулях і здійснюється програмування команд.

5.1.2 Структура головної програми Delphi

Код головної програми (проекту) створюється середовищем Delphi автоматично та, як правило, не потребує модифікації. Приклад такої головної програми наведено у табл. 5.1.

Таблиця 5.1 – Структура проекту програми в середовищі Delphi

Блок програми	Функціональне призначення блоку програми
program Projectl;	Програма починається з ключового слова «program», після якого вказується ім'я програми. Воно збігається з ім'ям файла, в якому був збережений проект. Це ж ім'я присвоюється виконавчому файлу програми. За замовчуванням використовується ім'я Project1.
Uses Forms, Unitl in 'Unitl.pas' {Forml};	У розділі «uses» перераховуються всі модулі, що завантажуються програмою з доступних бібліотек. Перший модуль «Forms» є системним, а наступні – модулями розроблених користувачем форм. Цей проект складається з однієї форми з ім'ям Form1, що розміщується в модулі Unit1. Після ключового слова «in» вказується ім'я файла, у якому міститься модуль Unit1.

Продовження таблиці 5.1

{$R *.res}	Цей рядок являє собою директиву компілятора, що поєднує програму з файлами ресурсів Windows («*.Dfm», «*.Res»). За замовчуванням для файлів ресурсів використовується розширення «*.Res».
begin Application.Initialize; Application.CreateForm (TForml, Forml); Application.Run; end.	Оператор Application.Initialize ініціалізує програму. Application.CreateForm створює об'єкти форми. Application.Run починає виконання програми.

Текст програми зберігається в модулях, назва яких має збігатися з ім'ям файлів. Модуль складається із трьох розділів: інтерфейсу, реалізації й ініціалізації. Структуру модуля програми Delphi подано в табл. 5.2.

Таблиця 5.2 – Структура модуля програми в середовищі Delphi

Розділ модуля	Функціональне призначення розділу модуля
Unit Unitl;	Назва модуля (ця назва використається в директиві «uses» при підключенні модуля до програми).
Interface …	Розділ інтерфейсу
Implementation …	Розділ реалізації (частина для команд)
Begin …	Розділ ініціалізації
End.	

Розділ *інтерфейсу* починається ключовим словом «Interface». Він повідомляє компілятору, яка частина модуля є доступною для інших модулів програми.

Тут можуть розміщуватися списки бібліотек, які підключаються до модуля, оголошення типів, констант, змінних, заголовки функцій і процедур, до яких має бути доступ з інших модулів. Тобто, у цьому розділі перераховується все, що має бути «видимим» для програми, яка використовує цей модуль.

Розділ *реалізації* починається ключовим словом «Implementation» і містить оголошення локальних змінних, процедур і функцій, що підтримують роботу форми.

Починається розділ директивою {$R *.dfm}, яка вказує компілятору, що в цей розділ реалізації потрібно вставити інструкції установлення

значень властивостей форми, які перебувають у файлі з розширенням «*.dfm», ім’я якого збігається з іменем модуля.

Далі в розділі реалізації може міститися директива «uses», блоки оголошення типів, констант, змінних, до яких не буде доступу з інших модулів. Тут само розташовуються всі тексти процедур і функцій, оголошених у розділі «Interface». Заголовки процедур і функцій можуть повністю збігатися із заголовками в інтерфейсному розділі або можуть відрізнятися від них повною відсутністю параметрів. Якщо в цій частині набраний текст функції або процедури, яка не оголошена в «Interface», то ця функція або процедура буде локальною.

Розділ *ініціалізації* дозволяє виконати ініціалізацію змінних модуля. Розташовується після розділу реалізації між ключовими словами «Begin» і «End».

У цій частині розміщуються оператори, які мають виконуватися один раз при першому звертанні до модуля (після ключового слова «Initialization») або оператори, які виконуються в будь-якому разі при завершенні роботи модуля (після ключового слова «Finalization»). Ця частина є необов’язковою. На практиці такий підхід використовується рідко, тому зазвичай після «Implementation» відразу розміщується «End» з крапкою (без «Begin»).

Таблиця 5.3 – Структура процедури дії

Розділ процедури	Зміст розділу процедури
Procedure <назва процедури>	Заголовок процедури. Назва процедури складається з двох частин: назви об’єкта та назва події
Const <ім’я константи> = <значення константи>; …	Розділ опису констант
Type <ім’я типу> = <тип>; …	Розділ типів
Var <ім’я змінної>:<тип>; …	Розділ опису змінних
<Тексти локальних процедур і функцій із заголовками	Розділ процедур і функцій

Begin
<інструкція>;
<інструкція>;
...
End;
	Розділ, у якому пишуться інструкції процедури обробки подій

5.1.3 Структура проекта в Delphi

Проект Delphi складається з набору програмних одиниць – модулів, які зберігаються в окремих файлах.

В Delphi існують файл проекту та файли модулів. Файл проекту – це файл головної програми з розширенням «*.dpr», файли модулів – це набір всіх файлів програми.

Файл з розширенням «*.dpr» містить основну інформацію про проект. За замовчуванням цей файл називається «Project1.dpr». Новачкам редагувати його не рекомендується.

Файл з розширенням «*.pas» – це файл модуля. В ньому зберігається текст програми мовою Object Pascal. За замовчуванням цей файл називається «Unit1.pas».

Файл з розширенням «*.dfm» містить інформацію про зовнішній вигляд форми. Інформація подана в кодованому вигляді.

Файл з розширенням «*.res» – це ресурсний файл, у якому зберігається інформація про курсори, іконки тощо.

Файл з розширенням «*.exe» – виконавчий файл програми.

Файли з розширеннями «*.~df», «*.~pa» – файли зі старими версіями програми.

У файлів з розширеннями «*.pas», «*.dfm», «*.~df», «*.~pa» завжди однакові імена (за замовчуванням Unit1) .

У файлів з розширеннями «*.dpr», «*.exe», «*.res» – також однакові імена (за замовчуванням Project1).

5.1.4 Загальний опис середовища Delphi

Після запуску Delphi на екрані комп’ютера з’являється основне вікно інтегрованого середовища розробки (ICP) (рис. 5.1).

У верхній частині вікна ICP відображається смуга головного меню. Нижче дві інструментальні панелі:

• ліва панель містить два ряди кнопок, що дублюють деякі команди меню, які найбільш часто використовуються (рис. 5.2);

• права панель містить бібліотеки візуальних компонентів (Visual Component Library, VCL) – палітри компонентів (рис. 5.3).

Палітра компонентів за допомогою іконок дозволяє вибрати візуальні та інші компоненти, з яких, як з «будівельних блоків», складається програма.

Палітра містить низку сторінок, назви яких відображаються в її верхній частині. Всього сторінок тринадцять. Найбільш актуальні з них: «Standard» (стандартні компоненти) і «Additional» (додаткові компоненти).

Стандартні компоненти – це класи, що реалізовують інтерфейсні елементи середовища Windows. Серед них: головне меню («MainMenu»), контекстне меню («PopupMenu»), позначка або мітка («Label»), текстове поле («Edit») та інші.

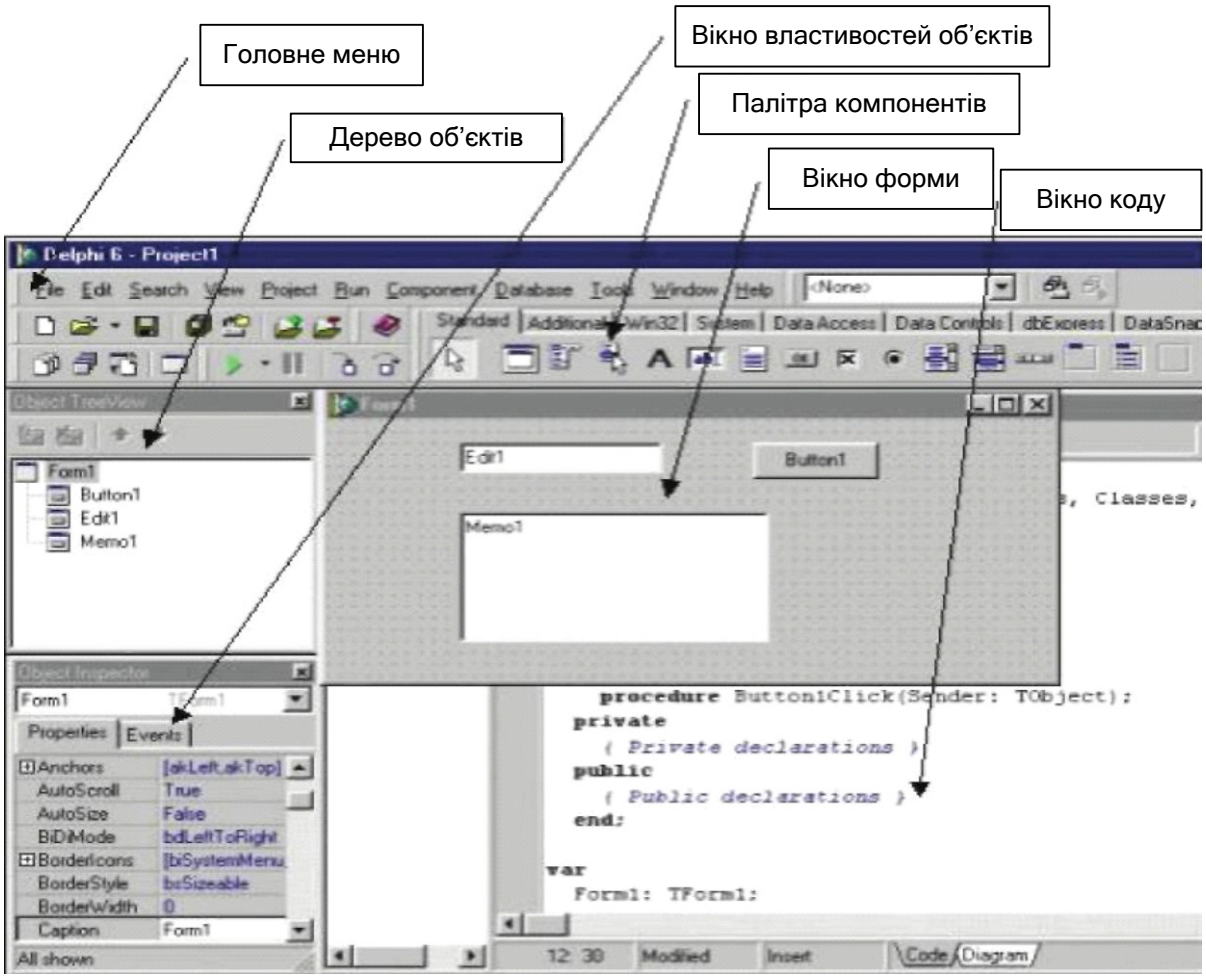


Рисунок 5.1 – Інтерфейс середовища Borland Delphi

Рисунок 5.2 – Панель швидкого доступу до функцій

Рисунок 5.3 – Палітра компонентів Delphi

Додаткові компоненти – це класи різних додаткових інтерфейсних елементів – графічні кнопки («BitBtn»), редактор із введенням за шаблоном («MaskEdit») та інші.

Правіше смуги головного меню розташовується невелика інструментальна панель, що слугує для збереження й вибору різних конфігурацій вікна ІСР.

На основному вікні інтегрованого середовища розробки розташовано ще три вікна.

Вікно «Форма» (Form1) – це макет вікна програми, яка розробляється в ICP Delphi (рис. 5.4).

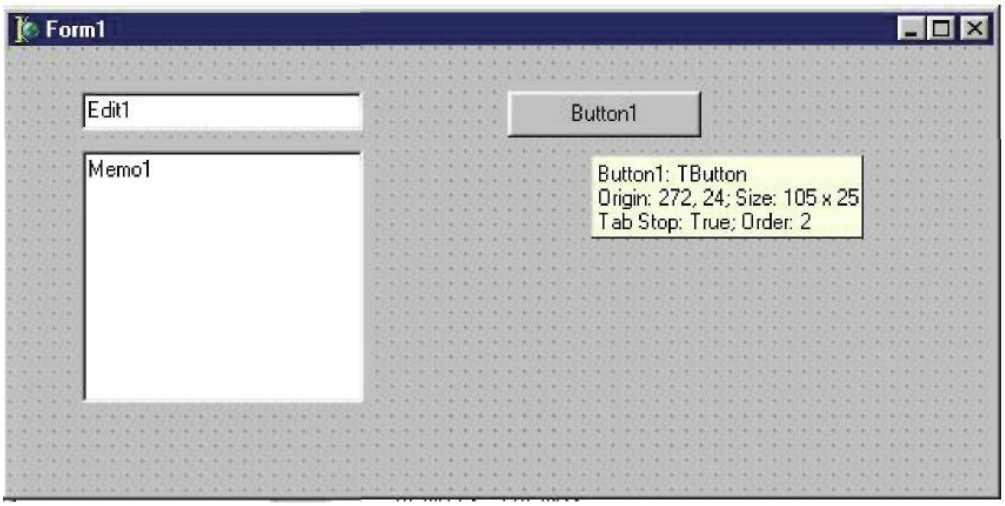


Рисунок 5.4 – Вікно форми

Вікно «Інспектор об'єктів» (Object Inspector) дозволяє змінювати властивості (характеристики) об'єктів: форму, командні кнопки, поля введення і т. д. (рис. 5.5, *а*).

Інспектор об'єктів складається з двох сторінок: «Properties» (властивості) та «Events» (події).

Кожна сторінка розділена на дві частини. На сторінці властивостей у лівій частині відображаються назви властивостей, а у правій – їх значення.

Знак «+» ліворуч від назви вказує на те, що властивість складається з декількох параметрів. Параметрами властивостей можуть бути слова, числа, значення з нисхідного списку.

Натиснувши на знак «**v**» праворуч від поля значення, з'явиться випадаючий список значень.

При натисканні на знак «...» з'являється спеціальне діалогове вікно.

Якщо значенням параметру є число або текст, то після його набору краще натиснути «Enter», інакше воно може не бути зафіксовано. При натисканні на «Esc» введення скасовується.

Сторінка подій використовується для створення реакції на деяку подію (рис. 5.5 *б*). Сторінка також складається з двох частин. В першій – назва події, а в другій – назва процедури, яка обробляє цю подію. Якщо в правій частині нічого не написано, то програма на подію не реагує. Назва процедури складається із двох частин: назви об'єкта та назви події.

Для створення нової реакції на подію необхідно двічі клацнути в правій половині напроти потрібної події. У цьому місці з'явиться назва нової процедури, а на екрані – вікно редактора коду.

Наприклад:

```
procedure TForml.Button2Click(Sender: TObject);
begin
Close;
```

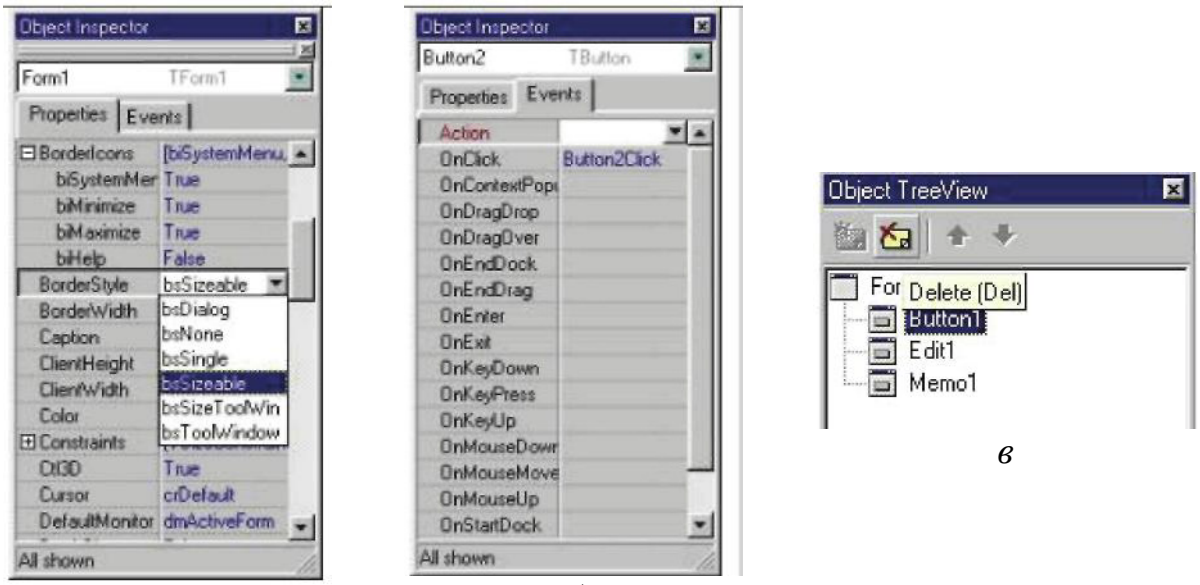
end;

Увага!

1. Якщо Ви випадково створили непотрібну реакцію на подію, то не варто звертати на неї увагу: при збереженні або компіляції програми дана помилка виправляється автоматично ICP Delphi.

2. Забороняється виправляти текст програми, написаний самим ICP Delphi, інакше вся програма може бути зіпсована.

Вікно «Дерево об'єктів» (Object Tree View) відображає ієрархію компонентів програми з точки зору їхньої належності один одному (рис. 5.5, *в*).



а *б* *в*

Рисунок 5.5 – Інспектор об'єктів (*а, б*) та дерево об'єктів (*в*)

У дереві об'єктів можна здійснювати операції натискання та перетаскування, переміщуючи дочірні компоненти в інші контейнери. При цьому зміни синхронно відображаються в редакторі форм.

У вікні «Редактор коду» (Code Editor) між ключовими словами «Begin» і «End» можна друкувати інструкції Object Pascal, які реалізовують процедури обробки подій (рис. 5.6).

5.1.5 Коротка характеристика деяких компонентів

Нижче перераховано компоненти, їхні основні властивості та події, при здійсненні яких будуть виконуватися запрограмовані дії.

Компонент «Form» (екранна форма). Форма – це не лише зовнішній вигляд вікна програми, але й повноцінна компонента з власними властивостями та подіями (табл. 5.4).

Значення властивостей можна задати або у вікні властивостей об'єкта, або в самій програмі, наприклад:

Form1.Color := clRed; {встановлення червоного кольору форми}

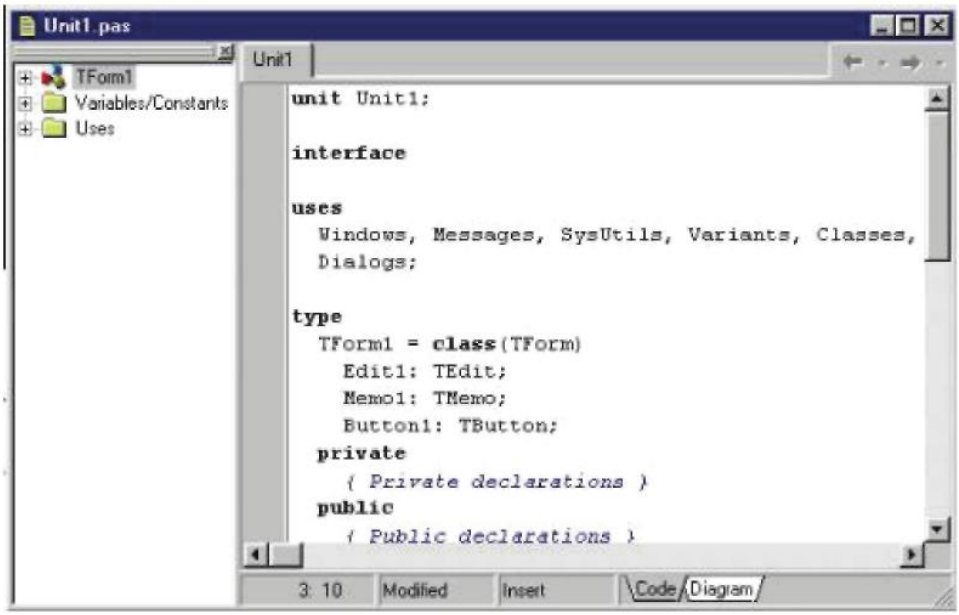


Рисунок 5.6 – Вікно редактора коду

Таблиця 5.4 – Основні властивості та подія компонента «Form»

Властивість, подія	Призначення властивості або події
Властивості	
ALIGN	Задає режим вирівнювання об’єктів всередині форми.
BORDERSTYLE	Задає стиль обрамлення форми, а також поведінки форми (можливість змінювати розміри вікна).
CAPTION	Задає заголовок вікна форми.
COLOR	Задає кольори форми.
FONT	Задає атрибути шрифту форми.
Подія	
LOAD	Відбувається при завантаженні форми.

Компонент «Label» (напис, позначка або мітка) призначений для створення написів. Його можна використати для виведення відповіді або для пояснення даних, що вводяться. Належить до групи «Standard». Властивості та подія компонента наведені у табл. 5.5.

Приклад використання компоненту «Label» в програмі:

Labell.Caption: = ‘Сума=‘ +IntToStr(a); {оформлення виведення результату}

Компонент «Edit» (поле редагування). Використовується для введення/виведення чисел і тексту в програмі. Належить до групи «Standard». Властивості та подія компонента наведені у табл. 5.6.

Таблиця 5.5 – Основні властивості та подія компонента «Label»

Властивість, подія	Призначення властивості або події
Властивості	
CAPTION	Задає заголовок напису, що виводиться на екран.
LOAD	Відбувається при завантаженні форми.
ALIGNMENT	Задає режим вирівнювання тексту мітки.
AUTOSIZE	Дозволяє автоматично змінювати розміри мітки, щоб компонент відповідав розмірам напису (значення True).
FONT	Задає шрифт, що використовується для відображення тексту.
VISIBLE	Задає видимість напису на екрані. Має два значення. Якщо значення True, то напис видимий, якщо False – ні.
WORDWRAP	Дозволяє перенесення слів і рядків, які не помістились в задану область за шириною. Необхідно узгоджувати значення цієї властивості з властивістю Auto Size.
Подія	
CHANGE	Відбувається, коли користувач змінює текст

Таблиця 5.6 – Основні властивості та подія компонента «Edit»

Властивість, подія	Призначення властивості або події
Властивості	
AUTOSIZE	Задає необхідність зміни розміру компонента при зміні розміру шрифту (якщо True).
BORDERSTYLE	Задає стиль обрамлення поля.
TEXT	Задає вміст рядка редагування.
MAXLENGTH	Обмежує число символів, що вводяться в поле.
READONLY	Забороняє редагувати текст, який міститься в компоненті (якщо True).
Подія	
CHANGE	Відбувається, коли користувач змінює текст

Приклад використання компонента «Edit» в програмі:

Edit3.Text := FloatToStr(c, f f Fixed, 10, 4) {оформлення виведення результатів}

Компонент «Button» (командна кнопка) використовується для створення реакції на подію. Належить до групи «Standard». Основні властивості та подія компонента наведені у табл. 5.7.

Приклад використання компоненту «Button» в програмі:

```
procedure TForm1.Button2Click(Sender: TCtoject);
begin
Close          {припиняє виконання програми}
end;
```

Таблиця 5.7 – Основні властивості та подія компонента «Button»

Властивість, подія	Призначення властивості або події
Властивості	
CAPTION	Задає назву кнопки.
HEIGHT	Задає висоту кнопки.
WIDTH	Задає ширину кнопки.
LEFT	Відстань від лівої границі кнопки до лівої границі форми.
TOP	Відстань від верхньої границі кнопки до верхньої границі форми.
Подія	
ONCLICK	Відбувається, коли користувач клацає основною (лівою) кнопкою миші на об’єкті.

5.1.6 Помилки виконання додатка

Під час компіляції текст програми перевіряється на відсутність синтаксичних *помилок* (Errors). Компілятор переглядає програму від початкового рядка до останнього. Якщо при цьому виявляється помилка, то процес компіляції припиняється, у вікні редактора коду червоним кольором виділяється той рядок, який, за рішенням компілятора, містить помилкову конструкцію (рис. 5.7).

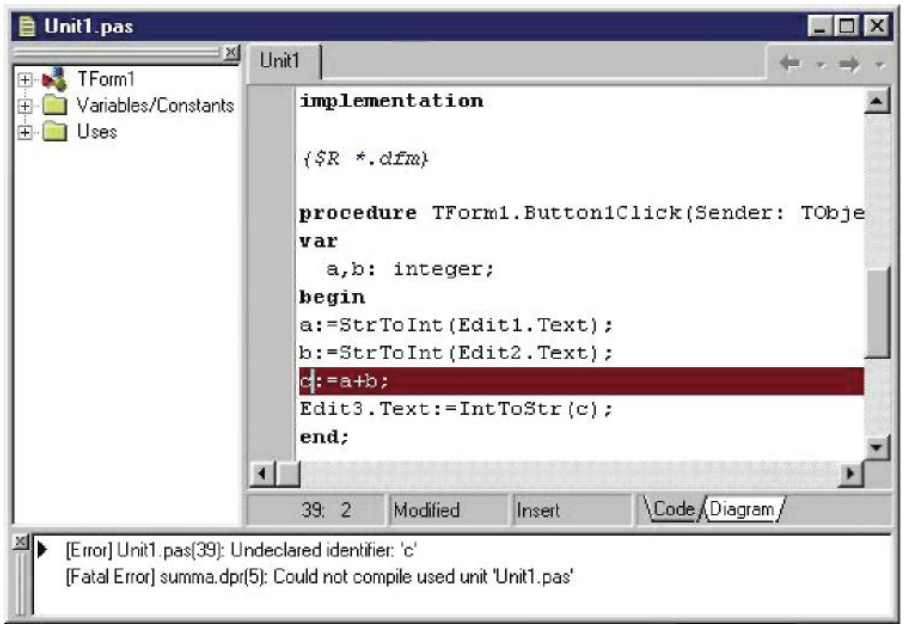


Рисунок 5.7 – Виділення помилки, допущеної в коді програми

В нижній частині вікна редактора коду компілятор виводить повідомлення про помилку. Перша помилка – це перша від початку тексту програми синтаксична помилка, виявлена компілятором (табл. 5.8). Наявність у тексті навіть однієї синтаксичної помилки призводить до виникнення другої, фатальної помилки (Fatal Error) – неможливості генерації виконавчого файла програми, яка компілюється.

Увага!

Рядок, виділений компілятором, не завжди містить помилку. Досить часто помилковою є інструкція, що перебуває в попередньому рядку.

Таблиця 5.8 – Найбільш типові помилки компіляції

Повідомлення компілятора	Імовірна причина
Undeclared identifier (Неоголошений ідентифікатор)	Використовується змінна, яка не оголошена в розділі «var» програми; помилка при написанні імені змінної; помилка при написанні імені інструкції (оператора).
Unterminated string (Незавершений рядок)	При записі рядкової константи не поставлені завершальні лапки.
Incompaible types ... and ... (Несумісні типи)	В операторі присвоєння тип виразу не відповідає або не може бути поставлений у відповідність до типу змінної, що одержує значення виразу.
Missing operator or semicolon (Відсутній оператор або крапка з комою)	Не поставлена крапка з комою після інструкції програми.

Якщо синтаксичних помилок у програмі немає, компілятор створить виконавчий файл програми, який пізніше можна буде запустити з системи Windows. Ім’я виконавчого файла таке саме, як і файла проекту, а розширення – «*.exe». Delphi поміщає виконавчий файл в ту папку, де вже перебуває файл проекту.

При виявленні в програмі неточностей, які не є помилками, компілятор виводить підказки та попередження.

Найбільш часто виникають підказки та попередження у вигляді повідомлення про оголошену, але не використану в програмі змінну, наприклад:

- [Hint] Unitl.pas(35): Variable ‘d’ is declared but never used in ‘ TForml. ButtonlClick (повідомлення про оголошену змінну, що не використовується в програмі) (рис. 5.8);
- [Warning] Unitl.pas(39): Variable ‘a’ might not have been initialized (змінній, імовірно, не присвоєно початкове значення).

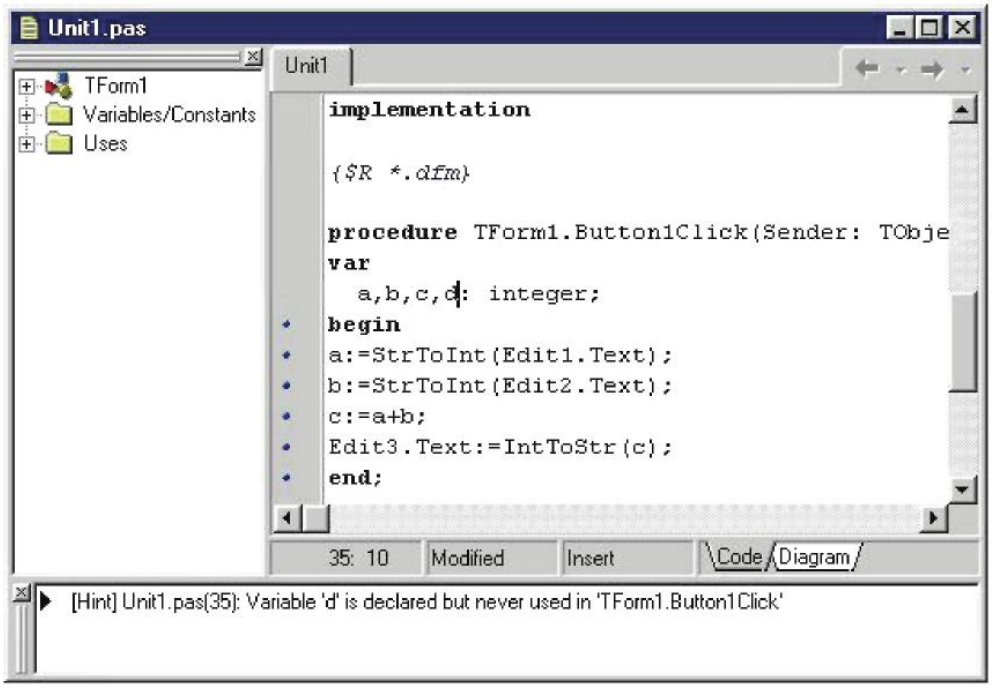


Рисунок 5.8 – Зразок тексту програми

Під час роботи програми можуть виникати помилки, які називаються помилками часу виконання. У більшості випадків причинами є неправильні вхідні дані.

Створену програму можна запустити на виконання в ОС Windows та із середовища Delphi, при цьому повідомлення про одну і ту ж помилку часу виконання, що будуть виводитися на екран, будуть мати різний вигляд.

Наприклад, якщо одне із цілих чисел ввести не як число, а як літеру або як дійсне число, то на екран буде виведено таке вікно з повідомленням про помилку, виконання програми при цьому припиняється (рис. 5.9).

Рисунок 5.9 – Повідомлення про помилку (при виконанні проекту)

Якщо програма запускається з ICP Delphi за допомогою вбудованого налагоджувача, то при виникненні помилки також з’являється вікно з повідомленням про помилку, виконання програми припиняється, але в цьому вікні, крім повідомлення про помилку, вказується тип помилки. Наприклад, у вікні на рис. 5.10 повідомляється про невідповідність сподіваного і одержаного типу даних (в цьому випадку – неможливість перетворення значення «y» в ціле число.

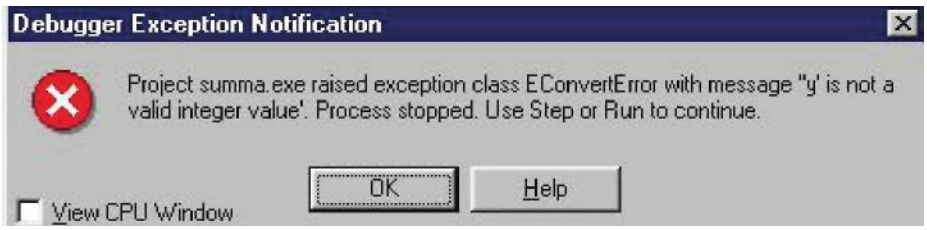


Рисунок 5.10 – Повідомлення про помилку (при компіляції проекту)

Щоб припинити роботу програми, в якій виникла помилка, потрібно виконати команду меню «Run – Program Reset».

При розробленні проекту програміст повинен передбачити всі можливі варіанти некоректних дій користувача, які можуть призвести до виникнення помилок часу виконання і забезпечити способи захисту від них.

5.1.7 Програмування алгоритмів лінійної структури. Оператор присвоювання

В результаті виконання оператора присвоювання змінна одержує значення. Формат оператора присвоювання:

<Ім’я змінної> := <Вираз> ;

де двокрапка й наступний за нею знак рівності – це символ оператора присвоювання. Після кожної інструкції програми ставиться символ «крапка з комою». Приклади:

```
Counter:=0;
d:=b*b-4*a*c;
pi:=3.1415926;
z:=(rl + r2)/(rl*r2);
```

Вираз складається з операндів та операцій, які виконуються над операндами. При обчисленні значень виразів дотримується пріоритет операцій. Операції з більш високим пріоритетом виконуються раніше тих операцій, у яких пріоритет нижчий. Якщо пріоритет операцій однаковий, то операція, що розміщується лівіше, виконується раніше. Для реалізації будь-якого потрібного порядку виконання операцій у виразі можна використовувати круглі дужки.

Вираз, який взято в дужки, трактується як один операнд. Це значить, що операції, взяті в дужки, будуть виконуватися в звичайному порядку, але раніше, ніж операції, які перебувають поза дужками. При записі виразів, що містять дужки, має дотримуватися парність дужок, тобто кількість дужок, що відкриваються, має дорівнювати кількості дужок, які закриваються. Порушення парності дужок – найпоширеніша помилка при записі виразів.

5.1.7.1 Арифметичні вирази

Операндами арифметичних виразів можуть бути числа (константи) та змінні, функції й арифметичні вирази, взяті в круглі дужки. Операції позначають дії, які виконуються над операндами (табл. 5.9).

В найпростішому випадку арифметичний вираз – це константа або змінна. Тип результату залежить від типу операндів. Тип константи, що входить у вираз, визначається видом самої константи. Наприклад, константи 0, 1 і 512 – цілого типу (INTEGER), а константи 1.0, 56.356, 0.01, 3.2e-05 – дійсного типу (REAL).

Таблиця 5.9 – Арифметичні операції мови Pascal-Delphi

Операція	Дія	Вираз, що використовує арифметичні операції	Тип операндів	Тип результату
+	додавання	a + b, a - b, a · b	real	real
–	віднімання		integer	integer
·	множення		integer, real	real
/	ділення	a / b	real	real
			integer	real
			integer, real	real
div	ділення націло (відкидання залишку)	a div b	integer	integer
mod	залишок від ділення	a mod b	integer	integer

Приклади арифметичних виразів:

- a:=123;
- b:=0.001;
- x:=i;
- y:=A+B/3;
- Rezultat:=Summa·0.7 5;
- Bzvazhene:=(B1+B3+B3)/100;
- UCena:=Cena MOD 1000;
- Vyraz:=(Cena DIV 1000)+Cena;
- opory:=(rl+r2+r3)/(rl·r2·r3).

В арифметичних виразах прийнято такі правила послідовності виконання операцій: «·», «/», «div», «mod» мають більш високий пріоритет, ніж «+» і «–».

Співвідношення математичних виразів та їх програмних записів наведено у табл. 5.10.

Таблиця 5.10 – Приклади програмних записів математичних виразів

Вираз	Програмний запис
$ab + cd$	A · B + C · D
$\frac{a+b}{c+d}$	(a + b) / (c+ d)
$a + \frac{cd}{b} + \frac{e}{fg}$	A + C · D / B + E / (F · G) або A + C · D / B + E / F / G

5.1.7.2 Рядкові вирази

Операндом рядкових виразів може бути послідовність символів або символи, взяті в одинарні лапки. Така послідовність називається рядком. Докладніше рядкові вирази та операції з ними будуть розглянуті в одній з наступних лабораторних робіт цього практикуму.

5.1.7.3 Логічні вирази

Результатом виконання логічних виразів є значення «True» або «False». Для запису простих логічних виразів використовуються знаки відношення (=, <, >, <=, >=, <>), а для написання складних – логічні операції (NOT, AND, OR). Докладніше логічні операції та логічні вирази будуть розглянуті в одній з наступних лабораторних робіт цього практикуму.

5.1.8 Виконання оператора присвоювання

Виконання інструкції присвоювання відбувається таким чином: спочатку обчислюється результат виразу, що перебуває праворуч від символа присвоювання, потім обчислене значення записується в ту ділянку пам’яті, що призначена для зберігання змінної, ім’я якої розміщується ліворуч від символа присвоювання (інакше кажучи, обчислене значення присвоюється цій змінній) (табл. 5.11).

Таблиця 5.11 – Варіанти використання оператора присвоєння

Запис	Дія оператора
I:=0	Значення змінної «I» дорівнює нулю.
A:=b+z;	Значенням змінної «A» буде число, що дорівнює сумі значень змінних «b» і «z».
j:=j+1;	Значення змінної «j» збільшується на одиницю.
Label1.Caption:=‘Приклад’;	Задається заголовок напису.
Edit1.Text:=FloatToStr(a);	Задається значення текстового поля.

Оператор присвоювання вважається правильним, якщо тип виразів відповідає або може бути поставлений у відповідність типу змінної. Змінним типу REAL можна присвоїти значення виразів типу REAL або

INTEGER. Змінним типу INTEGER можна присвоїти значення виразів лише типу INTEGER.

Під час трансляції вихідного коду програми у виконавчу програму компілятор перевіряє відповідність типів виразів і змінних. Якщо тип виразу не відповідає типу змінної, то компілятор видає повідомлення про помилку.

5.2 Хід роботи

5.2.1 Покрокова інструкція до виконання лабораторної роботи

Створити програму, яка забезпечує введення двох цілих чисел, обчислює їхню суму та виводить значення результату.

Рекомендації

1. Запустіть ICP Delphi за допомогою команди головного меню Windows «Пуск – Программы – Borland Delphi 6 – Delphi 6».

2. Створіть новий проект за допомогою команди «File – New – Application». У відкритому вікні форми можна розміщувати візуальні компоненти для реалізації проекту програми.

3. Збережіть новий проект за допомогою команди меню «File – Save Project As». У вікні, що з'явилося, натисніть «Save Unit1 As» за допомогою кнопки  . Створіть нову папку для файлів проекту з назвою «Сума чисел» у вказаній викладачем папці. Відкрийте створену папку та натисніть кнопку «Зберегти». Після збереження файла модуля Unit1.pas відкриється вікно «Save Project As». Задайте ім'я файла проекту «Summa» та натисніть кнопку «Зберегти» (рис. 5.11).

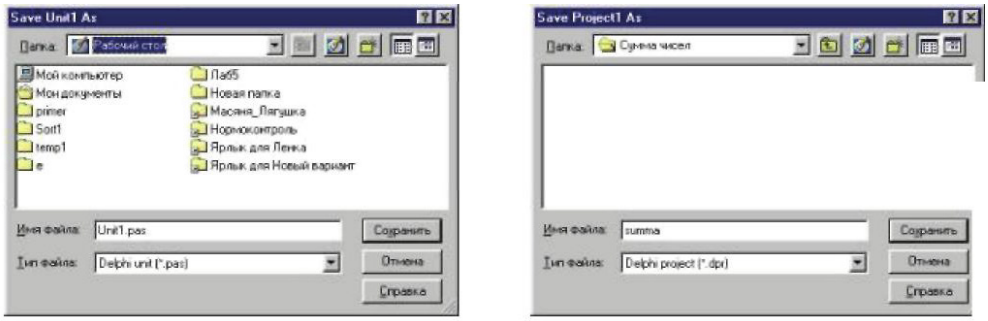
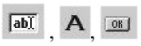
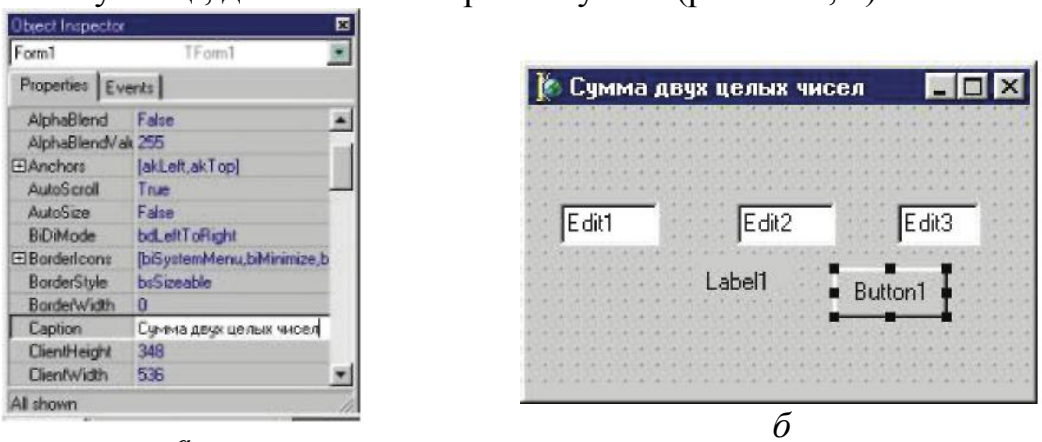


Рисунок 5.11 – Діалог збереження файла проекту

4. Змініть заголовок форми Form1. Для цього у вікні інспектора об'єктів відкрийте сторінку «Властивості» («Properties») інспектора об'єктів («Object Inspector»), виберіть властивість «Caption» (заголовок) і надайте її нове значення – «Сума двох цілих чисел» (див. приклад на рис. 5.12, *а*).

5. Розмістіть на формі компоненти «Edit1», «Edit2», «Edit3», «Label1», «Button1», використовуючи відповідні піктограми  , групи «Standard». Для цього клацніть на вкладці «Standard» палітри компонентів,

потім виберіть піктограму необхідного компонента та клацніть у вікні форми в тому місці, де хочете його розташувати (рис. 5.12, *б*).



а *б*

Рисунок 5.12 – Зміна властивостей і розміщення компонентів на формі

Увага! Виділений компонент форми оточений вісьмома маркерами. Саме його властивості відображаються у вікні інспектора об’єктів. Виділений компонент можна переміщувати по формі, змінюючи його місце розташування. Можна також змінювати його розміри, потягнувши за один з маркерів.

6. Змініть властивості компонента «Label1»: встановіть властивість «Caption» (заголовок) як «+», у списку властивостей «Font» (шрифт) властивості «Size» (розмір) присвойте значення 20.

7. Змініть властивості компонента «Button1»: задайте властивість «Caption» (заголовок) як «=», у списку властивостей «Font» (шрифт) властивості «Size» (розмір) присвойте значення 20.

8. Вирівняйте компоненти на формі. Для цього виділіть їх всі при натиснутій клавіші «Shift», потім у контекстному меню виберіть команду «Position – Align».

9. Змініть властивості компонентів «Edit1», «Edit2», «Edit3»: видаліть текст з властивості «Text».

10. Додайте на форму ще три об’єкти «Labe1», розмістіть їх над об’єктами «Edit1» – «Edit3» та присвойте їхнім властивостям «Caption» значення «Доданок», «Доданок» і «Сума», відповідно.

11. Активізуйте вікно редактора коду, натиснувши клавішу «F12». Переглянете згенерований Delphi модуль опису форми та розміщених на ній компонентів.

12. Збережіть зміни, які внесені в проект, за допомогою команди меню «File – Save All».

13. Відкомпілюйте створений проект командою меню «Project – Compile summa», де summa – це ім’я проекту.

14. За допомогою файлового менеджера перегляньте папку проекту (рис. 5.13):

- Summa.cfg – файл конфігурації проекту;
- Summa.dof – файл параметрів проекту;
- Summa.dpr – файл проекту Delphi;
- Summa.exe – відкомпільований проект, файл програми;
- summa.res – файл ресурсів Windows;
- Unitl.~dfm, unit1.~pas – файли резервних копій;
- Unitl.dcu – відкомпільований модуль;
- Unitl.dfm – двійковий файл форми містить початкові дані компонентів;
- Unitl.pas – вихідний код модуля форми.

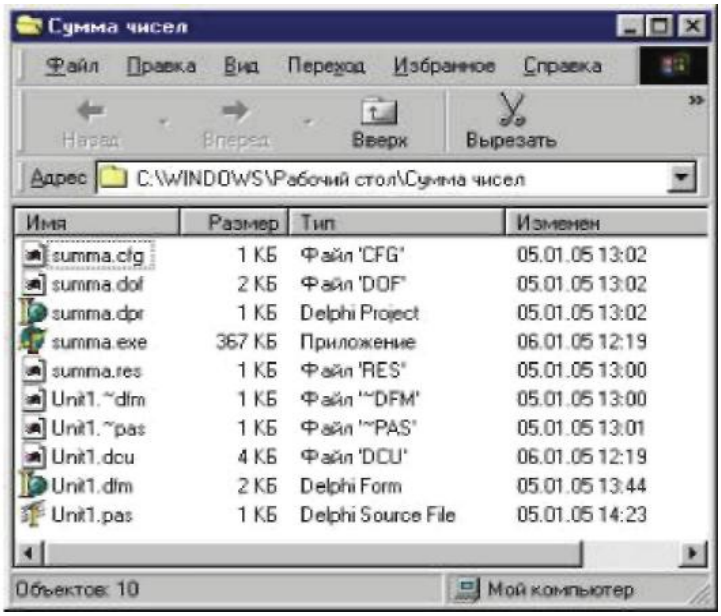


Рисунок 5.13 – Вміст папки проекту

15. Закрийте вікно файлового менеджера, активізуйте вікно Delphi та запустіть проект на виконання командою меню «Run – Run».

16. Розгляньте вікно створеної форми. Зверніть увагу на те, що воно має стандартні атрибути вікна Windows.

17. Завершіть роботу додатка кожним зі стандартних методів.

18. Використайте код обробника подій: при натисканні на кнопку «=» два введених числа мають додаватися.

Для цього виділіть кнопку «Button1», у вікні інспектора об’єктів перейдіть на вкладку «Events», виберіть подію «OnClick» і двічі клацніть лівою кнопкою миші на порожньому полі списку. У вікні редактора коду, яке відкрилося, між операторами «begin» і «end» розмістіть необхідні оператори.

Остаточна процедура обробки події «клік» на кнопці Button1 має виглядати так:

```
procedure TForml.ButtonlClick(Sender: TObject);
var
```

```
a,b,c: integer;                {Оголошення змінних}
begin
a:=StrToInt(Edit1.Text);       {Перетворення текстового рядка на число}
b:=StrToInt(Edit2.Text);
c:=a+b;
Edit3.Text:=IntToStr(c);       {Перетворення цілого числа на рядок}
end;
```

19. Збережіть зміни в проекті.
20. Запустіть програму на виконання.
21. Після перевірки роботи програми, закрийте її.
22. Запустіть програму з ОС Windows, використовуючи виконавчий файл програми.

5.2.2 Приклад структури модуля програми

Створимо таку програму: ввести два дійсних числа, знайти їхню суму та показати результат.

Форма цього додатка буде містити такі компоненти: три текстових вікна «TEdit» для введення даних та виведення результату, одну мітку «TLabel» для відображення у вікні знака «+» і дві кнопки «TButton», одна – для реалізації завдання, інша – для завершення роботи програми (рис. 5.14).

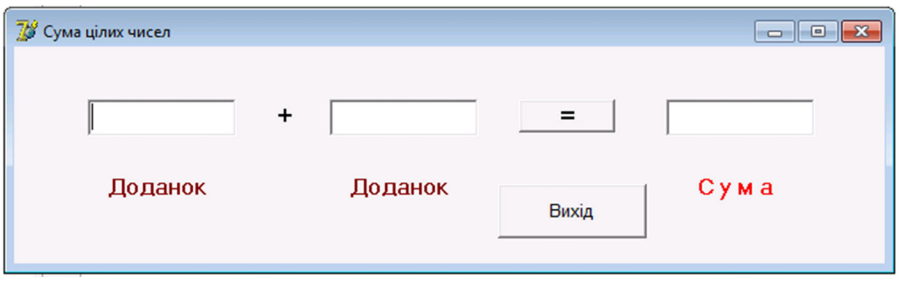


Рисунок 5.14 – Зразок виконання програми (вікно головної форми)

Модуль цієї програми практично весь генерується автоматично Delphi в процесі її проектування (табл. 5.12). Користувач лише вводить свій код для обробки події, що настає при натисканні кнопки (в тексті модуля він виділений жирним шрифтом).

Таблиця 5.12 – Структура модуля програми

Розділ програми	Зміст розділу програми
unit Unitl;	Модуль починається з ключового слова «unit», після якого вказується ім’я модуля, що збігається з ім’ям файла, в якому збережений даний модуль.
interface	Відкритий інтерфейс модуля.

Продовження таблиці 5.12

Розділ програми	Зміст розділу програми
uses Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls; type	Список бібліотек, що підключаються до програми.
TForml = class(TForm) Editl:Edit; Edit2: TEdit; Edit3: TEdit; Labell: TLabel; Buttonl: TButton; Button2: TButton;	Оголошення класу форми, а також типів всіх компонентів, що входять до класу форми.
procedure ButtonlClick(); procedure Button2Click();	Оголошення процедур, що описують дії, які мають відбуватися при натисканні кнопок.
private {Private declarations}	Закритий розділ класу. Оголошення змінних, функцій і процедур, що входять до класу форми, але недоступні для інших модулів.
public {Public declarations} end;	Відкритий розділ класу. Оголошення змінних, функцій і процедур, що входять до класу форми і доступні для інших модулів.
var Forml: TForml;	Оголошення типів, констант, змінних, функцій і процедур, до яких буде доступ з інших модулів, але які не входять до класу форми.
implementation	Реалізація модуля. Тут можуть міститися пропозиції «uses», оголошення типів, констант, змінних, до яких не буде доступу з інших модулів. Тут автоматично розташовуються всі реалізації процедур і функцій, оголошені в розділі «Interface».
{$R*.dfm}	Директива компілятора, що забезпечує компонування файлів ресурсів форми. Видаляти її з тексту модуля не можна, оскільки модуль не буде створений.

Продовження таблиці 5.12

Розділ програми	Зміст розділу програми

Procedure TForml.ButtonlClick();
Var a,b,c: real;
begin
a:=strtofloat(editl.Text);
b:=strtofloat(edit2.Text);
c:=a+b;
Edit3.Text:=floattostr(c);
end;
	Реалізація процедури дії, що реалізовує обробку події «Click» на кнопці «Button1», в результаті виконання якої додаються два введених числа, а отримана сума виводиться на екран.

Procedure
TForml .Button2Click();
begin
Close;
end;
	Реалізація процедури дії, що реалізовує обробку події на кнопці «Button2», в результаті виконання якої завершується робота програми.

end.	

5.3 Контрольні питання

1. В чому полягає основна відмінність процедурного та об’єктно-орієнтованого програмування?
2. Назвіть основні принципи об’єктно-орієнтованого програмування.
3. Що таке об’єкт?
4. Що таке інкапсуляція та для чого вона застосовується?
5. В чому полягає поліморфізм операцій?
6. Що таке метод?
7. Що таке властивість об’єкта, яким чином її можна змінювати?
8. Що таке події? Яке призначення обробника подій?
9. Яке призначення повідомлень?
10. В чому полягають переваги візуального програмування інтерфейсу?
11. Опишіть структуру та призначення окремих елементів головної програми ICP Delphi.
12. Яке призначення модуля в проекті Delphi? Опишіть призначення окремих розділів модуля.
13. Чим відрізняється доступність об’єктів, описаних у розділах «interface» та «implementation»?
14. Як вказати посилання на властивості та методи об’єкта в тексті програми?
15. Які компоненти входять до інтегрованого середовища розробки програм Delphi?
16. Перелічіть основні компоненти вікна середовища Delphi та вкажіть їхнє призначення.
17. Як одержати підказку Delphi щодо елементів вікна та компонентів графічного інтерфейсу?

18. Яке призначення сторінок «Properties» та «Events» у вікні «Object Inspector»?

19. Як розмістити компонент на формі?

20. Якими способами можна змінювати властивості компонента? Наведіть приклади.

21. Перелічіть зміст проекту Delphi. Опишіть призначення різних файлів.

22. Яке призначення обробника подій? Яким чином можна ініціювати створення процедури-обробника події?

23. Опишіть порядок одержання довідки про синтаксис мови Object Pascal в ICP Delphi.

24. Вивчіть і опишіть інформацію про компоненти палітри «Standard» за допомогою стандартної довідки Delphi.

25. Вивчіть і опишіть інформацію про числові типи даних мови Object Pascal за допомогою стандартної довідки Delphi.

26. Вивчіть і опишіть інформацію про арифметичні операції мови Object Pascal за допомогою стандартної довідки Delphi.

27. Вивчіть і опишіть інформацію про функцію «Random» мови Object Pascal за допомогою стандартної довідки Delphi.

28. Що зберігається в файлі проекту з розширенням «*.dfm»?

29. Що зберігається в файлі проекту з розширенням «*.pas»?

30. Що зберігається в файлі проекту з розширенням «*.dpr»?

31. Що зберігається в файлі проекту з розширенням «*.res»?

ЛАБОРАТОРНА РОБОТА № 6
ПРОГРАМУВАННЯ АЛГОРИТМІВ РОЗГАЛУЖЕНОЇ СТРУКТУРИ

Мета: набути навичок роботи з операторами «If...Then...Else» та «Case».

6.1 Теоретичні відомості

Оператори в програмі-обробнику подій виконуються в тій послідовності, в якій вони записані. Однак досить часто потрібно змінити порядок виконання операторів, залежно від виконання (або невиконання) певної умови. Існують керівні конструкції, призначені для керування порядком виконання операторів. Підставою для прийняття рішень у керівних операторах є істинність або хибність умовного або логічного виразу.

Умовні вирази – це такі вирази, які повертають одне із двох значень «True» (істина) або «False» (хибність). Прості логічні вирази містять відносні операції (операції порівняння): = (дорівнює), > (більше), < (менше), <> (не дорівнює), >= (більше або дорівнює), <= (менше або дорівнює).

Логічні вирази дозволяють здійснювати операції над змінними логічного типу (табл. 6.1).

Таблиця 6.1 – Логічні операції (A і B) в Delphi

Операція	Позначення	Тлумачення
Заперечення (інверсія) (логічне НЕ)	NOT A	Не A; Неправильно, що A
Кон'юнкція (логічний добуток, логічне І)	A AND B	A і B; A, але B; A, а B; як A, так і B; A разом з B; A в той час, як B
Диз'юнкція (логічне додавання, логічне АБО)	A OR B	A або B; A або B, або обидва
Диз'юнкція (виключне АБО)	A XOR B	A або B; A або B, але не обидва

Таблицю істинності для основних логічних операцій наведено у табл. 6.2.

Таблиця 6.2 – Таблиця істинності для основних логічних операцій

A	B	NOT A	A AND B	A OR B	A XOR B
False	False	True	False	False	False
False	True	True	False	True	True
True	False	False	False	True	True
True	True	False	True	True	False

Пріоритети виконання логічних операцій:

1. Заперечення (NOT).
2. Логічний добуток (AND).
3. Логічне додавання (OR), виключне або (XOR).

Дужки змінюють порядок виконання операцій.

6.1.1 Оператор умови «If...Then...Else»

Умовні оператори призначені для вибору на виконання однієї з можливих дій (операторів) залежно від деякої умови, при цьому одна з дій може бути відсутня.

Вибір дії, залежно від виконання умови, може бути реалізований за допомогою оператора «If».

Розрізняють два типи умовних операторів: «If...Then» і «If…Then...Else». Конструкція «If...Then» застосовується тоді, коли необхідно виконати певні дії, однак тільки в тому випадку, якщо значення деякої умови дорівнює «істина».

Синтаксис оператора «If...Then»:

If < умова > Then < інструкція для обробки умови >

Спочатку обчислюється значення умови (вираз логічного типу). У випадку, якщо значення умови дорівнює «істина», виконується інструкція, що розміщується за словом «Then».

Якщо в результаті виконання умовного оператора мають виконуватися 2, 3 або більше інструкцій, то необхідно використати складений оператор.

Пари «Begin...End» часто називають операторними дужками, тому в програмі для наочності їх записують на одному рівні («End» під відповідним йому «Begin»). Всередині одного складеного оператора може міститися інший складений оператор.

Для реалізації двох можливих варіантів послідовностей виконання операторів у коді програми використовується оператор «If...Then...Else».

Синтаксис оператора «If...Then...Else»:

If
< умова >
Then
< Інструкція для обробки умови >
Else
< Інструкція для обробки хибної умови >

Оператор «If...Then...Else» виконується в такий спосіб:

1. Обчислюється значення умови (вираз логічного типу).

2. Якщо значення умови дорівнює «істина», то виконується інструкція, зазначена за словом «Then». Якщо значення умови дорівнює «хибність», то виконується інструкція, записана за словом «Else».

6.1.2 Оператор «Case»

Оператор «Case» дозволяє реалізувати множинний вибір (більше двох варіантів розвитку подій). Перехід виконується на одну із гілок (інструкцій), залежно від значення заданого виразу.

Оператор «Case» існує також у двох варіантах:

1. Case k of
A1: < інструкція 1 >;
A2: < інструкція 2 >;
A*N*: < інструкція *N*>;
End;

2. Case k of
Al: < інструкція 1 >;
A2: < інструкція 2 >;
A*N*: < інструкція *N* >
Else
< інструкція, яка виконується у випадку, якщо значення виразу не потрапило ні в один зі списків констант Al, A2, ..., A*N* >
End;

У наведених прикладах k – вираз-селектор, від його значення залежить подальший хід програми, він може мати тільки простий порядковий тип (цілий, символьний, логічний).

Список констант (A1, ..., A*N*) – константи того ж типу, що й селектор, які виконують роль міток-гілок. Якщо константи – це діапазон чисел, то замість списку можна вказати першу та останню константу діапазону, розділивши їх двома послідовними крапками.

Виконання оператора починається з обчислення виразу, отримане значення порівнюється з константами (мітками), в результаті чого виконується відповідний оператор. Наприклад: для знаходження найбільшого із двох неоднакових дійсних чисел, використовується такий оператор:

```
var
Max, a, b: real;
begin
a:=strtofloat(editl.Text);
b:=strtofloat(edit2.Text);
case   a > b   of
true:      Max:=a;
false:     Max:=b;
end;
edit3.text:= floattostr(Max);
```

6.1.3 Компоненти «RadioGroup» та «CheckBox»

Компонент «RadioGroup» (виключний перемикач) з групи «Standard» дозволяє відображати поля з обмеженою множиною значень (табл. 6.3).

Таблиця 6.3 – Основні властивості компонента «RadioGroup»

Властивість	Функціональне призначення
Caption	Задає назву групи перемикачів.
Items	Визначається кількість перемикачів у групі та написи біля них.
Itemlndex	Задає номер кнопки, обраної за замовчуванням: 0 – перша, –1 – жодна кнопка не обрана.

Компонент «CheckBox» (кнопка з незалежною фіксацією – прапорець Windows) дозволяє користувачу вибрати/скасувати певну опцію. Стан кнопки міститься у властивості «Checked» (табл. 6.4). Компонент належить до групи «Standard».

Таблиця 6.4 – Основні властивості компонента «CheckBox»

Властивість	Функціональне призначення
Caption	Задає текст, що супроводжує кнопку.
AllowGrayed	Задає наявність у кнопки третього стану. Якщо значення цієї властивості False, то кнопка може перебуває у двох станах – увімкненому або вимкненому. Якщо значення властивості дорівнює True, то додається третій стан, коли кнопка неактивна.
Checked	Повертає або задає наявність «галочки» прапорця. True – кнопка увімкнена, False – вимкнена.

6.2 Хід роботи

6.2.1 Завдання до лабораторної роботи

6.2.1.1 Написати програму «Калькулятор», використовуючи оператори If...Then...Else та Case (табл. 6.5).

Таблиця 6.5 – Індивідуальні завдання на лабораторну роботу

Завдання
1. Виводити повідомлення в тому випадку, коли результат додавання більший, ніж будь-який операнд.
2. Виводити повідомлення, коли один з операндів від’ємний.
3. Виводити повідомлення, коли результат множення дорівнює будь-якому операнду.
4. Виводити повідомлення, коли перший операнд більший, ніж другий.
5. Виводити повідомлення, коли обидва операнди однакові.

Продовження таблиці 6.5

Завдання
6. Виводити повідомлення, коли результат множення і додавання збігаються.
7. Виводити повідомлення, коли квадрат першого операнда менший або дорівнює значенню другого операнда.
8. Виводити повідомлення, коли результат віднімання операндів менший, ніж результат ділення.
9. Виводити відповідні повідомлення в тому випадку, коли результат віднімання менший нуля, дорівнює нулю чи більший нуля.
10. Виводити повідомлення, коли обидва операнди додатні.
11. Виводити повідомлення про попередньо виконану арифметичну дію.
12. Виводити повідомлення про відношення першого операнда до результату.
13. Виводити повідомлення, коли значення обох операндів і результату арифметичної дії збігаються.
14. Виводити повідомлення, коли операнди діляться націло.
15. Виводити повідомлення, коли другий операнд більший і в той же час дорівнює нулю.

6.2.1.2 Програма підрахунку повинна виглядати як на рис. 6.1.

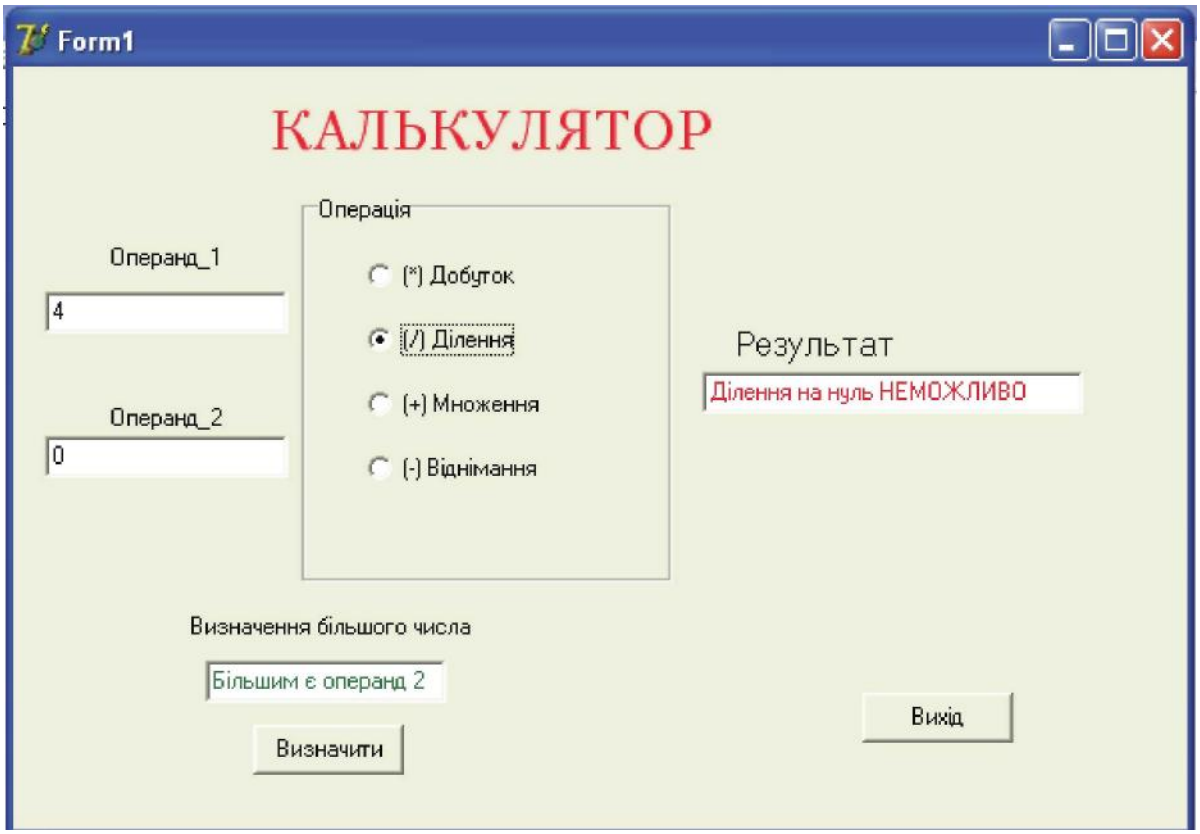


Рисунок 6.1 – Зовнішній вигляд вікна програми

6.2.2 Лістинг програми

```
unit CALCULATOR;

interface

uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, ExtCtrls;

type TForml = class(TForm)
Edit 1: TEdit;
Edit2: TEdit;
Edit3: TEdit;
RadioGroupl: TRadioGroup;
RadioButton6: TRadioButton;
RadioButton7: TRadioButton;
RadioButtonl: TRadioButton;
RadioButton2: TRadioButton;
Button 1: TButton;
Label 1: TLabel;
Label2: TLabel;
Edit4: TEdit;
Label3: TLabel;
Label4: TLabel;
Button2: TButton;
Label5: TLabel;
procedure RadioButton6Click(Sender: TObject);
procedure RadioButton7Click(Sender: TObject);
procedure RadioButtonlClick(Sender: TObject);
procedure ButtonlClick(Sender: TObject);
procedure RadioButton2Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
private
  { Private declarations }
  public
  {Public declarations }
  end;

  var
 Forml: TForml;

 implementation
```

```
{$R*.dfm}

{Множення}
procedure TForml.RadioButton6Click(Sender: TObject);
var
a,b,c : real;
begin
a:= Strtofloat(Editl.Text);
b:= Strtofloat(Edit2.Text);
  c:=a*b;
  if c>0 then begin
Edit3.Font.Color:=clbrue; // заміна кольору шрифту
Edit3.Text:=floattostr (c);
 end else
begin
Edit3.Font.Color:=clred; // заміна кольору шрифту
Edit3.Text:=floattostr (c);
   end;
   end;

{ділення}
procedure TForml.RadioButton7Click(Sender: TObject);
var a,b,c : real;
begin
a:= Strtofloat(Editl.Text);
  b:= Strtofloat(Edit2.Text);
  if b<>0 then
begin
c:=a/b;
Edit3.Text:=floattostr (c);
 end else
begin
Edit3.Font.Color:=clRed; // заміна кольору шрифту
Edit3.Width:=190; // заміна довжини рядка
Edit3.Text:= 'Ділення на нуль НЕМОЖЛИВО';
end;
end;

{додавання}
procedure TForml.RadioButtonlClick(Sender: TObject);
var
a,b,c : real; begin
a:= Strtofloat(Editl.Text);
b:= Strtofloat(Edit2.Text);
```

```
c:=a+b;
 ifc>0then begin
Edit3.Font.Color:=clblue; // заміна кольору шрифту
Edit3.Text:=floattostr (c);
  end else
begin
Edit3.Font.Color:=clred; // заміна кольору шрифту Edit3.Text:=floattostr (c);
end;
   Edit3.Text:=floattostr (c);
   end;

   {вихід}
procedure TForml.ButtonlClick(Sender: TObject);
begin
close;
end;

{віднімання}
procedure TForml.RadioButton2Click(Sender: TObject);
var a,b,c : real;
begin
a:= Strtofloat(Editl.Text);
b:= Strtofloat(Edit2.Text);
c:=a-b;
 if c>0 then begin
Edit3.Font.Color:=clblue; // заміна кольору шрифту
Edit3.Text:=floattostr (c);
  end else
begin
Edit3.Font.Color:=clred; // заміна кольору шрифту
Edit3.Text:=floattostr (c);
end;
   Edit3.Text:=floattostr (c);
   end;

procedure TForml.Button2Click(Sender: TObject);
var
 a,b,Max : real;
 begin
a:= Strtofloat(Editl.Text);
b:= Strtofloat(Edit2.Text);
case a > b of
true: Max:=a;
False: Max:=b;
```

```
end;
edit4.Font.Color:=clgreen; // заміна кольору шрифту
Edit4.Width:=120; // заміна довжини рядка
If Max=b then
begin
  Edit4.Text:= ‘Більшим є операнд 2’;
  end else
  Edit4.Text:= ‘Більшим є операнд 1;
  end;
  end.
```

6.3 Контрольні питання

1. За допомогою якого оператора реалізовується алгоритмічна структура «розвилка»? Нарисуйте її блок-схему.
2. За допомогою якого оператора реалізовується алгоритмічна структура «Вибір»? Нарисуйте її блок-схему.
3. Перерахуйте знаки відношень.
4. Перерахуйте основні логічні операції.
5. Коли застосовується умовний оператор?
6. Назвіть два види умовного оператора.
7. Що дозволяє робити оператор вибору?
8. Що таке селектор?
9. Наведіть приклад використання оператора умови.
10. Наведіть приклад використання оператора вибору.
11. Для чого слугує кнопка з незалежною фіксацією?
12. Опишіть властивість «Checked».
13. Вивчіть й опишіть інформацію про компонент «RadioButton» палітри «Standard» за допомогою стандартної довідки.
14. У чому розходження між об’єктами «Checkbox» і «RadioButton»?
15. Яке призначення об’єкта «RadioGroup»?

ЛАБОРАТОРНА РОБОТА № 7
РОБОТА З ОДНОВИМІРНИМИ (ЛІНІЙНИМИ) МАСИВАМИ

Мета: набути навичок роботи з лінійними (одновимірними) масивами.

7.1 Теоретичні відомості

Багато завдань, які вирішуються за допомогою комп'ютера, пов'язані з обробкою великих обсягів інформації, що є сукупністю даних, об'єднаних єдиним математичним змістом або пов'язаних між собою за змістом. Такі дані зручно зображати у вигляді лінійних або прямокутних таблиць.

У лінійній таблиці кожному її елементу відповідає порядковий номер. Для елемента прямокутної таблиці мають бути зазначені два номери: номер по вертикалі (номер рядка) і номер по горизонталі (номер стовпця).

У вищій математиці табличні величини називають відповідно векторами й матрицями.

У програмі для подання таких даних використовуються масиви.

Масив (array) – це впорядкована сукупність однотипних даних, з кожним із яких пов'язаний впорядкований набір цілих чисел, які називаються *індексами* (index). Масив характеризується ім'ям, розмірністю та розміром.

Ім'я масиву утворюється відповідно до загального правила утворення імен, тобто являє собою ідентифікатор, наприклад: A, B1, C8 і т. д. Однак воно не може збігатися з іменем ні однієї простої змінної, що використовується в тій самій програмі.

Робота з масивом зводиться до дій над його елементами. Для того, щоб вказати, який елемент у цей момент використовується, досить задати його порядковий номер, що приписується до імені відповідного масиву. Таким чином, елементи масиву позначаються змінними з індексами. Запис змінної з індексами складається з імені масиву та наступного за ним, у квадратних дужках, списку індексів, наприклад: A[1], A[i], B1[K], C8[I,J], C8[2,1].

Індекси визначають положення елемента в масиві. Число індексів визначає розмірність масиву, тобто форму його компонування: одновимірний, двовимірний і т. д. Одновимірний масив відповідає лінійній таблиці. Його елемент позначається змінною з одним індексом: A[1], A[i] – відповідно перший та і-й елементи одновимірного масиву A.

Двовимірний масив описує в програмі прямокутну таблицю. Його елементи позначаються змінною з двома індексами: C8[I,J], C8[2,1], де перший індекс позначає номер рядка, а другий – номер стовпця.

Таким чином, для звертання до конкретного елемента масиву необхідно вказати ім'я масиву та значення індексів.

Для запису елементів масиву у пам'ять комп'ютера потрібно виділити для їхнього зберігання необхідну кількість (масив) комірок пам'яті, що

визначається розміром масиву. Розміри масиву задаються границями зміни індексів кожного з вимірів (мінімальне й максимальне значення індексу).

За замовчуванням застосовується нумерація з нульовою базою, тобто елементи масиву нумеруються, починаючи з 0.

У програмі для кожного масиву мають зазначатися його параметри: ім'я, розмірність і розміри. Ця інформація потрібна для резервування необхідного обсягу пам'яті задля зберігання числових значень; вона задається спеціальним оператором опису масивів.

Опис статичного масиву визначає ім'я, розмір масиву й тип даних, які в ньому зберігаються. Формат опису в розділі змінних:

Var <ім'я_масиву>: array <[тип_індексу]> of <тип_даних>;

Найчастіше як тип індексу використовується інтервальний цілий тип (тип-діапазон). Інтервальний тип задається початковим і кінцевим значеннями, які розділяються двома крапками. Наприклад

Var A : array [1..10] of real;

Описується одновимірний масив чисел A, що максимально може складатися з 10 елементів. Нижня границя індексу дорівнює 1, верхня – 10.

Починаючи з версії Delphi 4, можна використовувати також і динамічні масиви – коли кількість елементів може змінюватись під час виконання програми.

Динамічні масиви відрізняються від звичайних статичних тим, що для них заздалегідь не оголошується довжина – число елементів. Оголошення такого масиву містить тільки ім'я та тип елементів.

Var <ім'я_масиву>: array of <тип_даних>;

При оголошенні динамічного масиву місце під нього не виділяється. Перш ніж використати такий масив, необхідно задати в програмі його розмір процедурою «SetLength». Параметри такої процедури – кількість елементів за кожною розмірністю. Наприклад, SetLength(10,20) – для двовимірного масиву.

Лінійний (одновимірний) масив – це просто список елементів даних. Приклади опису одновимірних масивів:

```
Var
B : array [0..5] of real;
R : array [1..34] of char;
N : array ['A'..'Z'] of integer;
M : array of integer; {динамічний масив}
```

Для доступу до елемента масиву потрібно вказати ім’я масиву з числом (індексом), взятим у квадратні дужки.

Елементи масиву можна використати в будь-якому виразі, так само, як і значення константи або змінної. Наприклад:

```
a[0]=11.2;
a[1]=10.2;
a[3]=22.1;
a[4]=1.1;
Y = a[0] *2 – a[1];
```

Під час виконання лабораторної роботи студенти повинні навчитися здійснювати формування масиву, обчислення суми, добутку, кількості елементів та середнього арифметичного елементів масиву, визначення максимального й мінімального елементів масиву (рис. 7.1).

Компоненти додатка

«Edit»
«Label»
«CheckBox»
«Button»
«GroupBox»

Загальний вигляд вікна програми

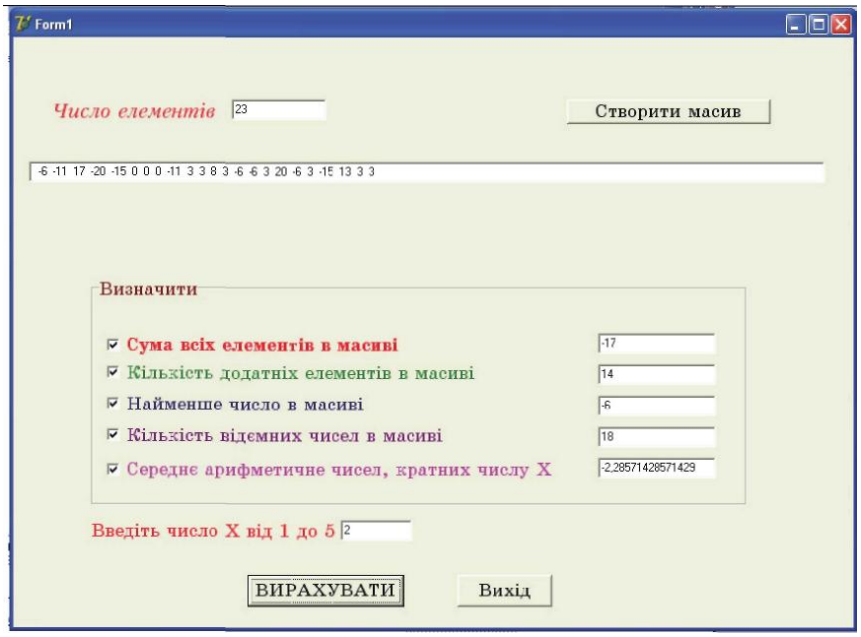


Рисунок 7.1 – Загальний вигляд вікна програми

Компонент «GroupBox» групи «Standard» – це контейнер з рамкою та написом, що поєднує групу пов’язаних органів керування: перемикачів «RadioButton», прапорців «CheckBox» і т. ін.

Основна властивість панелі «GroupBox» – «Caption» – задає напис для рамки, яка виділяє групу об’єднаних компонентів.

Якщо компоненти, розташовувані на панелі, виявляються під панеллю та не відображаються, то варто виділити панель і вибрати в контекстному меню команду «Control → Send to Back» («Порядок → На задній план»).

7.2 Хід роботи

7.2.1 Програмний код додатка

Індивідуальні завдання до лабораторної роботи наведено у табл. 7.1.

Зверніть увагу, що розмір масиву N і сам масив M описані в розділі оголошення типів, констант, змінних, функцій і процедур, доступному для всіх модулів програми тому, що ці змінні будуть використовуватись у різних подіях-процедурах: у процедурі заповнення масиву випадковими числами TForml.ButtonlClick, а також у процедурі обробки масиву TForml. Button2Click.

Таблиця 7.1 – Індивідуальні завдання на лабораторну роботу

Завдання
1. Виводити повідомлення в тому випадку, якщо число елементів в масиві більше або дорівнює 10.
2. Виводити повідомлення, яке відображає кількість елементів в масиві.
3. Виводити повідомлення в тому випадку, якщо сума елементів більша, ніж кількість додатних чисел в масиві.
4. Виводити повідомлення в тому випадку, якщо сума елементів більша за число елементів в масиві.
5. Виводити повідомлення в тому випадку, якщо кількість від'ємних чисел не дорівнює найменшому числу масиву.
6. Виводити повідомлення в тому випадку, якщо найменше число в масиві – додатне.
7. Виводити повідомлення в тому випадку, якщо кількість від'ємних чисел в масиві дорівнює 0.
8. Виводити повідомлення в тому випадку, якщо кількість додатних чисел дорівнює кількості від'ємних чисел.
9. Виводити повідомлення в тому випадку, якщо довжина масиву менша за 10.
10. Виводити повідомлення в тому випадку, якщо найменше число в масиві дорівнює числу елементів.
11. Виводити повідомлення в тому випадку, якщо середнє арифметичне не дорівнює 0.
12. Виводити повідомлення в тому випадку, якщо сума елементів менша за 0.
13. Виводити повідомлення в тому випадку, якщо кількість від'ємних чисел менша 5.
14. Виводити повідомлення про найменше число в масиві.
15. Виводити повідомлення про стан числа X.

Всі підзадачі вирішуються після одного перегляду елементів масиву.

Для прикладу розглянемо пошук мінімального елемента в масиві. Спочатку встановлюється поточний мінімум за нульовим елементом масиву min := M[0]. Потім починається перегляд елементів масиву: вибирається черговий елемент M[i] і порівнюється з min. Якщо елемент M[i] менший поточного min, то виконується переприсвоювання min := M[i].

7.2.2 Лістинг програми

```
unit MASIV;
interface
uses
   Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
   Dialogs, StdCtrls, ExtCtrls;
type
   TForml = class(TForm)
   Labell: TLabel;
   Editl: TEdit;
   Buttonl: TButton;
   Edit2: TEdit;
   RadioGroupl: TRadioGroup;
   CheckBoxl: TCheckBox;
   CheckBox2: TCheckBox;
   CheckBox3: TCheckBox;
   CheckBox4: TCheckBox;
   CheckBox5: TCheckBox;
   Edit3: TEdit;
   Edit4: TEdit;
   Edit5: TEdit;
   Edit6: TEdit;
   Edit7: TEdit;
   Button2: TButton;
   Button3: TButton;
   Edit8: TEdit;
   Label2: TLabel;
      procedure ButtonlClick(Sender: TObject);
      procedure Button3Click(Sender: TObject);
      procedure Button2Click(Sender: TObject);
      private
       { Private declarations }
      public
       { Public declarations }
      end;
var
   Forml: TForml;
```

```
    N: integer;
    M: array of integer; {опис динамічного масиву цілих чисел}
implementation

 {$R *.dfm}

procedure TForml.ButtonlClick(Sender: TObject);
Var i: integer;
begin
Randomize;
N:=StrToInt (Editl. text) ; {число елементів масиву}
SetLength (M,N); {задати масиву M довжину N}
Edit2.text:=‘‘;
For I :=0 to N-l do {записати масив випадковими числами}
    begin
    M[i]:=Round(Sin(Random(20))*20); {присвоїти і-му елементу масиву
                                          випадкове число від -20,20}
    Edit2.Text:=Edit2.Text+’  ‘+IntToStr(M[i]); {вивести елементи масиву}
    end;
    end;

procedure TForml.Button3Click(Sender: TObject);
begin
close;
end;

procedure TForml.Button2Click(Sender: TObject);
var i:Integer;
Suml, X, Kol, Sum, Min, Koldod, Kolvid, CountP: integer;
begin
X:=StrToInt(Edit8.text) ;
begin
If CheckBox3.Checked Then Min:=M[0]; {нехай мінімальний елемент
                                          перший елемент масиву}
Edit3.Text:=‘‘;
 Edit4.Text:=‘‘;
Edit5.Text:=‘‘;
Edit 6.Text: = ‘‘;
Edit7.Text:=‘‘;
Suml:=0; Sum:=0; Koldod:=0; Kolvid:=0; CountP:=0;
For i:=0 to N-l do begin if CheckBoxl.Checked then {визначити суму}
    Sum:=Sum+M[i]; if CheckBox2.Checked then {визначити кількість
                                          додатних чисел}
```

```
    if M[i]>=0 Then Koldod:=Koldod+l; if CheckBox3.Checked then
                                                    {визначити
                                                    мінімальний
                                                    елемент}
    if M[i]<0 Then Kolvid:=Kolvid+l; if CheckBox4.Checked then {визначити
                                                кількість від'ємних
                                                чисел}
   if M[i]<0 Then Kolvid:=Kolvid+l; if CheckBox5.Checked then {визначити
                                кількість і суму елементів масиву, кратних 5}
    If (M[i] mod X = 0) and (M[i]<>0) then
          begin
          CountP:=CountP+M[i];
          Suml:=Suml+l;
          end;
    end;
 {виведення результатів обробки масиву}
if CheckBoxl.Checked then Edit3.Text:=IntToStr(Sum);
if CheckBox2.Checked then Edit4.Text:=IntToStr(Koldod) ;
if CheckBox3.Checked then Edit5.Text:=IntToStr(Min);
if CheckBox4.Checked then Edit6.Text:=IntToStr(Kolvid);
if CheckBox5.Checked then Edit7.Text:=FloatToStr(CountP/Suml);
end;
end;
end.
```

7.3 Контрольні питання

1. Що таке масив даних?
2. Як і в якому розділі програмного коду описуються масиви?
3. Як визначити місце розташування елемента в масиві?
4. Що таке індекс? Які вимоги він має задовольняти?
5. Як здійснюється доступ до елементів масиву?
6. Що таке динамічний масив?
7. Опишіть відмінності статичних і динамічних масивів. Який порядок опису та застосування динамічних масивів?
8. Коли та для чого використовується процедура «SetLength»?
9. Яке призначення компонента «GroupBox»? Як задати напис у цьому компоненті?
10. Чим відрізняються компоненти «RadioGroup» і «GroupBox»? Як задати список елементів в об'єкті «RadioGroup»?

ЛАБОРАТОРНА РОБОТА № 8
РОБОТА З БАГАТОВИМІРНИМИ МАСИВАМИ

Мета: набути навичок роботи з багатовимірними масивами.

8.1 Теоретичні відомості

Масив – це впорядкована сукупність однотипних даних, з кожним елементом яких пов'язаний упорядкований набір цілих чисел, що називаються індексами. Масив характеризується ім'ям, розмірністю й розміром.

Ім'я масиву утворюється за загальним правилом утворення імен, тобто є ідентифікатором, наприклад A, Bl, C8 і т. д. Однак воно не може збігатися з іменем жодної простої змінної, що використовується в тій самій програмі.

Робота з масивом зводиться до дій над його елементами. Для того, щоб вказати, який елемент у цей момент використовується, досить задати його порядковий номер, що приписується до імені відповідного масиву. Таким чином, елементи масиву позначаються змінною з індексами. Позначення змінної з індексами складається з імені масиву та списку індексів у квадратних дужках, наприклад: A[1], A[i], B1[k], C8[i, j], C8[2, 1].

Індекси визначають положення елемента в масиві. Число індексів визначає розмірність масиву, тобто форму його компонування: одновимірний, двовимірний і т. д. Одновимірний масив відповідає лінійній таблиці. Його елемент позначається змінною з одним індексом: A[l], A[i] – відповідно перший та і-й елементи одновимірного масиву A.

Двовимірний масив описує в програмі прямокутну таблицю. Його елементи позначаються змінною з двома індексами: C8[I, J], C8[2, 1], де перший індекс позначає номер рядка, а другий – номер стовпця.

Таким чином, для звертання до конкретного елемента масиву необхідно вказати ім'я масиву та значення індексів.

Для запису елементів масиву в пам'ять комп'ютера потрібно виділити для їхнього зберігання необхідну кількість (масив) комірок пам'яті, що визначається розміром масиву. Розміри масиву задаються границями зміни індексів кожного з вимірів (мінімальне й максимальне значення індексу).

За замовчуванням застосовується нумерація з нульовою базою, тобто елементи масиву нумеруються, починаючи з нуля.

У програмі для кожного масиву мають бути зазначені його параметри: ім'я, розмірність і розміри. Ця інформація потрібна для резервування необхідного обсягу пам'яті для зберігання числових значень; вона задається спеціальним оператором опису масивів.

Часто буває необхідно зобразити таблиці даних у форматі рядків і стовпців. Для цього використовують багатовимірні масиви.

Доступ до елементів, наприклад, двовимірного масиву – *матриці* (matrix) здійснюється за допомогою двох індексів. Перший індекс вказує на рядок, другий – на стовпець: A[1,3], A[0,4] і т. д. (рис. 8.1).

a_{11}	a_{12}	a_{13}	a_{14}
a_{21}	a_{22}	a_{23}	a_{24}
a_{31}	a_{32}	a_{33}	a_{34}
a_{41}	a_{42}	a_{43}	a_{44}

Рисунок 8.1 – Приклад двомірного масиву

Формат опису двовимірного масиву в розділі оголошення змінних:

Var <ім'я_масиву>: array <[тип_індексу,тип_індексу]> of <тип_даних>;

Наприклад:

```
Var A : array [1..10, 1..10] of real;
```

Описується двовимірний масив дійсних чисел A, що максимально може складатися з 10 рядків і 10 стовпців. Нижня границя індексів по обох вимірах дорівнює 1, верхня – 10.

До типових завдань із двовимірними масивами відносяться задачі на обчислення суми, добутку, кількості, середнього арифметичного, максимуму, мінімуму елементів кожного рядка або кожного стовпця, заданого рядка або стовпця.

Обчислення подібного роду здійснюються за допомогою стандартних прийомів, особливість яких полягає в організації вкладених циклів при обробці масивів.

Схема вкладеного *циклу* (cycle) з параметром така:

```
For i := a1 to b1 do
begin
  For j := a2 to b2 do
   begin
   ...
   end;
end;
```

Цикл, що містить інший цикл, називають зовнішнім циклом, а цикл, що перебуває в тілі іншого циклу, називають внутрішнім (вкладеним). Всі

оператори внутрішнього циклу мають повністю розташовуватися в тілі зовнішнього циклу.

Щораз, коли внутрішній цикл завершується, зовнішній цикл збільшує значення свого лічильника циклу на 1, а внутрішній цикл виконується знову.

Тому, якщо як лічильник зовнішнього циклу використати індекс рядка, а як лічильник внутрішнього – номер стовпця, то обробка двовимірного масиву буде йти по рядках, а якщо навпаки – по стовпцях.

У середовищі Delphi для введення й виведення масивів використовується компонент «StringGrid», тому робота з масивами в Delphi дуже відрізняється від роботи в Pascal.

8.1.1 Коротка характеристика компонента «StringGrid»

Компонент «StringGrid» групи «Additional» має вигляд таблиці, що містить рядки й колонки. Властивості компонента наведені у табл. 8.1.

Таблиця 8.1 – Властивості компонента «StringGrid»

Cells	Дана властивість зберігає всі елементи таблиці. Має тип «String».
FixedCols	Задає кількість фіксованих (нерухомих) стовпців у таблиці
FixedRows	Задає кількість фіксованих рядків у таблиці
Options → GoEditing	За замовчуванням дані в таблицю вводити не можна. Щоб зняти цю заборону, значення властивості має бути True.
ColCount	Задає загальну кількість стовпців таблиці
RowCount	Задає загальну кількість рядків таблиці

Таблиця може мати смуги прокручування, причому задане число перших рядків і стовпців може бути фіксованим і не піддаватися прокручуванню. Таким чином можна задати заголовки стовпців і рядків, постійно присутніх у вікні компонента. Кожній комірці таблиці може бути поставлений у відповідність деякий об’єкт.

8.2 Хід роботи

8.2.1 Індивідуальне завдання та робочий інтерфейс програми

Під час виконання лабораторної роботи необхідно розробити програму введення, обробки та виведення двовимірного масиву цифрових даних.

Орієнтовний вигляд вікна програми та індивідуальні завдання для виконання лабораторної роботи наведено на рис. 8.2 та у табл. 8.2.

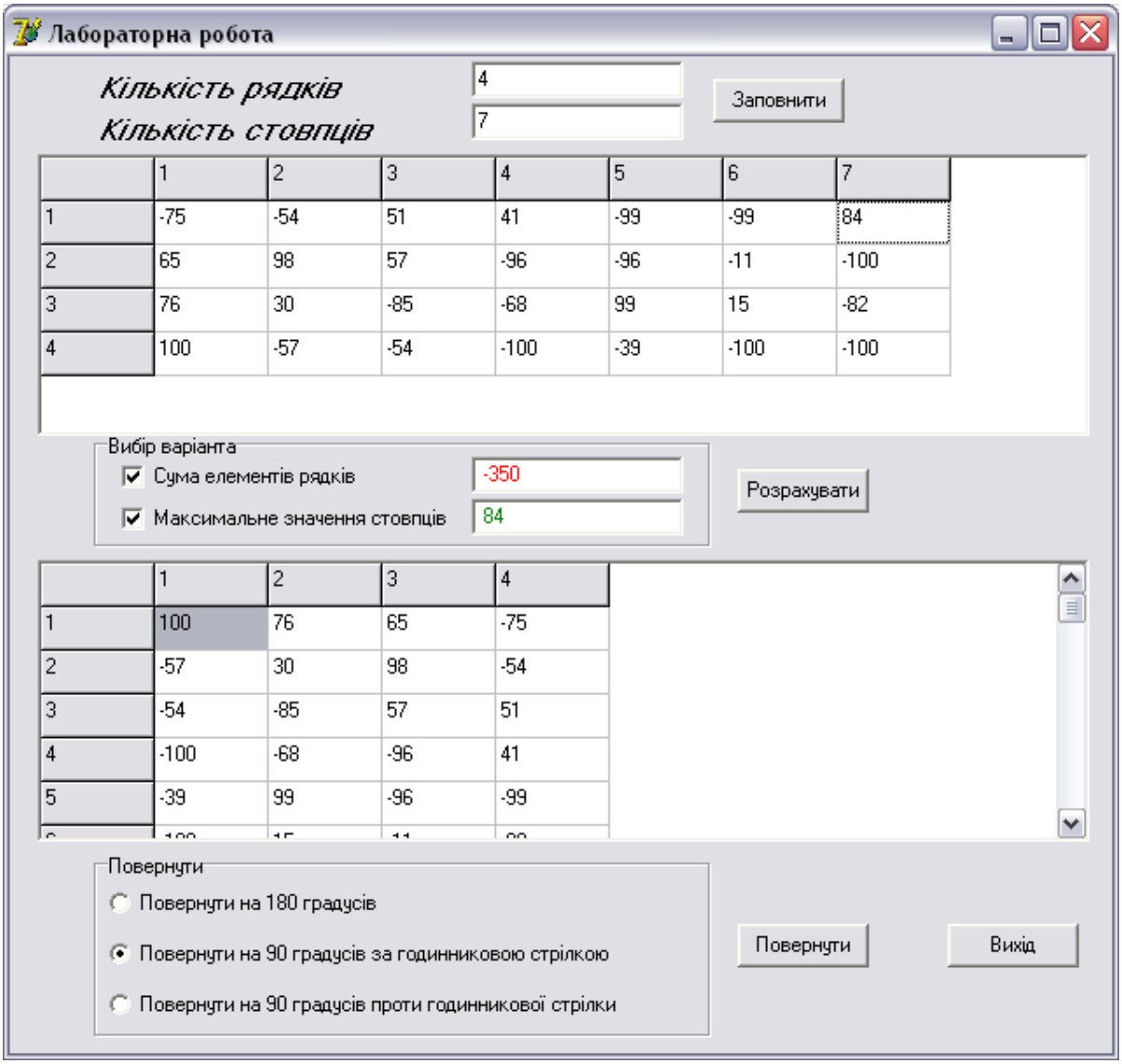


Рисунок 8.2 – Орієнтовний вигляд вікна програми

Таблиця 8.2 – Індивідуальне завдання

Варіант	**Завдання**
1	Виводити в заголовок вікна значення позиційного перемикача RadioGroup2 (регіон «Повернути»).
2	Виводити в заголовок вікна розмірність сформованого масиву.
3	Виводити в заголовок вікна суму елементів головної діагоналі матриці
4	Виводити в заголовок вікна значення комірки StringGrid2, яка вибрана у цей момент
5	Виводити в заголовок вікна максимальний елемент масиву і координати його комірки в масиві
6	Виводити в заголовок вікна кількість мінімальних елементів
7	Виводити в заголовок вікна кількість елементів, які менші середнього значення

Продовження табл. 8.2

Варіант	Завдання
8	Виводити в заголовок вікна середнє значення елементів масиву
9	Виводити в заголовок вікна число стовпців або рядків залежно від того, яке з них більше
10	Виводити в заголовок вікна кількість натискань кнопки «Повернути»
11	Виводити в заголовок вікна значення добутку елементів третього стовпця
12	Виводити в заголовок вікна другий за величиною елемент масиву
13	Виводити в заголовок вікна мінімальний елемент масиву і координати його комірки в масиві
14	Виводити в заголовок вікна кількість максимальних елементів комірки
15	Виводити в заголовок вікна значення суми елементів другого рядка

8.2.2 Лістинг програми

```
unit Unit1;
interface
uses
    Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
    Dialogs, StdCtrls, Grids, ExtCtrls;
type
    TForm1 = class(TForm)
        Edit1: TEdit;
        Edit2: TEdit;
        Label1: TLabel;
        Label2: TLabel;
        Button1: TButton;
        StringGrid1: TStringGrid;
        Button2: TButton;
        Edit3: TEdit;
        Edit4: TEdit;
        Button3: TButton;
        RadioGroup1: TRadioGroup;
        CheckBox1: TCheckBox;
        CheckBox2: TCheckBox;
        StringGrid2: TStringGrid;
        Button4: TButton;
        RadioGroup2: TRadioGroup;
        procedure Button1Click(Sender: TObject);
```

```
        procedure Button2Click(Sender: TObject);
        procedure Button3Click(Sender: TObject);
        procedure Button4Click(Sender: TObject);
        private
        { Private declarations }
         public
        { Public declarations }
        end;

    var
    Form1: TForm1;
    i,j,n,m: integer;
    a: array [0..50,0..50] of integer;
    b: array [0..50,0..50] of integer;

    implementation
{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
begin
Randomize;  {ініціюємо генератор випадкових чисел}
n:=StrToInt(Edit1.Text);
m:=StrToInt(Edit2.Text);
for i:=0 to n-1 do
for j:=0 to m-1 do
a[i,j]:=Round(Sin(Random(100))*100); {заповнюємо масив
                                      випадковими числами від -100 до +100}
StringGrid1.RowCount:=n+1; {кількість рядків в заголовку}
StringGrid1.ColCount:=m+1; {кількість стовпців в заголовку}
   {формування заголовків рядків і стовпців}
  with StringGrid1 do
  begin i:=0;
      for j:=1 to RowCount do
      Cells[i,j]:=IntToStr(j);
      j:=0;
for i:=1 to ColCount do
Cells[i,j]:=IntToStr(i);
end;
{виведення елементів масиву в таблицю}
with StringGrid1 do
for i:=1 to n do
for j:=1 to m do
    Cells[j,i]:=IntToStr(a[i-1,j-1]);
    end;
```

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  close;
  end;

  procedure TForm1.Button3Click(Sender: TObject);
  var
s, max,k,p: integer;
begin
  edit3.Text:=' ';
  edit4.Text:=' ';
  for i:=0 to n-1 do
  {вкладений цикл знаходження суми}
  begin s:=0;
      For j:=0 to m-1 do
       s:=s+a[i,j];
      end;
    For j:=0 to m-1 do
    begin
    max:=a[i,j];
for i:=0 to n-1 do
if a[i,j]>max then
max:=a[i,j];
end;
If CheckBox1.Checked then
edit3.Text:=edit3.Text+IntToStr(s)+'  ';
If CheckBox2.Checked then
edit4.Text:=edit4.Text+IntToStr(max)+'  ';
end;

procedure TForm1.Button4Click(Sender: TObject);
begin
If RadioGroup2.ItemIndex=2 then
begin
StringGrid2.RowCount:=m+1; {кількість рядків в заголовку}
      StringGrid2.ColCount:=n+1; {кількість стовпців в заголовку}
      {формування заголовків рядків і стовпців}
      with StringGrid2 do
      begin i:=0;
          for j:=1 to RowCount do
          Cells[i,j]:=IntToStr(j);
          j:=0;
for i:=1 to ColCount do
```

```
Cells[i,j]:=IntToStr(i);
end;
begin
for i:=0 to m-1 do
 for j:=0 to n-1 do
b[i,j]:=a[j,m-1-i];
with StringGrid2 do
for i:=1 to m do
for j:=1 to n do
Cells[j,i]:=IntToStr(b[i-1,j-1]);
end;
end;
if RadioGroup2.ItemIndex=1 then
begin begin
StringGrid2.RowCount:=m+1; {кількість рядків в заголовку}
      StringGrid2.ColCount:=n+1; {кількість стовпців в заголовку}
      {формування заголовків рядків і стовпців}
      with StringGrid2 do
      begin
      i:=0;
          for j:=1 to RowCount do
          Cells[i,j]:=IntToStr(j); j:=0;
            for i:=1 to ColCount do
            Cells[i,j]:=IntToStr(i);
        end;
        begin
        for i:=0 to m-1 do
        for j:=0 to n-1 do
            b[i,j]:=a[n-j-1,i];
            with StringGrid2 do
            for i:=1 to m do
            for j:=1 to n do
  Cells[j,i]:=IntToStr(b[i-1,j-1]);
  end;
  end;
   end;
  if RadioGroup2.ItemIndex=0 then
  begin
  StringGrid2.RowCount:=n+1;
  StringGrid2.ColCount:=m+1;
  with StringGrid2 do
  begin
  i:=0;
      for j:=1 to RowCount do
```

```
      Cells[i,j]:=IntToStr(j);
      j:=0; for i:=1 to ColCount do
      Cells[i,j]:=IntToStr(i);
      end;
      begin
      for i:=0 to n-1 do
      for j:=0 to m-1 do
b[i,j]:=a[n-i-1,m-j-1];
with StringGrid2 do
for i:=1 to n do
for j:=1 to m do
Cells[j,i]:=IntToStr(b[i-1,j-1]);
end;
end;
end;

end.
```

8.3 Контрольні питання

1. Що таке масив даних?
2. Як і в якому розділі програмного коду описуються масиви?
3. Як визначити місце розташування елемента в масиві?
4. Що таке індекс? Які вимоги він має задовольняти?
5. Як здійснюється доступ до елементів масиву?
6. Що таке динамічний масив?
7. Опишіть відмінності статичних і динамічних масивів. Який порядок опису й застосування динамічних масивів?
8. Коли й для чого використовується процедура «SetLength»?
9. Які компоненти зручно застосовувати при роботі з одновимірними, двовимірними масивами?
10. Як у середовищі Delphi здійснюється введення й виведення елементів масиву?
11. Що варто зробити, щоб у компоненті «StringGrid» можна було вводити дані?
12. Як використовується властивість «Cells» компоненти «StringGrid»?
13. Для чого використовуються властивості «FixedRows» і «FixedCols» в StringGrid?
14. Яке призначення компонента «GroupBox»? Як задати напис у цьому компоненті?
15. Чим відрізняються компоненти «RadioGroup» і «GroupBox»? Як задати список елементів в об’єкті «RadioGroup»?
16. З якою метою використається оператор «with…do» для звертання до елементів об’єкта?

ЛАБОРАТОРНА РОБОТА № 9
ОБРОБКА РЯДКОВИХ ТИПІВ ДАНИХ. РОБОТА З ФАЙЛАМИ

Мета: набути навичок роботи з рядковими типами даних в Delphi.

9.1 Теоретичні відомості

У реальних завданнях часто зустрічаються об’єкти символьного типу – *рядки* (string). Рядок в Pascal трактується як послідовність символів. До складу рядка можуть входити букви латинського алфавіту, кирилиці, цифри, різноманітні знаки, дужки, пропуск та інші символи. Кожний символ рядкової величини займає 1 байт пам’яті (десятковий код від 0 до 255, зафіксований у кодовій таблиці ASCII).

Таблиця 9.1 – Фрагмент таблиці символьного кодування ASCII

32 –	48 – 0	64 – @	80 – P	96 – `	112 – p
33 – !	49 – 1	65 – A	81 – Q	97 – a	113 – q
34 – "	50 – 2	66 – B	82 – R	98 – b	114 – r
35 – #	51 – 3	67 – C	83 – S	99 – c	115 – s
36 – $	52 – 4	68 – D	84 – T	100 – d	116 – t
37 – %	53 – 5	69 – E	85 – U	101 – e	117 – u
38 – &	54 – 6	70 – F	86 – V	102 – f	118 – v
39 – ’	55 – 7	71 – G	87 –W	ЮЗ – g	119 – w
40 – (	56 – 8	72 – H	88 – X	104 – h	120 – x
41 –)	57 – 9	73 – I	89 – Y	105 – i	121 – y
42 – *	58 – :	74 – J	90 – Z	106 – j	122 – z
43 – +	59 – ;	75 – К	91 – [	107 – k	123 – {
44 – ,	60 – <	76 – L	92 – \	108 – 1	124 – |
45 – -	61 – =	77 – M	93 –]	109 – m	125 – }
46 – .	62 – >	78 – N	94 – ^	110 – n	126 – ~
47 – /	63 – ?	79 – O	95 – _	111 – o	127 – □

Кількість символів у рядку називається її довжиною. Довжина рядка може динамічно змінюватися від 0 до 255. Пустий рядок має нульову довжину.

Рядкова константа – послідовність символів, взятих в одинарні лапки. Наприклад:

’Це рядкова константа’, ’123’.

Таким чином, при використанні у виразах рядок обов’язково береться в одинарні лапки. Два послідовні апострофа (’’) позначають пустий рядок, тобто рядок з нульовою довжиною.

Рядкова змінна описується в розділі опису змінних:

Var <ім’я> : string [<максимальна довжина рядка>];

Наприклад:

Var Name: string[20]; St: string;

Якщо максимальна довжина рядкової змінної не зазначена, то вважається, що вона дорівнює 255 (за замовчуванням).

Елементи рядка ідентифікуються іменем рядка з індексом, взятим у квадратні дужки. Тобто до будь-якого символа в рядку можна звернутися точно так само, як до елемента одновимірного масиву за його номером. Наприклад:

N [5], S[i], slovo[k+l]

Індекс може бути додатною константою, змінною, виразом цілого типу. Значення індексу не має виходити за межі опису.

Для зберігання й обробки окремих символів використовують змінні типу «char». Значенням змінної такого типу може бути будь-який один символ або кілька символів. Оголошується як var a: char. Для змінної такого типу резервується 1 байт пам’яті.

Вирази, у яких операндами виступають рядкові дані, називаються рядковими. З рядковими даними допустимі операції зчеплення та відношення.

Операція зчеплення або конкатенації (+) застосовується для з’єднання декількох рядків в один результувальний рядок. Зчеплювати можна як рядкові константи, так і рядкові змінні. Наприклад:

’Паскаль’ + ’ і ’ + ’Сі’ + ’ –’ + ’ мови програмування високого рівня’

В результаті вийде рядок: ’Паскаль і Сі – мови програмування високого рівня’.

Довжина результувального рядка не може перевищувати 255 символів.

Операції відношення (=, <, >, <=, >=, <>) дозволяють здійснювати порівняння двох рядків, в результаті чого повертається логічне значення (True або False). Операції відношення мають нижчий пріоритет, ніж операції зчеплення. Порівняння рядків здійснюється зліва направо до першого незбіжного символа, і більшим вважається той рядок, у якого перший незбіжний символ має більший номер у таблиці символьного

кодування. Якщо рядки мають різну довжину, але в загальній частині символи збігаються, вважається, що коротший рядок менший, ніж довший. Рядки однакові, якщо вони повністю збігаються по довжині й містять ті самі символи (табл. 9.2).

Таблиця 9.2 – Приклади логічних виразів

Вираз	Результат
’True1’ < ’True2’	True
’Student’ > ’STUDENT’	True
’студент’ <> ’Студент’	True
’група’ = ’група’	True

Усі інші дії над рядками та символами реалізовуються за допомогою вбудованих функцій (табл. 9.3) і процедур (табл. 9.4).

Таблиця 9.3 – Функції для роботи з даними рядкового типу

Звертання до функції	Дія	Приклад
Copy (S, Poz, N)	Виділяє з рядка S підрядок довжиною N символів, починаючи з позиції Poz. Тут N і Poz – цілі числа.	S := ’IBM-PC’; Sl := Copy(S,5,2); Результат: S1=’PC’
Concat (Sl,...,Sn)	Виконує зчеплення (конкатена-цію) рядків S1, S2, ..., Sn в один рядок.	Sl:= ’Test’; S2:= ’-’; S3:= ’5’; S:=Concat (S1,S2,S3); Результат: S = ’test-5’
Length (S)	Визначає поточну довжину рядка S. Результат – значення цілого типу.	S := ’test-5’; n := Length (S); Результат: n = 5
Lowercase (S)	Повертає копію рядка S, у якій усі символи перетворені на символи нижнього регістра. Працює тільки з літерами латинського алфавіту.	S := ’STUDENT’; S := Lowercase (S); Результат: S = ’student’
Uppercase(S)	Повертає копію рядка S, у якому усі символи перетворені на символи верхнього регістра (на великі літери). Працює тільки з літерами латинського алфавіту.	S := ’student’; S := Uppercase (S); Результат: S = ’STUDENT’
Trim(S)	Видаляє з рядка S початкові та кінцеві пропуски, а також керівні символи.	S := ’ Сту ’; S := Trim(S); Результат: S = ’Сту’

Продовження таблиці 9.3.

Звертання до функції	Дія	Приклад
Trimleft(S)	Повертає копію рядка S з видаленими початковими пропусками й керівними символами.	S := ’ Студент ’; S := Trimleft(S); Результат: S = ’Студент ’
Trimright (S)	Повертає копію рядка S з видаленими кінцевими пропусками й керівними символами.	S := ‘ Студент ‘; S := Trimright(S); Результат: S = ‘ Студент’
Chr(X)	Повертає символ за позначенням його ASCII-кодом. Тут X – число цілого типу. Результат – значення типу «Char».	for i := 65 to 71 do S := S+Chr(i); Результат: S = ’ABCDEFG’
Ord(S)	Повертає ASCII-код зазначеного символа. Тут S – змінна типу «Char». Результат – значення цілого типу.	S : = ’F’; Cod := Ord(S); Результат: Cod = 70
Pos(Sl, S2)	Виявляє першу появу в рядку S2 підрядка S1. Результат – ціле число, що дорівнює номеру позиції, в якій перебуває перший символ підрядка S1. Якщо поява не виявлена, то результат = 0.	S := ’abodef’; n := Pos(’od’,S); Результат: n = 3

Таблиця 9.4 – Процедури для роботи з даними рядкового типу

Звертання до процедури	Дія	Приклад
Delete(S,Poz,N);	Видалення N символів з рядка S, починаючи з позиції Poz.	S := ’abodef’; Delete(S,3,2); Результат: S = ‘abef’
Insert(Sl,S2,Poz);	Вставка рядка Sl у рядок S2, починаючи з позиції Poz.	S := ’3BM PC’; Insert(’IBM-’,S,5); Результат: S = ’3BM IBM-PC
Str(X[:N[:M]],S);	Перетворює числове значення X у рядок S. Параметр X – вираз цілочислового або дійсного типу, а параметри N і M – це цілочислові значення, що визначають формат рядка S (рядок з N символів, з яких M – після коми).	X := -23,60; Str(X:6:1,S); Результат: X = ’-23.6’ a := -23.60; b := 20.5; Str(a+b:6:1;S); Результат: S = ’-3.1’

Продовження таблиці 9.4

Звертання до процедури	Дія	Приклад
Val(S,x,Cod);	Перетворює рядок S на числове значення. Параметр S має містити послідовність символів, яка може бути сприйнята як дійсне число зі знаком. Параметр x може бути як дійсним, так і цілочисловим. У змінну-параметр Cod заноситься код виконання перетворення: 0, якщо перетворення виконане успішно, або ж номер позиції, у якій відбулася помилка.	S := ’-23.60’; val(S,x,Cod); Результат: x = -23,6 Cod = 0

9.2 Хід роботи

9.2.1 Завдання на лабораторну роботу

Під час виконання лабораторної роботи необхідно розробити програму, яка дозволяє виконувати операції з рядком тексту. Орієнтовний інтерфейс програмного засобу наведено на рис. 9.1.

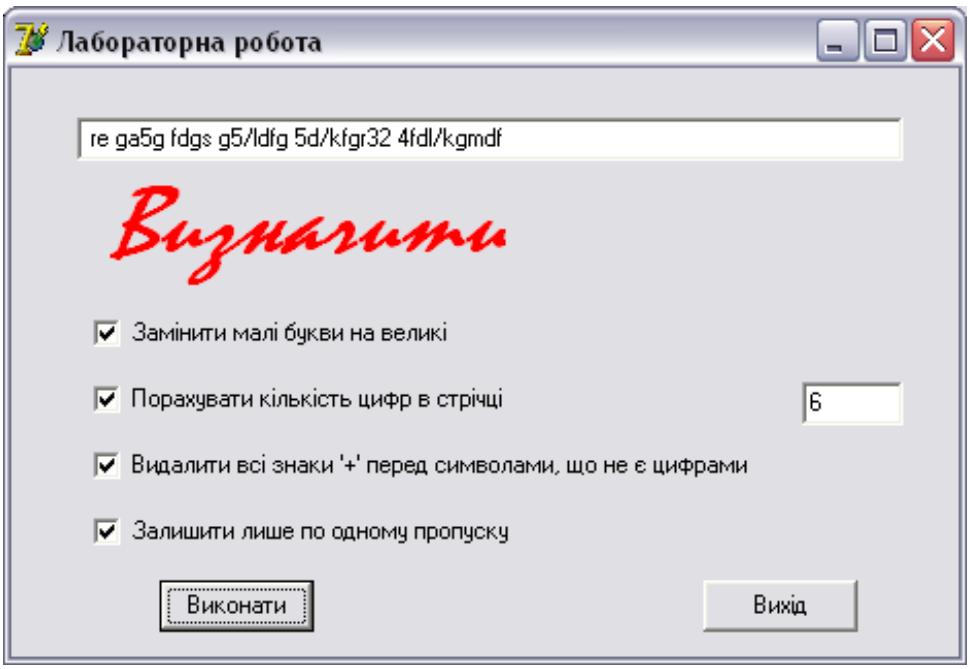


Рисунок 9.1 – Орієнтовний вигляд вікна програми

9.2.2 Лістинг програми

```
unit DIYI_NAD_SLOVAMY;
interface
```

```
uses
    Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
    Dialogs, StdCtrls;
type
    TForml = class(TForm)
        Editl: TEdit;
        Labell: TLabel;
        CheckBoxl: TCheckBox;
        CheckBox2: TCheckBox;
        CheckBox3: TCheckBox;
        CheckBox4: TCheckBox;
        Buttonl: TButton;
        Button2: TButton;
        Edit2: TEdit;
        procedure Button2Click(Sender: TObject);
        procedure ButtonlClick(Sender: TObject);
        private { Private declarations }
         public { Public declarations }
         end;
var
Forml: TForml;
var  f,fl, s,sl: string; i,n,k: integer;

implementation
 {$R *.dfm}

procedure TForml.Button2Click(Sender: TObject);
begin
close;
end;

procedure TForml.ButtonlClick(Sender: TObject);
begin
// робимо всі літери великими
s:=Editl.Text;
s1: = '' ;
    for i:=l to length(s) do
      case s[i] of
       'a'..'я' : s1:=sl + chr(ord(s[i])-32) ;
  else sl:=sl+s[i];
  if CheckBoxl.Checked  then
  Editl.text:=sl;
  end;
```

```
// підраховуємо кількість цифр в рядку
n:=0;
for i:=l to length(s) do
f (S[i]>=’0’) and (S[i]<=’9’) then
n:=n+l;
 if CheckBox2.Checked then
Edit2.text:=inttostr(n); // забираємо +
S:=Editl.Text+’ ’; {для роботи з останнім символом}
sl:= ’’;  {вихідний рядок}
for i:=l to length (s) do
if (S[i]=‘+’) then begin
    if (s[i+l]>=‘0’) and (s[i+l]<=‘9’)  then
    si:=sl + s [i]
    end else
    si:=sl + s [i];
      if CheckBox3.Checked  then
      Editl.text:=sl; // забираємо непотрібні пропуски
      s:=Trim(Editl.Text) ;
      Sl: = ’ ’ ;
    for i:=l to length(s) do
        if (S[i]<>’ ’) then
        Sl:=Sl+S[i] else
    if (S[i+1]<>’ ’) then
    S1:=S1+’ ’;
    K:=Length(s)-Length (si);
    if CheckBox4.Checked  then
    Editl.text:=S1;
    end;
end.
```

9.3 Контрольні питання

1. Як описуються змінні символьного типу?
2. Опишіть стандартні функції для роботи із символьними змінними.
3. Як описуються рядкові змінні?
4. Як записується рядкова константа?
5. Яка максимальна довжина рядка?
6. Що спільного між рядком і символьним масивом?
7. Які операції застосовні до рядкових змінних і констант?
8. Опишіть із прикладами стандартні функції обробки рядкових даних.
9. Опишіть роботу стандартних процедур обробки рядкових даних.
10. Яким чином відбувається звертання до елементів рядка?
11. Як в полі компонента «Memo» можна обробити кілька рядків?

ЛАБОРАТОРНА РОБОТА № 10
РОБОТА З ВЕКТОРНОЮ ГРАФІКОЮ В СЕРЕДОВИЩІ DELPHI ТА ГРАФІЧНЕ ОФОРМЛЕННЯ ІНТЕРФЕЙСУ

Мета: вивчити можливості графічної підсистеми Delphi, її властивості та принципи побудови прикладних програм із використанням графіки.

10.1 Теоретичні відомості

10.1.1 Створення зображень

У Win32 для зображенння об'єктів, які ми бачимо на екрані комп'ютера, використовується інтерфейс GDI (Graphic Device Interface). До появи Delphi програміст безпосередньо працював з функціями та інструментами GDI (пером, пензлем та шрифтом) через графічний контекст вікна.

Для використання графічних інструментів Windows у Delphi створено відповідні класи: для графічного контексту – клас «TCanvas», для пера – «TPen», для пензля – «TBrush», для шрифту – «TFont». Об'єкти цих класів автоматично створюються для форми та більшості видимих елементів на формі та стають доступними через властивості «Canvas», «Pen», «Brush» та «Font».

Наступні два приклади демонструють наскільки ICP Delphi спрощує роботу з графікою у Win32. У прикладі 1 наведено програму створення еліпса, яка використовує стандартні GDI-функції, а в прикладі 2 – програму, що створює аналогічний рисунок засобами класів Delphi.

Приклад 1:

```
procedure TForm1.FormPaint(Sender: TObject);
var
PaintDC : HDC;
PenHandle, OldPenHandle: HPEN;
BrushHandle, OldBrushHandle: HBRUSH;
begin
PaintDC: = GetDC(Handle); {отримати графічний контекст форми}
PenHandle: = CreatePen(PS_SOLID, 1, RGB(0, 0, 255)); {створити синє перо}
OldPenHandle: = SelectObject(PaintDC, PenHandle); {використати його у графічному контексті}
BrushHandle: = CreateSolidBrush(RGB(255, 255, 0)); {створити жовтий пензлик}
OldBrushHandle: = SelectObject(PaintDC, BrushHandle);
{використати його у графічному контексті}
Ellipse(PaintDC, 10, 10, 50, 50); {нарисувати еліпс}
SelectObject(PaintDC, OldBrushHandle);
```

```
{відновити попередній пензлик}
DeleteObject(BrushHandle); {знищити жовтий пензлик}
SelectObject(PaintDC, OldPenHandle); {відновити попереднє перо }
DeleteObject(PenHandle); {знищити синє перо}
end;
```

Приклад 2:

```
procedure TForm1.FormPaint(Sender: TObject);
begin
with Canvas do {працюємо з властивістю Canvas класу TForm}
begin
Pen.Color: = clBlue; {створити синє перо}
Brush.Color: = clYellow; {створити жовтий пензлик}
Ellipse(10, 10, 50, 50); {нарисувати еліпс}
end;
end;
```

Канва («Canvas») або полотно – це площа видимого елемента на формі, на якій можна рисувати (графічний контекст вікна). Вона є однією з властивостей видимого елемента. «Font», «Pen» та «Brush» є підвластивостями «Canvas».

Деякі видимі елементи не мають властивості «Canvas», наприклад, панелі – компоненти класу «TPanel». На таких компонентах можна рисувати, якщо розмістити на них спеціальний компонент класу «TPaintBox». Крім того, об’єкти «PaintBox» можна використовувати для обмеження одним чи декількома прямокутниками (на формі чи на деякій компоненті ділянки рисування).

У процесі розробки проекту властивість «Canvas» недоступна у вікні інспектора об’єктів. Щоб використати «Canvas», необхідно присвоїти її підвластивостям необхідні значення та викликати потрібні методи.

Клас «TCanvas» інкапсулює графіку Windows різного рівня (табл. 10.1). До нього входять функції високого рівня для рисування ліній, фігур або тексту, можливості середнього рівня для роботи з графічними інструментами та можливості найнижчого рівня – безпосереднього доступу до об’єктів Win32 GDI.

Pen (перо) – це лінія, яка виділяє зовнішній контур фігури. Можна задавати товщину (підвластивість «Width»), колір («Color»), вигляд («Style»), спосіб малювання лінії (накладання кольору лінії на поверхню – «Mode»).

Brush (пензлик) – це площина фігури, тобто внутрішня відносно контуру частина. Можна задавати колір («Color») та спосіб зафарбовування («Style»), наприклад, горизонтальними чи діагональними лініями.

Font (шрифт) – використовують для опису шрифтів, які застосовують під час виведення тексту (за допомогою методів «TextOut» або «TextRect» класу «TCanvas»). Має підвластивості «Name» (назва), «Style» (стиль), «Color» (колір), «Size» (розмір).

Таблиця 10.1 – Функціональні рівні класу «TCanvas»

Рівень	Операції	Інструменти
Високий	Рисування ліній та фігур	MoveTo, LineTo, Rectangle, Ellipse (методи)
	Зміна розмірів та виведення тексту	TextOut, TextHeight, TextWidth, TextRect (методи)
	Зафарбовування площини фігури	FillRect, FloodFill (методи)
Середній	Маніпулювання графічними інструментами	Pen, Brush, Font (властивості)
	Маніпулювання пікселами	Pixels (властивість)
	Копіювання та переміщення зображень	Draw, StretchDraw, BrushCopy, CopyRect (методи)
		CopyMode (властивість)
Низький	Виклик функцій Win32 GDI	Handle (властивість)

10.1.2 Опрацювання зображень

Клас «TGraphic» у Delphi забезпечує базові інструменти для використання графічних файлів. Для опрацювання графічних зображень різних форматів введено дочірні класи: «TBitmap» – растрові файли («*.bmp»), «TMetafile» – метафайли Windows («*.wmf», «*.emf») та «TIcon» – піктограми («*.ico»). Також можна вводити власні дочірні класи.

Растрові файли (bitmaps) – це файли з двійковою інформацією у вигляді бітів, з їх послідовності і складається графічний образ. В цих бітах зберігається інформація про колір пікселів. Растри бувають двох типів: залежні від пристроїв (device dependent bitmaps – DDB) і незалежні (device independent bitmaps – DIB).

Метафайли – це *векторні* (vector images) графічні зображення, що надають можливість довільної зміни розмірів без впливу на якість відображення ліній і дуг, на відміну від растрових зображень.

Піктограми (icons) – це ресурси, що, як правило, зберігаються у файлі з розширенням «*.ico» або у файлах ресурсів з розширенням «*.res». Існує два типових розміри піктограм у 32-бітових версіях Windows: великі (32×32) та малі (16×16). У всіх прикладних програмах для Windows використовуються обидва розміри. Малі піктограми розміщуються в лівому верхньому куті головного вікна прикладної програми, а також в

елементах управління. Великі піктограми можна розмістити на панелі задач прикладної програми або у вікні повідомлення.

Для виведення рисунків на екран використовується клас «TImage». За допомогою відповідної компоненти (сторінка палітри «Additional») можна завантажити та відобразити в будь-якому місці екрана чи форми файл будь-якого формату, що опрацьовуються класами-нащадками «TGraphic». Графічні дані зберігаються у властивості «Picture» (класу «TPicture») об'єкта «TImage». Клас «TPicture» є контейнером для графіки, тобто може містити будь-який зі згаданих класів графічних файлів. Для завантаження та збереження зображення використовують відповідні методи класу «TPicture»**,** наприклад:

```
MyImage.Picture.LoadFromFile('FileName.BMP');
```

Для копіювання зображень використовуються методи класу «TCanvas» «Draw» (копіювання всього рисунка), «StretchDraw» (копіювання всього рисунка зі зміною розміру (стискання або розтяг), «CopyRect» (копіювання частини рисунка), «BrushCopy» (копіювання рисунка разом із растровими операціями). Для копіювання одного растра в інший можна застосувати метод «TBitmap Assign»**.**

10.2 Хід роботи

10.2.1 Завдання до лабораторної роботи

Створити програму, яка будує криві за заданим параметричним поданням (індивідуальні варіанти 1, 3, 5, 7, 9, 11, 13, 15, 17) або за їх рівняннями у полярних координатах (індивідуальні варіанти 2, 4, 6, 8, 10, 12, 14, 16, 18).

Параметричне подання кривої l на площині з координатами x, y – це функції $x = x(t)$ та $y = y(t)$, визначені на одній і тій самій числовій множині.

Полярні координати ρ, φ точки M на площині – це відстань $\rho = OM$ від фіксованої точки O (полюса) до точки M і кут $\varphi = \angle POM$ між OM і полярною віссю (променем) OP.

Додатково для всіх варіантів!

За допомогою стандартної функції LineTo(X,Y) навколо кривої нарисувати рамку.

В процесі захисту лабораторної роботи оцінюється графічне оформлення інтерфейсу вікна розробленої програми.

Таблиця 10.2 – Індивідуальні завдання

Варіант	Назва кривої	Рівняння
1	Трактриса	$x = a \cdot \cos(t) + a \cdot \ln(tg(t/2))$, $y = a \cdot \sin(t)$, $a > 0$, $t \in [0, \pi)$
2	Спіраль «жезл»	$\rho = a / \sqrt{\varphi}$, $a > 0$
3	Суперколо	$x = a \cdot \cos^n(t)$, $y = a \cdot \sin^n(t)$, $a > 0$, $t \in [0, 2\pi)$
4	Лемніската	$\rho = a \cdot \sqrt{2 \cdot \cos(2 \cdot \varphi)}$, $a > 0$, $\varphi = (-\pi/4; \pi/4), (3\pi/4; 5\pi/4)$
5	Цисоїда	$x = a / (1 + t^2)$, $y = a / (t \cdot (1 + t^2))$, $t = (-\infty; \infty)$, $a > 0$
6	Синусоїдальна спіраль	$\rho = a \cdot \sqrt{\cos(n \cdot \varphi)}$, $a > 0$, $n > 0$
7	Кардіоїда	$x = a \cdot \cos(t) \cdot (1 + \cos(t))$, $y = a \cdot \sin(t) \cdot (1 + \cos(t))$, $a > 0$, $t \in [0, 2\pi)$
8	Спіраль Кейлі шостого порядку	$\rho = a / \cos^3(\varphi/3)$, $a > 0$
9	Циклоїда	$x = a \cdot (t - \sin(t))$, $y = a \cdot (1 - \cos(t))$, $a > 0$, $t \in [0, 2\pi)$
10	Спіраль Архімеда	$\rho = a \cdot \varphi$, $a > 0$
11	Строфоїда	$x = a \cdot (t^2 - 1) / (t^2 + 1)$, $y = a \cdot t \cdot (t^2 - 1) / (t^2 + 1)$, $t = (-\infty; \infty)$, $a > 0$
12	Квадратична спіраль	$\rho = a \cdot \varphi^2$, $a > 0$
13	Дельтоїд	$x = 2 \cdot a \cdot \cos(t) + a \cdot \cos(2 \cdot t)$, $a > 0$, $y = 2 \cdot a \cdot \sin(t) - a \cdot \sin(2 \cdot t)$, $t \in [0, 2\pi)$
14	Трисектриса	$\rho = a / \cos(\varphi/3)$, $a > 0$
15	Астроїда	$x = b \cdot \cos^3(t)$, $y = b \cdot \sin^3(t)$, $t \in [0, 2\pi)$
16	Спіраль Ферма	$\rho = a \cdot \sqrt{\varphi}$, $a > 0$
17	Евольвента кола	$x = a \cdot (\cos(t) + t \cdot \sin(t))$, $y = a \cdot (\sin(t) - t \cdot \cos(t))$, $a > 0$, $t \in [0, 2\pi)$
18	Логарифмічна спіраль	$\rho = a \cdot e^{b \cdot \varphi}$, $a > 0$, $b > 0$

10.3 Контрольні питання

1. Які класи відповідають за роботу з графікою в Delphi?
2. Які графічні інструменти наявні в Delphi?
3. Охарактеризуйте властивості та методи класу «TCanvas».
4. Які існують типи графічних зображень?
5. Як завантажити на форму зображення з файла?

ГЛОСАРІЙ

Атрибут – це параметр файла, що визначає певні властивості файлів і встановлюється як властивість конкретного файла.

Векторні зображення – зображення в комп'ютерній графіці з сукупності геометричних примітивів (точок, ліній, кривих, полігонів), тобто об'єктів, які можна описати математичними виразами. Векторна графіка для опису зображення використовує вектори, на відміну від растрової графіки, яка описує зображення як масив пікселів (точок). Векторні зображення нечутливі до масштабування.

Відеокарта – пристрій, призначений для обробки, генерації зображень з подальшим їх виведенням на екран периферійного пристрою. Відеокарта, зазвичай, є платою розширення (дискретна відеокарта) і під'єднується через слот розширення, універсальний (PCI-Express, PCI, ISA ...) або спеціалізований (AGP) порт, проте відеокарта може бути вбудованою (інтегрованою) у материнську плату (як у вигляді окремого елемента, так і як складова частина північного мосту чипсета або центрального процесора). Сучасні відеокарти не обмежуються лише звичайним виведенням зображень, вони мають вбудований графічний мікропроцесор, котрий може проводити додаткову обробку, звільняючи від цих задач центральний процесор.

Вікно – візуально відокремлена область екрана з певним інтерфейсом користувача. Зазвичай вікно має прямокутну форму. В ньому відображаються елементи вводу і, можливо, виводу для одного або декількох процесів. Вікнами можна управляти за допомогою курсора миші та клавіатури.

Електронні таблиці – клас програмних засобів для роботи з даними, які сформовані у вигляді таблиці. Сучасний табличний редактор дає можливість з'єднувати кілька таблиць для спільної роботи і обробки, включати текстові, графічні та інші мультимедійні елементи. Інструментарій електронних таблиць містить потужні математичні функції, що дозволяють проводити складні статистичні, фінансові та інші розрахунки.

Жорсткий диск, твердий диск або накопичувач на магнітних дисках (на комп'ютерному сленгу – «вінчестер») – магнітний диск, основа якого виконана з твердого матеріалу. У більшості ПК виконує функцію енергонезалежного носія інформації (комп'ютерної пам'яті чи накопичувача інформації) з довільним доступом (random access).

Індекс – число або інша комбінація символів, що вказує місце елемента в сукупності або характеризує стан деякої системи (в масиві – елемент, який вказує адресу комірки в загальній послідовності).

Інспектор об'єктів – інструмент середовища розробки Delphi, який дає змогу повністю визначати вигляд та поведінку компонента, що розташований на формі проекту, а також самої форми. З його допомогою можна задавати потрібні значення властивостей компонентів об'єктів), а також реакцію на стандартні події. Інспектор об'єктів розташований в окремому

вікні, яке створюється автоматично під час завантаження Delphi. Вікно інспектора об'єктів містить список компонентів поточної форми, а також дві сторінки: властивостей Properties та подій Events.

Інтерфейс апаратний – сукупність алгоритмів обміну і технічних засобів, що забезпечують обмін інформацією між пристроями. Апаратний інтерфейс є системою шин, роз'ємів, узгоджувальних пристроїв, алгоритмів і протоколів, що забезпечують зв'язок всіх частин мікропроцесорної системи між собою. Від характеристик інтерфейсу залежить швидкодія і надійність системи. У розгорнутих мікропроцесорних системах для розвантаження процесора апаратний інтерфейс забезпечується контролерами.

Інтерфейс користувача – різновид інтерфейсу, в якому одна сторона представлена людиною (користувачем), інша – машиною/пристроєм. Це сукупність засобів і методів, за допомогою яких користувач взаємодіє з різними, найчастіше складними, машинами, пристроями, апаратурою і програмним забезпеченням. Часто термін застосовується до комп'ютерних програм, однак він може застосовуватися і до набору засобів, методів і правил взаємодії будь-якої системи, керованої людиною. Інтерфейс вважається двонаправленим (інтерактивним), якщо пристрій, отримавши команди від користувача і виконавши їх, видає інформацію користувачеві наявними у нього засобами – візуальними, звуковими, тактильними тощо. Інтерфейс – це сукупність елементів, які також можуть складатися з елементів.

Інтерфейс програмний – набір готових класів, процедур, функцій, структур і констант, що надаються інформаційною системою (бібліотекою, сервісом) для використання у зовнішніх програмних продуктах. Використовується програмістами для написання власних програмних засобів.

Канва або полотно – це площа видимого елемента на формі (або всієї форми) програмного засобу, на якій можна рисувати (графічний контекст вікна) штатними засобам середовища програмування.

Кеш-пам'ять – особлива швидкісна пам'ять або частина ОЗП, де зберігаються копії даних, що часто використовуються. Кеш забезпечує швидкий доступ до них. Кеш-пам'ять зберігає вміст і адреси даних, до яких часто звертається процесор. Це дозволяє скоротити тривалість обміну даними, оскільки швидкодія кешу більша за швидкодію звичайної пам'яті.

Кластер – в деяких типах файлових систем логічна одиниця зберігання даних у таблиці розміщення файлів, що об'єднує групу секторів. Наприклад, на дисках з розміром секторів в 256 байтів, 256-байтовий кластер містить один сектор, тоді як 2-кілобайтовий кластер містить вісім секторів. Як правило, це найменший обсяг пам'яті на диску, який може бути виділений для зберігання файла.

Комп'ютер – багатозначний термін, найчастіше вживається для означення програмно керованого електронного пристрою обробки інформації. Разом з тим, це може бути будь-який механічний, немеханічний (електронний) пристрій, призначений для проведення обчислень. Обчислення можуть відбуватися дискретно або безперервно у часі. У вузькому значен-

ні – електронний цифровий програмований пристрій (електронна обчислювальна машина) для проведення обчислень, а також приймання, оброблення, зберігання і видачі інформації за заздалегідь визначеним алгоритмом. Термін «комп'ютер» і «ЕОМ», «ПЕОМ» (електронно-обчислювальна машина, персональна ЕОМ), прийняті в українській науковій та технічній літературі, є синонімами. Останнім часом терміни ЕОМ і ПЕОМ вважаються застарілими.

Масив – впорядкований набір фіксованої кількості однотипних елементів, що зберігаються в послідовно розташованих комірках оперативної пам'яті, мають порядковий номер і спільне ім'я, що надає їм користувач. Характеристика масиву: розмірність (кількість індексів елемента – одновимірний, двовимірний, багатовимірний); розмір (загальна кількість елементів у масиві). За типом масиви поділяються на числові та символьні. В програмуванні масив – це сукупність елементів одного типу даних, впорядкованих за індексами, які зазвичай репрезентовані натуральними числами, що визначають положення елемента в масиві. Масив може бути одновимірним (вектором), та багатовимірним (наприклад, двовимірною таблицею), тобто таким, де індексом є не одне число, а кортеж (сукупність) з декількох чисел, кількість яких збігається з розмірністю масиву.

Материнська плата – складна багатошарова друкована плата, що є основою побудови обчислювальної системи (комп'ютера). У деяких складних електронних приладах і пристроях (наприклад: стільниковий телефон, телевізор) основна (найбільша, найбільш значуща) плата пристрою також може називатися материнською або системною. Як основні (нез'ємні) частини, материнська плата містить роз'єм процесора, мікросхеми чипсета, завантажувальний ПЗП, контролери шин і інтерфейсів введення-виведення та периферійних пристроїв. ОЗП у вигляді модулів пам'яті встановлюються в спеціально призначені роз'єми; в слоти розширення встановлюються карти розширення.

Матриця – математичний об'єкт, записаний у вигляді прямокутної таблиці чисел, він допускає операції (додавання, віднімання, множення та множення на скаляр). Зазвичай матриці зображаються у вигляді двовимірних (прямокутних) таблиць.

Монітор – конструктивно завершений пристрій, призначений для візуального відображення інформації. Сучасний монітор складається з екрана (дисплея), блока живлення, плат управління і корпусу. Інформація для відображення на моніторі надходить з електронного пристрою, що формує відеосигнал (у комп'ютері – відеокарта).

Мости північний і південний – набори мікросхем, що керують роботою пристроїв персонального комп'ютера. Північний міст – системний контролер чипсета на материнській платі з архітектурою x86, до якого в рамках організації взаємодії підключені: мікропроцесор; оперативна пам'ять (якщо у складі процесора немає контролера пам'яті); відеоадаптер (в материнських платах нижнього цінового діапазону відеоадаптер часто вбудо-

ваний). Південний міст – це функціональний контролер, також відомий як контролер-концентратор введення-виведення. Зазвичай, це одна мікросхема, яка пов'язує «повільні» взаємодії на материнській платі з ЦПУ через північний міст. Функціонально до складу південного мосту входять: контролери шин PCI, PCI Express, SMBus, I2C, LPC, Super I/O; DMA контролер; контролер переривань; PATA (IDE) і SATA контролери; годинник реального часу (Real Time Clock); керування живленням (Power management, APM і ACPI); незалежна пам'ять BIOS (CMOS); звуковий контролер (зазвичай AC'97 або Intel HDA). Опціонально південний міст також може містити в собі контролер Ethernet, RAID-контролери, контролери USB, контролери FireWire та аудіо-кодек.

Оперативний запам'ятовувальний пристрій (ОЗП) – енергозалежна частина системи комп'ютерної пам'яті, в якій тимчасово зберігаються вхідні, вихідні та проміжні дані програми процесора. Найпоширеніші типи – DDR, DIMM і SIMM. Це технічний пристрій, що реалізує функції оперативної пам'яті. ОЗП може виготовлятися як окремий зовнішній модуль або розташовуватися на одному кристалі з процесором, наприклад, в однокристальних ЕОМ або однокристальних мікроконтролерах.

Основна система введення/виведення (BIOS) – набір спеціальніх підпрограм, які використовуються комп'ютерами архітектури x86 для ініціалізації компонентів персональної платформи, необхідних для її первинного завантаження та подальшої роботи. Це перше програмне забезпечення, що виконується процесором, оскільки на початковому етапі завантаження комп'ютера зовнішні пристрої недоступні.

Панель завдань – елемент графічного інтерфейсу, який використовується для запуску інших програм та керування вже запущеними. Має вигляд панелі інструментів. Використовується, зокрема, для керування вікнами програмних засобів. Панель завдань може бути компонентом операційної системи (наприклад, вона є в операційних системах Microsoft Windows, починаючи з версії Windows 95), елементом середовища робочого столу (наприклад, у KDE або GNOME) або сторонньою програмою.

Піктограма – значок, елемент графічного інтерфейсу користувача; невелике зображення на екрані, що служить для ідентифікації деякого об'єкта: файла, програми і т. п. Вибір і активізація піктограми викликає дію, пов'язану з обраним об'єктом.

Помилка – позаштатна ситуація, яка приводить до збою нормальної роботи програми або взагалі до зупинення її роботи, а також спеціалізоване повідомлення, що виводиться при виникненні в процесі роботи програмних засобів такої ситуації.

Постійний запам'ятовувальний пристрій (ПЗП) – пристрій з постійним збереженням певної інформації, що не допускає її зміни в процесі роботи ЕОМ, тобто такий, що не здатний виконувати операцію запису. Зміна вмісту даної пам'яті проводиться шляхом заміни носія інформації.

Програма – послідовність інструкцій, призначених для виконання пристроєм управління обчислювальної машини. Програма – один з компонентів програмного забезпечення. Залежно від контексту цей термін може відноситися також і до вихідних текстів програми.

Програмне забезпечення – сукупність програм, процедур і правил, а також документації, що стосуються функціонування системи оброблення даних з використанням електронних обчислювальних машин.

Растрове зображення – зображення у вигляді мережі пікселів (зазвичай прямокутної), відображенних на моніторі, папері та інших матеріалах. Характеристиками растрового зображення є: кількість пікселів (зазвичай вказують кількість пікселів по ширині і висоті (наприклад, 1024×768, 1920×1080); кількість кольорів або глибина кольору (обсяг пам'яті в бітах, що використовуються для одного піксела); колірний простір (RGB, CMYK, XYZ, YCbCr та ін.).

Робоча книга – це багатосторінковий табличний документ MS Excel, кожна сторінка якого називається робочим листом (аркушем). У вікні документа відображається активна сторінка.

Робочий стіл або стільниця – це головна область екрана, яка відображається після ввімкнення комп'ютера та входу до Windows. Водночас це папка, до якої можна одержати доступ через «Провідник» або файлові менеджери. Він є поверхнею для роботи, як і звичайний стіл. Відкриті програми чи папки відображаються на робочому столі.

Рядок – тип даних, який використовується для зберігання набору символів (як літер, так і цифр), що інтерпретуються згідно з кодом таблиці ANSI.

Сектор – мінімальна одиниця зберігання інформації на дискових запам'ятовувальних пристроях, що адресується. Є частиною доріжки диска. У більшості пристроїв розмір сектора становить 512 байтів (наприклад, на жорстких і гнучких дисках) або 2048 байтів (на оптичних дисках).

Система числення – сукупність правил і знаків, за допомогою яких можна відобразити (кодувати) будь-яке число. До систем числення висуваються певні вимоги, серед яких найбільш важливими є вимоги однозначного кодування невід'ємних чисел 0, 1, … до деякої їх скінченної множини за скінченне число кроків і можливості виконання з числами арифметичних і логічних операцій. Крім того, системи числення розв'язують задачу нумерації, тобто ефективного переходу від зображень чисел до номерів, які в такому випадку повинні мати мінімальну кількість цифр.

Центральний процесорний пристрій – електронний блок або інтегральна схема (мікропроцесор), що виконує машинні інструкції (код програм), головна частина апаратного забезпечення комп'ютера або програмованого логічного контролера. Іноді називають мікропроцесором або просто процесором.

Цикл – різновид керівної конструкції у високорівневих мовах програмування, призначена для організації багаторазового виконання набору інструкцій. Також циклом може називатися будь-яка послідовність інструк-

цій, що виконується багаторазово і організована довільним чином. Послідовність інструкцій, призначена для багаторазового виконання, називається тілом циклу. Одиничне виконання тіла циклу називається ітерацією. Вираз, що визначає, чи буде в черговий раз виконуватися ітерація або ж цикл завершиться, називається умовою виходу або умовою закінчення циклу. Змінна, що зберігає поточний номер ітерації, називається лічильником ітерацій циклу або просто лічильником циклу.

Файл – впорядкована сукупність даних, що зберігається на диску і займає іменовану область зовнішньої пам'яті. Згідно з термінологією, прийнятою в програмуванні, файл — це довільний блок інформації, своєрідний пристрій введення-виведення. Файл обов'язково має ім'я і може містити будь-який розмір інформації (максимальна довжина імені та обсяг файла обмежується властивостями конкретної файлової системи). Файл може мати набір атрибутів.

Файлова система – спосіб організації даних, який використовується операційною системою для збереження інформації у вигляді файлів на носіях інформації. Також цим поняттям позначають сукупність файлів та директорій, які розміщуються на логічному або фізичному пристрої. Створення файлової системи відбувається в процесі форматування. Сучасні файлові системи – це ієрархічні структури каталогів.

Форма – основний елемент користувацького інтерфейсу в ОС Windows. У Delphi майже кожен проект має принаймні одне вікно – головне вікно програми. Всі вікна в Delphi основані на об'єкті TForm. Форми мають свої властивості, події та методи, за допомогою яких можна управляти виглядом і поведінкою форми. Форма – це звичайний компонент Delphi, але на відміну від інших, її немає на панелі компонентів.

Ярлик – це спеціальний файл, у якому зберігається посилання на об'єкт (інформація про місце розташування об'єкта на жорсткому диску).

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Азарова А. О. Інформатика та комп'ютерна техніка. Ч. 1 : навчальний посібник / А. О. Азарова, А. В. Поплавський. – Вінниця : ВНТУ, 2012. – 361 с.
2. Архангельский А. Я. Программирование в Delphi 7 / Архангельский А. Я. – М. : ООО «Бином–Прес», 2003. – 1152 с.
3. Архітектура, принцип функціонування і керування ресурсами IBM PC : [навчальний посібник] / А. М. Гуржій, С. Ф. Коряк, В. В. Самсонов, О. Я. Скляров. – Харків : ТОВ «Компанія СМІТ», 2003. – 512 с.
4. Баженова И. Ю. Delphi 7: самоучитель программиста / Баженова И. Ю. - М. : Кудиц-Образ, 2003 – 448 с.
5. Басюк Т. М. Основи інформаційних технологій : навчальний посібник / Басюк Т. М., Думанський Н. О., Пасічник О. В.; за ред. В. В. Пасічника. – Львів : Новий Світ-2000, 2011. – 390 с.
6. Бобровский С. И. Delphi 7 : учебный курс / Бобровский С. И. – СПб. : Питер, 2004. – 736 с.
7. Ващук О. М. Програмне забезпечення комп'ютера : навчальний посібник / О. М. Ващук, В. О. Нелюбов. – Ужгород : ЗакДУ, 2008. – 132 с.
8. Веселовська Н. Р. Комп’ютерна техніка та програмування / Веселовська Н. Р. – Вінниця : ОЦ ВДАУ, 2000. – 249 с.
9. Злобін Г. Г. Архітектура та апаратне забезпечення ПЕОМ : навчальний посібник / Г. Г. Злобін, Р. Є. Рикалюк . – [2-ге вид., переробл. й допов.]. – К. : Каравела, 2012. – 224 с.
10. Інформатика: візуальне програмування : [навчально-методичний посібник] / О. С. Булгакова, В. В. Зосімов, Н. А. Броницька, Н. В. Танкова. – Херсон : Олді-плюс, 2014. – 312 с.
11. Інформатика. Комп'ютерна техніка. Комп'ютерні технології : підручник / [В. А. Баженов, П. С., Венгерський, В. С. Гарвона та ін.]. – [4-те вид.]. – К. : Каравела, 2012. – 496 с.
12. Козак Л. І. Основи програмування : навчальний посібник / Козак Л. І., Костюк І. В., Стасевич С. Л. – Львів : ЛДІНТУ ім. В. Чорновола, Новий Світ-2000, 2011. – 328 с.
13. Кузьмичов А. І. Оптимізаційні методи і моделі: моделювання засобами MS Excel : навчальний посібник / Кузьмичов А. І. – К. : Ліра-К, 2015. – 215 с.
14. Культин Н. Б. Основы программирования в Delphi 7 / Культин Н. Б. – СПб. : БХВ–Петербург, 2003. – 608 с.
15. Матвієнко М. П. Архітектура комп'ютера : навчальний посібник / Матвієнко М. П., Розен В. П., Закладний О. М. – К. : Ліра-К, 2013. – 264 с.
16. Матвієнко М. П. Комп'ютерна схемотехніка : навчальний посібник / М. П. Матвієнко, В. П. Розен. – К. : Ліра-К, 2013. – 192 с.

17. Мельник А. О. Архітектура комп’ютера / Мельник А. О. – Луцьк : Волинська обласна друкарня, 2008. – 470 с.

18. Назаров С. В. Архитектура и проектирование программных систем : монография / Назаров С. В. – М. : Инфра-М, 2013. – 351 с.

19. Пекарський Б. Г. Основи програмування : навчальний посібник / Пекарський Б. Г. – К. : Кондор, 2008. – 364 с.

20. Семеренко В. П. Системне програмне забезпечення. Системне об'єктно-орієнтоване програмування мовою С++. Курсове проектування. Самостійна та індивідуальна робота студентів : навчальний посібник / В. П. Семеренко. – Вінниця : ВНТУ, 2015. – 195 с.

21. Сергієнко І. В. Елементи загальної теорії оптимальних алгоритмів та суміжні питання : монографія / Сергієнко І. В., Задірака В. К., Литвин О. М. – К. : Наукова думка, 2012. – 400 с.

22. Шищук В. В. Основи програмування на алгоритмічній мові Pascal : навчальний посібник / Шищук В. В. – К. : Кондор, 2006. – 224 с.

Навчальне видання

Костішин Сергій Володимрович
Штофель Дмитро Хуанович
Белзецький Руслан Станіславович

ІНФОРМАТИКА
Частина 1

Лабораторний практикум

Рукопис оформлено: Д. Штофель

Редактор Т. Старічек

Оригінал-макет підготовлено: О. Ткачук

Підписано до друку 23.04.2018
Формат 29,7×42¼. Папір офсетний.
Гарнітура Times New Roman.
Друк різографічний. Ум. друк. арк. 6,78.
Наклад 50 (1-й запуск 1-20) пр. Зам. № 2018-076.

Видавець та виготовлювач
інформаційний редакційно-видавничий центр.
ВНТУ, ГНК, к. 114.
Хмельницьке шосе, 95,
м. Вінниця, 21021.
Тел. (0432) 65-18-06.
press.vntu.edu.ua;
E-mail: kivc.vntu@gmail.com.
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.