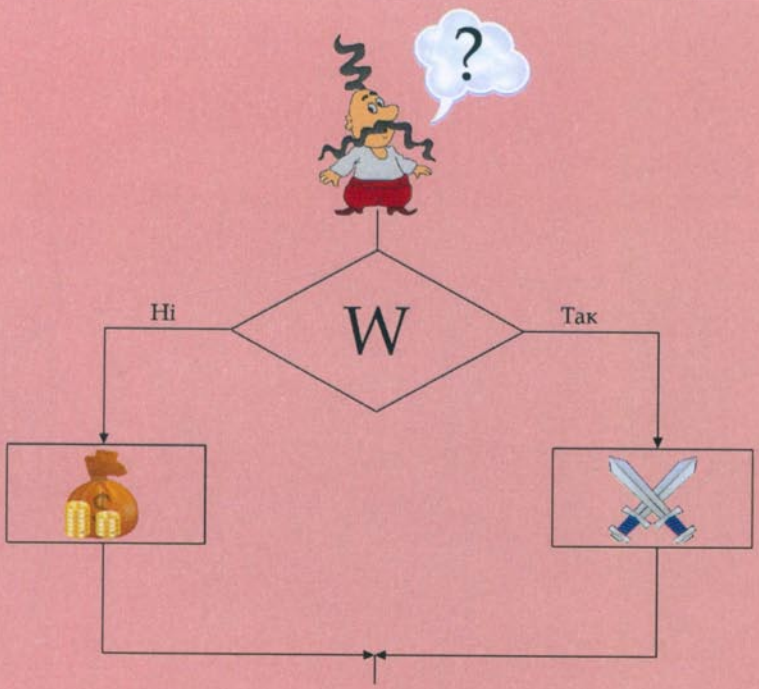
С. М. Москвіна, Т. В. Грищук

Комп'ютерні технології та програмування

Алгоритмічні основи програмування



Лабораторний практикум

Частина 1

Міністерство освіти і науки України
Вінницький національний технічний університет

С. М. Москівна, Т. В. Грищук

Комп'ютерні технології та програмування
Частина 1
Алгоритмічні основи програмування
Лабораторний практикум

Вінниця
ВНТУ
2014

УДК 681.3.06(075)
ББК 32.973я73
М82

Рекомендовано до друку Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України (протокол № 9 від 25.04.2013 р.).

Рецензенти:

В. М. Лисогор, доктор технічних наук, професор
В. А. Лужецький, доктор технічних наук, професор
О. М. Ткаченко, кандидат технічних наук, доцент

Москвіна, С. М.

М82 Комп'ютерні технології та програмування. Частина 1. Алгоритмічні основи програмування : лабораторний практикум / С. М. Москвіна, Т. В. Грищук – Вінниця : ВНТУ, 2014. – 116 с.

У лабораторному практикумі наведено теоретичні відомості, завдання та пояснення до практичного виконання лабораторних робіт, що охоплюють матеріал з математичних основ ЕОМ, теорії алгоритмів та програмування на мові С.

Призначений для студентів денної форми навчання напряму підготовки 6.050202 – «Автоматизація та комп'ютерно-інтегровані технології».

УДК 681.3.06(075)
ББК 32.973я73

ЗМІСТ

ВСТУП

Одним з головних шляхів оволодіння майстерністю програмування (*programming*) будь-якою мовою програмування (*programming language*) є виконання лабораторних завдань. Лабораторний практикум має допомогти студенту на простих прикладах оволодіти навичками розробки програмних продуктів мовою програмування високого рівня.

Практикум охоплює матеріал з математичних основ ЕОМ (*mathematical basics of computers*), теорії алгоритмів (*theory of algorithms*) та основ програмування мовою С. Лабораторний практикум складається з 13 лабораторних робіт, які призначені для виконання студентами у першому та другому триместрах вивчення дисципліни «Комп'ютерні технології та програмування».

Під час занять студенти повинні отримати корисний для майбутньої роботи досвід розробки консольних програм (*console applications*) відповідно до структурного підходу (*structural approach*) розробки програмного забезпечення, які на сьогодні широко використовуються в сучасних комп'ютеризованих системах управління (*computer control systems*).

Вимоги до лабораторних робіт

Лабораторні роботи повинні бути оформлені відповідно до ДСТУ 3008-95 і мати такі основні структурні елементи:

1. Титульний аркуш;
2. Текст завдання;
3. Хід виконання завдання з детальними поясненнями, що містять:
   - 3.1. Алгоритм розв'язання завдання: опис вхідних та вихідних даних, описовий алгоритм розв'язання завдання з доцільними поясненнями, схема алгоритму розв'язання завдання;
   - 3.2. Лістинги програм;
   - 3.3. Розробку тестів;
   - 3.4. Лістинги вхідних даних та результатів розв'язання завдання на ЕОМ;
   - 3.5. Аналіз отриманих результатів;
4. Висновки.

Вимоги до програм

Програмне забезпечення лабораторної роботи повинно ґрунтуватись на принципах структурного програмування. Програма має містити: блок опису вхідних та вихідних даних, введення початкових даних, програмний код зі змістовними коментарями, виведення результатів обчислень. Результати досліджень на ЕОМ необхідно зводити в таблиці та показувати графічно.

ЛАБОРАТОРНА РОБОТА № 1
ПОДАННЯ ІНФОРМАЦІЇ В ЕОМ

Мета роботи: навчитись переводити числа в системи числення, що використовуються в ЕОМ, підраховувати кількість інформації та вміти переводити значення кількості інформації з одних одиниць вимірювання в інші.

1.1 Порядок виконання роботи

1.1.1 Ознайомтесь з теоретичними відомостями до лабораторної роботи.

1.1.2 Візьміть у викладача номер варіанта для виконання індивідуального завдання. Цифра X в числах буде позначати номер вашого завдання. Всі розрахунки треба робити двома способами: вручну та з використанням спеціалізованих програм (в калькуляторі та/або в програмі MSExcel).

1.1.3 Переведіть з довільної системи числення в десяткову:

$X721{,}172_8$;
$X1011{,}001_2$;
$XD1A4{,}F3_{16}$.

1.1.4 Переведіть з десяткової системи числення в довільну:

$X64935_{10}$ → в систему числення з основою 16;
$X29_{10}$ → в систему числення з основою 2;
$X13_{10}$ → в систему числення з основою 2;
$X613_{10}$ → в систему числення з основою 8.

1.1.5 Переведіть десяткові дроби в довільну систему числення

$0{,}X25_{10}$ → в систему числення з основою 2;
$0{,}X75_{10}$ → в систему числення з основою 8;
$0{,}X28125_{10}$ → в систему числення з основою 2;
$0{,}X140625_{10}$ → в систему числення з основою 2.

1.1.6 Переведіть з бітів в Кбайти:

X429217 бітів;
X424719 бітів.

1.1.7 Переведіть з Кбайтів в біти:

X301 Кбайт;
X274 Кбайтів 317 байтів 2 біти.

1.1.8 Підрахуйте кількість інформації у вашому прізвищі, імені та по батькові, якщо вони між собою розділені пропуском і закодовані в коді ASCII та Unicode.

1.1.9 Напишіть висновки до лабораторної роботи.

1.2 Теоретичні відомості

Способи подання чисел

Система числення – це спосіб подання чисел цифровими знаками та правила дій над числами.

Системи числення можна розділити на непозиційні та позиційні системи числення. В **непозиційній системі числення** значення (величина) символу (цифри) не залежить від розташування в числі. **Позиційна система числення** має обмежену кількість символів і значення кожного символу чітко залежить від його позиції у числі. Кількість таких символів q називають **основою позиційної системи числення**. Головна перевага позиційної системи числення – це зручність виконання арифметичних операцій.

Окремі позиції в записі числа називають розрядами, а номер позиції – номером розряду. Число розрядів у записі числа називається його розрядністю і збігається з довжиною числа.

В двійкових числах (*binary numbers*) кожна цифра відповідає значенню одного біта (0 або 1), старший біт завжди записується зліва, після числа ставиться буква «b». Для зручності сприйняття тетроди можуть бути розділені пропусками. Наприклад, 1010 0101b.

Шістнадцяткові числа (*hexadecimal numbers*) – кожна тетрада подається одним символом 0...9, A, B, ..., F. Позначатись таке подання може порізному, тут використовується тільки символ «h» після останньої шістнадцяткової цифри. Наприклад, A5h. В текстах програм це ж число може позначатись і як 0xA5, і як 0A5h, в залежності від синтаксису мови програмування. Незначущий нуль (0) додається зліва від старшої шістнадцяткової цифри, щоб розрізняти числа і символічні імена.

Десяткові числа (*decimal numbers*) – кожний байт (слово, подвійне слово) подається звичайним числом, а ознаку десяткового подання (букву «d») зазвичай опускають. На відміну від двійкового і шістнадцятково запису, за десятковим записом важко в голові визначити значення кожного біта, що іноді необхідно робити.

Вісімкові числа (*octal numbers*) – кожна трійка бітів (розділення починається з молодшого розряду) записується у вигляді цифр 0–7, в кінці ставиться ознака «o». Вісімкова система є незручною через те, байт неможливо розділити рівно навпіл.

Алгоритм переведення чисел з однієї системи числення в іншу

1. З десяткової системи числення:

- розділити число на основу нової системи числення;
- знайти остачу від ділення цілої частини числа;
- записати всі остачі від ділення у зворотному порядку.

2. З двійкової системи числення:

- для переведення в десяткову систему числення необхідно знайти суму добутків основи 2 на відповідний степінь розряду;

- для переведення числа у вісімкову систему числення необхідно розбити число на тріади. Наприклад, $1000110 = 1\ 000\ 110 = 106_8$.

- для переведення числа з двійкової системи числення у шістнадцяткову необхідно розбити число на групи по чотири розряди. Наприклад, $1000110 = 100\ 0110 = 46_{16}$.

Переведення чисел в MSExcel

В таблиці 1.1 перераховані основні функції (для інтерфейсу російською та англійською мовами) для переведення чисел в програмі MSExcel.

Таблиця 1.1 – Функції переведення чисел

Конвертування числа у двійкову систему числення	
ВОСЬМ.В.ДВ (число) OCT2BIN (число)	конвертування числа з вісімкової системи числення
ДЕС.В.ДВ (число) DEC2BIN (число)	конвертування числа з десяткової системи числення
ШЕСТН.В.ДВ (число) HEX2BIN (число)	конвертування числа з шістнадцяткової системи числення
Конвертування числа у десяткову систему числення	
ДВ.В.ДЕС (число) BIN2DEC (число)	конвертування числа з двійкової системи числення
ВОСЬМ.В.ДЕС (число) OCT2DEC (число)	конвертування числа з вісімкової системи числення
ШЕСТН.В.ДЕС (число) HEX2DEC (число)	конвертування числа з шістнадцяткової системи числення
Конвертування числа у вісімкову систему числення	
ДВ.В.ВОСЬМ (число) BIN2OCT (число)	конвертування числа з двійкової системи числення
ДЕС.В.ВОСЬМ (число) DEC2OCT (число)	конвертування числа з десяткової системи числення
ШЕСТН.В.ВОСЬМ (число) HEX2OCT (число)	конвертування числа з шістнадцяткової системи числення
Конвертування числа у шістнадцяткову систему числення	
ДВ.В.ШЕСТН (число) BIN2HEX (число)	конвертування числа з двійкової системи числення
ВОСЬМ.В.ШЕСТН (число) OCT2HEX (число)	конвертування числа з вісімкової системи числення
ДЕС.В.ШЕСТН (число) DEC2HEX (число)	конвертування числа з десяткової системи числення

Переведення чисел в програмі Калькулятор ОС Windows

Щоб перевести число в іншу систему числення:

1. В меню **Вид** оберіть команду **Инженерный**;

2. Введіть число для перетворення;

3. Оберіть систему числення, в яку його потрібно перевести. Доступні шістнадцяткова (Hex), десяткова (Dec), вісімкова (Oct) та двійкова (Bin)

системи числення. Відповідні перемикачі розташовані під полем введення зліва;

4. Оберіть необхідну розрядність результату (під полем введення справа).

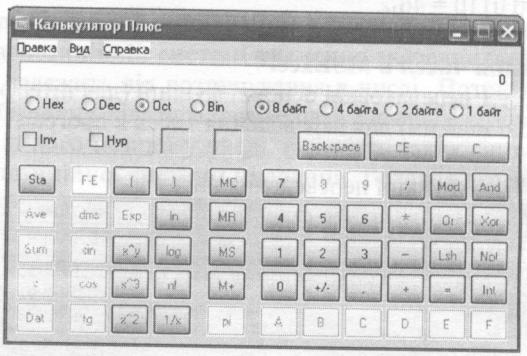


Рисунок 1.1 – Калькулятор Windows

При переведенні нецілого числа в іншу систему числення його дробова частина відкидається. Шістнадцяткові, вісімкові та двійкові числа, переведенні у десяткові, відображаються як цілі.

Розрахунок кількості інформації

Кількість інформації, яку вміщує один символ N-елементного алфавіту, дорівнює $i = \log_2 N$. Це відома формула Р. Хартлі. В 32-значному алфавіті кожний символ несе $i = \log_2 32 = 5$ (бітів) інформації. В кодуванні **ASCII** на кожний символ відводиться 1 байт = 8 бітів. В кодуванні **Unicode** на кожний символ відводиться 2 байти = 16 бітів.

1.3 Контрольні питання

1. Що таке «система числення»?
2. Класифікація систем числення.
3. Які системи числення використовуються в обчислювальній техніці?
4. Які символи використовуються як ознаки систем числення в програмуванні?
5. Які переваги і недоліки двійкової системи числення?
6. Алгоритм переведення чисел з десяткової системи числення в інші системи.
7. Функції програми MS Excel для переведення чисел.
8. Які обмеження є в програмі Калькулятор для переведення чисел з однієї системи числення в іншу?

ЛАБОРАТОРНА РОБОТА № 2
РОБОТА В ПРОГРАМІ FAR MANAGER

Мета роботи: здобути практичні навички роботи з об'єктами файлової системи Windows та навчитись користуватись спеціалізованими файловими менеджерами (*file managers*).

2.1 Порядок виконання роботи

2.1.1 Ознайомтесь з теоретичними відомостями до лабораторної роботи.

2.1.2 Запустіть програму Far Manager.

2.1.3 Зайдіть у спадне меню (*drop-down menu*) оболонки (F9). Ознайомитись з командами усіх пунктів меню.

2.1.4 Перевірте, скільки кілобайтів (*kilobytes*) займає кожний файл (*file*) на диску (*dick*) G:\. Яка місткість флеш-карти (*flash card*) та скільки на ній вільної пам'яті?

2.1.5 Створіть у кореневому каталозі диску G:\ файл VEL.TXT будь-якого змісту. Змініть ім'я файлу VEL.TXT на нове ім'я OPIS.TXT. Переконайтесь, що зміна імені файлу відбулася. Перегляньте вміст файлу OPIS.TXT.

2.1.6 Створіть у кореневому каталозі (*root directory*) вашого диску підкаталог (*subdirectory*) ROM. Скопіюйте файл OPIS.TXT у підкаталог ROM з іменем VEL.TXT.

2.1.7 Зробіть поточним каталогом каталог ROM. В каталозі ROM створіть підкаталог TER. В цьому каталозі створіть файл ASD.TXT будь-якого змісту. Скопіюйте файл зі зміною імені. Потім знищіть підкаталог TER. Переконайтесь, що ваші дії виконані правильно.

2.1.8 Вимкніть панелі оболонки з екрана і відновіть їх. Вимкніть, а потім відновіть тільки ліву панель оболонки, потім поміняйте панелі оболонки місцями.

2.1.9 Установіть на лівій панелі оболонки коротку форму інформації, а на правій – повну.

2.1.10 Перейдіть на другу активну панель оболонки. Виділіть та відмініть виділення групи файлів.

2.1.11 Викличте на першу панель оболонки дерево каталогів (*tree panel*). Вимкніть на екрані дерево каталогів.

2.1.12 Виведіть на екран інформацію про диск і каталог.

2.1.13 Визначити інформацію про пам'ять комп'ютера з панелі інформації (*info panel*).

2.1.14 Виведіть файли в алфавітному порядку їх імен, в алфавітному порядку їх розширень, у порядку зменшення їх розміру, у порядку зменшення дати останньої модифікації.

2.1.15 Виведіть у ліву панель оболонки тільки файли з розширенням TXT.

2.1.16 Виділіть файли за маскою. Відмітьте виділення файлів за маскою.

2.1.17 Здійсніть пошук файлів типу cpp на диску.

2.2 Теоретичні відомості

Для простоти використання можливостей ОС існує клас спеціальних програм, які поєднують переваги графічного інтерфейсу (*graphical interface*) і функціональність командного рядка (*command line*) і називаються файловими менеджерами. Програма Far Manager є яскравим прикладом такого класу службових програм.

Для запуску програми необхідно клацнути на його значку (*shortcut*) на робочому столі (*desktop*).

На російську мову Far перемикається через пункт головного меню **Options –> Languages**, у вікні **Main language** (Основна мова) і **Help language** (Мова довідки) потрібно обрати пункт **Russian** (Російська).

Інтерфейс Far manager

Зовнішній вигляд програми Far показаний на рисунку 2.1. У вікні програми відображаються дві панелі, на яких виводиться інформація про файли, каталоги та інші додаткові дані в залежності від режиму. В Far використовується кольорове виділення типів файлів и папок; наприклад, каталоги виділяються білим кольором, виконувані файли – зеленим і т. д.

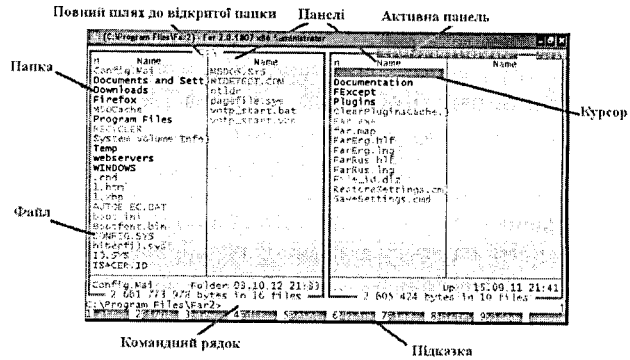


Рисунок 2.1 – Інтерфейс Far Manager

Під панелями знаходиться командний рядок, за допомогою якого можна вводити команди ОС. Нижче розміщується стисла довідка про призначення функціональних клавіш (*functional keys*). За кожною функціональною клавішею закріплено декілька дій, які викликаються при простому на-

тисканні на функціональну клавішу або у поєднанні з клавішами-модифікаторами Shift, Ctrl та Alt.

Розглянемо основні функції програми Far Manager.

Виклик довідки – F1 (Help).

Переміщення курсора: клавішами Page Up (на екран вгору), Page Down (на екран униз), Home (на початок каталогу) і End (у кінець каталогу) та мишкою.

Перегляд вмісту каталогу: для входу в каталог необхідно навести курсор (*cursor*) на його ім'я і натиснути Enter або двічі клацнути лівою кнопкою миші; для виходу – навести курсор на «..» і натиснути Enter.

Перехід у кореневий каталог – Ctrl+\.

Перехід на протилежну панель – клавіша Tab.

Приховати/вивести панелі – Ctrl+O.

Перехід на інший диск

1. Натисніть Alt+F1 для переходу на інший диск у лівій панелі чи Alt+F2 – у правій.

2. Наведіть курсор миші на символ поточного режиму сортування панелі, натисніть ліву кнопку миші та оберіть ім'я потрібного диска, як показано на рисунку 2.2.

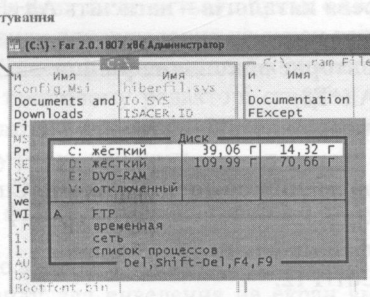


Рисунок 2.2 – Перехід на інший диск

Виділення файлів

1. Для виділення одного файлу наведіть на нього курсор.

2. Для виділення групи файлів наведіть курсор на кожний з них і натисніть клавішу Insert або натисніть на імені файлу правою кнопкою миші. Повторне натискання Insert або натискання правою кнопкою миші знімає виділення.

3. Для виділення групи файлів за шаблоном (*template*) (за маскою) натисніть клавішу «+» на додатковій клавіатурі, введіть шаблон і натисніть

Enter. Для зняття виділення натисніть «-» на додатковій клавіатурі, введіть шаблон і натисніть Enter.

Копіювання файлів: виділіть потрібний файл або групу файлів на одній з панелей. На іншій панелі відкрийте каталог, в який потрібно скопіювати обрані файли та натисніть клавішу F5 або перетягніть файл, утримуючи ліву кнопку миші.

Швидкий перегляд текстових файлів (*text files*): натисніть Ctrl+Q: у неактивній панелі буде виводитись вміст виділеного курсором файлу.

Пошук файлів

1. Натисніть Alt+F7.
2. Введіть ім'я файлу або шаблон і натисніть Enter.
3. У вікно буде виведено список знайдених файлів.
4. Щоб перейти до одного з них, виділіть його курсором і натисніть Enter.

Створення каталогу

1. Перейдіть у каталог, у якому потрібно створити підкаталог.
2. Натисніть F7.
3. Введіть ім'я каталогу і натисніть Enter.

Перегляд дерева каталогів – натисніть Alt+F10.

Перегляд і повторне виконання раніше введених команд

1. Натисніть Alt+F8.
2. Виділіть курсором потрібну команду і натисніть Enter.

Перегляд інформації про диск: натисніть Ctrl+L.

Сортування файлів у панелях

1. Натисніть Ctrl+F12.
2. У переліку режимів сортування оберіть потрібний.

2.3 Контрольні питання

1. Як запустити програму на комп'ютері?
2. Яка інформація міститься на кожній з панелей?
3. Яка з панелей є поточною? Чому?
4. Де розміщується рядок міністатусу? Для чого він призначений?
5. Де розміщений командний рядок? Для чого він використовується?
6. Що таке рядок функціональних клавіш? Для чого він призначений?
7. Що таке меню? Опишіть призначення кожного з пунктів меню.

ЛАБОРАТОРНА РОБОТА № 3
КОМАНДНИЙ РЯДОК WINDOWS. РОБОТА З ФАЙЛОВОЮ СИСТЕМОЮ

Мета роботи: оволодіти основними принципами роботи з командним рядком для автоматизації роботи з файловою системою (*file system*).

3.1 Порядок виконання роботи

3.1.1 Ознайомтесь з теоретичними відомостями до лабораторної роботи. Коротко занотуйте синтаксис запису основних команд для роботи з операційною системою.

3.1.2 Скопіюйте у каталог своєї бригади по декілька файлів з різними розширеннями.

3.1.3 Перейдіть до командного рядка меню **Пуск** або програми **Far Manager** та перейдіть у каталог своєї бригади.

3.1.4 У командному рядку:

- введіть команду для виведення на екран всіх файлів поточного каталогу;
- введіть команду для виведення на екран всіх файлів поточного каталогу, що мають розширення *.txt або *.doc;
- введіть команду для виведення на екран всіх файлів поточного каталогу, ім'я яких починається з букви «А»;
- введіть команду для виведення на екран всіх файлів поточного каталогу, ім'я яких складається з 4-х символів.

3.1.5 У будь-якому текстовому редакторі створіть текстовий файл з розширенням *.bat. Запишіть в цей файл всі команди, які Ви записували для п. 3.1.4, в командному рядку напишіть

ім'я командного файлу, пропуск, > файл_з_результатами.txt

та проаналізуйте файл, що з'явиться після виконання цієї інструкції.

3.1.6 Запишіть команду для виведення на екран вмісту файлу, який Ви отримати в попередньому пункті.

3.1.7 Запишіть команду для створення нового каталогу всередині робочого каталогу.

3.1.8 Запишіть команду, щоб зробити поточним щойно створений каталог.

3.1.9 Поверніться на рівень вище. Оберіть будь-який файл та виконайте над ним операції перейменування, копіювання (в каталог, який Ви створили в п. 3.1.7) та видалення (з каталогу, який Ви створили в п. 3.1.7).

3.1.10 Знаходячись в робочому каталозі, запишіть команду для видалення каталогу, який Ви створили в п. 3.1.7.

3.1.11 Запишіть команди для перевірки поточної дати та часу, проана-

лізуйте формат запиту (*query*) операційної системи, у відповідь на запит введіть правильно поточні дату та час.

3.1.12 Запишіть команду для перевірки версії операційної системи, з якою Ви працюєте, та проаналізуйте результат обробки вашого запиту.

3.1.13 Запишіть команду очищення консолі.

3.1.14 Складіть звіт до лабораторної роботи. У звіті вкажіть назву, мету, завдання та хід лабораторної роботи. В ході роботи вказуйте як команди з шляхами і назвами файлів, які Ви набираєте, так і всі повідомлення, які виводяться на екран в результаті введених команд. Запишіть висновки до роботи.

3.2 Теоретичні відомості

Файлова система – сукупність програм, що забезпечують роботу з файлами і каталогами, а також самі файли і каталоги, що зберігаються на пристроях зовнішньої пам'яті.

Файл – це програма (*program*) або організована сукупність даних, що має свою назву і зберігається на пристроях зовнішньої пам'яті як єдине ціле. Розрізняють файли даних і програмні файли.

Назва файлу – це ідентифікатор, що використовується для звертання до файлу.

Назва файлу	Ім'я файлу	.	Розширення (тип) файлу

Ім'я файлу (*file name*) найчастіше характеризує внутрішній вміст файлу.

За способом найменування файлів розрізняють «**коротке**» і «**довге**» ім'я файлу.

Коротке ім'я (*short name*): на ім'я файлу виділяється 8 символів, а на його розширення – 3 символи.

Довге ім'я (*long name*) може містити до 255 символів. Не дозволяється використовувати такі спеціальні символи: / \ ^ * ? " < > |. В імені дозволяється використовувати пропуски і кілька крапок.

Ім'я файлу надає користувач (*user*), а розширення (*extension*) надається програмою автоматично.

За символи в іменах файлів допускаються латинські літери (великі і малі), літери інших алфавітів, цифри, пропуски і спеціальні символи: ~ # $ % & () _ {} ; = [].

Маска файлу (*file mask*) – подання імені та розширення файлу загальними символами. Двома основними символами, що використовуються в масках файлів, є:

* - будь-яка кількість будь-яких символів;

? - будь-який один символ.

Розширення (тип) файлу використовується для класифікації файлів, визначення належності до певної групи із загальними ознаками, утворюється не більше ніж з 3-4 символів і є необов'язковим.

Тип файлу визначає користувач або програма, в якій створювався файл. Приклади розширень файлів:

TXT, RTF – текстові файли;
DOC – текстові файли, створені за допомогою редактора;
HLP – файли довідки;
C, CPP, PAS – файли, що містять програми мовами програмування;
BAT – командні файли;
COM, EXE – виконувані файли (готові до виконання програми);
JPG, BMP, CDR, TIFF – графічні файли.

Каталог (директорія (*directory*), папка (*folder*) – це іменований запис в таблиці файлової системи диску, в якому реєструються відомості про файли (ім'я, розмір, властивості і т. д.). Каталог може бути вкладений в інший каталог (тобто ім'я каталогу зберігатися в іншому каталозі разом з іменами звичайних файлів). У цьому випадку утворюється ієрархічна деревоподібна структура:

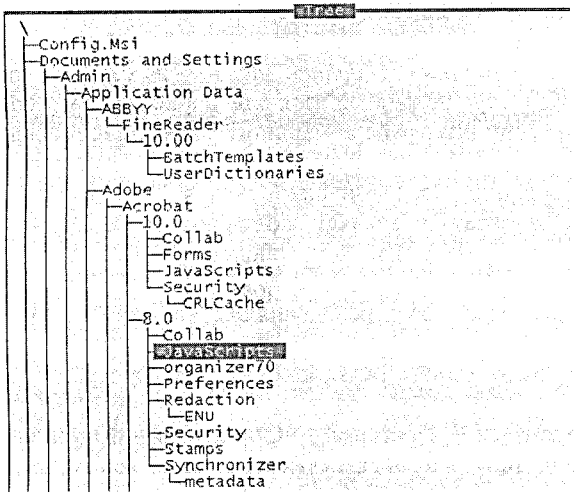


Рисунок 3.1 – Дерево каталогів

Кореневий каталог – це головний каталог кожного диска. Імена каталогів, поміщених один в другий, відокремлюються символом «\».

Шлях (маршрут) до файлу (*file path*) – це послідовність з імен від кореневого каталогу до того каталогу, в якому знаходиться необхідний файл.

Накопичувач (*drive*) – пристрій зовнішньої пам'яті, на якому розміщується файлова система диску. Накопичувачі прийнято позначати літерами англійського алфавіту з наступною двокрапкою:

A: і B: – дисководи для гнучких магнітних дисків;

C:...Z: – логічні диски вінчестера, пристрої для оптичних, магнітооптичних, змінних дисків, логічні мережеві диски.

Повне ім'я файлу (*file full name*) – це назва накопичувача, шлях до файлу і сама назва файлу.

Наприклад: C:\USER\ABC\MYFILE.EXE

Поточний каталог (*current directory*) – каталог, з яким у даний час працює користувач.

Операційна система здійснює доступ (*access*) до файлу або каталогу через шлях до нього. Існує два види шляхів: абсолютні та відносні.

Абсолютний шлях (*absolute path*) – це послідовність імен каталогів, що починається з імені кореневого каталогу і, проходячи по дереву імен каталогів, закінчується іменем каталогу або файлу, з яким нам потрібно працювати.

Відносний шлях (*relative path*) може бути вказаний від поточного або робочого каталогу, що може бути значно стисліше і зручніше, ніж використання абсолютного шляху. Коли ми знаходимось в робочому каталозі, то нам не потрібно вказувати шляхи до каталогів і файлів, які в ньому знаходяться. До інших каталогів потрібно вказувати шлях, але не абсолютний, а набагато коротший.

Командна оболонка (*command shell*) — це окремий програмний продукт, що забезпечує прямий зв'язок між користувачем і операційною системою. Текстовий інтерфейс командного рядка надає середовище, в якому виконуються прикладні та службові програми з текстовим інтерфейсом. В командній оболонці програми виконуються, і результат виконання відображається на екрані у вигляді, який схожий на вигляд інтерпретатора Command.com MS-DOS. Командна оболонка Windows використовує інтерпретатор команд Cmd.exe, що завантажує програми та направляє потік даних між програмами для переведення введеної команди у зрозумілий системі вигляд.

Доступ до командного рядка можна отримати через:

1. Меню Пуск → Програми → Стандартні → Командний рядок;
2. Діалог **Запуск програми** (меню Пуск), в якому ввести **cmd**;
3. Будь-який файловий менеджер.

Командний рядок наведений на рисунку 3.2. Перше питання, що виникає після входу в систему: «Де я?». Відповідь на це питання міститься в запрошенні командного рядка.

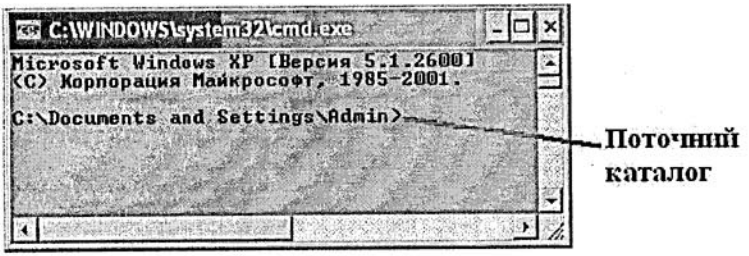


Рисунок 3.2 – Командний рядок

Команди

Користувач може керувати операційною системою шляхом введення команд (*commands*). Для того, щоб ввести команду в командний рядок, треба набрати її з клавіатури та натиснути клавішу «Enter». Стрілочками «←» та «→» можна пересуватись по тексту, який вводиться. Стрілочками «↑» та «↓» можна пересуватись по пам'яті введених команд. Наприклад, натискання клавіші управління курсором «↑» виводить на екран попередню команду. Для її запуску слід знову натиснути «Enter».

Будь-яка команда складається з таких складових:

команда [шляхи\назви файлів] [/параметри].

Квадратні дужки означають необов'язковий параметр (*optional parameter*). Наприклад, шлях може не вказуватись, якщо команда застосовується до файлів в поточному каталозі. Наприклад, «**ren aaa bbb**» — змінити в поточному каталозі назву файлу «aaa» на назву «bbb». Якщо ж він вказується, то він складається з назви диска з файлом (A: E: D: C:), назви каталогу та назв підкаталогів — всі ці складові набираються в такому ж порядку і відокремлюються похилою рискою «слешем». Далі через такий же «слеш» йде назва файлу. Наприклад, шлях до цього файлу, що Ви зараз читаєте, має такий вигляд (може бути трохи інший шлях тому \...\ ): **D:\...\ 1KURS\AC-05\ms_dos1.doc**.

Назва файлів може не вказуватись, якщо ведеться робота з усіма файлами поточного каталогу. Наприклад, команда «**dir**» виводить всі файли поточного каталогу, що показано на рисунку 3.3.

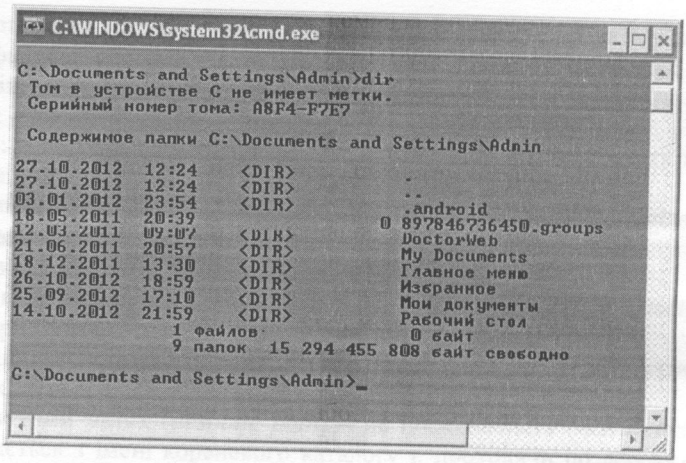


Рисунок 3.3 – Результат виконання команди dir

З іншого боку, може вказуватись не тільки один файл, а група файлів, схожих за якоюсь загальною ознакою, як правило, схожих за назвою або розширенням. Для виділення цих груп файлів використовують маску.

Параметри вводять додаткові відомості щодо застосування команди і вказуються лише в деяких командах. Як правило, це робиться у вигляді «/параметр», тобто через «пропуск» та нахилену риску «/», після якої одразу записується параметр. Наприклад, команда з параметром «**dir /p**» — виводить всі файли поточного каталогу з розбиттям на сторінки — виведе сторінку і чекає, поки користувач ознайомиться з текстом і натисне будь-яку клавішу.

З повним переліком команд можна ознайомитись за допомогою команди «**help**». Для виведення довідки про конкретну команду потрібно ввести «**help назва команди**».

Для виведення результатів виконання команд не на екран, а у файл, можна використати перенаправлення виведення. При використанні командного рядка стандартним пристроєм введення є клавіатура, а пристроєм виведення – дисплей, однак ці пристрої можна перепризначити з використанням символьного перенаправлення:

< – перенаправлення введення;

> – перенаправлення виведення (або >> – перенаправлення в існуючий файл, коли дані дописуються в кінець файлу).

Для виведення потоку даних команди **help** в файл help.txt командний рядок буде таким: **help > help.txt**.

Основні команди (якщо команда застосовується до файлу в поточному каталозі, тоді «шлях» не вказується).

1) робота з файлами:

copy шлях1\назва файлу1 шлях2 — копіювання файлу1 зі шляхом1 по шляху2 під тією ж назвою «файл1»;

copy шлях1\назва файлу1 шлях2\назва файлу2 — копіювання файлу1 зі шляхом1 по шляху2 під назвою «файл2»;

ren стара назва файлу1 нова назва файлу1 — заміна старої назви файлу1 на нову;

type [шлях\назва файлу] — виведення на екран вмісту файлу;

del назва файлу1 — видалення файлу1;

2) робота з каталогами:

dir [назва каталогу] — виведення на екран назв всіх файлів в каталозі;

cd [назва нового каталогу] — змінити поточний каталог (повернутися назад на один каталог можна, набравши дві крапки ..);

md [назва нового каталогу] — утворити новий каталог;

rd [назва каталогу] — видалення каталогу;

3) загальносистемні команди:

cls — очищення екрана;

time — виведення на екран поточного часу;

date — виведення на екран поточної календарної дати;

exit — завершення роботи з командним рядком;

ver — виведення на екран номера версії операційної системи.

Командні файли – це текстові файли з розширенням **bat** або **cmd**, рядки яких являють собою команди або імена виконуваних файлів. Коли Ви запускаєте на виконання командний файл, то управління отримує командний процесор операційної системи, який послідовно зчитує та інтерпретує рядки командного файлу.

3.3 Контрольні питання

1. Що таке операційна система? Основні функції ОС.
2. Що таке файлова система?
3. Що таке розширення файлу? Що визначає розширення файлу.
4. Які існують обмеження на імена файлів в операційній системі Windows? Чим відрізняються довгі імена від коротких?
5. На які групи можна розділити всі команди командної мови?
6. Перерахуйте відомі Вам загальносистемні команди.
7. Як отримати довідку про команду?
8. Що означають символи [ ] в синтаксисі запису команди?
9. Як створити командний файл?
10. Як записати результати виконання команд в текстовий файл?

ЛАБОРАТОРНА РОБОТА № 4
РОЗРОБКА БЛОК-СХЕМ АЛГОРИТМІВ

Мета роботи: набути практичні навички розробки блок-схем (*flow chart*) алгоритмів в середовищі MS Visio.

4.1 Порядок виконання роботи

4.1.1 Ознайомтесь з теоретичними відомостями до лабораторної роботи.

4.1.2 Відповідно до номера індивідуального завдання виконайте завдання з побудови блок-схеми у середовищі MS Visio.

4.1.3 Відповідно до номера індивідуального завдання виконайте пошук чисельного алгоритму та запишіть його на псевдокоді та у вигляді блок-схеми.

4.1.4 Складіть звіт за результатами лабораторної роботи.

4.1.5 Зробіть висновки.

4.2 Теоретичні відомості

Алгоритм (*algorithm*) – це спосіб розв'язання задачі у вигляді точної інструкції для отримання результату на основі вихідних даних (*input data*).

Слово «алгоритм» походить від імені великого вченого середньовічного Сходу Аль-Хорезмі. Він жив приблизно з 780 по 850 р. і сформулював правила виконання чотирьох арифметичних дій. Ці правила і називали алгоритмами.

Базові структури алгоритму (*algorithm basic structures*) – структури (*structures*), за допомогою яких створюється алгоритм для розв'язання певної задачі.

Основна особливість базових алгоритмічних структур – їх повнота (*completeness*), тобто цих структур достатньо для створення найскладнішого алгоритму.

Розрізняють структури слідування, вибору та повторення.

Лінійний алгоритм (*linear algorithm*) – це такий алгоритм, в якому всі операції виконуються послідовно одна за одною.

Алгоритми розгалуженої структури (*conditional structure*) застосовуються, коли в залежності від деякої умови необхідно виконати одну чи іншу дію.

Циклом (*cycle*) називають повторення одних і тих самих дій (кроків). Послідовність дій, які повторюються в циклі, називають тілом циклу.

Циклічні алгоритми підрозділяють на алгоритми з передумовою, післяумовою і алгоритми з кінцевим числом повторів. В алгоритмах з передумовою спочатку виконується перевірка умови закінчення циклу і потім, в

залежності від результату перевірки, виконується (або не виконується) так зване тіло циклу.

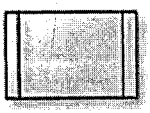
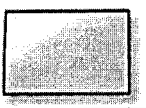
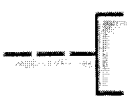
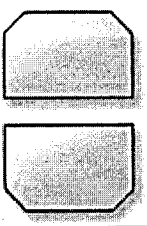
Процес алгоритмізації – це визначення елементарних дій та порядку їх виконання для розв'язання поставленої задачі. Різні способи записування алгоритмів застосовуються для подання алгоритму у вигляді, який однозначно розуміється і розробником, і виконавцем алгоритму.

Алгоритми можна описувати у словесній або словесно-формульній формі, у графічному вигляді та за допомогою мов програмування.

Блок-схема алгоритму – це графічне зображення алгоритму у вигляді спеціальних блоків з необхідними словесними поясненнями.

Кожний блок в межах однієї роботи повинен мати однакові розміри. Блоки об'єднуються між собою лініями потоку. Лінії потоку будуть зверху вниз і зліва направо. Якщо потік рухається у протилежному напрямку то його рух вказується стрілками. В таблиці 4.1 наведено основні блоки, з яких складаються блок-схеми алгоритмів.

Таблиця 4.1 – Спеціальні блоки

Блок	Назва	Блок	Назва
	Початок (кінець)		Виклик підпрограми
	Введення, виведення даних		Цикл за параметром
	Перетворення		З'єднувач
	Розгалуження		Коментар
	Межа циклу		

Microsoft Visio — це професійний програмний продукт для розробки різного виду схем.

Пакет Microsoft Visio може стати помічником у розв'язанні трьох основних задач: у аналізі складних даних, в графічному поданні даних і в обміні цими даними між користувачами. Основний засіб подання даних в Visio — це векторні фігури, на основі яких будується діаграма або план. Для зручності фігури згруповані за тематичними категоріями, в кожній з яких можна побачити схожі на вигляд або за темою елементи. Фігури є основним, але не єдиним засобом для подання даних в Visio. Окрім них можна також використовувати текст і числові дані, графічні елементи і форматування кольором. Основні елементи графічного інтерфейсу програми показано на рисунку 4.1.

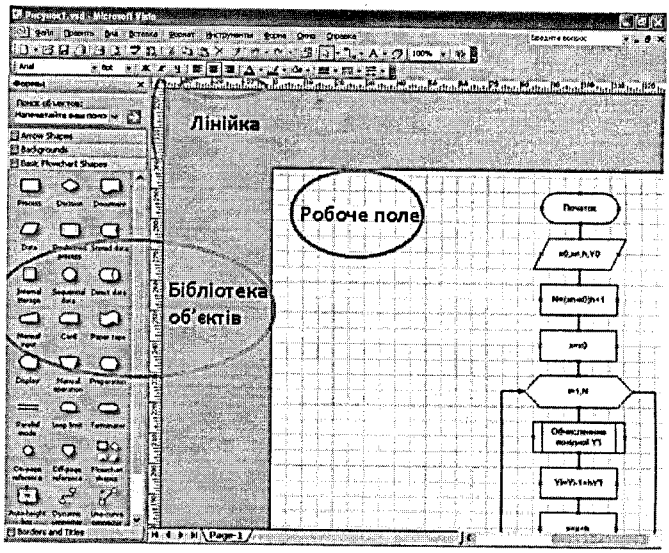


Рисунок 4.1 – Інтерфейс MS Visio

На панелі **Формы** (Shapes) відображаються доступні для використання в схемі фігури. Приклад бібліотеки основних фігур блок-схем наведено на рисунку 4.2.

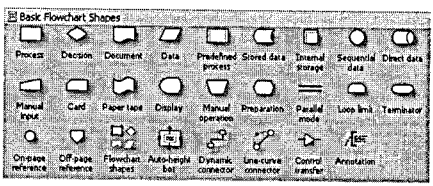


Рисунок 4.2 – Фігури для розробки блок-схем алгоритмів

Існує багато типів схем Visio, але для створення практично всіх документів можна скористатись трьома основними діями:

1. Вибір і відкриття шаблону;
2. Перетаскування та з'єднання фігур;
3. Додання тексту у фігури.

Розглянемо дії для створення простої блок-схеми.

Дія 1. Вибір і відкриття шаблону.

1. Запустіть Visio.
2. Натисніть кнопку категорії **Блок-схема (Flowchart)**.
3. Оберіть шаблон **Простая блок-схема (Basic Flowchart)**.

Дія 2. Перетаскування та з'єднання фігур.

Щоб створити блок-схему, перетаскуйте фігури з панелі **Формы** (Shapes) на робочу область та з'єднуйте їх одна з одною.

1. Перетягніть на робочу область фігуру **Начало/конец** (Terminator). За допомогою маркерів розміру та повороту встановіть потрібні розміри фігури, як показано на рисунку 4.3.

Рисунок 4.3 – Приклад рисування фігури

2. Перетягніть на робочу область наступну фігуру.
3. З'єднайте фігури за допомогою інструмента з'єднання, як показано на рисунку 4.4.

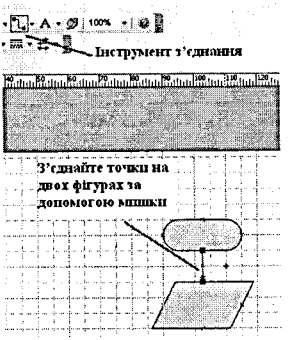


Рисунок 4.4 – З'єднання фігур

З'єднання фігур здійснюється в трьох формах:
- з'єднування з правильними кутами;
- пряме з'єднування;
- з'єднування по кривій.

Вид з'єднувача можна обирати через контекстне меню, як показано на рисунку 4.5.

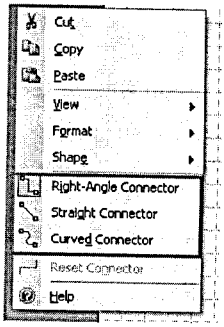


Рисунок 4.5 – Види з'єднувачів

З'єднувач правильного кута з'єднує два об'єкти лініями, кут яких кратний 90, та оминає інші об'єкти на шляху. Прямий з'єднувач з'єднує два об'єкти прямою лінією. З'єднувач по кривій закруглює лінію, що з'єднує два об'єкта.

Дія 3. Додавання тексту до фігури

1. Клацніть фігуру і почніть вводити текст.
2. Після завершення введення клацніть на пустому місці сторінки або натисніть клавішу Esc.

Цей спосіб дозволяє додавати текст практично до будь-якої фігури, навіть до з'єднувальних ліній.

Групування фігур

При створенні складних схем часто виникає необхідність об`єднати декілька простих фігур в одну. Дана операція називається **групуванням**. Для групування фігур необхідно спочатку вибрати декілька фігур.

Інструменти, які дозволяють користуватись функцією групування, у середовищі Visio доступні через головне меню **Форма** (Shape) – **Группировка** (Grouping) – **Группировать** (Group) або **Разгруппировать** (Ungroup) або через контекстне меню **Форма** (Shape).

Використання сітки

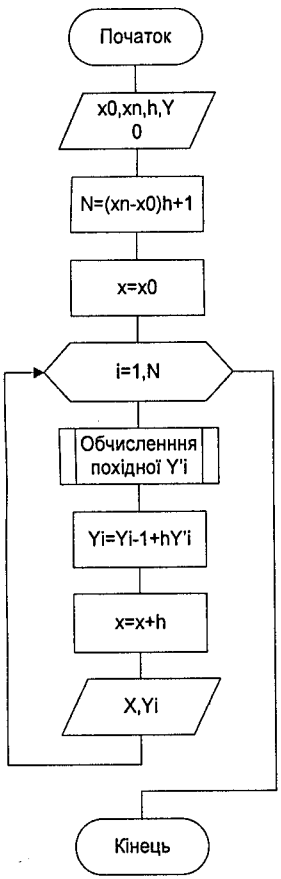
Сітка допомагає точно розташувати фігури та скорегувати їх розміри.

Сітка вмикається/вимикається через головне меню **Вид** (View) – **Сетка** (Grid).

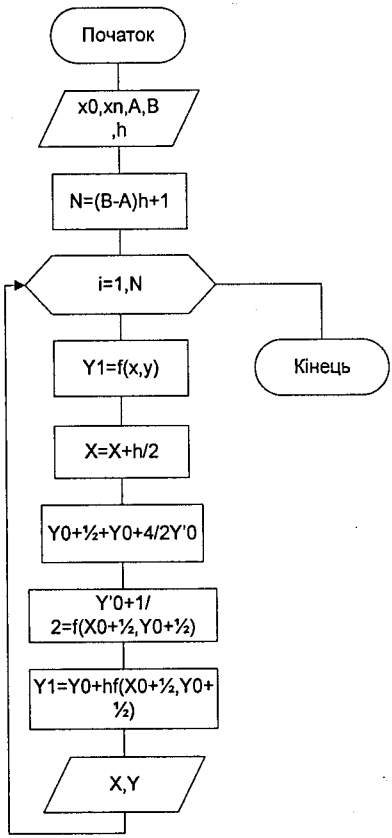
Крок сітки встановлюється автоматично згідно з масштабом зображення екрана в кратній формі, таким чином, при збільшенні масштабу ми отримуємо більш деталізовану сітку, не втрачаючи попередніх ліній і місця їх розташування.

4.3 Варіанти завдання 1

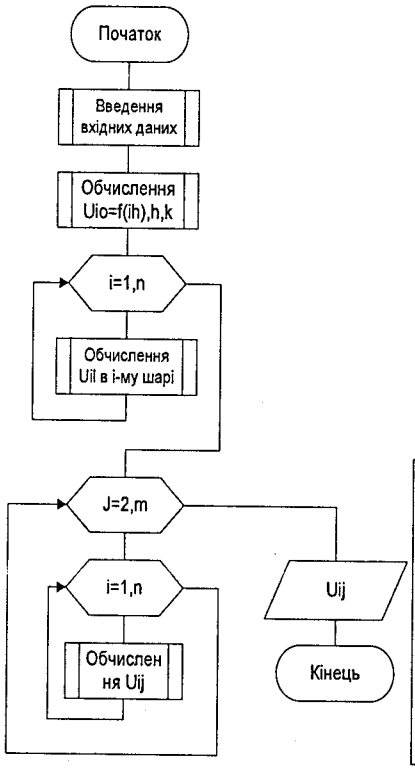
Варіант № 1



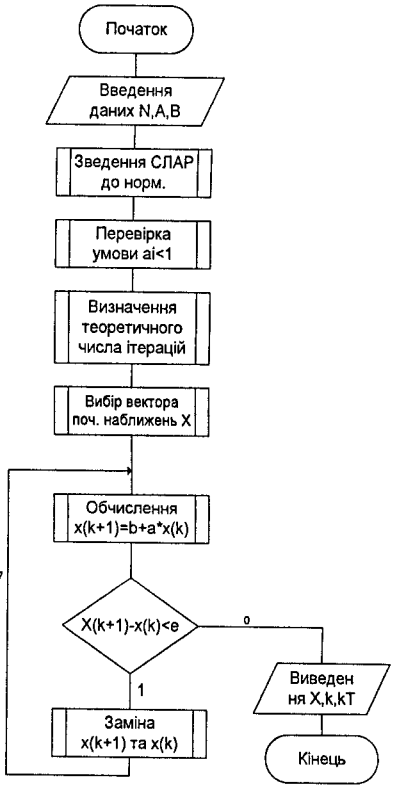
Варіант № 2



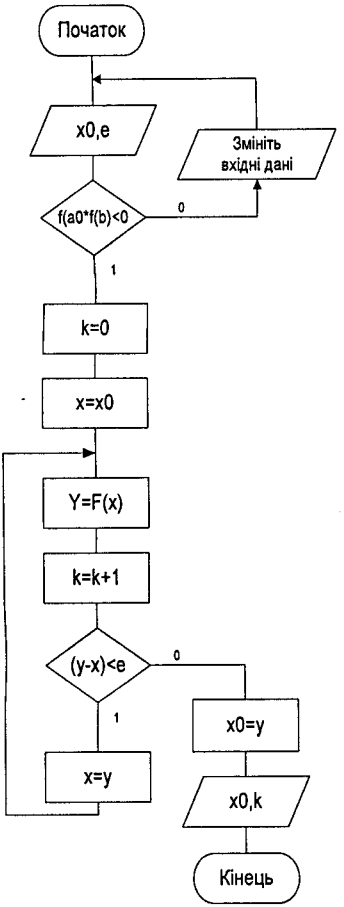
Варіант № 3



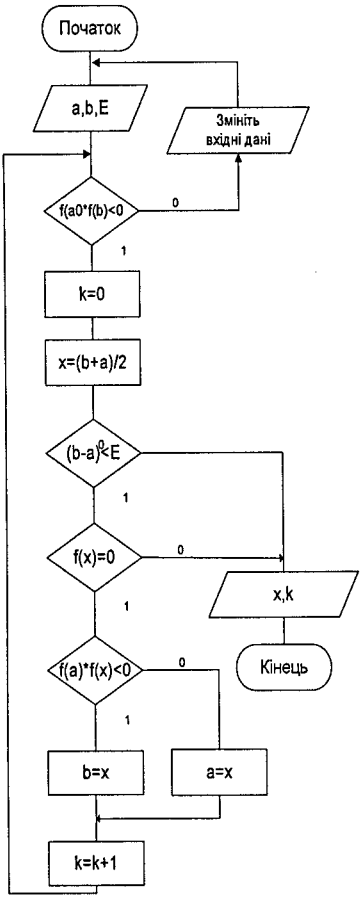
Варіант № 4



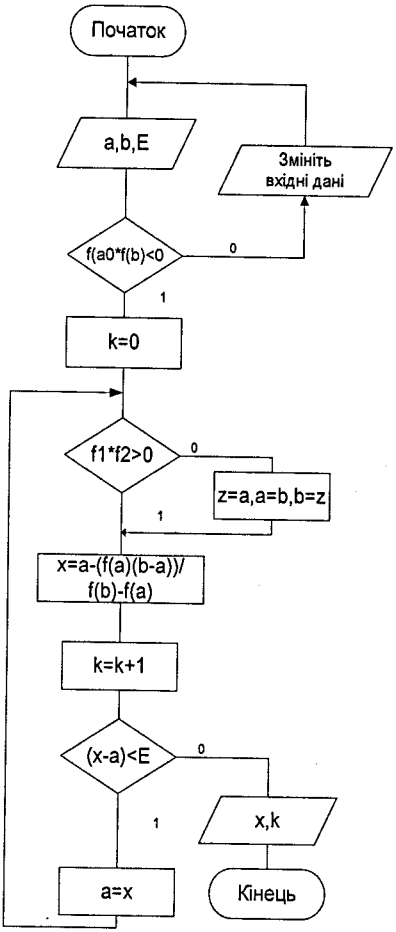
Варіант № 5



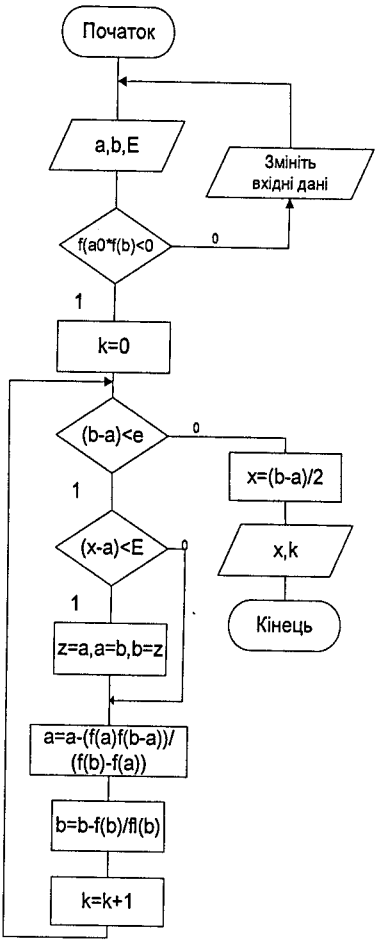
Варіант № 6



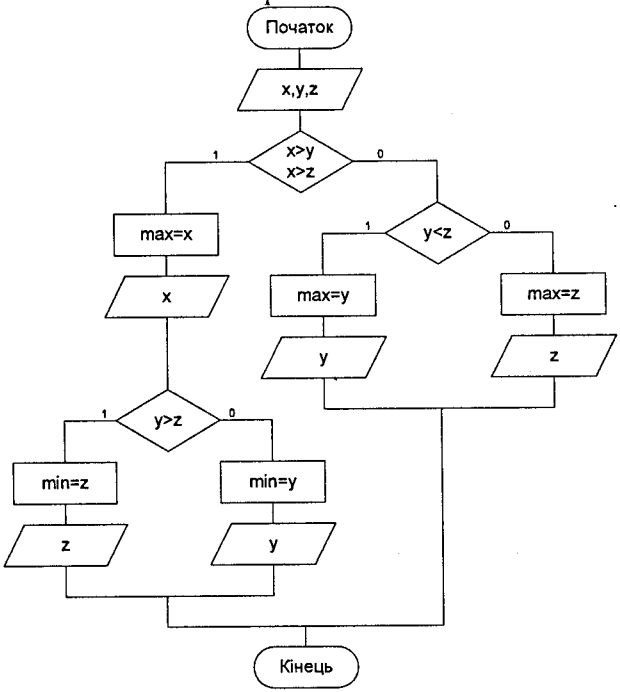
Варіант № 7



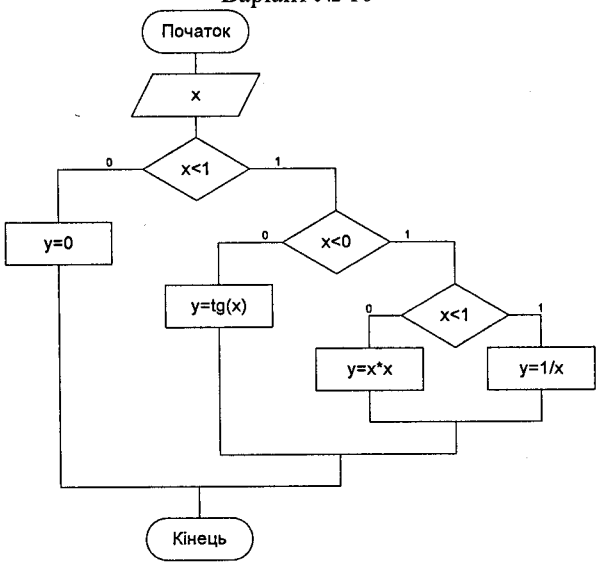
Варіант № 8



Варіант № 9



Варіант № 10



4.4 Варіанти завдання 2

Згідно з таблицею 4.2 в усіх варіантах необхідно знайти описовий алгоритм розв'язання інженерної задачі в Інтернеті з обов'язковими зазначенням джерел та розробити схему алгоритму в середовищі Microsoft Visio або в іншому спеціалізованому редакторі.

Таблиця 4.2 – Чисельні методи розв'язання інженерних задач

№	Назва методу
1	Метод половинного ділення для розв'язання нелінійних рівнянь
2	Метод ітерацій для розв'язання нелінійних рівнянь
3	Метод Гаусса-Зейделя
4	Метод Гаусса-Жордана
5	Метод Гаусса з вибором головного елемента
6	Метод послідовних наближень (метод Якобі)
7	Метод Гаусса з послідовним вилученням змінних
8	Метод хорд для розв'язання нелінійних рівнянь
9	Комбінаційний метод для розв'язання нелінійних рівнянь
10	Метод Гаусса з одиничною діагоналлю

4.5 Контрольні питання

1. Дайте означення алгоритму.
2. Що таке блок-схема алгоритму?
3. Навіщо потрібно графічне подання алгоритму у вигляді схеми алгоритму?
4. Які основні символи блок-схем Ви знаєте?
5. В чому полягає різниця між базовими структурами «повне розгалуження» та «неповне розгалуження»?
6. Перерахуйте особливості структур «цикл з передумовою», «цикл з післяумовою» та «цикл за параметром».
7. Як виконується запис алгоритму псевдокодом?
8. Що таке групування та для чого воно використовується при побудові блок-схем?
9. Що таке прив'язка до ліній та прив'язка до символів? Яка між ними різниця? Для чого вони використовуються при побудові блок-схем?
10. Які символи блок-схем були використані в вашому алгоритмі? Які дії вони виконують?

ЛАБОРАТОРНА РОБОТА № 5
ПРИНЦИПИ РОБОТИ В СИСТЕМІ ПРОГРАМУВАННЯ TURBO C ++

Мета роботи: отримання базових практичних навиків використання середовищ програмування (*programming environment*), необхідних для подальшого виконання циклу лабораторних робіт.

5.1 Порядок виконання роботи

5.1.1 Ознайомтесь з теоретичними відомостями про роботу в середовищі програмування Turbo C ++.

5.1.2 Виконайте налаштування параметрів середовища програмування, а саме: задайте шляхи до основних каталогів середовища, задайте параметри компіляції (*compilation*), програми, компонувальника (*linker*) та процесу відлагодження (*debugging*). Збережіть встановлені параметри.

5.1.3 Наведіть приклади довідкової інформації з контекстної довідкової системи (*context help*) середовища програмування.

5.1.4 Згідно з п. 5.2.4 теоретичних відомостей напишіть просту програму мовою програмування С та збережіть на жорсткому диску у власній папці. Виконайте компіляцію програми. Виправте необхідні помилки та запустіть програму на виконання.

5.1.5 Запустіть програму в покроковому режимі та прослідкуйте за зміною значень змінних в процесі відлагодження програми. Встановіть декілька контрольних точок та виконайте відлагодження програми.

5.1.6 Підготуйте звіт та зробіть висновки.

В ході цієї лабораторної роботи студенту пропонується виконати ряд дій, які повинні познайомити його з використанням середовища програмування для розробки, компіляції, виконання і відлагодження програм. Використання засобів, що автоматизують зазначені етапи є основою успішної розробки професійних програмних систем в будь-якому іншому середовищі.

5.2 Теоретичні відомості

5.2.1 Основи роботи в середовищі *Turbo C ++*

Підготовка каталогу і запуск середовища програмування

Середовище програмування ***Turbo C ++*** складається з великого числа файлів, зберігання яких структуроване по каталогах. Кореневий каталог середовища програмування в лабораторії: ***D:\STUDY\TC***. Файл запуску середовища ***D:\STUDY\TC\BIN\TC.EXE***.

Решта каталогів є підкаталогами в цьому каталозі. Основні з них:

D:\STUDY\TC\BIN - у цьому підкаталозі розміщені програмні модулі середовища програмування, підказка і т. д.

D:\STUDY\TC\INCLUDE - у цьому підкаталозі розміщені файли-заголовки з описами стандартних функцій середовища програмування.

D:\STUDY\TC\LIB - у цьому підкаталозі розміщені бібліотеки стандартних функцій середовища програмування.

5.2.2 Настроювання робочого середовища

Нижче наведені параметри середовища програмування, які потрібно встановити для його роботи.

Виберіть в ***Головному Меню*** пункт ***Options****(Параметри)*, і в підменю, що відкрилось, виберіть пункт ***Directories****...(Каталоги...)* (далі послідовність виборів з меню ми подаємо у вигляді: ***Головне Меню -> Options -> Directories****...)*.

В полях діалогу, що з'явився на екрані, встановіть такі значення:

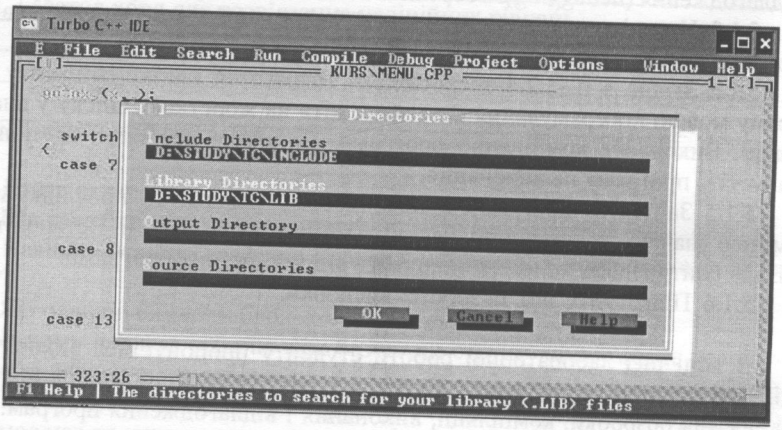


Рисунок 5.1 – Встановлення шляхів середовища

Після встановлення значень натисніть кнопку ***OK***.

Встановлені значення задаватимуть пошук файлів-заголовків і бібліотек в стандартних каталогах середовища програмування. Ті ж програмні модулі, які створюватимете Ви – текстові, об'єктні і завантажувальні, – розміщуватимуться у Вашому поточному каталозі.

Для налаштування параметрів компіляції виконайте такі дії:

Виберіть команду головного меню: ***Options -> Compiler -> Code generation...***

В полях діалогу, що з'явився на екрані, встановіть такі значення:

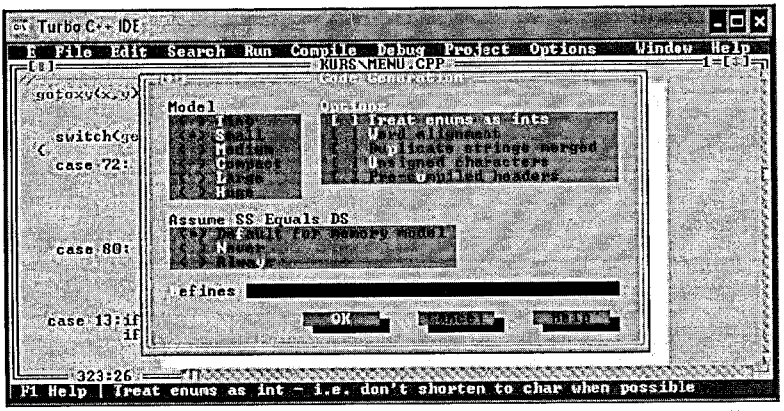


Рисунок 5.2 – Налаштування параметрів компіляції

Після встановлення значень натисніть кнопку ***OK***.

Виберіть команду головного меню: ***Options - > Compiler -> Advanced code generation...***

В полях діалогу, що з'явився на екрані, встановіть такі значення:

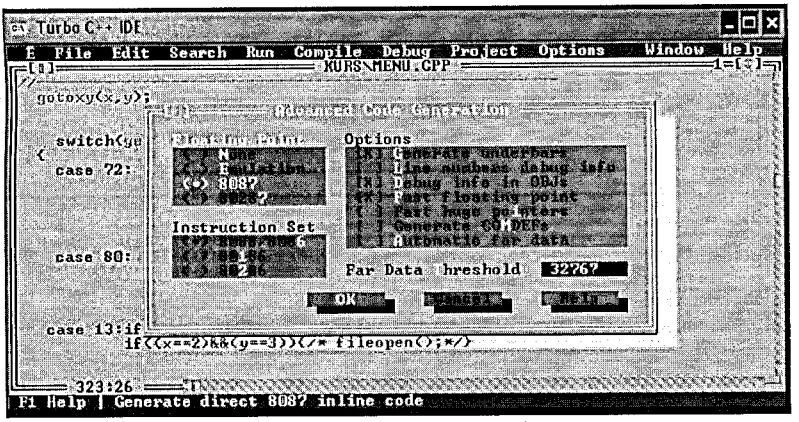


Рисунок 5.3 – Розширені налаштування компіляції

Після встановлення значень натисніть кнопку ***OK***.

Виберіть команду головного меню: ***Options -> Compiler -> Entry/Exit Code...***

В полях діалогу, що з'явився на екрані, встановіть такі значення:

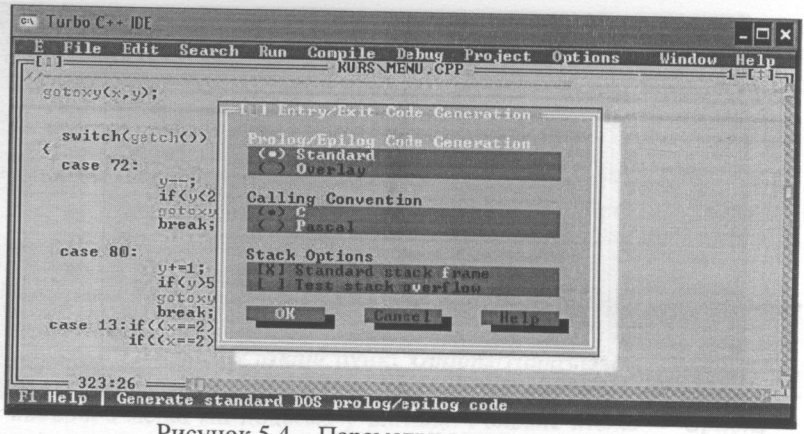


Рисунок 5.4 – Параметри генерування коду

Після встановлення значень натисніть кнопку ***OK***.
Виберіть команду головного меню: ***Options -> Compiler -> Source...***
В полях діалогу, що з'явився на екрані, встановіть такі значення:

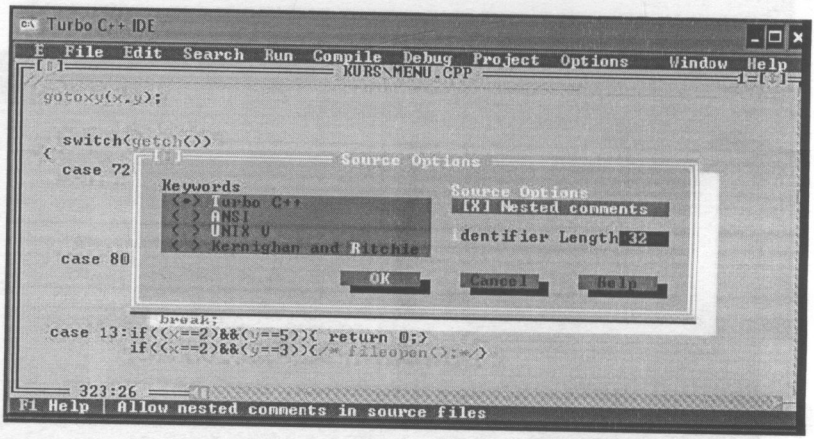


Рисунок 5.5 – Вибір стандарту

Після встановлення значень натисніть кнопку ***OK***.
Виберіть команду головного меню ***Options -> Make ...***
В полях діалогу, що з'явився на екрані, встановіть такі значення:

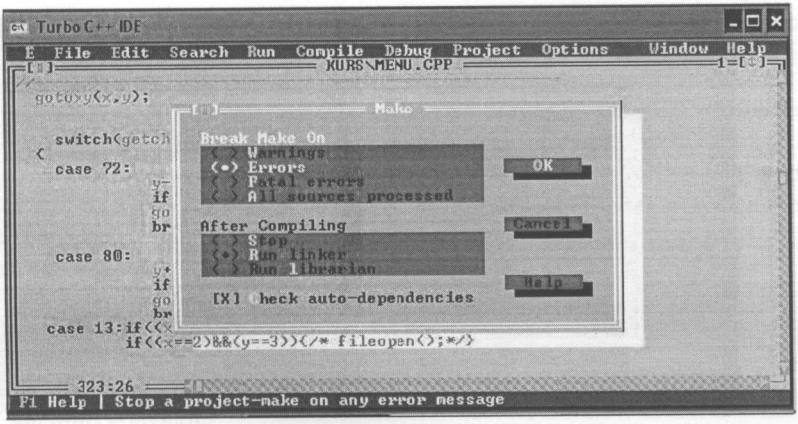


Рисунок 5.6 – Вибір налаштувань побудови

Після встановлення значень натисніть кнопку ***OK***.
Виберіть команду головного меню: ***Options -> Linker -> Settings ...***
В полях діалогу, що з'явився на екрані, встановіть такі значення:

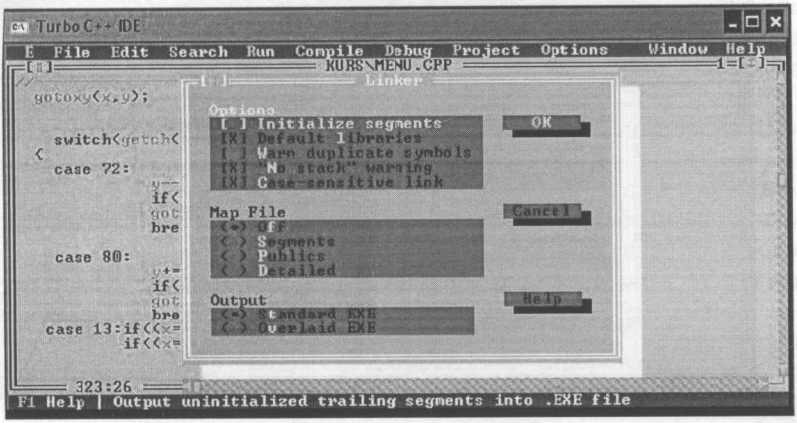


Рисунок 5.7 – Налаштування лінкування

Після встановлення значень натисніть кнопку ***OK.***
Виберіть в головному меню команду ***Options -> Linker -> Libraries ...***
В полях діалогу, що з'явився на екрані, встановіть такі значення:

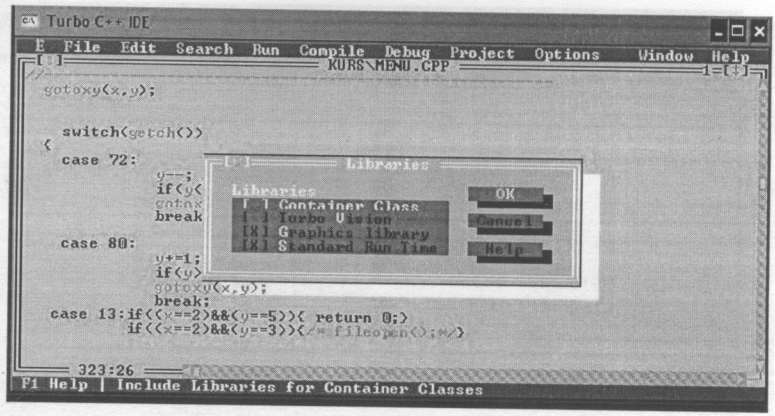


Рисунок 5.8 – Налаштування лінкування для бібліотек

Після встановлення значень натисніть кнопку ***OK***.
Виберіть в головному меню команду ***Options - > Debugger...***
В полях діалогу, що з'явився на екрані, встановіть такі значення:

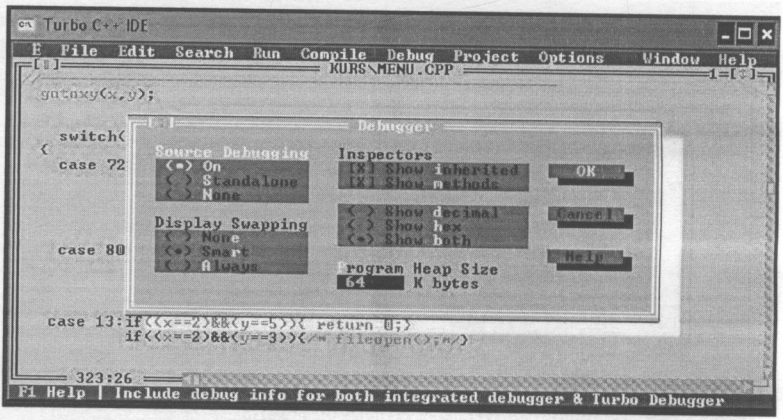


Рисунок 5.9 – Налаштування відлагодження

Після встановлення значень натисніть кнопку ***OK***.
Виберіть в головному меню команду ***Головне Меню -> Options -> Save...***. У вікні, що з'явилось на екрані, натисніть кнопку ***OK***.

5.2.3 Користування підказкою

Середовище програмування ***Turbo C++*** має систему онлайн-підказки, яка дуже корисна, особливо для програміста-початківця.

Підказка є гіпертекстовою, тобто, в усіх текстах, які з'являються на екрані при виведенні підказки, є деякі виділені слова. Вибравши одне з виділених слів, можна одержати підказку, пов'язану з цим словом.

Перша можливість звернення до підказки – через ***Головне Меню -> Help***. В меню ***Help*** найкориснішими є пункти ***Contents*** і ***Index***.

Пункт ***Contents*** виводить на екран тематичний список розділів, з яких можна одержати підказку. Зі всіх цих розділів перш за все ми рекомендуємо:

How to Use Help – як користуватися підказкою;
Menus and Hot Keys – меню і гарячі клавіші;
Editor Commands – команди редактора;
Turbo C++ Language – мова ***Turbo C++***;
Error Messages – повідомлення про помилки;
Functions – функції;
Header Files – файли заголовків.

Вибір будь-якого з цих розділів приводить до появи списку підрозділів і так далі.

Пункт ***Index*** виводить на екран алфавітний список всіх ключових слів і імен функцій мови і середовища програмування ***Turbo C++***. Вибираючи пункт цього списку, можна одержати докладну підказку щодо нього.

Інша можливість звернення до підказки – через клавіші ***F1*** і комбінацію клавіш ***Ctrl+F1***. Ця підказка – контекстно-залежна, тобто, підказка, яка з'являється на екрані, стосується саме того стану середовища програмування, в якому вона зараз знаходиться.

За клавішею ***F1*** Ви одержуєте підказку стосовно активного на даний момент меню або вікна.

Комбінацією клавіш ***Ctrl+F1*** можна користуватися тільки в активному вікні редактора. Якщо при натисканні цієї комбінації курсор знаходиться на якомусь ключовому слові або на функції, то на екран виводиться докладна підказка щодо цього ключового слова або функції. Інакше дія цієї комбінації аналогічна ***Головне Меню -> Help -> Index.***

5.2.4 Створення, редагування і збереження програми

Виберіть в головному меню команду ***File -> New.*** На екрані відкриється порожнє вікно редактора із заголовком ***00.CPP***. У цьому вікні наберіть текст програми, наведений нижче (нумерацію рядків, яка дана зліва, набирати не потрібно).

```
1  #include <stdio.h>
2  int main(void){
3   int a, b, c, d, x, y;
4     a=1;
5     b=2;
6     c=a+b;
7     d=a*b;
```

```
 8    if (c==d) {
 9 x=100;
10 y=200;
11 }
12    else {
13 x=200;
14 y=100;
15 }
16    printf("%d %d\n",x,y);
17    return 0;
18  }
```

При наборі і подальшому редагуванні тексту намагайтесь якнайчастіше використовувати такі спеціальні клавіші і комбінації клавіш, як: ***Home***, ***End***, ***Ctrl<-***, ***Ctrl->***, ***Ctrl+Y*** і ін. Також корисними можуть бути блокові команди: ***Ctrl+K B*** (виділити початок блоку), ***Ctrl+K K*** (виділити кінець блоку), ***Ctrl+K B*** (копіювати блок), ***Ctrl+K V*** (перемістити блок), ***Ctrl+K H*** (відмінити виділення). Докладну підказку щодо спеціальних клавіш і комбінацій можна одержати, обравши в головному меню команду ***Help -> Contents -> Editor Commands***.

Текст, який Ви набираєте, зберігається в оперативній пам'яті, тому, якщо під час набору виникне якась аварія (наприклад, вимкнення живлення), все, що Ви набрали, буде втрачено. Щоб уникнути такої небезпеки, потрібно зберегти текст у файлі на зовнішній пам'яті. Набравши декілька перших рядків, збережіть текст. Для цього оберіть команду головного меню ***File -> Save as...*** . При цьому екрані з'являється діалог ***Save File As.*** У верхньому полі цього діалогу введіть ім'я, під яким Ви хочете зберегти текст програми. Якщо Ви введете тільки ім'я (без розширення), система автоматично додасть до імені Вашої програми розширення ***CPP*** – стандартне розширення для програм, написаних на мові С++. Ми рекомендуємо явно задавати розширення, наприклад, ***myfile.c***, оскільки програми, які ми пишемо в цій частині лабораторного практикуму, обмежуються можливостями базової мови С. Набравши ім'я, натисніть на клавішу ***Enter*** або натисніть на кнопку ***OK***.

При першому збереженні програми Ви дали їй ім'я (зверніть увагу – це ім'я тепер є заголовком вікна редактора). Далі після кожного додавання декількох рядків або при внесенні деякої кількості змін в текст зберігайте програму під тим же ім'ям. Для цього достатньо викликати команду головного меню ***File -> Save*** або натиснути клавішу ***F2***.

Згодом, якщо Вам потрібно буде знову завантажити в редактор текст тієї ж програми, оберіть команду головного меню ***File -> Open*** (або натисніть клавішу ***F3***). Відкриється діалог ***Open a File***, який дуже схожий на діалог ***Save File As***. У верхньому полі цього діалогу виводиться маска для файлів, з яких можна вибирати файл для відкриття, в нижньому – список файлів, імена яких відповідають цій масці. Змінюючи маску у верхньому

полі, Ви змінюєте склад списку в нижньому. Досягнувши того, що в нижньому полі буде саме той список, який Вам потрібен, перейдіть в нижнє поле. Виберіть потрібний файл і натисніть на клавішу ***Enter*** або натисніть на кнопку ***OK***. Файл, який Ви обрали, відкриється у вікні редактора.

5.2.5 Компіляція і виконання програми

Закінчивши набір тексту і зберігши файл програми на диску, виконайте компіляцію програми. Для цього оберіть в головному меню команду ***Compile -> Compile*** або натисніть комбінацію клавіш ***Alt+F9***. На екрані з'являється вікно ***Compiling***, в якому відображається хід компіляції. При нормальному завершенні компіляції в нижньому рядку цього вікна повинно бути виведено повідомлення ***Success: Press any key***

Якщо там виводиться повідомлення

Errors: Press any key

або

Warnings: Press any key,

то Ваша програма вимагає корекції.

Роботу з діагностикою помилок ми докладніше розглянемо в наступному розділі. Після того, як Ви відкомпілювали програму без помилок, запустіть її на виконання. Для цього оберіть в головному меню команду ***Run -> Run*** або натисніть комбінацію клавіш ***Ctrl+F9***. На екрані з'являється вікно ***Linking***, в якому відображається хід компонування. При нормальному завершенні компонування це вікно зникає само собою і виконується програма.

Якщо ж в цьому вікні виводиться повідомлення ***Errors: Press any key***, то Ваша програма вимагає корекції.

Якщо Ваша програма не вимагає введення даних (а саме така та програма, з якою ми зараз працюємо), Ви навіть не встигнете помітити, як ця програма виконується. Щоб подивитися результати, які програма видала на екран, оберіть в головному меню команду ***Window -> User screen*** або натисніть комбінацію клавіш ***Alt+F5***. Ви побачите чорний екран з тим, що вивела Ваша програма (у нашому випадку це повинне бути: 200 100). Щоб вийти з режиму перегляду результатів, натисніть будь-яку клавішу.

Перевірте зміст Вашого робочого каталогу. Якщо текст Вашої програми був збережений у файлі ***myfile.c***, то після компіляції в каталозі повинен з'явиться файл ***myfile.obj***, а після виконання – ще і ***myfile.exe.***

5.2.6. Діагностика помилок і попереджень компілятора і компонувальника

На цьому етапі виконання лабораторної роботи ми пропонуємо Вам поекспериментувати з повідомленнями компілятора і компонувальника. Заздалегідь рекомендуємо зробити копію файлу програми. Для цього оберіть в головному меню команду ***File -> Save as...*** і введіть якесь нове ім'я програми, наприклад: ***myfilex.c***. Тепер у Вас є дві копії програми в двох

файлах (*myfile.c* і *myfilex.c*). Одна копія (хай це буде *myfile.c*) зберігатиме правильну версію програми, а в іншу (це буде *myfilex.c*) ми навмисне вводитимо помилки.

Відкрийте в текстовому редакторі файл *myfilex.c*. Внесіть такі зміни в текст програми (тут і далі ми вказуємо номери рядків, в які потрібно внести зміни):

6 c=a+b1;
9 x=100; a123
18 /}*/*

Запустіть програму на компіляцію (*Ctrl+F9*). Ви одержите повідомлення *Errors: Press any key* у вікні *Compiling*. Коли Ви натиснете будь-яку клавішу, в нижній частині екрана відкриється вікно *Message* з таким вмістом:

Compiling MYFILEX.C:		
Error	*MYFILEX.C*	*6: Undefined symbol 'b1'*
Error	*MYFILEX.C*	*10: Undefined symbol 'a123'*
Error	*MYFILEX.C*	*10: Statement missing ;*
Error	*MYFILEX.C*	*17: Compound statement missing }*

Це повідомлення компілятора про помилки. Перший рядок – заголовок. У наступних рядках: ознака помилки, ім'я файлу, в якому знайдена помилка, номер рядка тексту програми, в якому була знайдена помилка, діагностика помилки.

Повідомлення до рядка 6 – «*Невизначений символ 'b1'*». У цьому операторі використовується змінна з таким ім'ям, якої немає серед оголошених змінних.

Перше повідомлення до рядка 10 – «*Невизначений символ 'a123'*». Текст *a123* схожий на ім'я змінної, але така змінна не оголошена. Інше повідомлення до рядка 10 – «*В операторі відсутній ;*» . Текст *a123* може бути окремим оператором, але в ньому немає ознаки кінця оператора. Зверніть увагу на те, що хоча помилку ми внесли в рядок 9, повідомлення видається до рядка 10, оскільки помилка була знайдена тільки при обробці цього рядка.

Повідомлення до рядка 17 – «*В складеному операторі відсутній }*» – компілятор виявив непарність операторних дужок { }. Де б не була пропущена закривальна операторна дужка, її відсутність може бути знайдена тільки на останньому операторі програми.

Коли Ви переміщаєтеся списком повідомлень про помилки у вікні *Message,* у вікні редактора те місце тексту програми, якого стосується поточне повідомлення, виділяється кольором. Коли Ви перемикаєтеся у вікно редактора (клавіша *F6*), курсор встановлюється на це саме місце.

Відновіть правильний вміст файлу ***myfilex.c***. (Це можна зробити, відкривши в редакторі файл ***myfile.c*** і знов зберігши його з ім'ям ***myfilex.c***. При цьому Ви одержите попередження про те, що файл ***myfilex.c*** буде змінений, на яке Вам потрібно відповісти ***Yes***.) Внесіть такі зміни в текст програми:

```
3   int a, b, з, d, x, y, z=2;
4     a=b;
8     if (c=d) {
```

Запустіть програму на компіляцію (***Ctrl+F9***). Ви одержите повідомлення: ***Warnings: Press any key*** у вікні ***Compiling***. Коли Ви натиснете будь-яку клавішу, в нижній частині екрана відкриється вікно ***Message*** з таким текстом:

```
Compiling MYFILEX.C:
Warning  MYFILEX.C      4: Possible use of 'b' before definition
Warning  MYFILEX.C      8: Possible incorrect assignment
Warning  MYFILEX.C      18: 'z' is assigned value that is never used
Warning  MYFILEX.C      18: 'c' is assigned value that is never used
```

Це попередження компілятора.

Попередження до рядка 4 – «***Можливе використовування 'b' до визначення***». У цьому операторі значення змінної ***b*** привласнюється змінній ***a***, але яке значення має ***b*** на цей момент виконання програми – невідомо.

Попередження до рядка 8 – «***Можливе некоректне привласнення***». Вираз ***c=d*** має сенс: «***привласнити змінній с значення змінної d***». Вираз ***c==d***, який застосований в правильній програмі, має сенс: «***порівняти змінні с і d***». Оскільки вираз внесений в умовний оператор, компілятор має підстави припускати, що тут повинно бути порівняння, а не привласнення.

Попередження до рядка 18 – ***«Змінній 'z' привласнюється значення, яке ніде не використовується»***. У операторі 3 ми дали змінній ***z*** початкове значення 2. Але далі в програмі значення ***z*** ніде нічому не привласнюється і ні з чим не порівнюється.

Ще одне таке ж попередження до рядка 18 стосується змінної ***c***. Привласнення значення цій змінній відбувається в рядку 8. Ситуація невикористання значення може бути виявлена тільки в кінці програми.

Ви можете одержати не всі попередження з тих, які тут перераховані. При налаштуванні середовища можна відмінити або відновити видачу тих або інших попереджень. Щоб зробити відповідне налаштування, оберіть в головному меню команду ***Options -> Compiler -> Messages***.

Відновіть правильний вміст файлу ***myfilex.c***. Внесіть такі зміни в текст програми:

```
4     a=1; abc();
```

Запустіть програму на виконання (***F9***).

Ви одержите повідомлення ***Errors: Press any key*** у вікні ***Linking***. У вікні ***Message*** буде:

Linking MYFILEX.EXE:
Linker Error: Undefined symbol _abc in module MYFILEX.C

Це повідомлення компонувальника. «***Невизначений символ _abc в модулі MYFILEX.C***». Синтаксично оператор ***abc();*** є звертанням до функції, саме так його трактує компілятор. Але коли компонувальник намагається знайти функцію з таким ім'ям в доступних йому модулях і бібліотеках, він її не знаходить, про що і повідомляє нас.

Якщо Ви запустите окремим кроком компіляцію тієї ж програми, Ви одержите:

Compiling MYFILEX.C:
Warning MYFILEX.C 4: Call to function 'abc' with no prototype

Компілятор попереджає про те, що в програмі є «***Виклик функції 'abc' без прототипу***».

5.2.7 Відлагодження програми

В середовищі програмування є відладник, який працює на рівні текстового коду. Використовуючи його, Ви можете виконувати програму в покроковому режимі, встановлювати брейкпойнти виконання і стежити за поточними значеннями змінних програми.

Відкрийте в текстовому редакторі файл ***myfile.c.***

Оберіть в головному меню команду ***Debug -> Watches -> Add watch*** (або натисніть комбінацію клавіш ***Ctrl+F7***). У вікні ***Add Watch***, яке з'явиться на екрані, введіть в полі ***Watch Expression*** ім'я змінної ***a***, натисніть кнопку ***OK***. У нижній частині екрана з'явиться вікно ***Watch***, а в ньому – «***a: Undefined symbol 'a'***».

Повторіть ці дії кілька разів, вводячи в полі ***Watch Expression*** імена ***b, z, d***. У вікні ***Watch*** додаватимуться аналогічні повідомлення. Цими діями ми даємо середовищу програмування інструкцію відстежувати і відображати у вікні ***Watch*** поточні значення вибраних змінних програми. Оскільки програма ще не виконується, ці змінні поки що «невідомі» системі програмування, про що і свідчать повідомлення.

Зробіть активним вікно редактора і натисніть клавішу ***F8***. У вікні редактора кольором (швидше за все – блакитним) буде виділений рядок 2 тексту програми. Це ми почали відлагодження нашої програми в покроковому режимі. Ще раз натисніть клавішу ***F8***. Виділення зміститься на рядок 4, а у вікні відображатимуться якісь значення змінних. Виконання програми по-

чалося і ці змінні вже «відомі», але їх значення ще не встановлені, тому вони – якісь випадкові числа. З наступним натисканням клавіші ***F8*** значення змінної ***a*** зміниться на 1. Кожне наступне натискання клавіші ***F8*** просуватиме виконання програми на оператор вперед і, відповідно до виконання операторів програми, мінятимуться значення змінних.

Зверніть увагу на те, що після рядка 8 виконання зразу ж «перестрибне» на рядок 13. Оскільки умова в умовному операторі 8 не виконується, виконання обходить рядки 9–12. Якщо Ви зміните: ***4 a=2;*** і знову виконаєте програму в покроковому режимі, умова в рядку 8 виконуватиметься, отже, і виконання програми пройде через рядки 9, 10, 11 і обійде рядки 12–15.

Закінчивши виконання програми, встановіть курсор на рядок 8 і оберіть в головному меню команду ***Debug -> Toggle breakpoint*** (або натисніть комбінацію клавіш ***Ctrl+F8***). Рядок 8 виділиться кольором (швидше за все – червоним). Цим ми задали брейкпойнт програми. Тепер запустіть програму на виконання (***F9***). Виконання зупиниться на рядку 8. При цьому у вікні ***Watch*** відображатимуться поточні значення змінних. Ви можете продовжити виконання в покроковому (***F8***) або в автоматичному режимі (***F9***).

Команди ***Debug*** дають можливість повністю керувати стеженням за значеннями і точками зупинки.

5.3 Контрольні питання

1. Яке основне призначення середовища Turbo C ++?
2. Опишіть структуру головного вікна середовища. Дайте характеристику складових елементів.
3. Опишіть головне меню, його основні пункти та їх призначення.
4. Як створити новий файл?
5. Які основні операції роботи з текстом в вікні текстового редактора? Які комбінації клавіш використовуються у редакторі?
6. Опишіть алгоритм запуску програми.
7. Опишіть основні етапи відлагодження програми.
8. Як переглянути значення програмних об'єктів під час роботи програми?
9. Розробіть найпростішу програму виведення привітання на екран та покажіть основні етапи відлагодження програми.

ЛАБОРАТОРНА РОБОТА № 6
РОБОТА У СЕРЕДОВИЩІ VISUAL STUDIO 2010 EXPRESS

Мета роботи: навчитись працювати у середовищі Visual Studio 2010 Express.

6.1 Порядок виконання роботи

6.1.1 Ознайомтесь з теоретичними відомостями.

6.1.2 Створіть новий проект та виконайте всі практичні кроки, як показано в теоретичних відомостях.

6.1.3 Стисло запишіть основні комбінації клавіш для редагування, компіляції, запуску та відлагодження програмного коду.

6.1.4 Зробіть висновки.

6.2 Теоретичні відомості

Visual Studio 2010 Express є безкоштовним середовищем для розробки програмного забезпечення від компанії Microsoft. Інтерфейс середовища наведений на рисунку 6.1.

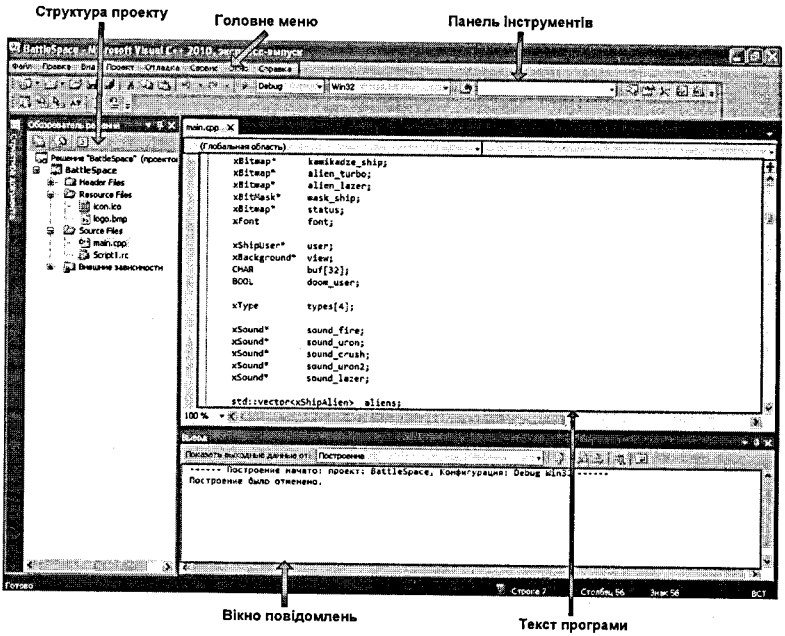


Рисунок 6.1 – Структура інтерфейсу середовища

Опис команд головного меню середовища наведено на рисунках 6.2–6.5.

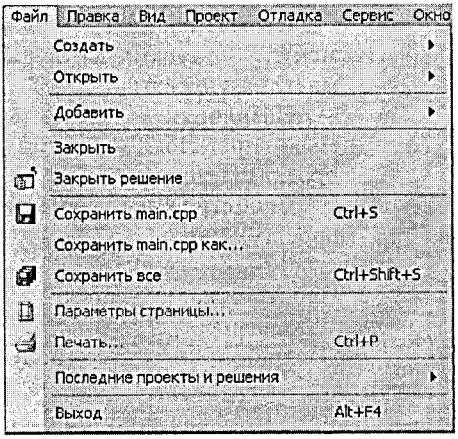


Рисунок 6.2 – Меню **Файл**

Создать – створити новий проект або модуль

Открыть – відкрити існуючий проект або модуль

Добавить – додати новий модуль до проекту

Закрыть – закрити поточний модуль

Закрыть решение – закрити проект

Сохранить main.cpp – зберегти поточний модуль

Сохранить все – зберегти всі модулі проекту

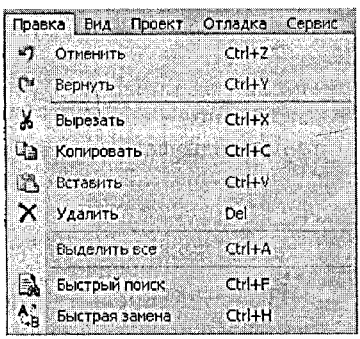


Рисунок 6.3 – Меню **Правка**

Отменить – відмінити останню дію

Вернуть – повторити останню відмінену дію

Вырезать – вирізати виділений фрагмент у буфер обміну

Копировать – скопіювати виділений фрагмент у буфер обміну

Вставить – вставити фрагмент з буфера обміну

Удалить – видалити виділений фрагмент

Выделить все – виділити все

Быстрый поиск – швидкий пошук

Быстрая замена – швидка заміна

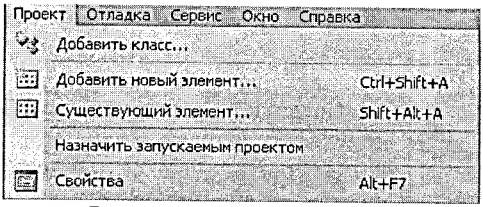


Рисунок 6.4 – Меню **Проект**

Добавить новый элемент... – додати новий модуль до проекту

Существующий элемент... – додати існуючий модуль до проекту

Назначить запускаемым проектом – зробити проект стартовим

Свойства – параметри проекту

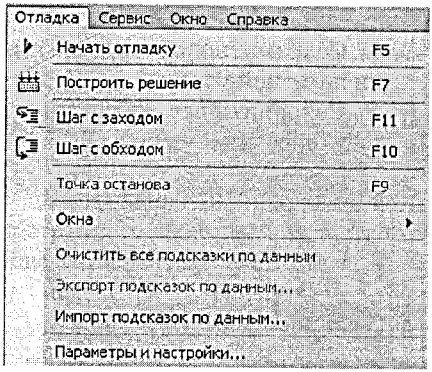


Рисунок 6.5 – Меню **Отладка**

Начать отладку – почати відлагодження програми

Построить решение – побудувати весь проект

Шаг с заходом – якщо наступний рядок коду, під час відлагодження, є викликом функції, то налагоджувальник зупинить виконання програми на першому рядку «всередині» цієї функції

Шаг с обходом – якщо наступний рядок коду, під час відлагодження, є викликом функції, то налагоджувальник виконає цю функцію та зупинить виконання програми на наступному рядку після її виклику, тобто не «входячи» у саму функцію

Точка остановка – поставити точку зупинки на виділеному рядку

Приклад створення простого консольного проекту

1. Виберіть пункт меню **Файл → Создать → Проект**...

2. У вікні, що з'явилось, виберіть категорію проектів **Win32**, тип проекту – **Win32 Консольное приложение**. Потім введіть шлях до проекту (**Расположение**) та вкажіть його ім'я (**Имя**).

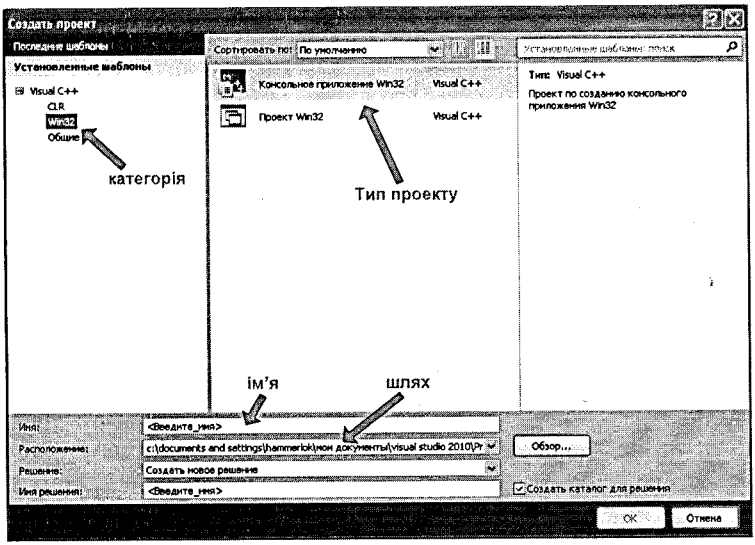


Рисунок 6.6 – Діалог створення нового проекту

3. Після натискання **ОК**, з'явиться помічник зі створення проектів.

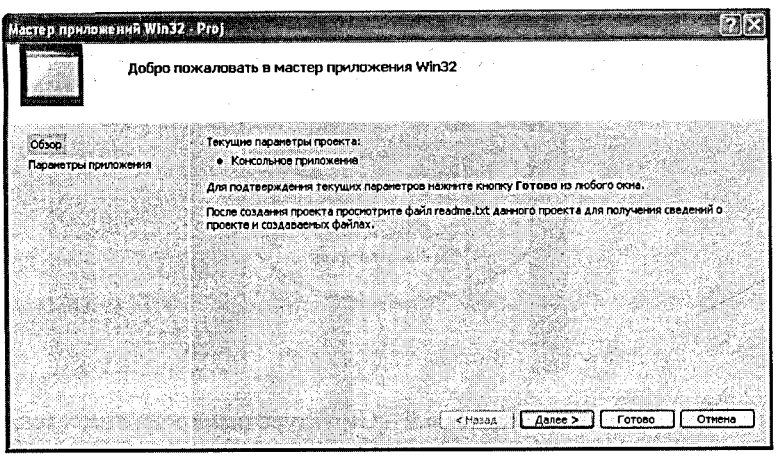


Рисунок 6.7 – Майстер створення додатків

4. Натисніть **Далее**.

5. З'явиться наступне вікно, у якому необхідно вказати основні параметри проекту.

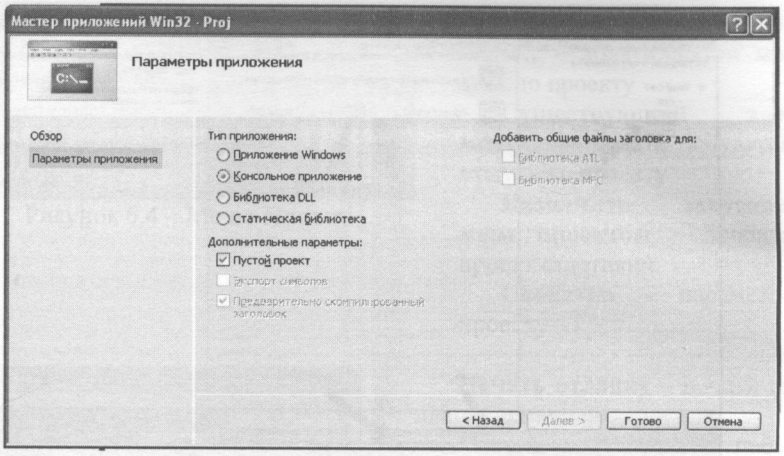


Рисунок 6.8 – Останній екран майстра створення додатків

6. Відмітьте прапорець **Пустой проект** та натисніть **Готово**.
7. На панелі **Обозреватель решений** з'явиться новий проект.

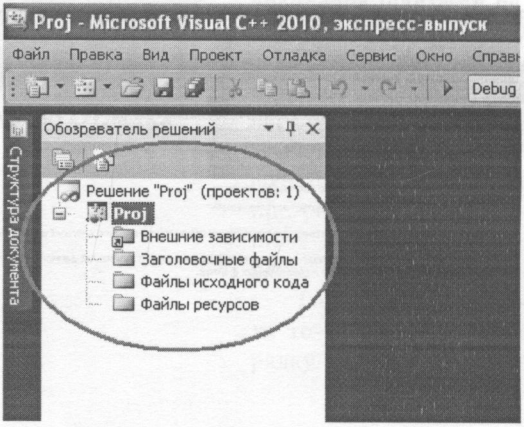


Рисунок 6.9 – Структура нового проекту

8. Додайте новий модуль до створеного проекту. Виберіть пункт меню **Проект → Добавить новый элемент…**
9. У вікні, що з'явилося, виберіть категорію **Код**, тип модуля – **Файл C++(.cpp)** та вкажіть ім'я – **main**. Натисніть **Добавить**.

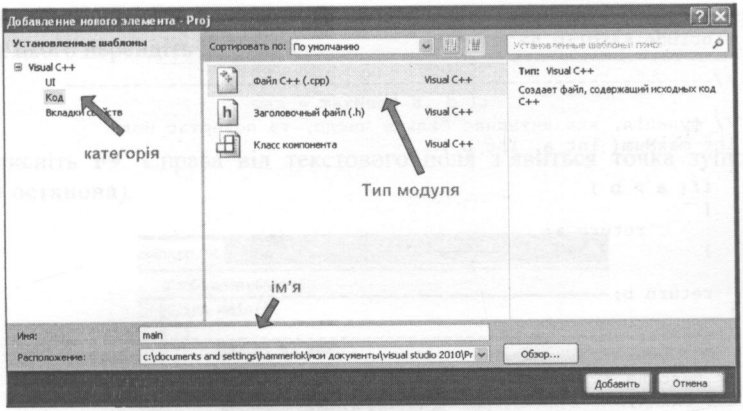


Рисунок 6.10 – Додавання файлу у проект

10. Новий модуль з'явиться у вікні **Обозреватель решений**.

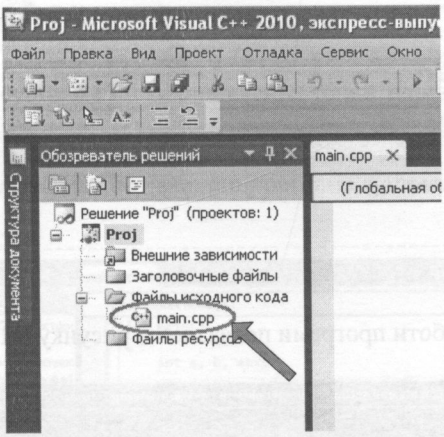


Рисунок 6.11 – Результат додавання нового файлу

Відлагодження програми

1. Створіть новий проект та додайте до нього модуль, як було вказано раніше.

2. Створіть просту програму, яка буде визначати найбільше число із двох введених:

Код модуля main.cpp

```
// підключаємо необхідні бібліотеки
```

```
#include <stdio.h>
#include <stdlib.h>

//------------------------------------------------------------

// функція, яка визначає більше число, та повертає його
int maxNum( int a, int b )
{
  if( a > b )
  {
        return a;
  }

  return b;
}
//------------------------------------------------------------

// головна функція
int main()
{
  int a, b, max;

  // введення даних
  printf( "Vvedite chislo a: " );
  scanf( "%d", &a );
  printf( "Vvedite chislo b: " );
  scanf( "%d", &b );

  // визначення більшого числа
  max = maxNum( a, b );

  // виведення результату
  printf( "\nBolshee chislo: %d\n\n", max );

  system( "pause" );
  return 0;
}
  //------------------------------------------------------------
```

3. Запустіть програму, натиснувши **F5.**

Результат роботи програми показано на рисунку 6.12.

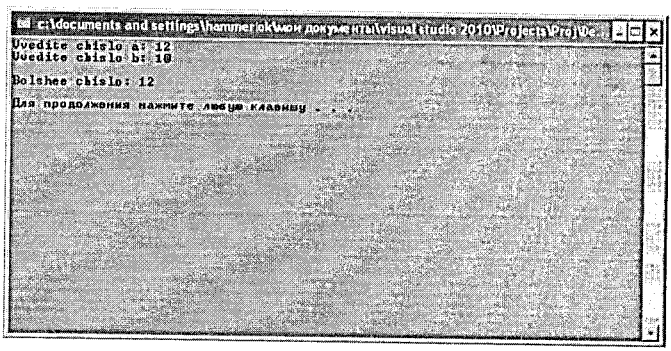


Рисунок 6.12 – Результат роботи програми

4. Поставте точку зупинки (**Точка останова**).

Для цього перейдіть до рядка:

```
max = maxNum( a, b );
```

Натисніть **F9**. Справа від текстового поля з'явиться точка зупинки (**Точка останова**).

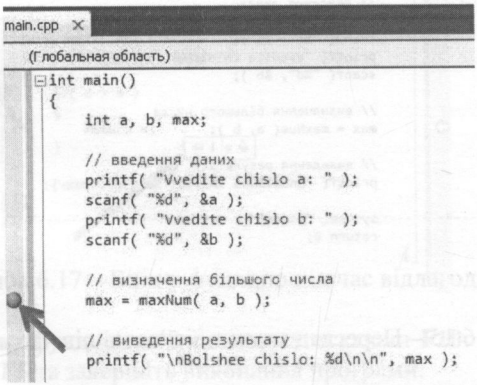


Рисунок 6.13 – Встановлення точки зупинки

5. Запустіть програму, натиснувши **F5.**

Після введення даних ми бачимо, що програма зупинилася на точці зупинки.

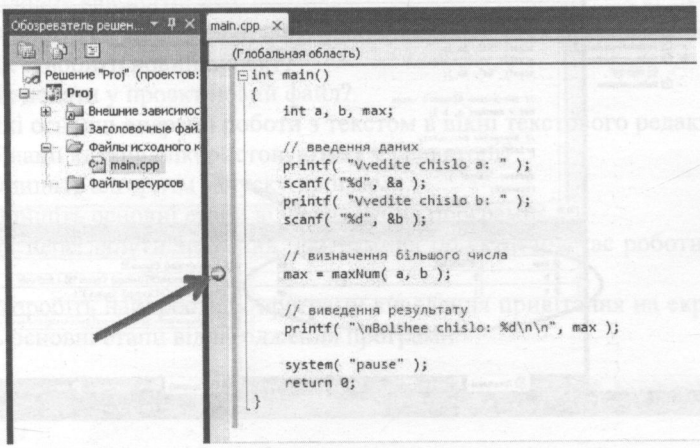


Рисунок 6.14 – Зупинка програми в точці зупинки

6. Перегляньте вміст змінних **a** та **b:**

1. Підвівши до них курсор миші

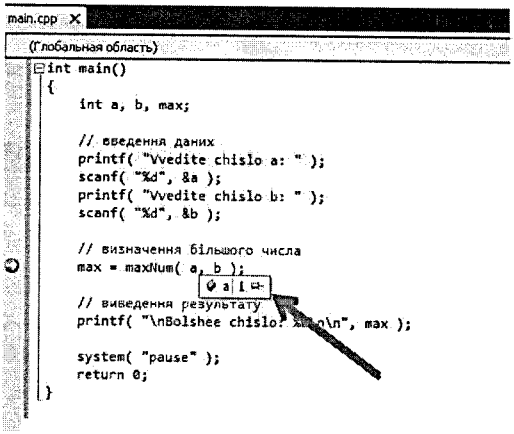


Рисунок 6.15 – Перегляд значень об'єктів під час роботи програми

2. Переглянувши у вікні **Локальные**

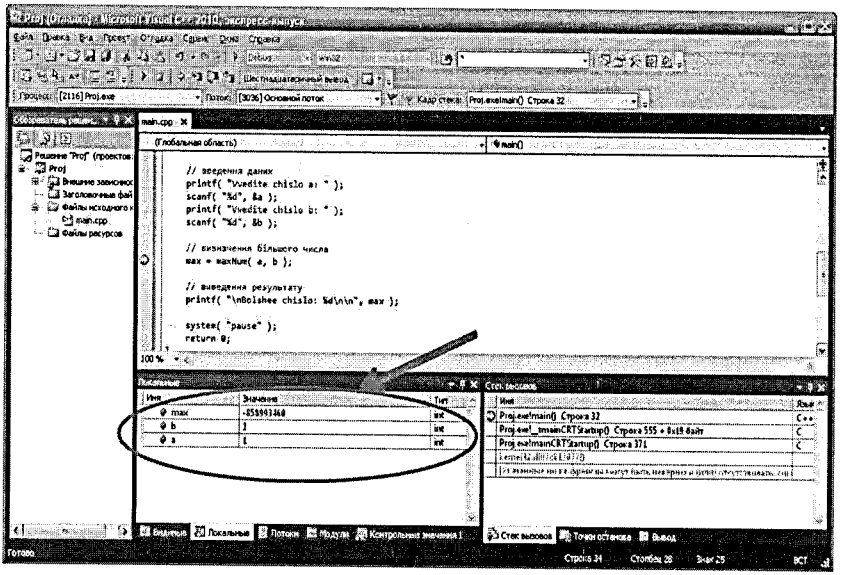


Рисунок 6.16 – Перегляд значень локальних змінних

7. Виконайте програму до наступного рядка, натиснувши **F10**, та перегляньте вміст змінної **max.**

8. Запустіть програму далі, натиснувши **F5.**

9. Після закінчення виконання програми, перезапустіть її.

10. На цей раз, після зупинки на точці зупинки, яку ми поставили у **п.4**, натисніть **F11.** Ми бачимо, що програма зупинилася на першому рядку функції **maxNum.**

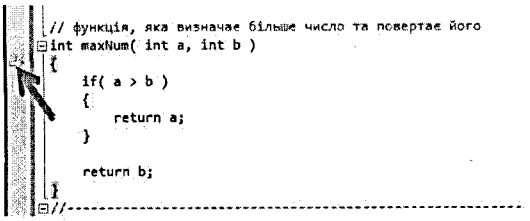


Рисунок 6.17 – Вхід у функцію під час відлагодження

11. Виконайте цю функцію покроково, натискаючи **F10**. Після виходу з функції натисніть **F5** та завершіть виконання програми.

6.3 Контрольні питання

1. Яке основне призначення середовища проектування Visual Studio 2010 Express?
2. Опишіть структуру головного вікна середовища. Дайте характеристику складових елементів.
3. Опишіть головне меню, його основні пункти та їх призначення.
4. Як створити новий проект?
5. Як додати у проект новий файл?
6. Які основні операції роботи з текстом в вікні текстового редактора? Які комбінації клавіш використовуються у редакторі?
7. Опишіть алгоритм запуску програми.
8. Опишіть основні етапи відлагодження програми.
9. Як переглянути значення програмних об'єктів під час роботи програми?
10. Розробіть найпростішу програму виведення привітання на екран та покажіть основні етапи відлагодження програми.

ЛАБОРАТОРНА РОБОТА № 7
БАЗОВІ ТИПИ ДАНИХ ТА ОРГАНІЗАЦІЯ ВВЕДЕННЯ-ВИВЕДЕННЯ ДАНИХ

Мета роботи: отримання практичних навиків роботи з типами даних (*data type*) мови С та використання функцій стандартного введення-виведення даних.

7.1 Порядок виконання роботи

7.1.1 Ознайомтесь з теоретичними відомостями за темами: типи даних мови С, оголошення (*declaration*) змінних (*variables*), функції (*functions*) стандартного введення-виведення даних printf() і scanf().

7.1.2 Розробіть програму, яка вводить фактичні дані з таблиці, подані у вашому варіанті індивідуального завдання і виводить на екран таблицю, яка знаходиться в індивідуальному завданні (охоплюючи заголовок і примітки).

7.1.3 Оформіть звіт та зробіть висновки.

7.2 Теоретичні відомості

Змінна – це поіменована область пам'яті, в якій зберігаються дані визначеного типу. Змінна має ім'я і значення. В процесі роботи значення змінних може змінюватися; перед застосуванням змінної її треба оголосити. Щоб оголосити змінну, потрібно вказати її тип.

Тип – це характеристика змінної, яка визначає, по-перше, сукупність значень, які може приймати змінна, по-друге, операції (*operations*), які можуть виконуватись над змінною, та, по-третє, об'єм пам'яті, що відводиться для зберігання значення змінної.

Базові типи даних мови С наведено у таблиці 7.1.

Таблиця 7.1 – Базові типи даних мови С

Тип даних	Розмір, байтів	Діапазон значень
unsigned char	1	0.....255
char	1	-128.....127
enum	2	-32768.....32767
unsigned int	2	0.....65535
short int	2	-32768.....32767
unsigned short	2	0.....65535
int	4	-32768.....32767
unsigned long	4	0.....4294967295
long	4	-2147483648.....2147483647
float	4	3.4e-38.... 3.4e+38
double	8	1.7e-308. 1.7e+308
long double	10	3.4e-49321.1e+4932

Наведемо приклади оголошення змінних:

1) **short a;** // оголошення змінної короткого цілого типу;

2) **char c='c';** // оголошення змінної символьного типу та одночасна ініціалізація значення даної змінної;

3) **char s, sf = 'f';** // оголошення двох символьних змінних; другій змінній одразу надається ініціальне значення.

Виведення даних

Для розробки програми потрібно використовувати стандартні функції введення-виведення бібліотеки ***stdio***.

Функція ***printf()***:

int printf(char *format, <список виведення>);

Перший параметр ***char *format*** є символьним рядком, який задає специфікації формату. Кожна специфікація формату має вигляд (параметри в квадратних дужках необов'язкові):

%[flags][width][.prec][F№N№h№l]type

де	***type –***	тип специфікації
	d або i	ціле десяткове число із знаком
	u	десяткове число без знака
	x	ціле 16-цяткове число без знака
	f	дійсне число
	e	число в Е-формі
	g	дійсне число або в Е-формі
	c	один символ
	s	рядок
	%	символ %
	flags –	ознака вирівнювання:
	+ або порожньо	вирівнювання по правому краю
	-	вирівнювання по лівому краю
	width –	ціле число – загальна ширина поля. Якщо це число починається з цифри 0, висновок доповнюється зліва нулями до заданої ширини. У задану ширину входять всі символи виведення, в тому числі знак, дробова частина і т. п.
	prec –	ціле число, кількість знаків після крапки при виведенні дійсних чисел

F – відповідний елемент списку виведення є дальнім вказівником

N – відповідний елемент списку виведення є близьким вказівником

l – відповідний елемент списку виведення є long int або double

Список виведення – це перелік змінних і виразів (*expressions*), значення яких виводяться.

Наприклад:

printf("x=2.3%f y=%d c=%c",x, y, c);

Дана функція виведе на екран змінну ***x*** типу ***float*** з заданою кількістю знаків, змінну ***y*** типу ***int*** і символьну змінну ***c***.

Функція ***putchar()*** – виведення одного символу на екран:

int putchar(int ch);

Параметр функції ***int ch*** – це код символу, який виводиться. При успішному виконанні функція повертає цей же код, при неуспішному – стан EOF.

Наприклад: ***x='c';***

putchar(x); // виведення значення змінної ***x*** на екран

Функція ***puts()*** – виведення рядка символів на екран:

int puts(char *string);

Параметр функції ***char *string*** – це вказівник (*pointer*) на початок того рядка (*string*), з якого виводяться дані. Функція повертає кількість виведених символів.

Наприклад: ***char string[] = "This is an example output string\n";***
puts(string);

Введення даних

Функція ***scanf()*** – введення формату з клавіатури:

int scanf(char *format, <список введення>);

Перший параметр ***char *format*** є символьним рядком, який задає специфікації формату (див. функцію ***printf()***). Список введення – це перелік

адрес змінних, в які вводяться дані. У цьому списку перед іменами всіх змінних, окрім тих, які вводяться за специфікацією типу *%s*, повинен стояти символ *&*.

Наприклад: ***scanf("%f %d %c",&x, &y, &c);***

Функція ***getchar()*** – введення одного символу з клавіатури:

int getchar(void);

Функція повертає код введеного символу.

Наприклад: ***while ((c = getchar()) != '\n') printf("%c", c);***

Функція ***gets()*** – введення рядка символів з клавіатури:

char *gets(char *string);

Наприклад:
```
printf("Input a string:");
gets(string);
```

Константи граничних значень типів

Розглянемо таблицю констант (*constants*) стандартної бібліотеки ***limits***, які наведено в таблиці 7.2.

Таблиця 7.2 – Граничні значення для цілочислових типів

Ім'я константи	Значення	Суть
CHAR_BIT	8	Число бітів в байті
SCHAR_MIN	-128	Мінімальне значення signed char
SCHAR_MAX	127	Максимальне значення signed char
UCHAR_MAX	255	Максимальне значення unsigned char
CHAR_MIN	'0' SHAR_MIN	Мінімальне значення для char
CHAR_MAX	USHAR_MAX SCHAR_MAX	Максимальне значення для char
MB_LEN_MAX	1	Мінімальне число байтів в багато-байтовому символі
SHRT_MIN	-32768	Мінімальне значення для short
SHRT_MAX	32767	Максимальне значення для short
USHRT_MAX	65535	Максимальне значення unsigned short
INT_MIN	-32768	Мінімальне значення для int
INT_MAX	32767	Максимальне значення для int
UINT_MAX	65535	Максимальне значення unsigned int
LONG_MIN	-2147483648	Мінімальне значення для long
LONG_MAX	2147483648	Максимальне значення для long
ULONG_MAX	4294967295	Максимальне значення unsigned long

Для визначення констант граничних значень дійсних даних використовують бібліотеку ***float***, які наведено в таблиці 7.3.

Таблиця 7.3 – Константи для дійсних типів

Ім'я константи	Значення	Суть
FLT_RADIX	2	Основа експоненціального подання, наприклад: 2,16
FLT_DIG	6	Кількість правильних десяткових цифр
FLT_EPSILON	1E - 5	Мінімальне x, таке, що 1.0 + x ≠ 1.0 (1,192093E-07)
FLT_MANT_DIG	24	Кількість цифр в мантисі за основою FLT_RADIX
FLT_MAX	1E + 37	Максимальне число з плаваючою точкою (3,402823E+38)
FLT_MAX_EXP	128	Максимальне n, таке, що FLT_RADIX^n – 1 подамо в вигляді числа типу float
FLT_MAX_10_EXP	38	Максимальне ціле n, таке, що 10^n подамо як float
FLT_MIN	1E - 37	Мінімальне нормалізоване число з плаваючою точкою типу float (1,175494E-38)
FLT_MIN_EXP	-125	Мінімальне ціле n, таке, що 10^n подамо у вигляді нормалізованого числа
FLT_MIN_10_EXP	38	Мінімальне ціле n, таке, що 10^n подамо як float
DBL_DIG	10	Кількість правильних десяткових цифр для типу double
DBL_EPSILON	1E - 16	Мінімальне x, таке, що 1.0 + x ≠ 1.0, де x належить до типу double
DBL_MANT_DIG	53	Кількість цифр за основою FLT_RADIX в мантисі для чисел типу double
DBL_MAX	1E +308	Максимальне число з плаваючою точкою типу double (1.797693E+308)
DBL_MAX_EXP	1024	Максимальне n, таке, що FLT_RADIX^n – 1 подамо у вигляді числа типу double
DBL_MAX_10_EXP	308	Максимальне ціле n, таке, що 10^n подамо як double
DBL_MIN	1E - 308	Мінімальне нормалізоване число з плаваючою точкою типу double
DBL_MIN_EXP	-1021	Мінімальне ціле n, таке, що 10^n подамо у вигляді нормалізованого числа типу double
DBL_MIN_10_EXP	-307	Мінімальне від'ємне число n, таке, що 10^n – в діапазоні визначення чисел типу double

Програма для визначення нижньої та верхньої границь кожного типу за допомогою стандартних констант, які описані в файлі ***limits.h*** і ***float.h***, може мати такі оператори:

```
#include "limits.h"
#include "float.h"
#include <stdio.h>
int main()
```

```
{
printf("CHAR_MIN=\t%d\n",CHAR_MIN);
printf("CHAR_MAX=\t%d\n",CHAR_MAX);

printf("INT_MAX=\t%d\n",INT_MAX);
printf("INT_MIN=\t%d\n",INT_MIN);

printf("LONG_MAX=\t%ld\n",LONG_MAX);

printf("FLT_MAX=\t%e\n",FLT_MAX);

printf("DBL_MIN=\t%e\n",DBL_MIN);

return 0;
}
```

Для використання завдання 6 можна також використовувати бібліотеки ***limits.h*** і ***float.h.*** Наприклад, аналіз можливості зміни нижньої та верхньої меж кожного типу виконуємо за допомогою функцій ***increment*** і ***decrement***:

```
char ch=127;
printf("max char ch=\t\t%d\n",ch);
ch++;
printf("after ch++ min ch=\t\t%d\n",ch);
unsigned char unch=255;
printf("max unsigned char unch=\t\t%d\n",unch);
unch++;
printf("after unch++ min unch=\t\t%d\n",unch);
```

Правила перетворення типів (*type cast*)

При обчисленні виразів деякі операції вимагають, щоб операнди мали відповідний тип, а якщо вимоги до типу не виконані, примусово викликають виконання потрібних перетворень. Та ж ситуація виникає при ініціалізації, коли тип виразу, що ініціалізується (*initialization*), приводиться до типу обумовленого об'єкта (*object*). Нагадаємо, що в мові С присвоювання є бінарною операцією, тому сказане щодо перетворення типів стосується всіх форм присвоювання, однак при присвоюваннях значення виразу з правої частини завжди приводиться до типу змінної з лівої частини, незалежно від співвідношення цих типів.

Правила перетворення в мові С для основних типів визначені стандартом мови. Ці стандартні перетворення охоплюють переклад «нижчих» типів в «вищі».

Серед перетворень типів виділяють:

- перетворення в арифметичних виразах;
- перетворення при присвоюваннях;

• перетворення вказівників.

При перетворенні типів потрібно розрізняти перетворення, що змінюють внутрішнє подання даних, і перетворення, що змінюють тільки інтерпретацію внутрішнього подання. Наприклад, коли дані типу ***unsigned int*** переводяться в тип ***int***, змінювати їх внутрішнє подання не потрібно – змінюється тільки інтерпретація. При перетворенні значень типу ***float*** у значення типу ***int*** недостатньо змінити тільки інтерпретацію, необхідно змінити довжину ділянки пам'яті для внутрішнього подання й кодування. При такому перетворенні з ***float*** в ***int*** можливий вихід за діапазон допустимих значень типу ***int***, і реакція на цю ситуацію істотно залежить від конкретної реалізації. Саме тому для збереження мобільності програм у них рекомендується з обережністю застосовувати перетворення типів.

Розглянемо послідовність виконання перетворення операндів в арифметичних виразах.

1. Всі короткі цілі типи перетворяться в типи не меншої довжини відповідно до таблиці 7.3. Потім обидва значення, що беруть участь в операції, приймають однаковий тип у відповідності з нижченаведеними правилами.

Таблиця 7.4 – Правила стандартних арифметичних перетворень

Вихідний тип	Перетворений тип	Правила перетворень
char	int	Розширення нулем або знаком залежно від замовчування для char
unsigned char	int	Старший байт заповнюється нулем
signed char	int	Розширення знаком
short	int	Зберігається те ж значення
unsigned short	unsigned int	Зберігається те ж значення
enum	int	Зберігається те ж значення
бітове поле	int	Зберігається те ж значення

2. Якщо один з операндів має тип ***long double***, то другий теж буде перетворений в ***long double***.

3. Якщо пункт 2 не виконується й один з операндів є ***double***, інший приводиться до типу ***double***.

4. Якщо пункти 2–3 не виконуються і один з операндів має тип ***float***, то другий приводиться до типу ***float***.

5. Якщо пункти 2–4 не виконуються (оба операнда цілі) і один операнд ***unsigned long int***, то обидва операнда перетворяться до типу ***unsigned long int***.

6. Якщо пункти 2–5 не виконуються і один операнд є ***long***, інший перетвориться до типу ***long***.

7. Якщо пункти 2–6 не виконуються й один операнд ***unsigned***, то інший перетвориться до типу ***unsigned***.

8. Якщо пункти 2–7 не виконуються, то обидва операнди належать до типу ***int***.

Використовуючи арифметичні вирази, варто враховувати наведені правила та не потрапляти в «пастки» перетворення типів, тому що деякі з них приводять до втрат інформації, а інші змінюють інтерпретацію бітового (внутрішнього) подання даних.

На рисунку 7.1 стрілками відзначені «безпечні» арифметичні перетворення, що гарантують збереження точності й незмінність числового значення.

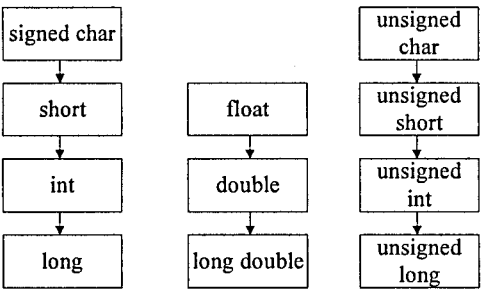


Рисунок 7.1 – Арифметичні перетворення типів, що гарантують збереження значимості

При перетвореннях, які не віднесені схемою до безпечних, можливі істотні інформаційні втрати. Для оцінювання значимості таких втрат рекомендується перевірити оборотність перетворення типів. Перетворення цілочислових значень у дійсні здійснюється настільки точно, наскільки це передбачено апаратурою. Якщо конкретне цілочислове значення не може бути точно подане як дійсне, то молодші значущі цифри губляться й оборотність неможлива.

Приведення дійсного значення до цілого типу виконується за рахунок відкидання дробової частини. Перетворення цілої величини в дійсну також може призвести до втрати точності.

Розглянемо фрагмент програми для визначення правил перетворення типів.

```
char ch=100;
short int shi=3200;
printf("ch=\t%c\n", ch);
printf("(short int)ch=\t%d\n", (short int) ch);// виведення символьної
                                    //змінної в форматі типу short int
printf("shi=\t%d\n", shi);
printf("error  (char)shi=\t%d\n", (char) shi); // виведення змінної shi
                                    // в форматі типу char
```

Зверніть увагу, що в другому випадку перетворення типів відбулась втрата інформації через те, що тип ***char*** має менший діапазон ніж тип ***short int.***

7.3 Варіанти індивідуальних завдань

Варіант № 1

Деякі види антилоп			
Назва	Група	Житло	Чисельність популяції
Джейран	А	Азія	30000
Гну	В	Африка	560000
Бейза	Н	Африка	2500
Групи: А – справжні антилопи, В – коров'ячі антилопи, Н – кінські антилопи			

Варіант № 2

Фірми - виробники СКБД			
Фірма	Кількість продуктів	Річний обсяг продажу ($)	Частина ринку (%)
Oracle	1	2488000000	31.1
IBM	3	2392000000	29.9
Microsoft	2	1048000000	13.1
Примітка. За даними Gartner Group за 1999 р.			

Варіант №3

Відділ кадрів			
Прізвище	Ініціали	Рік народження	Оклад
Іванов	І. І.	1975	517.50
Петренко	П. П.	1956	219.10
Паниковський	М. С.	1967	300.00
Примітка. Оклад встановлений за станом на 1 січня 2000 року			

Варіант № 4

Відомість деталей			
Найменування	Тип	Кількість	Вага 1 деталі (г)
Фланець	З	3	450
Перехідник	П	8	74
Станина	Про	1	117050
Примітка. Кодування типів: Про – оригінальний, П – купівельний, З – запозичений			

Варіант № 5

Характеристики ПЕОМ			
Процесор	Частота (MHz)	RAM (Mb)	Тип
Pentium-III	233	256	C
AMD-K6	166	512	C
PowerPC-620	2000	1024	R
Тип: C – CISC-процесор, R – RISC-процесор			

Варіант № 6

Каталог бібліотеки			
Автор книги	Назва	Рік випуску	Група
Сенкевич	Потоп	1978	Х
Ландау	Механіка	1989	Н
Дойль	Сумчасті	1990	Д
Примітка. Х – художня; Н – навчальна; Д– довідкова			

Варіант № 7

Відомість комплектуючих			
Позначення	Тип	Номінал	Кількість
RT-11-24	R	100000	12
RT-11-24	R	50000	10
CGU-12K	C	17.5	3
Примітка. R – резистор; C – конденсатор			

Варіант № 8

Проекти пошуку позаземних сигналів			
Рік	Науковий керівник	Діаметр антени (м)	Робоча частота (Мгц)
1960	Дрейк	26	1420
1970	Троїцький	14	1875
1978	Хоровіц	300	1665
Примітка. Спостерігалися об'єкти від 2 зірок до декількох галактик			

Варіант № 9

Офісні пакети			
Найменування	Виробник	Кількість скл. частин	Ціна ($)
Office	Microsoft	4	870
SmartSute	Lotus	5	1020
StarOffice	Sun	4	9
Примітка. Можна безкоштовно одержати продукт StarOffice через Internet			

Варіант № 10

Сільськогосподарські культури			
Найменування	Тип	Посівна площа (га)	Врожайність (ц/га)
Соя	Би	13000	45
Чумиза	З	8000	17
Рис	З	25650	24
Примітка. З – зернові, Би – боби			

Варіант № 11

Відомість спортивних змагань			
Прізвище учасника	Код команди	Кількість балів	Місце у результаті
Баландін	С	123.7	2
Шишков	Ш	79.98	3
Кравченко	Д	134.8	1
Примітка. Д – «Динамо», С – «Спартак», Ш – «Шахтар»			

Варіант № 12

Відомість суспільного транспорту			
Вид транспорту	№ маршруту	Протяжність маршруту (км)	Час в дорозі (хв)
Тр	12	27.55	75
Тс	17	13.6	57
А	12а	57.3	117
Примітка. Тр – трамвай, Тс – тролейбус, А – автобус			

Варіант № 13

Приблизна кількість зірок різних спектральних класів в Галактиці			
Спектральний клас	Приблизна маса, M_O	Частина, %	Чисельність
O	60	0.00003	55000
F	1.7	3.03398	12000000000
M	0.3	76.4563	293000000000
Примітка. Не показані дані для класів: B, A, G			

Варіант № 14

Конфігурація програмних засобів інформаційних систем				
Операційна система	СУБД	Мін. обсяг зовнішньої пам'яті (MB)	Мін. обсяг оперативної пам'яті (MB)	Приблизна ціна ($)
OS/2	DB2	130	22	3343
Windows/NT	SQLServer	230	24	2685
SCO/Unix	Oracle	110	48	3745
Примітка. Приймалася ціна ліцензії на 8 користувачів				

Варіант № 15

Сільськогосподарські підприємства			
Назва	Вид власності	Площа землі (га)	Кіл. працівників
Зоря	Д	300	120
Росинка	Ко	174	27
Петренко	П	56	6
Вид власності: Д – державна, П – приватна, Ко – кооперативна			

7.3 Контрольні питання

1. Які типи даних в мові С Ви знаєте? Навести приклади.
2. Які граничні значення вони мають?
3. Як можна перетворювати дані одного типу в інший?
4. Які бібліотеки використовуються в мові С?
5. Що таке бібліотека stdio.h?
6. Які функції в ній містяться? Навести приклади.
7. Які функції введення–виведення використовуються в мові С?
8. Які параметри вони мають?
9. Як вивести число з різною точністю?
10. Як вивести число в експоненціальній формі?
11. Які особливості введення–виведення символьних даних існують в мові С?
12. За допомогою яких функцій вводяться–виводяться окремі символи?
13. За допомогою яких функцій вводяться–виводяться рядки?
14. Що таке специфікатор?
15. Які специфікатори Ви знаєте? Навести приклади.

ЛАБОРАТОРНА РОБОТА № 8
АРИФМЕТИЧНІ ОПЕРАЦІЇ І МАТЕМАТИЧНІ ФУНКЦІЇ МОВИ С

Мета роботи: одержання практичних навичок у програмуванні математичних виразів і використанні математичних функцій бібліотеки math мови С.

8.1 Порядок виконання роботи

8.1.1 Ознайомтесь з теоретичними відомостями до лабораторної роботи.

8.1.2 Розробіть програму, що підраховує і виводить значення t1 і t2 за формулами, що наведені у вашому варіанті індивідуального завдання. Визначите області припустимих значень параметрів формул і задайте довільні значення з цих областей. Параметри, що мають імена: a і b – цілі, інші параметри – дійсні. Значення параметрів з іменами x і y повинні вводитися з клавіатури, значення інших – задаватися як початкові значення при оголошенні відповідних змінних. Допускається (і навіть бажано) спростити / розкласти формули для того, щоб забезпечити мінімізацію обсягу обчислень.

8.1.3 Оформіть звіт та зробіть висновки.

8.2 Теоретичні відомості

Пріоритет операцій та порядок обчислення виразів

В мові С операції з більшими пріоритетом (*priority*) обчислюються першими. Найвищим пріоритетом є пріоритет рівний 1. Пріоритети і порядок операцій наведено в таблиці 8.1.

Таблиця 8.1. – Пріоритет виконання операцій в мові С

Пріоритет	Знак операції	Типи операції	Порядок виконання
1	2	3	4
1	() [] . ->	Вирази	Зліва направо
2	- ~ ! * & ++ -- sizeof приведення типів	Унарні	Справа наліво
3	* / %	Мультиплікативні	Зліва направо
4	+ -	Адитивні	Зліва направо
5	<< >>	Зсув	Зліва направо
6	< > <= >=	Відношення	Зліва направо
7	== !=	Відношення (рівність)	Зліва направо

Продовження таблиці 8.1

1	2	3	4
8	&	Порозрядне І	Зліва направо
9	^	Порозрядне виключне АБО	Зліва направо
10	|	Порозрядне АБО	Зліва направо
11	&&	Логічне І	Зліва направо
12	||	Логічне АБО	Зліва направо
13	? :	Умовна	Зліва направо
14	= *= /= %= += -= &= |= >>= <<= ^=	Просте і складне присвоєння	Справа наліво
15	,	Послідовне обчислення	Зліва направо

В таблиці 8.2 наведено список функцій бібліотеки ***math.h***.

Таблиця 8.2. – Список функцій стандартної бібліотеки математичних функцій (файл math.h)

Функція	**Прототип та значення**
1	2
abs	**int abs(int i);** Повертає абсолютне значення цілого аргументу **i**.
acos	**double acos(double x);** Функція арккосинуса. Значення аргументу повинно знаходитись в діапазоні від –1 до +1.
asin	**double asin(double x);** Функція арксинуса. Значення аргументу повинно знаходитись в діапазоні від – 1 до +1.
atan	**double atan(double x);** Функція арктангенса.
atan2	**double atan2(double y, double x);** Функція арктангенса від значення **y/x**.
cabs	**double cabs(struct complex znum);** Обчислює абсолютне значення комплексного числа **znum**. Визначає структуру (типу) **complex** – в файлі **math.h**.

Продовження таблиці 8.2

cos	**double cos(double x);** Функція косинуса. Кут задається в радіанах.
cosh	**double cosh(double x);** Повертає значення гіперболічного косинуса *x*.
exp	**double exp(double x);** Обчислює значення e^x (експоненціальна функція).
fabs	**double fabs(double x);** Повертає абсолютне значення дійсного аргументу *x* подвійної точності.
floor	**double floor(double x);** Знаходить найбільше ціле, що не перевищує значення *x*. Повертає його в форматі **double**.
fmod	**double fmod(double x, double y);** Повертає остачу від ділення *x* на *y*.
hypot	**double hypot(double x, double y);** Обчислює гіпотенузу z прямокутного трикутника за значеннями катетів x, y ($z^2=x^2+y^2$).
labs	**double labs(long x);** Повертає абсолютне значення цілого аргументу **long x**.
ldexp	**double ldexp(double v, int e);** Повертає значення виразу $v \cdot 2^e$.
log	**double log(double x);** Повертає значення натурального логарифма (**ln x**).
log10	**double log10(double x);** Повертає значення десяткового логарифма ($\log_{10}$x).
poly	**double poly(double x, int n, double c[ ]);** Обчислює значення полінома: $c[n]x^n + c[n-1]x^n + \ldots + c[1]x + c[0]$
pow	**double pow(double x, double y);** Повертає значення x^y, тобто *x* в степені *y*.

Продовження таблиці 8.2

pow10	**double pow10(int p);** Повертає значення 10^p.
sin	**double sin(double x);** Функція синуса. Кут задається в радіанах.
sinh	**double sinh(double x);** Повертає значення гіперболічного синуса для x.
sqrt	**double sqrt(double x);** Повертає додатне значення квадратного кореня $\sqrt{x}$.
tan	**double tan(double x);** Функція тангенса. Кут задається в радіанах.
tanh	**double tanh(double x);** Повертає значення гіперболічного тангенса для x.

8.3 Варіанти індивідуальних завдань

Варіант 1

$$t1 = \frac{ax}{y} + \frac{b}{y^2}\lg(yx + c)$$

$$t2 = \frac{1}{2ab}\ln\frac{\sqrt{c^2 - b^2}\,\mathrm{tg}\,ax + 2}{\sqrt{c^2 - b^2}\,\mathrm{tg}\,ax - 2}$$

Варіант 2

$$t1 = \frac{1}{c}\left[\frac{b}{a}\ln(ax + b) + \frac{d}{y}\ln(yx + d)\right]$$

$$t2 = \frac{1}{a(n-1)}\frac{\sin ax}{\cos^{n-1} ax}$$

Варіант 3

$$t1 = \frac{1}{c}\left(\frac{1}{ax + b} + \frac{y}{c}\ln\frac{yx + a}{ax + b}\right)$$

$$t2 = \frac{\sin ax}{2a\cos^2 x} + \frac{1}{2a}\ln\mathrm{tg}\frac{ax}{2}$$

Варіант 4

$$t1 = \frac{b}{(a-b)(b+x)} - \frac{a}{(a-b)^2}\ln\frac{a+x}{b+x}$$

$$t2 = \frac{1}{a}\left(\ln\mathrm{tg}\frac{ax}{2} - \frac{1}{\sin ax}\right)$$

Варіант 5

$$t1 = \frac{-1}{(a-b)^2}\left(\frac{1}{a+x} + \frac{1}{1+x}\right) + \frac{2}{(a-b)^3}\ln\frac{a+x}{b+x}$$

$$t2 = -\frac{1}{2a}\left(\frac{\cos ax}{\sin^2 ax} - \ln\tan\frac{ax}{2}\right)$$

Варіант 6

$$t1 = \frac{1}{a}\left(\frac{-1}{(n-2)x^2} + \frac{b}{(n-1)x^2}\right)$$

$$t2 = \frac{2x}{a^2}\sin ax - \left(\frac{x^2}{a} - \frac{2}{a^3}\right)\cos ax$$

Варіант 7

$$t1 = \frac{1}{a^3}\left(\ln x + \frac{2b}{x} - \frac{b^2}{2x^2}\right)$$

$$t2 = \frac{\cos ax}{2a\sin^2 ax} + \frac{1}{2a}\ln \operatorname{tg}\frac{ax}{2}$$

Варіант 8

$$t1 = \frac{1}{a^4}\left(\frac{x^3}{3} - 3bx + 3b^2 \ln x + \frac{b^3}{x}\right)$$

$$t2 = \frac{1}{1-\sin ax} + \frac{1}{a}\operatorname{tg}\frac{ax}{2}$$

Варіант 9

$$t1 = \frac{1}{b^2}\left(\ln\frac{y}{x} + \frac{ax}{y}\right)$$

$$t2 = -\frac{x}{a}\operatorname{tg}\frac{ax}{2} + \frac{2}{a^2}\ln\sin\frac{ax}{2}$$

Варіант 10

$$t1 = \frac{1}{b^3}\left(\ln\frac{y}{x} - \frac{a^2b^2}{2y^2}\right)$$

$$t2 = \frac{1}{a}\operatorname{tg}\frac{ax}{2} + \frac{1}{a}\ln\operatorname{tg}\frac{ax}{2}$$

Варіант 11

$$t1 = -a\left(\frac{1}{b^2y} + \frac{1}{ab^2x} - \frac{2}{b^3}\ln\frac{y}{x}\right)$$

$$t2 = \frac{1}{2a}\operatorname{ctg}\frac{ax}{2} + \frac{1}{6a}\operatorname{ctg}^3\frac{ax}{2}$$

Варіант 12

$$t1 = \frac{1}{b^3}\left(a^2\ln\frac{y}{x} + \frac{2ax}{y} + \frac{y^2}{2x^2}\right)$$

$$t2 = \frac{1}{2\sqrt{2}a} + \frac{3\sin^2 ax - 1}{\sin^2 ax - 1}$$

Варіант 13

$$t1 = \frac{1}{b^4}\left(3a^3\ln\frac{y}{x} + \frac{a^2x}{y} - \frac{3ay}{x}\right)$$

$$t2 = \frac{2b\operatorname{tg}\frac{ax}{2}}{a\sqrt{b^2-c^2}}$$

Варіант 14

$$t1 = \frac{1}{2(n-1)x^{n-1}} + \frac{a}{2nx^n}$$

$$t2 = \frac{1}{2a}\operatorname{tg}^2 ax + \frac{1}{a}\ln\cos ax$$

Варіант 15

$$t1 = \frac{1}{4a^2x^2} + \frac{1}{2a^4x} + \frac{1}{2a^6}\ln\frac{y^2}{x}$$

$$t2 = \frac{x}{2} + \frac{1}{2a}\ln(\sin ax + \cos ax)$$

8.4 Контрольні питання

1. Яка бібліотека використовується для обчислення математичних операцій в мові С?
2. Що таке пріоритет операцій?
3. Як змінити пріоритет виконання операцій?
4. Перерахуйте назви функцій стандартної бібліотеки для обчислення тригонометричних математичних функцій.
5. Які функції використовуються для обчислення логарифмів?

ЛАБОРАТОРНА РОБОТА № 9
РОЗРОБКА С-ПРОГРАМ З ВИКОРИСТАННЯМ ОПЕРАТОРІВ ВИБОРУ

Мета роботи: освоїти прийоми використання операторів вибору (*conditional statements*) в програмах, написаних алгоритмічною мовою С.

9.1 Порядок виконання роботи

9.1.1 Ознайомтесь з теоретичними відомостями до лабораторної роботи.

9.1.2 Розробіть алгоритм, схему алгоритму і програму для розв'язання індивідуального завдання 1.

9.1.3 Розробіть алгоритм, схему алгоритму і програму для розв'язання завдання 2, що полягає в обчисленні функції f(x), заданої графічне. Розробіть 4-6 тестів для перевірки правильності розв'язання задачі.

9.1.4 Оформіть звіт та зробіть висновки

9.2 Теоретичні відомості

Оператор (*statement*) – це літерал, що змушує компілятор виконувати деякі дії.

Оператори мови С поділяються на:

1) оператори перетворення даних;
2) оператори керування програмою.

До **операторів перетворення даних** відносять:

- оператор присвоювання (**x = a + b**);
- оператор складеного присвоювання (**x *= i, A=У=З=40**);
- арифметичний вираз **i ++**.

До **операторів керування роботою програми** відносять:

- складені оператори (з використанням стандартних операцій: ***=; +=; – =; /=; %=**);
- оператори вибору (оператор **if** та ключ **switch**);
- оператори циклів (**while, do, for**);
- оператори переходу.

Оператори циклів в С є трьох видів:

- з передумовою **while**;
- з післяумовою **do**;
- параметричний **for**;

Оператори переходу виконують безумовну передачу керування:

goto – безумовний перехід;
continue – завершення поточної операції;

break – вихід з циклу чи перемикача;
return – повернення з функції;

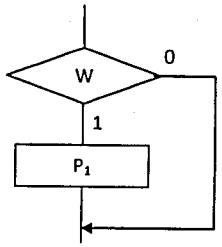
Оператор вибору if

У мові С використовуються такі конструкції оператора if:

1) **if** (вираз_умова)
оператор;

Наприклад: if (x>0)
y = x;

Даний оператор алгоритмічно відповідає базовій конструкції «перемикач»



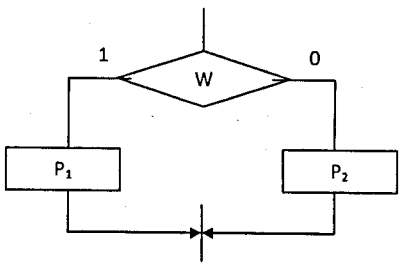
2) **if** (вираз_умова)
{
оператор1;
оператор2;
- - - - - - - -
оператор n;
}

Наприклад: if (x + y < 0)
{
y = x;
z = 2 * x;
}

3) **if** (вираз_умова)
оператор1;
else
оператор2;

Даний оператор алгоритмічно відповідає базовій конструкції «розподіл на два потоки»

Наприклад: if (x > 0)
y = x;
else
y = -x;



4) **if** (вираз_умова)
{
оператор1;
оператор2;
- - - - - - - -
оператор n;
}
else
{
оператор1;
оператор2;
- - - - - - - -
оператор n;
}

5) Вкладені оператори

```
if (вираз_умова 1)
        оператор1;
else  if (вираз_умова 2)
        оператор2;
else  if (вираз_умова 3)
        оператор3;
```

Програма демонстрації **оператора if**

```
#include <stdio.h>

int main()
{
  int number, okay;
  printf(" Enter a number from 1 to 10");
  scanf ("%d", & number);
  okay=(1<= number) &&( number<=10);
  if (!okay)
    printf("Incorrect answer!!! \n");
  return okay;
}
```

В даній програмі є логічний вираз, який має вигляд:

okay=(1<= number) &&(number<=10);

Результат цього виразу дорівнює **0**, якщо вираз «помилковий» (**false**) і **1**, якщо «істинний» (**true**).

Функція введення даних з клавіатури

scanf ("%d", & number),

де **"%d"** – форматний рядок;

& number – адреса змінної number (**&** – унарна операція визначення адреси змінної).

Програма демонстрації конструкції **if-else** для визначення високосного року.

```
#include <stdio.h>
int main()
{
  int leapYear;
  int year;
  printf (" Leap Year calculator \n");
  printf (" Year? \n");
  scanf("%d", & year);
  if (year>0);
  {
    if ((year %100) == 0)
      leapYear = ((year %4) == 0);
```

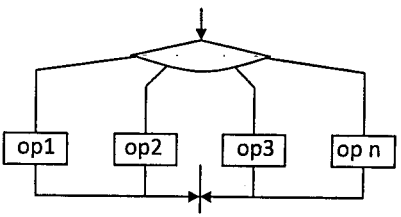
```
        else
          leapYear = ((year %4) == 0);
        if (leapYear)
          printf("%d is a leap Year \n", year);
        else
          printf("%d is not a leap Year \n", year);
    }
  return 0;
  }
```

Оператор switch (перемикач мультиканальний)

Вкладений набір операторів if-else може виглядати як заплутаний водопровід старого будинку. Система працює, але важко зрозуміти, яка труба куди веде.

Розглянемо вкладений набір операторів if-else, кожний з який порівнює вираз з певним значенням:

```
if (вираз = = значення 1)
          Оператор 1;
else if (вираз = = значення 2)
          Оператор 2;
else      if (вираз = = значення 3)
       Оператор 3;
else
  Оператор за умовчанням;
```



Цю конструкцію можна скоротити за допомогою оператора **switch**:

```
switch (вираз)
{
  case значення 1:
        оператор 1;
        break;        //виконує вихід з оператора switch
  case значення 2:
        оператор 2;
        break;
  case значення 3:
        оператор 3;
        break;
      default: оператор за умовчанням; // виконується, якщо
                                       // не було збігів
}
```

Приклад програми-демонстрації **switch** (розробка меню вибору)

```
# include <stdio.h>
```

```
# include <ctype.h>
int main()
{
  int choice;
  printf(" menu: A)dd  D)elete  S)ort  Q)uit:");
  choice = toupper(getchar());
  switch(choice)
  {
    case 'A':
      printf("You selected Add \n");
      break;
    case 'D':
      printf("You selected Delete \n");
      break;
    case 'S':
      printf("You selected Sort \n");
      break;
    case 'Q':
      printf("You selected Quit \n");
      break;
    default:
      printf("\n\n Illegal choice!!!/n");
  }
return choice;}
```

9.3 Варіанти завдання 1

1. Розрахувати площу прямокутника, якщо його сторони додатні.

2. Розрахувати *S=x-x3/6*, якщо *0,1<=x<=1*, інакше *y=sin(x)*.

3. Провести розрахунок за формулою *z=a2+b2+(a-b)2*, якщо *a* – додатне, *b* – від'ємне, інакше розрахунок не проводити.

4. Задані числа *a, b, c*. Знайти найбільше з них.

5. Розрахувати *S=x- x3/6 +x5/120*, якщо *0,1<=x<=1*, інакше $y=e^{2x}$.

6. Розрахувати вираз

$$\begin{cases} y=\sin(x), & x>0 \\ y=x^2/4, & x\le 0 \end{cases}$$

7. Написати програму, яка визначає, чи належить число *N* заданому інтервалу [*k, m*].

8. Серед чисел *A* и *B* знайти більше 20 і вивести результат на экран.

9. Задані числа *x* та *y*. Якщо їх сума додатна, вивести на экран ці числа, інакше вивести на екран їх різницю.

10. Задати значення змінним *C* і *D*. Якщо вони мають від'ємні значення, вивести на екран суму квадратів цих чисел, інакше – квадрат суми.

11. Задати значення змінним цілочислового типу *X* і *Y*. Якщо *X<Y* вивести остачу від ділення *X* на *Y*, в іншому випадку вивести на екран цілу частину від ділення *X* на *Y*.

12. Задати значення для цілочислової змінної X і для дробової змінної Y. Знайти суму цих чисел, якщо $X > 0$, в іншому випадку, знайти результат від ділення X на Y і помістити його в цілу змінну Z.

13. Серед заданих чисел X і Y знайти від'ємні і вивести на екран модулі цих чисел.

14. Вивести на екран цілу частину дробового числа x, визначеного за допомогою виразу $x = y + 20y$, якщо значення y від'ємне.

15. Серед чисел x і y, введених оператором введення, знайти додатні і вивести на екран цілу і дробову частини дробових чисел x і y окремо.

9.4 Варіанти завдання 2

Варіант 1

$$z = \begin{cases} (x-y)^2, \text{якщо } x > 0.1 \text{ і } y > 0.1 \\ \sin(x), \text{якщо } x < 0.1 \text{ і } y < 0.1 \\ \dfrac{x^2}{y}, \text{якщо } x = 0.1 \text{ і } y = 0.1 \end{cases}$$

Варіант 2

$$y = \begin{cases} |-x-1|^2, \text{якщо } x \le 0.3 \\ \sin^3(x) - \cos^2(x-3), \text{якщо } x \in [0.3; 0.9] \\ \dfrac{x^4}{2e^x}, \text{якщо } x \ge 0.9 \end{cases}$$

Варіант 3

$$t = \begin{cases} |v| - 2, \text{якщо } v \le 0 \\ 1/\sin(3v), \text{якщо } v \in [0;1] \\ 3v^3 + e^{2v+1}, \text{якщо } v \ge 1 \end{cases}$$

Варіант 4

$$g = \begin{cases} |x+1| + e^{3x}, \text{якщо } x \le -1 \\ -2x/\sin(x+3), \text{якщо } x \in [-1;1] \\ x^3 + 2|\cos(3x)|, \text{якщо } x \ge 1 \end{cases}$$

Варіант 5

$$y = \begin{cases} z * x^2, \text{якщо } x < 0 \, і \, z < 0 \\ \sqrt{z^2 + x^2}, \text{якщо } x > 0 \, і \, z > 0 \\ 100 \text{ в інших випадках} \end{cases}$$

Варіант 6

$$q = \begin{cases} a^2 + c^2 * x, \text{якщо } x \ge 0 \, і \, c \ge 0 \, і \, a \ge 0 \\ \dfrac{c^2 + x^2}{a}, \text{якщо } x < 0 \, і \, a < 0 \, і \, c < 0 \end{cases}$$

Варіант 7

$$g = \begin{cases} z^3, \text{якщо } z \le 1 \\ \dfrac{1}{z}, \text{якщо } z \in [-1;2] \\ \sqrt{z+1} - e^{5z}, \text{якщо } z \ge 2 \end{cases}$$

Варіант 8

$$p = \begin{cases} \dfrac{1}{|a + \sin(a)|}, \text{якщо } a < 0 \\ 0, \text{якщо } a = 0 \\ -a + \sqrt{3\sin(a)}, \text{якщо } a > 0 \end{cases}$$

Варіант 9

$$\mathrm{p} = \begin{cases} \mathrm{e^b + 7b^3}, \text{якщо } \mathrm{b} \le 4 \\ \frac{1.8}{\sin^3(\mathrm{b^2})}, \text{якщо } 4 < b < 6 \\ |6 - (\mathrm{b} - 2)^3|, \text{якщо } \mathrm{b} \ge 6 \end{cases}$$

Варіант 10

$$y = \begin{cases} z + 4|t^3|sin(t), \text{якщо } t < 0 \, i \, z < 0 \\ \sqrt{z^2 + 3cos(t^2)}, \text{якщо } t > 0 \, i \, z > 0 \\ 1 \text{ в інших випадках} \end{cases}$$

Варіант 11

$$\mathrm{f} = \begin{cases} |-\mathrm{x}| - 3\sin(\mathrm{x^3}), \text{якщо } \mathrm{x} < 1.3 \\ \frac{3\mathrm{x}}{\sin(\mathrm{x}+1)+\cos^2(\mathrm{x})}, \text{якщо } \mathrm{x} = 1.3 \\ (\mathrm{x} - 1.1)/\mathrm{e^{-2x+1}}, \text{якщо } \mathrm{x} > 1.3 \end{cases}$$

Варіант 12

$$s = \begin{cases} |cos^2(2b) - 3b^3|, \text{якщо } b \le -2 \\ \frac{8\sqrt{b+1}}{sin(3+b^2)}, \text{якщо } -2 < b < 2 \\ (b + 2)^4 + 6b, \text{якщо } x \ge 2 \end{cases}$$

Варіант 13

$$w = \begin{cases} |x - 2| - 5x^3, \text{якщо } x < 1.4 \\ 1.3sin(3x), \text{якщо } x = 1.4 \\ cos^2(x) + e^{-2x+1}, \text{якщо } x > 1.4 \end{cases}$$

Варіант 14

$$z = \begin{cases} x + ye^x, \text{якщо } x < 0 \, i \, y < 0 \\ \sqrt{sin(x)}cos(y), \text{якщо } x \ge 0.1 \, i \, y \ge 0.1 \\ y^3 - x \text{ в інших випадках} \end{cases}$$

Варіант 15

$$s = \begin{cases} z^2 + cos(2z), \text{якщо } z \le 0.8 \\ -2z|sin(z)|, \text{якщо } z \in [0.8; 2] \\ \frac{\sqrt{z+7}}{e^{4-z}}, \text{якщо } z \ge 2 \end{cases}$$

9.5 Контрольні питання

1. Для чого призначені оператори вибору?
2. Який алгоритм називається розгалуженим?
3. Яка умова називається складною?
4. Який вираз називають булевим? Чому?
5. Чи можуть виконуватись декілька розгалужень оператора вибору за один раз?
6. Скільки операторів можна записати після можливого значення змінної (виразу) в операторі вибору?

ЛАБОРАТОРНА РОБОТА № 10
РОЗРОБКА ЦИКЛІЧНИХ С-ПРОГРАМ

Мета роботи: придбати навички розробки програм з використанням циклічних операторів (*loop statements*) алгоритмічної мови С.

10.1 Порядок виконання роботи

10.1.1 Ознайомтесь з теоретичними відомостями до лабораторної роботи.

10.1.2 Розробити описовий алгоритм, схему алгоритму і програму для виконання завдання 1.

10.1.3 Розробити описовий алгоритм, схему алгоритму і програму для виконання завдання 2.

10.1.4 Для правильної роботи програм розробити 4-6 тестів.

10.1.5 Оформіть звіт та зробіть висновки.

10.2 Теоретичні відомості

Оператор while – оператор передумови

На рисунку 10.1 показано фрагмент блок-схеми, що відповідає циклу з передумовою.

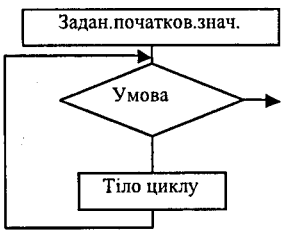


Рисунок 10.1 – Цикл з передумовою

Один із трьох операторів циклу, що виконують багаторазове повторення деяких дій, поки задана умова істинна.

```
while (вираз)
        оператор;
або
while (вираз)
{
        оператор 1;
        оператор 2;
        --------
```

оператор n;

}

Приклад 1. Виконання оператора while для виведення цілих чисел від 1 до 10:

```
#include <stdio.h>

int main()
{
   int counter;
   printf("while count\n");
   counter=1;
   while(counter<=10)
   {
      printf("%d\n",counter);
      counter++;  // збільшення значення на 1 на
                  // кожній ітерації циклу
   }
   return 0;
}
```

Приклад 2. Виведення букв алфавіту

```
#include <stdio.h>
int main()
{
   int c;
   printf("alphabet\n");
   c='A';
   while(c<='Z')
   {
      printf("%c\n",c);
      c++;
   }
   return 0;
}
```

Оператор do-while – оператор з післяумовою

На рисунку 10.2 показано фрагмент блок-схеми, що відповідає циклу з післяумовою.

Рисунок 10.2 – Цикл з післяумовою

Один із трьох операторів циклу, що виконують багаторазове повторення деяких дій, поки істинний вираз в while.

Формат простого оператора:

do оператор; while(вираз);

або

```
do
    оператор;
while(вираз);
```

або

```
do
{
    оператор 1;
    оператор 2;
     - - - - - - -
} while(вираз);
```

Приклад 1. Виведення цілих чисел від 1 до 10

```
#include <stdio.h>
int main()
{
  int counter;
  printf(" do while count \n");
  counter=1;
  do
  {
    counter++;
    printf("%d \n", counter);
  }while(counter<=10);
  return 0;
}
```

Приклад 2. Виведення букв алфавіту

```
#include <stdio.h>
int main()
{
  int c;
  printf("do while alphabet \n");
  c='A' - 1;
  do
  {
    c++;
    printf("%c",c);
  }while(c<'Z');
  return 0;
}
```

Оператор for

На рисунку 10.3 показано фрагмент блок-схеми, що відповідає циклу за параметром.

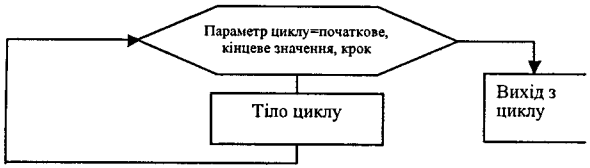


Рисунок 10.3 – Цикл за параметром

Циклічний оператор використовується, коли відоме число повторень тіла циклу. Формат оператора: **for(вираз1; вираз2; вираз3;)**

```
{
  оператор;
}
```

Вираз 1 – визначає дії, виконувані до початку циклу, тобто задає початкові умови для циклу; найчастіше це вираз присвоювання;

Вираз 2 (вираз умова) – звичайно логічний чи арифметичний. Він визначає умову закінчення чи продовження циклу. Якщо він «істинний» (тобто не дорівнює 0), то виконується тіло циклу, а потім обчислюється вираз 3.

Вираз 3 – звичайно задає необхідні для наступної ітерації зміни параметрів чи будь-яких змінних тіла циклу. Після виконання виразу обчислюється істинність виразу 2 і все повторюється.

Тобто вираз 1 обчислюється тільки один раз, а вирази 2 і 3 обчислюються після кожного виконання тіла циклу. Цикл продовжується доти, поки не стане помилковим вираз 2.

Вирази 1 і 3 можуть складатися з декількох виразів, розділених комами.

Кожен з трьох, будь-які два чи всі три вирази в операторі for можуть бути відсутні, але розділяючі їх символи «;» повинні бути присутніми завжди. Якщо відсутній вираз 2, то вважається, що він істинний і потрібні наступні спеціальні засоби для виходу з циклу.

Наприклад, для обчислення факторіала 5! використовується порожній оператор як тіло циклу, коли всі циклічно виконувані дії визначені в його заголовку:

Формула для обчислення факторіала має вид рекурентного співвідношення: p=p*i, i=1,2,...n

Фрагмент програми має вид: ***for(int i=0,p=1; i<5; i++,p*=i);***

Розглянемо різні записи оператора for для розв'язання задачі підсумовування квадратів перших к членів натурального ряду:

1) ***for(int i=1,s=0; i<=k; i++) s+=i*i;***
2) ***for(int i=1,s=0; i<=k; s+=++i*i) ;***
3) ***for(int i=1,s=0; i<=k; i++) s+=i*i;***
4) ***for(int i=1,s=0; i<=k)***
   { int j; j=++i; s+=j*j ; }

Оператор for аналогічний за результатом оператору while та відповідає алгоритмічній конструкції

```
вираз 1;
while(вираз 2)
  {
    оператор;
    вираз 3;
  }
```

Вираз 2 – зазвичай це вираз співвідношення. Наприклад, для виведення цілих чисел від 1 до10 можна побудувати такий цикл.

for(i=1; i<=10; i++) printf("i=%d\n",i);

Приклад для відображення набору видимих ANCII-символів (це символи зі значеннями від 32 до127)

```
#include <stdio.h>
int main()
{
  unsigned char c;
  for (c = 32; (c < 128); c++)
  {
    if ((c % 32) == 0)
      printf("\n");printf("%c",c);
  }
  printf("\n");
  return 0;
}
```

Нижченаведені оператори забезпечують нескінченне виконання порожніх операторів:

for(; ;); for(;1;);

Оператор break

Іноді необхідно переривати виконання циклу ***while, do-while, for***, для цього використовують оператор ***break***.

```
#include <stdio.h>
int main()
{
  int count;
  for(count=1;count<=10;count++)
  {
    if(count>5) break;
    printf("%d\n",count);
```

```
    }
    return 0;
}
```

Цей прийом звичайно використовується для відстеження різних (можливих) програмних переривань.

Оператор continue

Схожий на ***break***, але змушує цикл перервати попередню ітерацію і почати наступну.

```
#include <stdio.h>
int main()
{
    int count;
    for(count=1;count<=10; count++)
    {
        if(count>5) continue;
        printf("%d\n",count);
    }
    printf("%d\n",count);
    return 0;
}
```

У цьому прикладі оператор continue пропускає оператор ***printf()***, дозволяє змінній count досягти свого кінцевого значення, яке можна побачити за допомогою другого оператора ***printf***, що знаходиться поза циклом ***for***.

Оператор goto

Оператор дозволяє перейти до деякого іншого оператора (у будь-якому місці програми) і, починаючи з його позиції, продовжити виконання операторів, що стоять за ним. Фактично оператор ***goto*** дозволяє «стрибати» у будь-яке місце програми.

Для ініціалізації goto необхідно використовувати мітки (невикористаний раніше індикатор) і двокрапку.

```
#include <stdio.h>
int main()
{
    int count=1;
    TOP:
    printf("%d\n", count);
    count++;
    if(count<=10) goto TOP;
    return 0;
}
```

Мітка «***TOP:***» указує перехід за оператором ***goto***. Програма, звичайно, працює. Однак варто пам'ятати, що будь-який оператор goto можна замінити оператором циклу ***while*** або ***do-while***.

Програми, що використовують багато операторів goto, як правило, важкі для розуміння і налагодження, тому цей оператор краще не використовувати в програмах узагалі.

Зупинка програми за допомогою exit

Функція ***exit*** визначена в заголовному файлі ***<stdio.h>***.

Виконання оператора негайно завершує програму, закриває усі відкриті файли і виконує інші завершальні дії. Значення в круглих дужках повертається ОС (або в іншу програму, наприклад, у командний файл), яка використовує дану програму.

10.3 Варіанти завдання 1

Розробити описовий алгоритм, схему алгоритму і написати програму для друкування таблиці значень функції y(x) на заданому відрізку $[x_п, x_к]$ із кроком h. Розробити 4-6 тестів для перевірки правильної роботи програми.

1. $$y=\begin{cases}-0.05\lg x^3, & якщо\ \ x>5;\\ \dfrac{\cos^2 x^2}{\sqrt{x^2+4.2}}, & якщо\ \ x\le 2.\end{cases}$$

2. $$y=\begin{cases}x^2\sin 2x, & якщо\ \ x\ge 0{,}2;\\ \dfrac{4{,}73\lg x^2}{\sqrt{x^2+3.7x^4}}, & якщо\ \ x\le -2.6\end{cases}$$

3. $$y=\begin{cases}0.34\cos\dfrac{x}{2}, & якщо\ \ x\le 2.3;\\ \lg^2\left(|\sin x|+0.2\right), & якщо\ \ x>3.7\end{cases}$$

4. $$y=\begin{cases}\dfrac{\sin 2x}{1+\sqrt{\dfrac{x^2}{1+e^{-0.32x}}}}, якщо x>2.1\\ e^{-0.29x}+\cos 2.7x, якщо x\le 1.3\end{cases}$$

5. $y=\begin{cases}2.08x^2+\dfrac{\lg^3 2x^x}{1+x^2}, & якщо x \ge 1.72\\ \cos^3\lg\left(|x|^a+2.13\right), & якщо x < 0.97\end{cases}$

6. $y=\begin{cases}\dfrac{\sin^2 x \lg 0.34x}{1+x^2}, & якщо x > 4.09\\ \sqrt{\dfrac{2x^2+1}{e^{-0.32x}+7.2}}, & якщо x \le 2.32\end{cases}$

7. $y=\begin{cases}\dfrac{2.3\cos^2 x}{\sqrt{x+3.7x^2}}, & якщо x \ge 2.7\\ \ln^2\left(|\cos(2x+1.3)|+0.2\right), & якщо x < 1.8\end{cases}$

8. $y=\begin{cases}\cos\lg\left(1.3+|2.72x|\right), & якщо\ \ x \le -2.1;\\ \sqrt{\dfrac{e^{-0.34x}+5.1x^2}{\cos^2 x+4.193}}, & якщо\ \ x > 1.3\end{cases}$

9. $y=\begin{cases}\dfrac{4.7\cos 0.2x}{\sin 2x+\ln(x^2+1)}, & якщо\ \ x \le 1.2;\\ 8.3\cos 3x, & якщо\ \ x > 5.6\end{cases}$

10. $y=\begin{cases}9.74x^2+\cos 6.4x, & якщо\ \ x \le -3.1;\\ \sqrt{\dfrac{4.3x^2+2}{1+\ln^2(x^2+0.3)}}, & якщо\ \ x > -1.37\end{cases}$

11. $$y=\begin{cases}-0.51\lg|x|, & якщо\ \ x\geq 1.7;\\ \dfrac{\cos^2 x^3}{\sqrt{|x|+4.2x^2}}, & якщо\ \ x<0.9\end{cases}$$

12. $$y=\begin{cases}x^3\cos 0.7x, & якщо\ \ x\geq 0.7;\\ \dfrac{5.131\lg x^2}{\sqrt{2.1x^2+3.7}}, & якщо\ \ x\leq -1.31\end{cases}$$

13. $$y=\begin{cases}0.72\sin\dfrac{x}{3}, & якщо\ \ x<1.7;\\ \cos^2(\sin x+0.6), & якщо\ \ x\geq 4.1\end{cases}$$

14. $$y=\begin{cases}\dfrac{\lg 2x}{1+\sqrt{1.7+e^{-0.73x}}}, & якщо\ \ x>1.92;\\ tg3x+e^{-0.23x}, & якщо\ \ x<-0.35\end{cases}$$

15. $$y=\begin{cases}6.3x+\dfrac{\cos 2x}{1+x^2}, & якщо\ \ x<-0.37;\\ \cos^2\lg(x^2+0.29), & якщо\ \ x>0.62\end{cases}$$

10.4 Варіанти завдання 2

Варіант 1. Розробити описовий алгоритм, схему алгоритму і написати програму, що виводить на екран квадрат Піфагора – таблицю множення. Вигляд екрана, що його рекомендується використовувати під час виконання програми, наведений нижче.

```
    1  2  3 .....10
1   1  2  3 .....10
2   2  4  6 ..... 20
3   3  6  9.....30
.....
9   9 18 27.....90
```

Варіант 2. Розробити описовий алгоритм, схему алгоритму і написати програму, що обчислює часткову суму ряду: 1-1/3+1/5-1/7+1/9-... і порівнює отримане значення з π / 4 (при підсумовуванні достатньо великої кількості членів цього ряду величина часткової суми наближається до π / 4).

Варіант 3. Розробити описовий алгоритм, схему алгоритму і написати програму, що обчислює факторіал уведеного з клавіатури числа. (Факторіалом числа n називається добуток цілих чисел від 1 до n. Наприклад, факторіал 1 дорівнює 1,8 - 40320).

Варіант 4. Розробити описовий алгоритм, схему алгоритму і написати програму, що генерує послідовність з 10 випадкових чисел у діапазоні від 1 до 10, виводить ці числа на екран і обчислює їх середнє арифметичне. Рекомендований вигляд екрана наведений нижче.

```
                    ***Випадкові числа***
 1  3  4  2  7  4  9  6  2  1      серед. арифм. 3.9
```

Варіант 5. Розробити описовий алгоритм, схему алгоритму і написати програму, що виводить таблицю значень функції y=|x-2|+|x+1|. Діапазон зміни аргументу від -4 до 4, крок збільшення аргументу 0,5.

Варіант 6. Розробити описовий алгоритм, схему алгоритму і написати програму, що виводить таблицю степенів двійки від нульового до десятого. Нижче наведений рекомендований вигляд екрана.

```
************
   0   1
   1   2
   2   4
  -----
  10   1024
************
```

Варіант 7. Розробити описовий алгоритм, схему алгоритму і написати програму, що обчислює суму перших n членів ряду: +1/2+1/3+1/4+...Кількість підсумовуваних членів ряду задається під час роботи програми. Нижче наведений рекомендований вигляд екрана.

```
Обчислення часткової суми ряду: 1+1/2+1/3+1/4+...
Введіть кількість підсумовуваних членів ряду –>15
Сума перших 15 членів ряду дорівнює 3.3182
```

Варіант 8. Розробити описовий алгоритм, схему алгоритму і написати програму, що обчислює суму перших n членів ряду: 1,3,5,7... Кількість підсумовуваних членів ряду задається під час роботи програми. Нижче наведений рекомендований вигляд екрана.

Обчислення часткової суми ряду: 1 + 3 + 5 +...
Введіть кількість підсумовуваних членів ряду –>15
Сума перших 15 членів ряду дорівнює 330

Варіант 9. Розробити описовий алгоритм, схему алгоритму і написати програму, що обчислює суму перших n додатних парних цілих чисел. Кількість підсумовуваних чисел повинна вводитися під час роботи програми. Нижче наведений рекомендований вигляд екрана.

Обчислення суми парних додатних чисел.
Введіть кількість підсумовуваних чисел і натисніть <Enter> –>12
Сума перших 12 додатних парних чисел дорівнює 156

Варіант 10. Розробити описовий алгоритм, схему алгоритму і написати програму, що обчислює суму перших n додатних цілих чисел. Кількість підсумовуваних чисел повинна вводитися під час роботи програми. Нижче наведений рекомендований вигляд екрана.

Обчислення суми додатних чисел.
Введіть кількість підсумовуваних чисел –>20
Сума перших 20 додатних чисел дорівнює 210

Варіант 11. Розробити описовий алгоритм, схему алгоритму і написати програму, що виводить таблицю квадратів перших п'яти цілих додатних непарних чисел. Нижче наведений рекомендований вигляд екрана.

```
******************
 Число     Квадрат
******************
   1           1
   3           9
 ------------
   9          81
```

Варіант 12. Розробити описовий алгоритм, схему алгоритму і написати програму, що виводить таблицю квадратів перших десяти цілих додатних чисел. Нижче наведений рекомендований вигляд екрана.

```
******************
 Число     Квадрат
******************
   1           1
   2           4
   3           9
 -----------
  10         100
```

Варіант 13. Розробити описовий алгоритм, схему алгоритму і написати програму, що виводить таблицю значень функції y= -2,4x2 + 5x - 3 у діапазоні від - 2 до 2 із кроком 0,5.

Варіант 14. Розробити описовий алгоритм, схему алгоритму і написати програму, що виводить на екран таблицю множення, наприклад 7. Вигляд рекомендованого екрана наведений нижче.

```
7*2=14
7*3=21
------
7*9=63
```

Варіант 15. Розробити описовий алгоритм, схему алгоритму і написати програму, що виводить на екран таблицю вартості, наприклад, яблук у діапазоні від 100 г до 1 кг із кроком 100 г. Нижче наведений рекомендований вигляд екрана.

```
Введіть ціну одного кілограма і натисніть <Enter>  -> 16.50
    Вага (г)        Вартість (грн)
      100               1.65
      200               3.30
      ---------------
      1000             16.50
```

10.5 Контрольні питання

1. Що таке цикл? Наведіть приклад.
2. Які є типи циклів?
3. Назвіть оператори циклу, дозволені в мові програмування С.
4. Які правила організації циклу за параметром?
5. Які правила організації циклу з передумовою?
6. Які правила організації циклу з післяумовою?
7. Що таке крок? Що означає вираз «крок дорівнює 5»?
8. Як організувати нескінченний цикл?
9. Як організувати можливість дострокового виходу з циклу?

ЛАБОРАТОРНА РОБОТА № 11
ОБРОБКА ОДНОВИМІРНИХ ТА ДВОВИМІРНИХ МАСИВІВ

Мета роботи: Освоїти прийоми обробки одновимірних і двовимірних масивів і придбати навички розробки С-програм для обробки масивів.

11.1 Порядок виконання роботи

11.1.1 Ознайомтесь з теоретичними відомостями до лабораторної роботи.
11.1.2 Розробіть схему алгоритму, програму, що працює в режимі простого меню, для обробки одновимірного масиву відповідно до завдання 1 за варіантами, і тести, що підтверджують правильність роботи програми.
11.1.3 Розробіть алгоритм, схему алгоритму, програму, що працює в режимі простого меню, для обробки двовимірного масиву відповідно до завдання 2 за варіантами, і тести, що підтверджують правильність роботи програми.
11.1.4 Оформіть звіт та зробіть висновки.

11.2 Теоретичні відомості

Масиви і змінні з індексами. Математичним поняттям, що привело до появи в мовах програмування поняття «масив» (*array*), є матриця та її окремі випадки: вектор-стовпець або вектор-рядок. В програмі, також як у математиці, елементи матриці прийнято позначати з використанням індексів. Всі елементи матриці або цілі, або дійсні і т. п. Така «однорідність» елементів властива і масивові. Визначення масиву включає тип елементів, ім'я масиву, його розмірність.

Наприклад, **int** a[10]; визначає масив з 10 елементів a[0], a[1], ..., a[9]. **float Z** [13][[6]; визначає двовимірний масив, перший індекс якого приймає 13 значень від 0 до 12, другий індекс приймає 6 значень від 0 до 5. Таким чином, елементи двовимірного масиву 2 можна перелічити так:

Z[0][0], Z[0][1], Z[0][2],...,Z[12][4], Z[12][5]

В стандарті алгоритмічної мови С представлені тільки одновимірні масиви, однак елементами одновимірного масиву, у свою чергу, можуть бути масиви. Тому двовимірний масив визначається як масив масивів. Таким чином, у прикладі визначений масив Z з 13 елементів-масивів, кожний з яких, у свою чергу, складається з 6 елементів типу **float.** Зверніть увагу, що

нумерація елементів будь-якого масиву завжди починається з 0, тобто індекс змінюється від 0 до N-1, де N – кількість значень індексу.

Обмежень на розмірність масивів (тобто на число індексів його елементів) у мові С теоретично немає. Стандарт мови С вимагає, щоб транслятор (*translator*) міг обробляти визначення масивів з розмірністю до 31. Однак найчастіше використовуються одновимірні і двовимірні масиви.

Ініціалізація масивів. При визначенні масивів можлива їх ініціалізація, тобто присвоювання початкових значень їх елементам. По суті (точніше за результатом), ініціалізація – це об'єднання визначення об'єкта з одночасним присвоюванням йому конкретного значення. Використання ініціалізації дозволяє змінити формат визначення масиву. Наприклад, можна явно не вказувати кількість елементів одновимірного масиву, а тільки перерахувати їх початкові значення в списку ініціалізації:

double d[ ]={1.0, 2.0, 3.0, 4.0, 5.0};

У даному прикладі довжину масиву компілятор обчислює за кількістю початкових значень, перерахованих у фігурних дужках. Після такого визначення елемент d[0] дорівнює 1.0, d[1] дорівнює 2.0 і т. д. до d[4], який дорівнює 5.0.

Якщо у визначенні масиву явно вказаний його розмір, то кількість початкових значень не може бути більше кількості елементів в масиві. Якщо кількість початкових значень менша, ніж оголошена довжина масиву, то початкові значення одержать тільки перші елементи масиву (з меншими значеннями індексу):

int M[8]={8,4,2);

У даному прикладі визначені значення тільки змінних M[0], M[1] і M[2], що дорівнюють 8, 4 і 2. Елементи M[3], ..., M[7] не ініціалізуються.

Правила ініціалізації багатовимірних масивів відповідають визначенню багатовимірного масиву як одновимірного, елементами якого служать масиви, розмірність яких на одиницю менша, ніж у початкового масиву. Одновимірний масив ініціалізується укладеним у фігурні дужки списком початкових значень. У свою чергу, початкове значення, якщо воно належить масиву, також є поміщений у фігурні дужки список початкових значень. Наприклад, присвоїти початкові значення дійсним елементам двовимірного масиву A, що складається з трьох «рядків» і двох «стовпців», можна таким чином:

double A[3][2]={ {10,20}, {30,40}, {50,60} };

Цей запис еквівалентний послідовності операторів присвоювання:

A[0][0]=10; A[0][1]=20; A[1][0]=30, A[1][1]=40; A[2][0]=50; A[2][1]=60;. Той же результат можна одержати з одним списком ініціалізації:

double A[3][2]={ 10, 20, 30, 40, 50, 60 };

За допомогою ініціалізації можна присвоювати значення не всім елементам багатовимірного масиву. Наприклад, щоб ініціалізувати тільки елементи першого стовпця матриці, її можна описати так:

double Z[4][6]={ {1}, {2}, {3}, {4} };

Наступний опис формує «трикутну» матрицю в цілочисловому масиві з 5 рядків і 4 стовпців:

int x[5][4]={{1}, {2,3}, {4,5,6}, {7,8,9,10}};

В даному прикладі останній п'ятий рядок x[4] залишається незаповненим. Перші три рядки заповнені не до кінця. Схема розміщення елементів масиву зображена в таблиці.

Рядок	0				1				2				3				4			
Стовпець	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
Значення	1	-	-	-	2	3	-	-	4	5	6	-	7	8	9	10	-	-	-	-

11.3 Варіанти завдання 1

Варіант 1

В одновимірному масиві, який складається з *n* дійсних елементів, обчислити:

1) суму від'ємних елементів масиву;

2) добуток елементів масиву, розташованих між максимальним і мінімальним елементами.

Впорядкувати елементи масиву за зростанням.

Варіант 2

В одновимірному масиві, в який складається з *n* цілих елементів, обчислити:

1) суму додатних елементів масиву з парними номерами;

2) суму елементів масиву, розташованих між першим і останнім нулевими елементами.

Змінити масив таким чином, щоб спочатку розташовувалися всі додатні елементи, а потім – всі від'ємні (елементи, що дорівнюють 0, вважати додатними).

Варіант 3

В одновимірному масиві, який складається з *n* дійсних елементів, обчислити:

1) максимальний елемент масиву;

2) суму елементів масиву, розташованих до останнього додатного елемента.

Стиснути масив, видаливши з нього всі елементи, модуль яких знаходиться в інтервалі [a,b]. Елементи, які звільнилися в кінці масиву, заповнити нулями.

Варіант 4

В одновимірному масиві, який складається з *n* дійсних елементів, обчислити:

1) мінімальний елемент масиву;

2) суму елементів масиву, розташованих між першим і останнім додатними елементами.

Змінити масив таким чином, щоб спочатку розташовувались всі елементи, рівні нулю, а потім – всі інші.

Варіант 5

В одновимірному масиві, який складається з *n* дійсних елементів, обчислити:

1) максимальний за модулем елемент масиву;

2) суму елементів масиву, розташованих між першим і другим додатними елементами.

Змінити масив таким чином, щоб елементи, рівні нулю, розташовувались після всіх інших.

Варіант 6

В одновимірному масиві, який складається з *n* дійсних елементів, обчислити:

1) номер мінімального за модулем елемента масиву;

2) суму модулів елементів масиву, розташованих після першого від'ємного елемента.

Стиснути масив, видаливши з нього всі елементи, величина яких знаходиться в інтервалі [a, b]. Елементи, що звільнилися в кінці масиву, заповнити нулями.

Варіант 7

В одновимірному масиві, який складається з *n* дійсних елементів, обчислити:

1) номер максимального за модулем елемента масиву;

2) суму елементів масиву, розташованих після першого додатного елемента.

Змінити масив таким чином, щоб спочатку розташовувались всі елементи, ціла частина яких лежить в інтервалі [*a*, *b*], а потім всі інші.

Варіант 8

В одновимірному масиві, який складається з *n* дійсних елементів, обчислити:

1) кількість елементів масиву, що знаходяться в діапазоні від A до B;

2) суму елементів масиву, розташованих після максимального елемента.

Впорядкувати елементи масиву за зниженням модулів елементів.

Варіант 9

В одновимірному масиві, який складається з *n* дійсних елементів, обчислити:

1) кількість елементів масиву, більших C;

2) добуток елементів масиву, розташованих після максимального за модулем елемента.

Змінити масив таким чином, щоб спочатку розташовувались всі від'ємні елементи, а потім – всі додатні (елементи, що дорівнюють 0, вважати додатними).

Варіант 10

В одновимірному масиві, який складається з *n* дійсних елементів, обчислити:

1) кількість від'ємних елементів масиву;

2) суму модулів елементів масиву, розташованих після мінімального за модулем елемента.

Замінити всі від'ємні елементи масиву їх квадратами і розташувати елементи масиву за зростанням.

Варіант 11

В одновимірному масиві, який складається з *n* дійсних елементів, обчислити:

1) кількість елементів масиву, менших C;

2) суму цілих частин елементів масиву, розташованих після останнього від'ємного елемента.

Змінити масив таким чином, щоб спочатку розташовувались всі елементи, що відрізняються від максимального не більше, ніж на 20%, а потім – всі інші.

Варіант 12

В одновимірному масиві, який складається з n дійсних елементів, обчислити:

1) добуток від'ємних елементів масиву;

2) суму додатних елементів масиву, розташованих до максимального елемента.

Змінити порядок розташування елементів на зворотний.

Варіант 13

В одновимірному масиві, який складається з n дійсних елементів, обчислити:

1) суму додатних елементів масиву;

2) добуток елементів масиву, розташованих між максимальним (за модулем) і мінімальним (за модулем) елементами.

Впорядкувати елементи масиву за спаданням.

Варіант 14

В одновимірному масиві, який складається з n дійсних елементів, обчислити:

1) суму елементів масиву з непарними номерами;

2) суму елементів масиву, розташованих між першим і останнім від'ємними елементами.

Стиснути масив, видаливши з нього всі елементи, модуль яких не перевищує 1. Елементи, які звільнилися в кінці масиву, заповнити нулями.

Варіант 15

В одновимірному масиві, який складається з n цілих елементів, обчислити:

1) номер максимального елемента масиву;

2) добуток елементів масиву, розташованих між першим і другим нулевими елементами.

Змінити масив таким чином, щоб в першій його половині розташовувались елементи, які стоять в непарних позиціях, а в другій половині – елементи, які стоять на парних позиціях.

11.4 Варіанти завдання 2

Варіант 1

Дана цілочислова прямокутна матриця. Визначити:

1) кількість рядків, які не містять жодного нульового елемента;

2) максимальне з чисел, яке зустрічається в заданій матриці більше одного разу.

Варіант 2

Дана цілочислова прямокутна матриця. Визначити:

1) кількість стовпців, які не містять жодного нульового елемента;

2) характеристикою рядка цілочислової матриці назовемо суму його додатних парних елементів. Переміщуючи рядки заданої матриці помістити їх згідно зі зростанням характеристик.

Варіант 3

Дана цілочислова прямокутна матриця. Визначити:

1) кількість стовпців, які містять хоча б один нульовий елемент;

2) номер рядка, в якому знаходиться найдовша серія однакових елементів.

Варіант 4

Дана цілочислова квадратна матриця. Визначити:

1) добуток елементів в тих рядках, які не мають від'ємних елементів;

2) максимум серед сум елементів діагоналей, паралельних головній діагоналі матриці.

Варіант 5

Дана цілочислова квадратна матриця. Визначити:

1) суму елементів в тих стовпцях, які не мають від'ємних елементів;

2) мінімум серед сум модулів елементів діагоналей, паралельних побічній діагоналі матриці.

Варіант 6

Дана цілочислова прямокутна матриця. Визначити:

1) суму елементів в тих рядках, які мають хоча б один від'ємний елемент;

2) номери рядків і стовпців всіх сідловин точок матриці.

Примітка. Матриця A має сідлову точку A_{ij}, якщо A_{ij} є мінімальним елементом в i-му рядку і максимальним у j-му стовпці.

Варіант 7

1) Для заданої матриці розміром 8×8 знайти таке k, що k-й рядок збігається з k-им стовпцем;

2) знайти суму елементів в тих рядках, які мають хоча б один від'ємний елемент.

Варіант 8

1) Характеристикою стовпця цілочислової матриці назвемо суму модулів його від'ємних непарних елементів. Переміщуючи стовпці заданої матриці, розташуйте їх згідно зі зростанням характеристик;

2) знайти суму елементів в тих стовпцях, які мають хоча б один від'ємний елемент.

Варіант 9

1) Елемент матриці називається локальним мінімумом, якщо він строго менший від розташованих біля нього сусідів. Підрахувати кількість локальних мінімумів заданої матриці розміром 10×10;

2) знайти суму модулів елементів, що розташовані вище головної діагоналі.

Варіант 10

1) Коефіцієнт системи лінійних рівнянь заданий у вигляді прямокутної матриці. За допомогою допустимих перетворень привести систему до трикутного вигляду.

2) знайти кількість рядків, середнє арифметичне елементів яких менше заданої величини.

Варіант 11

1) Ущільнити задану матрицю, видаляючи з неї рядки і стовпці, що заповнені нулями.

2) найти номер першого рядка, що містить хоча б один додатний елемент.

Варіант 12

1) Дана цілочислова прямокутна матриця. Визначити номер першого зі стовпців, які мають хоча б один нульовий елемент.

2) характеристикою рядка цілочислової матриці назвемо суму його від'ємних парних елементів. Виводячи рядки заданої матриці, розмістити їх відповідно за зниженням характеристик.

Варіант 13

1) Впорядкувати рядки цілочислової прямокутної матриці за зростанням кількості однакових елементів в кожному рядку;

2) знайти номер першого зі стовпців, що не містить жодного від'ємного елемента.

Варіант 14

Дана цілочислова прямокутна матриця. Визначити:

1) кількість рядків, що містять хоча б один нульовий елемент;

2) номер стовпця, в якому знаходиться найдовша серія однакових елементів.

Варіант 15.

Дана цілочислова квадратна матриця. Визначити:

1) суму елементів в тих рядках, які не мають від'ємних елементів;

2) мінімум серед сум елементів діагоналей, паралельних головній діагоналі матриці.

11.5 Контрольні питання

1. Поняття одновимірного масиву. Навести приклади.
2. Поняття багатовимірного масиву. Навести приклади.
3. Поняття ініціалізації масивів.
4. Ініціалізація одновимірних масивів. Навести приклади.
5. Ініціалізація багатовимірних масивів. Навести приклади.
6. Особливості роботи з масивами в С. Навести приклади.
7. Організація введення одновимірних та двовимірних масивів. Навести приклади.
8. Організація виведення одновимірних та двовимірних масивів. Навести приклади.
9. Особливості організації обробки одновимірних та двовимірних масивів. Навести приклади.
10. Сортування масивів. Навести приклади.
11. Реалізація вкладених циклів у С-програмах. Навести приклади.
12. Запишіть можливі способи ініціалізації масиву ***Array***, якщо він записаний у вигляді ***Array[2][3][4]***.
13. Які види сортування масивів Вам відомі?
14. Як Ви гадаєте, що відбувається при звертанні до елемента масиву за межами допустимого діапазону індексів?

ЛАБОРАТОРНА РОБОТА № 12
ОСОБЛИВОСТІ РОБОТИ З ТЕКСТОВИМИ РЯДКАМИ В МОВІ С

Мета роботи: набуття навичок роботи з текстовими рядками та масивами рядків.

12.1 Порядок виконання роботи

12.1.1 Ознайомтесь з теоретичними відомостями до лабораторної роботи.

12.1.2 Розробіть описовий алгоритм, схему алгоритму та програму обробки рядків з використанням стандартних функцій бібліотеки string.h відповідно до завдання.

12.1.3 Розробіть описовий алгоритм, схему алгоритму та програму обробки рядків без використання стандартних функцій бібліотеки string.h відповідно до завдання.

12.1.4 Для правильної роботи програм розробіть 4-6 тестів.

12.1.5 Оформіть звіт та зробіть висновки.

12.2 Теоретичні відомості

Для подання текстової інформації в мові С використовуються символи (константи), символьні змінні і рядки (рядкові константи), для яких в мові С не введено окремого типу на відміну від деяких інших мов програмування.

Для символьних даних введений базовий тип ***char.*** Опис символьних змінних має такий вигляд:

char список_імен_змінних;

Наприклад:

char a,z;

Введення-виведення символьних даних. Для введення і виведення символьних значень у форматних рядках бібліотечних функцій ***printf()*** і ***scanf()*** використовується специфікація перетворення ***%c.***

Розглянемо таку задачу.

Ввести речення, слова в якому розділені пропусками і в кінці якого стоїть крапка. Видалити пропуски, які повторюються між окремими словами (залишити по одному пропуску), вивести відредаговане речення на екран.

Текст програми:

```
/* Видалення повторюваних пропусків */
#include "stdafx.h"
```

```
void main( )
{
  char z,s; /*  z – поточний символ, що вводиться */
  printf("\n Напишіть речення з крапкою в кінці:\n");
  for (z=s=' ';z!='.';s=z)
        {/*  s – попередній символ */
        scanf("%c",&z);
        if  (z==' '&&  s==' ') continue;
        printf ("%c",z) ;
        } /* Кінець циклу обробки */
}  /* Кінець програми */
```

В програмі є дві символьні змінні: *z* – для читання попереднього символу і *s* – для збереження попереднього. В заголовку циклу змінні *z* і *s* отримують значення *«пропуск»*. Черговий символ вводиться як значення змінної *z*, і пара *z*, *s* аналізується. Якщо хоча б один із символів значень пари відмінний від пропуску, то значення *z* друкується. В заголовку циклу *z* порівнюється із символом *«крапка»* і при розбіжності запам'ятовується як значення *s*. Далі цикл повторюється до появи на вході крапки, причому поява двох пропусків *(z* і *s)* приводить до пропускання оператора друку.

Приклад роботи програми:

Напишіть речення з крапкою в кінці:

YYYYY yyyyy hhhhh ttttt.

YYYYY yyyyy hhhhh ttttt.

Крім ***scanf()*** і ***printf()*** для введення і виведення символів в бібліотеці передбачені спеціальні функції обміну:

getchar() – функція без параметрів. Дозволяє читати з вхідного потоку (звичайно з клавіатури) по одному символу за звертання. Як і при використанні функції ***scanf()***, читання даних, що вводяться, починається після натискання клавіші <***Enter***>. Це дає можливість виправляти інформацію, що вводиться (стираючи останні введені символи за допомогою клавіші <***Backspace***>) до початку їх читання програмою;

putchar(X) – виводить символьне значення ***X*** в стандартний потік (звичайно на екран дисплея).

Внутрішні коди й упорядкованість символів. В мові С прийнята домовленість, що скрізь, де синтаксис дозволяє використовувати цілі числа, можна використовувати і символи, тобто дані типу ***char***, що при цьому подаються числовими значеннями своїх внутрішніх кодів. Така домовленість дозволяє порівняно просто упорядковувати символи, працюючи з ними як з цілочисловими величинами. Наприклад, внутрішні коди десяткових цифр в таблицях кодів ASCII упорядковані за числовим значенням, тому нескладно перебирати символи десяткових цифр у потрібному порядку.

Рядки або рядкові константи. У програмі рядки або рядкові константи подаються послідовністю символів, які взяті в лапки (не в апострофи), наприклад *«будь-які символи»*. Серед символів рядка можуть бути керувальні послідовності, що відповідають кодам незображуваних (спеціальних) символьних констант.

Приклади рядків:

"1234567890"

"\t Склад президії"

"Початок рядка \n і кінець рядка".

Однак у рядків є деякі особливості. Транслятор відводить кожному рядку окреме місце в пам'яті ЕОМ навіть у тих випадках, коли кілька рядків повністю збігаються (стандарт мови С припускає, що в конкретних реалізаціях це правило може не виконуватися). Розміщуючи рядок у пам'яті, транслятор автоматично додає в його кінці символ *'\0'*, тобто нульовий байт. У записі рядка може бути й один символ: *"A"*, однак, на відміну від символьної константи *'A'* (використані апострофи), довжина рядка *"A"* дорівнює двом байтам. На відміну від інших мов (наприклад, від Паскаля) у мові С немає окремого типу для рядків. Прийнято, що рядок – це масив символів, тобто він завжди має тип ***char[ ]***. Таким чином, рядок вважається значенням типу *«масив символів»*. Кількість елементів у такому масиві на 1 більша, ніж у зображенні відповідної рядкової константи, тому що в кінець рядка доданий нульовий байт *'\0'*.

Присвоїти значення масиву символів (тобто рядку) за допомогою звичайного оператора присвоювання не можна. Помістити рядок у масив можна або за допомогою ініціалізації (при визначенні символьного масиву), або за допомогою функцій введення. У функції ***scanf()*** чи ***printf()*** для символьних рядків використовується специфікація перетворення ***%s***.

При визначенні масиву типу ***char*** з одночасною ініціалізацією можна не вказувати межі зміни індексу.

В мові С існують стандартні функції для обробки рядків, які спрощують розробку програм. Ці функції знаходяться в бібліотеці string.h. Розглянемо основні з них.

Функція ***strcat()*** – конкатенує (об'єднує) вихідний рядок ***src*** і ініціалізований підсумковий рядок ***dest***, приєднуючи останній до кінця першого. Повертає ***des.(string.h)***

Синтаксис

char *strcat(char *dest, const char *src);

Функція ***strchr()*** – шукає в рядку ***s*** перше входження символу ***c***, починаючи з початку рядка. У випадку успіху повертає вказівник на знайдений символ, інакше повертає нуль.

Синтаксис

char *strchr(const char *s, int c);

Функція ***strcmp()*** – порівнює два рядки. Повертає негативне значення, якщо ***s1 < s2***; нуль, якщо ***s1 == s2***; позитивне значення, якщо ***s1 > s2***.

Синтаксис

int strcmp(const char *s1, const char *s2);

Функція ***strcmpi*** - це функція аналогічна функції ***strcmp()***, але ігнорує різницю між малими і великими літерами. Для сумісності з іншими компіляторами мови С функція ***strcmpi()*** реалізована у вигляді макросу, що безпосередньо викликає функцію ***strcmpi()***.

Синтаксис

int strcmpi(const char *s1, const char*s2)

Функція ***strcpy()*** – копіює вихідний рядок ***src*** і нульовий символ, який його завершує, у рядок результату ***dst***, перезаписуючи символи підсумкового рядка, розташовані в місці копіювання. Повертає ***dest***. (***string.h***)

Синтаксис

char *strcpy(char *dest,const char *src);

Функція ***strncpy(), _fstrncpy()*** – копіює максимум ***maxlen*** символів із вихідного рядка ***src*** у підсумковий рядок ***dest***, перезаписуючи символи підсумкового рядка. Якщо ***maxlen*** дорівнює розміру в байтах підсумковому рядку (і вихідний рядок має не меншу довжину), то підсумковий рядок не завершується нульовим символом. Якщо ***maxlen*** і довжина вихідного рядка перевищують розмір підсумкового рядка, кінець підсумкового рядка перезаписується, що, можливо, зруйнує інші дані чи код у цьому місці пам'яті. Щоб уникнути подібних помилок, ніколи не встановлюйте ***maxlen*** більше максимального числа символів, що їх може містити підсумковий рядок.

Синтаксис

char *strncpy (char *dest, const char *src, size _t maxlen);

Функція ***strstr()*** – шукає рядок ***(s2)*** в іншому рядку ***(s1)***. Повертає адресу першого символу входження рядка або, якщо підрядок ***s2*** не знайдений в рядку ***s1***, – повертає нуль (***string.h***)

Синтаксис

char *strstr (const char *s1, const char *s2);

Функція ***strtod()*** – перетворює символьне подання числа з плаваючою крапкою, що міститься в рядку, у його двійкове подання типу ***double*** чи ***long double***. У випадку успіху повертає отриманий при перетворенні результат, а у випадку помилки повертає додатне або від'ємне значення ***HUGE_VAL***.

Синтаксис

double strtod (const char *s, char **endptr);
long double strtold (const char *s, char ** endptr);

Функція ***strtol()*** – перетворює символьне подання значення типу ***long*** у його двійкове подання. Значення в рядку може бути подано в десятковому, вісімковому або шістнадцятковому вигляді, що використовує стандартні правила форматування мови С (що діють для ***printf()***, ***scanf()*** й інших аналогічних функцій). Розпізнаються також інші основи системи числення, від 2 до 36. У випадку успіху функція повертає перетворений результат і встановлює ненульовий вказівник ***endptr*** рівним адресі, що слідує за останнім символом, який входить у символьне подання числа. У випадку помилки функція повертає нуль і встановлює ненульовий вказівник ***endptr*** рівним ***s***.

Синтаксис

long strtol (const char *s, char **endptr, int radix);

12.3 Варіанти завдання

Варіант 1

Розробити схему алгоритму, програму інвертування заданого рядка і тести, що підтверджують правильність роботи програми.

Варіант 2

Розробити схему алгоритму, програму визначення кількості слів у тексті, що вводиться з клавіатури, і тести, що підтверджують правильність роботи програми.

Варіант 3

Розробити схему алгоритму, програму копіювання з заданого тексту даної частини в рядок і тести, що підтверджують правильність роботи програми.

Варіант 4

Розробити схему алгоритму, програму злиття заданих рядків з тексту, що вводиться з клавіатури, і тести, що підтверджують правильність роботи програми.

Варіант 5

Розробити схему алгоритму, програму визначення рядка максимальної довжини в тексті, що вводиться з клавіатури, і тести, що підтверджують правильність роботи програми.

Варіант 6

Розробити схему алгоритму, програму пошуку в рядку STR кількості входжень підрядків STR1 і тести, що підтверджують правильність роботи програми.

Варіант 7

Розробити схему алгоритму, програму поділу рядка на підрядки довжиною в 5 символів кожен, не враховуючи пропуск, і тести, що підтверджують правильність роботи програми.

Варіант 8

Розробити схему алгоритму, програму знаходження кількості однакових символів у слові, що вводиться з клавіатури, і тести, що підтверджують правильність роботи програми.

Варіант 9

Розробити схему алгоритму, програму перевірки, чи є заданий рядок паліндромом (таким, що читається в обох напрямках однаково), і тести, що підтверджують правильність роботи програми.

Варіант 10

Розробити схему алгоритму, програму визначення кількості цифр в рядку, введеному з клавіатури, і тести, що підтверджують правильність роботи програми.

12.4 Контрольні запитання

1. Як відбувається введення рядків, які функції та бібліотеки при цьому використовуються?
2. За допомогою яких функцій відбувається виведення рядків?
3. Які оператори використовуються для визначення довжини рядка?
4. Як відбувається пошук в рядку входження підрядка?
5. Основні стандартні функції для роботи з рядками.
6. Наведіть приклад використання функції порівняння рядків.
7. Яка функція призначена для копіювання рядка?
8. За допомогою якої функції відбувається інвертування рядка?

ЛАБОРАТОРНА РОБОТА № 13
РОЗРОБКА ПРОГРАМ З ІНТЕГРОВАНИМИ ТИПАМИ ДАНИХ

Мета роботи: набуття практичних навичок розробки програм з інтегрованими типами даних – структурами.

13.1 Порядок виконання роботи

13.1.1 Ознайомтесь з теоретичними відомостями до лабораторної роботи.
13.1.2 Розробіть описовий алгоритм, схему алгоритму і програму для виконання індивідуального завдання.
13.1.3 Для правильної роботи програм розробіть 4-6 тестів.
13.1.4 Оформіть звіт та зробіть висновки.

13.2 Теоретичні відомості

Структура – це іменована сукупність даних різних типів.
Існують різні способи введення структурного типу
Перший спосіб:

struct ім'я_структурного_типу { визначення елементів };

struct – службове слово (ключове) або специфікатор структурного типу;
ім'я_структурного_типу – відіграє ту ж роль, що і специфікатори типів ***int, double, float*** (вибирається довільно програмістом);
визначення елементів – сукупність одного чи більше описів об'єктів структури.

Другий спосіб: ввести структурний тип можна за допомогою ***typedef***, що дозволяє ввести власне позначення для будь-якого визначення типу.
Формат визначення структурного типу:
typedef struct {визначення елементів} позначення структурного типу;

Третій спосіб: ввести структурний тип можна за допомогою директиви ***#define***.
Формат визначення структурного типу:
#define позначення структурного типу ***struct*** \
{ визначення елементів структури}

Оголошення структур в програмі
Перший спосіб.

Формат оголошення конкретних структур має вигляд:

1) ***struct* ім'я_структурного_типу список_імен_структур;**

2) ***struct* ім'я_структурного_типу {визначення елементів} список структур;**

Другий спосіб.

Якщо структурний тип був визначений через ***typedef***, то структури можуть вводиться за допомогою імені структури (***avto***).

Третій спосіб.

Після визначення структури в ***#define*** визначити структури можна за допомогою препроцесорного ідентифікатора.

Четвертий спосіб.

Можна визначити структури без імені структурного типу, тобто так званий «безіменний» структурний тип. Такий тип, зазвичай, використовується в програмі для однократного визначення структур:

***struct* {визначення елементів} список структур;**

Виділення пам'яті для структур

При кожному визначенні структури їй виділяється пам'ять у такій кількості, щоб могли розміститися дані всіх елементів.

Реальний розмір пам'яті в байтах, що виділяється для структури, можна визначити за допомогою операції

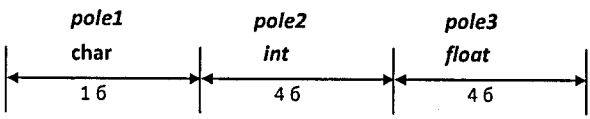
***sizeof* (ім'я структури)**

***sizeof* (ім'я структурного типу)**

Елементи структури в пам'яті розміщуються підряд:

Наприклад, ***struct str***

```
{char pole1;
 int  pole2 ;
 float pole3
};
```



Ініціалізація і присвоювання структур:

Ініціалізація структур схожа на ініціалізацію масивів.

Перший спосіб.

Безпосередня ініціалізація при визначенні структури. Наприклад,

struct student stud1={"Іванов", "ФАКСУ", 1}

struct st2={'X', 12, 97.34};

Другий спосіб.

Стандарт мови С дозволяє присвоювання структур (на відміну від масивів). Наприклад,

struct student {char FIO[10];char fak[6]; int kurs;} stud1,stud2;

syud1=stud2; – якщо визначена ***stud1***.

Доступ до елементів структури

Доступ до елементів структур забезпечується за допомогою уточнених імен:

Формат організації доступу:

ім'я структури . ім'я елемента

Наприклад,

stud1.FIO – змінна-масив типу ***char[10]***;

stud1.fak – змінна-масив типу ***char[6]***;

stud1.kurs – змінна типу ***int***;

Операція (.) називається операцією доступу до елемента структури і має найвищий ранг в порівнянні з дужками () та операцією → (операція доступу до елементів структури через її вказівник).

13.3 Завдання

Варіант 1

1. Описати структуру з ім'ям STUDENT, яка містить такі поля:
   - NAME – прізвище та ініціали;
   - GROUP – номер групи;
   - SES – успішність (масив з 5 елементів).
2. Написати програму, яка виконує такі дії:
   - введення з клавіатури даних в масив STUD1, який складається з 10 структур типу STUDENT; записи повинні бути впорядковані за зростанням поля GROUP;
   - виведення на дисплей прізвищ та номерів груп для всіх студентів, які внесені в масив, якщо середній бал студента більше 4.0; якщо таких студентів немає, вивести відповідне повідомлення.

Варіант 2

1. Описати структуру з ім'ям STUDENT, яка містить такі поля:
   - NAME – прізвище та ініціали;
   - GROUP – номер групи;
   - SES – успішність (масив з 5 елементів).
2. Написати програму, яка виконує такі дії:
   - введення з клавіатури даних в масив STUD1, який складається з 10 структур типу STUDENT; записи повинні бути впорядковані за зростанням середнього бала;

– виведення на дисплей прізвищ та номерів груп для всіх студентів, які мають оцінки 4 і 5; якщо таких студентів немає, вивести відповідне повідомлення.

Варіант 3

1. Описати структуру з ім'ям STUDENT, яка містить такі поля:
   – NAME – прізвище та ініціали;
   – GROUP – номер групи;
   – SES – успішність (масив з 5 елементів).
2. Написати програму, яка виконує такі дії:
   – введення з клавіатури даних в масив STUD1, який складається з 10 структур типу STUDENT; записи повинні бути впорядковані за алфавітом;
   – виведення на дисплей прізвищ та номерів груп для всіх студентів, які мають хоча б одну оцінку 2; якщо таких студентів немає, вивести відповідне повідомлення.

Варіант 4

1. Описати структуру з ім'ям AEROFLOT, яка містить такі поля:
   – NAZM – назва пункту призначення рейсу;
   – NUMR – номер рейсу;
   – TIP – тип літака.
2. Написати програму, яка виконує такі дії:
   – введення з клавіатури даних в масив AIRPORT, який складається з 7 елементів типу AEROFLOT; записи повинні бути впорядковані за зростанням номера рейсу;
   – виведення на екран номерів рейсів та типів літаків, які вилітають в пункт призначення, назва якого збіглася з назвою, введеною з клавіатури; якщо таких рейсів немає, вивести на дисплей відповідне повідомлення.

Варіант 5

1. Описати структуру з ім'ям AEROFLOT, яка містить такі поля:
   – NAZM – назва пункту призначення рейсу;
   – NUMR – номер рейсу;
   – TIP – тип літака.
2. Написати програму, яка виконує такі дії:
   – введення з клавіатури даних в масив AIRPORT, який складається з 7 елементів типу AEROFLOT; записи повинні бути розміщенні в алфавітному порядку за назвою пунктів призначення;
   – виведення на екран пунктів призначення та номерів рейсів, які обслуговуються літаком, тип якого введений з клавіатури; якщо таких рейсів немає, вивести на дисплей відповідне повідомлення.

Варіант 6

1. Описати структуру з ім'ям WORKER, яка містить такі поля:
   - NAME – прізвище та ініціали;
   - POS – назва посади;
   - YEAR – рік вступу на роботу.
2. Написати програму, яка виконує такі дії:
   - введення з клавіатури даних в масив TABL, який складається з десяти структур типу WORKER, записи повинні бути розміщені за алфавітом;
   - виведення на екран прізвищ працівників, стаж роботи яких в організації перевищує значення, введене з клавіатури; якщо таких робітників немає, вивести відповідне повідомлення.

Варіант 7

1. Описати структуру з ім'ям TRAIN, яка містить такі поля:
   - NAZM – назва пункту призначення рейсу;
   - NUMR – номер потягу;
   - TIME – час відправлення.
2. Написати програму, яка виконує такі дії:
   - введення з клавіатури даних в масив RASP, який складається з 8 елементів типу TRAIN; записи повинні бути розміщенні в алфавітному порядку за назвою пунктів призначення;
   - виведення на екран інформації про потяги, які були відправлені після введеного з клавіатури часу; якщо таких потягів немає, вивести на дисплей відповідне повідомлення.

Варіант 8

1. Описати структуру з ім'ям TRAIN, яка містить такі поля:
   - NAZM – назва пункту призначення рейсу;
   - NUMR – номер потягу;
   - TIME – час відправлення.
2. Написати програму, яка виконує такі дії:
   - введення з клавіатури даних в масив RASP, який складається з 6 елементів типу TRAIN; записи повинні бути впорядковані за часом відправлення потягу;
   - виведення на екран інформації про потяги, які направляються в пункт, назва якого введена з клавіатури; якщо таких потягів немає, вивести на дисплей відповідне повідомлення.

Варіант 9

1. Описати структуру з ім'ям TRAIN, яка містить такі поля:
   - NAZM – назва пункту призначення рейсу;
   - NUMR – номер потягу;
   - TIME – час відправлення.

2. Написати програму, яка виконує такі дії:

– введення з клавіатури даних в масив RASP, який складається з 8 елементів типу TRAIN; записи повинні бути впорядковані за номерами потягів;

– виведення на екран інформації про потяг, номер якого введений з клавіатури; якщо таких потягів немає, вивести на дисплей відповідне повідомлення.

Варіант 10

1. Описати структуру з ім'ям MARSH, яка містить такі поля:
   - BEGST – назва початкового пункту маршруту;
   - TERM – назва кінцевого пункту маршруту;
   - NUMER – номер маршруту.
2. Написати програму, яка виконує такі дії:

– введення з клавіатури даних в масив TRAFIC, який складається з 8 елементів типу MARSH; записи повинні бути впорядковані за номерами маршрутів;

– виведення на екран інформації про маршрут, номер якого введений з клавіатури; якщо такого маршруту немає, вивести на дисплей відповідне повідомлення.

Варіант 11

1. Описати структуру з ім'ям MARSH, яка містить такі поля:
   - BEGST – назва початкового пункту маршруту;
   - TERM – назва кінцевого пункту маршруту;
   - NUMER – номер маршруту.
2. Написати програму, яка виконує такі дії:

– введення з клавіатури даних в масив TRAFIC, який складається з 8 елементів типу MARSH; записи повинні бути впорядковані за номерами маршрутів;

– виведення на екран інформації про маршрути, які починаються або закінчуються в пункті, назва якого введена з клавіатури; якщо таких маршрутів немає, вивести на дисплей відповідне повідомлення.

Варіант 12

1. Описати структуру з ім'ям NOTE , яка містить такі поля:
   - NAME – прізвище та ім'я;
   - TELE – номер телефону;
   - BDAY – дата народження (масив з трьох чисел: день, місяць і рік).
2. Написати програму, яка виконує такі дії:

– введення з клавіатури даних в масив BLOCKNOTE, який складається з 8 елементів типу NOTE; записи повинні бути впорядковані за алфавітом;

– виведення на екран інформації про людину, номер телефону якого введений з клавіатури; якщо такого немає, вивести на дисплей відповідне повідомлення.

Варіант 13

1. Описати структуру з ім'ям NOTE , яка містить такі поля:
   - NAME – прізвище та ім'я;
   - TELE – номер телефону;
   - BDAY – дата народження (масив з трьох чисел: день, місяць і рік).
2. Написати програму, яка виконує такі дії:

– введення з клавіатури даних в масив BLOCKNOTE, який складається з 8 елементів типу NOTE; записи повинні бути впорядковані за алфавітом;

– виведення на екран інформації про людей, дні народження яких припадають на місяць, значення якого введено з клавіатури; якщо таких немає, вивести на дисплей відповідне повідомлення.

Варіант 14

1. Описати структуру з ім'ям NOTE , яка містить такі поля:
   - NAME – прізвище та ім'я;
   - TELE – номер телефону;
   - BDAY – дата народження (масив з трьох чисел).
2. Написати програму, яка виконує такі дії:

– введення з клавіатури даних в масив BLOCKNOTE, який складається з 8 елементів типу NOTE; записи повинні бути впорядковані за трьома першими цифрами номера телефону;

– виведення на екран інформації про людину, прізвище якої введено з клавіатури; якщо такої немає, вивести на дисплей відповідне повідомлення.

Варіант 15

1. Описати структуру з ім'ям ZNAK, яка містить такі поля:
   - NAME – прізвище та ім'я;
   - ZNAK – знак зодіаку;
   - BDAY – дата народження (масив з трьох чисел).
2. Написати програму, яка виконує такі дії:

– введення з клавіатури даних в масив BOOK, який складається з 8 елементів типу ZNAK; записи повинні бути впорядковані за датами народження;

– виведення на екран інформації про людину, прізвище якої введено з клавіатури; якщо такої немає, вивести на дисплей відповідне повідомлення.

13.4 Контрольні питання

1. Що таке структура даних?
2. Які Ви знаєте основні способи визначення структурного типу?
3. Як можна давати імена структурам? Наведіть приклади (не менше 3-х варіантів).
4. Способи ініціалізації структур даних.
5. Як розподіляється пам'ять під структури?
6. Що таке вкладені структури? Наведіть приклади.
7. Як визначаються масиви структур?
8. Як розподіляється пам'ять під масиви структур?
9. Які є способи ініціалізації масивів структур? Наведіть приклади.
10. Пояснити метод доступу до масивів структур.
11. Принципи роботи з елементами структури даних.
12. Як можна організувати введення елемента структури даних?
13. Як можна організувати виведення елемента структури даних? Наведіть приклади.
14. Як можна звернутися до елемента структури даних? Наведіть приклади.

ЛІТЕРАТУРА

Основна література

1. Давыдов В. Г. Программирование и основы алгоритмизации : учеб. пособие / Давыдов В. Г. – М. : Высш. шк., 2003. – 447 с.
2. Керниган Б. Язык программирования С / Б. Керниган, Д. Ритчи ; пер. с англ. – 2-е издание. – М. : Издательский дом «Вильямс», 2009. – 304 с.
3. Мюррей У. Visual C++ / У. Мюррей, К. Паппас. Руководство для профессионалов: пер. с англ. – Спб. : BHV-Санкт-Петербург, 1996. – 912 с.
4. Павловская Т. А. С/С++: Программирование на языке высокого уровня : учебник / Павловская Т. А. – СПб. : Питер, 2001. – 464 с.
5. Подбельский В. В. Программирование на языке Си : учебное пособие / В. В. Подбельский, С. С. Фомин. – [2-е изд., доп]. – М. : Финансы и статистика, 2002. – 600 с.

Додаткова

1. Златопольский Д. М. Сборник задач по программированию / Златопольский Д. М. – СПб. : БХВ-Петербург, 2007. – 240 с.
3. Макконнелл Дж. Основы современных алгоритмов / Дж. Макконнелл. – [2-е дополненное издание]. – Москва : Техносфера, 2004. – 368 с.
4. Седжвик Р. Фундаментальные алгоритмы на С++. Анализ/Структуры данных/Сортировка/Поиск / Р. Седжвик : пер. с англ. – К. : Издательство «ДиаСофт», 2001. – 688 с.

Інформаційні ресурси

1. Бібліотека MSDN [сайт]. Режим доступу : http://msdn.microsoft.com/ru-ru/library/default.aspx (дата звертання 15.09.2012). – Назва з екрана.
2. Программирование на языке С: [сайт]. Режим доступу : http://www.cyberguru.ru/home/root/programming/cpp/cpp-programming-guide.html (дата звертання 15.09.2012). — Назва з екрана.
3. Электронный учебно-методический комплекс по информатике. [сайт]. Режим доступу : http://informatics.ssga.ru/home (дата звертання 15.09.2012). – Назва з екрана.

СЛОВНИК НАЙЧАСТІШЕ ВЖИВАНИХ ТЕРМІНІВ

Абсолютний шлях – absolute path
Алгоритм – algorithm
Базові структури алгоритму – algorithm basic structures
Блок-схема – flow chart
Вираз – expression
Вихідні дані – input data
Відлагодження – debugging
Відносний шлях – relative path
Вісімкове число – octal number
Вказівник – pointer
Графічний інтерфейс – graphical interface
Двійкове число – binary number
Дерево каталогів – tree panel
Десяткове число – decimal number
Директорія – directory
Диск – dick
Довге ім'я – long name
Доступ – access
Запит – query, request, inquiry
Змінна – variable
Значок – shortcut
Ім'я файлу – file name
Ініціалізація – initialization
Кілобайт – kilobyte
Команда – command
Командна оболонка – command shell
Командний рядок – command line
Комп'ютеризована система управління – computer control system
Компіляція – compilation
Компонувальник – linker
Консольна програма – console application
Константа – constant
Контекстна довідкова система – context help
Кореневий каталог – root directory
Користувач – user
Коротке ім'я – short name
Курсор – cursor
Лінійний алгоритм – linear algorithm
Масив – array
Маска файлу – file mask
Математичні основи ЕОМ – mathematical basics of computers

Мова програмування – programming language
Накопичувач – drive
Необов'язковий параметр – optional parameter
Об'єкт – object
Оголошення – declaration
Оператор – statement
Оператор вибору – conditional statement
Операція – operation
Панель інформації – info panel
Папка – folder
Перетворення типів – type cast
Підкаталог – subdirectory
Повне ім'я файлу – file full name
Повнота – completeness
Поточний каталог – current directory
Програма – program
Пріоритет – priority
Програмування – programming
Робочий стіл – desktop
Розгалужена структура – conditional structure
Розширення – extension
Рядок – string
Середовище програмування – programming environment
Спадне меню – drop-down menu
Структура – structure
Структурний підхід – structural approach
Текстовий файл – text file
Теорія алгоритмів – theory of algorithms
Тип даних – data type
Транслятор – translator
Файл – file
Файловий менеджер – file manager
Файлова система – file system
Флеш-карта – flash card
Функція – function
Функціональна клавіша – functional key
Циклічний оператор – loop statement
Цикл – cycle
Шаблон – template
Шістнадцяткове число – hexadecimal number
Шлях до файлу – file path

Навчальне видання

Москвіна Світлана Михайлівна
Грищук Тетяна Вікторівна

КОМП'ЮТЕРНІ ТЕХНОЛОГІЇ ТА ПРОГРАМУВАННЯ ЧАСТИНА 1 АЛГОРИТМІЧНІ ОСНОВИ ПРОГРАМУВАННЯ

Лабораторний практикум

Редактор В. Дружиніна

Оригінал-макет підготовлено Т. Грищук

Підписано до друку 22.10.2014 р.
Формат 29,7×¼. Папір офсетний.
Гарнітура Times New Roman.
Друк різографічний Ум. друк. арк. 7,3.
Наклад 75 прим. Зам. № 2014-084.

Вінницький національний технічний університет,
навчально-методичний відділ ВНТУ.
21021, м. Вінниця, Хмельницьке шосе, 95,
ВНТУ, к. 2201.
Тел. (0432) 59-87-36.
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.

Віддруковано у Вінницькому національному технічному університеті
в комп'ютерному інформаційно-видавничому центрі.
21021, м. Вінниця, Хмельницьке шосе, 95,
ВНТУ, ГНК, к. 114.
Тел. (0432) 59-85-32.
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.

Москвіна Світлана Михайлівна,
кандидат технічних наук, професор кафедри комп'ютерних систем управління Вінницького національного технічного університету. Автор понад 120 наукових робіт з напрямку математичне моделювання та обчислювальні методи.

Грищук Тетяна Вікторівна,
кандидат технічних наук, доцент кафедри комп'ютерних систем управління Вінницького національного технічного університету. Головний науковий напрямок: "Розробка моделей та методів проектування голосових інтерфейсів". Автор понад 25 публікацій.