

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**  
**ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ**

**Методичні рекомендації  
до виконання лабораторних робіт  
з навчальної дисципліни**  
**"ОСНОВИ ПРОГРАМУВАННЯ  
ТА АЛГОРИТМІЧНІ МОВИ"**  
**для студентів напряму підготовки**  
**"Комп'ютерні науки"**  
**всіх форм навчання**

**Частина 1**

**Харків. Вид. ХНЕУ, 2008**

Затверджено на засіданні кафедри інформаційних систем.  
Протокол №6 від 24.01.2008 р.

М54                   Методичні рекомендації до виконання лабораторних робіт з навчальної дисципліни "Основи програмування та алгоритмічні мови" для студентів напряму підготовки "Комп'ютерні науки" всіх форм навчання. Ч. 1 / Укл. Ю. Е. Парфьонов, В. М. Федорченко, М. Ю. Лосєв, О. В. Щербаков. – Харків: Вид. ХНЕУ, 2008. – 128 с. (Укр. мов.)

Подано методичні рекомендації до виконання лабораторних робіт з першої частини даної навчальної дисципліни з метою розвитку алгоритмічного мислення та навчання на практиці написанню і відладці програм на мові C/C++, використовуючи середовище візуальної розробки програм Microsoft Visual Studio. NET.

Рекомендовано для студентів напряму підготовки "Комп'ютерні науки".

## Загальні положення

Методичні рекомендації призначені для виконання лабораторних робіт з першої частини навчальної дисципліни "Основи програмування та алгоритмічні мови".

Перед виконанням кожної роботи необхідно вивчити відповідний лекційний матеріал і звернути особливу увагу на загальні положення, передуючі опису лабораторних завдань.

Наведені приклади програм слід розглядати лише як один із можливих варіантів розв'язання задачі.

Методичні рекомендації містять опис 6 лабораторних робіт. Кожен розділ, який відповідає окремій лабораторній роботі, складається з таких підрозділів:

- мета роботи й вимоги до теоретичної та практичної підготовки, що необхідна для виконання лабораторної роботи;

- рекомендації щодо підготовки до виконання лабораторної роботи, основні теоретичні відомості, необхідні для її виконання;

- суть роботи – загальна постановка завдання до лабораторної роботи (необов'язково);

- індивідуальні варіанти завдань;

- контрольні запитання.

При проведенні всіх лабораторних робіт використовується єдина конфігурація програмно-апаратних засобів: персональна ЕОМ типу IBM-PC з процесором не нижче Pentium III, операційна система Windows XP або Windows Vista, середовище візуальної розробки програм Microsoft Visual Studio .NET.

Під час проведення лабораторних робіт студент повинен продемонструвати:

- творчий, індивідуальний підхід до розробки проектів (програмної коди);

- грамотне використання існуючого програмного забезпечення;

- навики програмування на мовах високого рівня C/C++.

Студент повинен уміти перетворити свою програму в програмний продукт, використовувати якісний аналіз програми, виконувати оцінку отриманих результатів. Велике значення має зручний інтерфейс з користувачем і поясненнями до програми.

Варіант завдання до лабораторної роботи вибирається відповідно до номера студента в журналі групи.

**Типовий порядок виконання роботи й методичні рекомендації до її виконання:**

уважно ознайомитися з методичними рекомендаціями до конкретної лабораторної роботи (теоретичними відомостями, прикладами, формулюванням завдань);

створити заготівку консольного застосування, скористатися для цього майстром створення додатків Microsoft Visual Studio;

заповнити отриману заготівку консольного застосування конкретним змістом відповідно до запропонованого завдання (див. приклади);

усунути всі помилки, що виникли на етапі компіляції початкового тексту програми;

виконання програми здійснити в покроковому режимі;

вивести у вікно попереднього перегляду значення всіх проміжних змінних;

знайти свою папку проекту й ознайомитися з її вмістом (за допомогою Блокнота відкрити файл ReadMe.txt і перекласти текст, який у ньому записаний);

підсумковий запуск додатка виконати за допомогою виконуваного модуля;

відповісти на контрольні запитання;

за допомогою динамічної довідки з'ясувати призначення основних службових слів у програмі;

виконати експериментальну частину роботи згідно з отриманим завданням;

оформити звіт і здати викладачеві.

**Звіт з будь-якої лабораторної роботи повинен містити:**

**1. Титульний лист:**

назва дисципліни;

тема лабораторної роботи;

дата виконання роботи;

П.І.Б. студента, курс, номер групи;

П.І.Б., посада викладача.

**2. Лист змісту** (нумерований перелік назв пунктів роботи із зазначенням номерів сторінок).

### 3. Опис виконаних завдань:

Умова завдання (завдання).

Опис архітектури програми – специфікація програмних вимог (склад, структура модулів, зв'язки між ними, алгоритми):

формулювання завдання;

специфікація даних;

математична модель обробки даних;

програмний інтерфейс;

план тестування.

Початковий код програми з коментарями для бібліотек (призначення кожної бібліотеки), що підключаються, оголошень, інструкцій, що управляють, і функцій (призначення кожної функції та інструкції, опис параметрів і зворотного значення).

Приклади результатів роботи програми на тестових початкових даних.

### 4. Висновки за роботою з урахуванням усіх виконаних завдань:

аналіз отриманих результатів за кожним пунктом завдання;

аналіз результатів тестування програм;

ступінь відповідності розроблених програм постановці завдання;

інша інформація.

Приклад звіту наведений у додатку В. Викладач може вносити корективи до оформлення звіту.

## **Вимоги до оформлення звіту**

Поля сторінки: ліве – 2,5 см, праве – 1,5 см, верхнє й нижнє – 2 см; шрифт Times New Roman (висота – 14 пт), міжрядковий інтервал – множник 1,1.

Номери сторінок повинні знаходитися у правому верхньому куті, титульний лист не нумерується.

Листи звіту мають бути з'єднані скріпкою або іншим загальноприйнятим способом.

# Лабораторна робота 1

## Частина 1 (4 години)

### Середовище візуальної розробки програм Microsoft Visual Studio .NET

**Мета лабораторної роботи** – придбання первинних практичних навиків роботи в середовищі візуальної розробки програм Microsoft Visual Studio .NET.

Перед виконанням лабораторної роботи студент повинен знати:  
призначення основних елементів вікна Microsoft Visual Studio .NET;  
основні органи управління вікном Microsoft Visual Studio .NET;  
структуру й призначення основних елементів формату С-програми;  
призначення директив препроцесора, операторів програми і коментарів;

призначення й способи використання динамічної довідки.

Після виконання лабораторної роботи студент повинен уміти:  
створювати заготовку проекту;  
здійснювати первинне введення й редагування тексту програми;  
знаходити й усувати помилки, які виникли на етапі трансляції програми.

#### Розробка простого консольного додатка

Запустіть Microsoft Visual Studio 2005, для чого виберіть меню

Пуск > Програми > Microsoft Visual Studio 2005.

Після цього на екрані з'явиться головне вікно Інтегрованого Середовища Розробки Microsoft Visual Studio 2005 (рис.1.1).

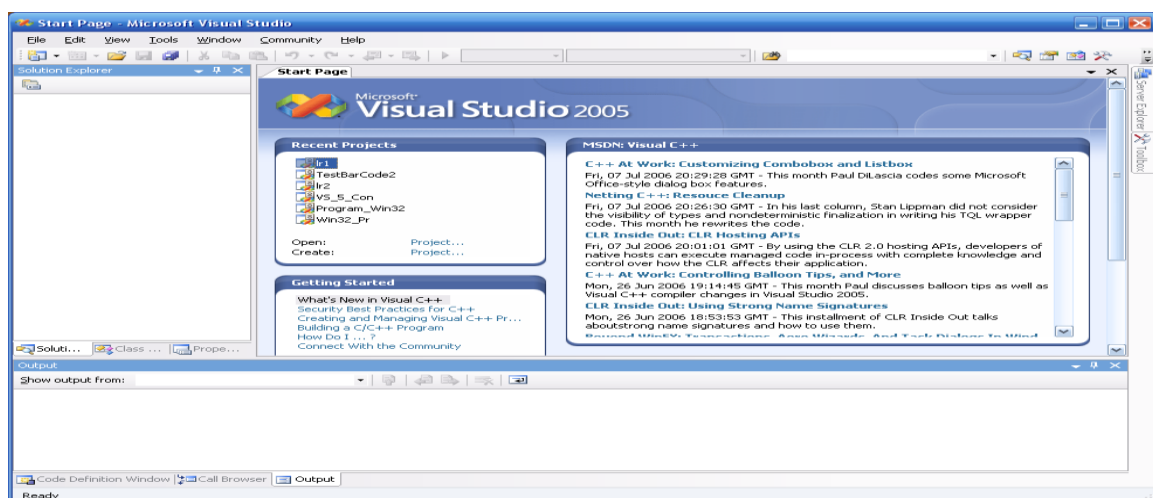


Рис. 1.1. Головне вікно Інтегрованого Середовища Розробки

Для створення консольного додатка оберіть меню File, в ньому – команду New, а потім – команду Project (рис. 1.2).

З'явиться діалогове вікно New Project (рис.1.3). У цьому діалоговому вікні вкажіть у розділі Project Types (Тип проекту) значення Win 32, у розділі Templates (Шаблон) – значення Win 32 Console Application (Консольний додаток), вкажіть у полі Name (Ім'я) ім'я проекту, наприклад, first (перший) і виберіть теку, в яку буде поміщений новий проект. Для цього в полі Location (Розміщення) виберіть диск і теку. Натисніть кнопку OK і на екрані з'явиться діалогове вікно Win 32 Application Wizard (рис.1. 4).

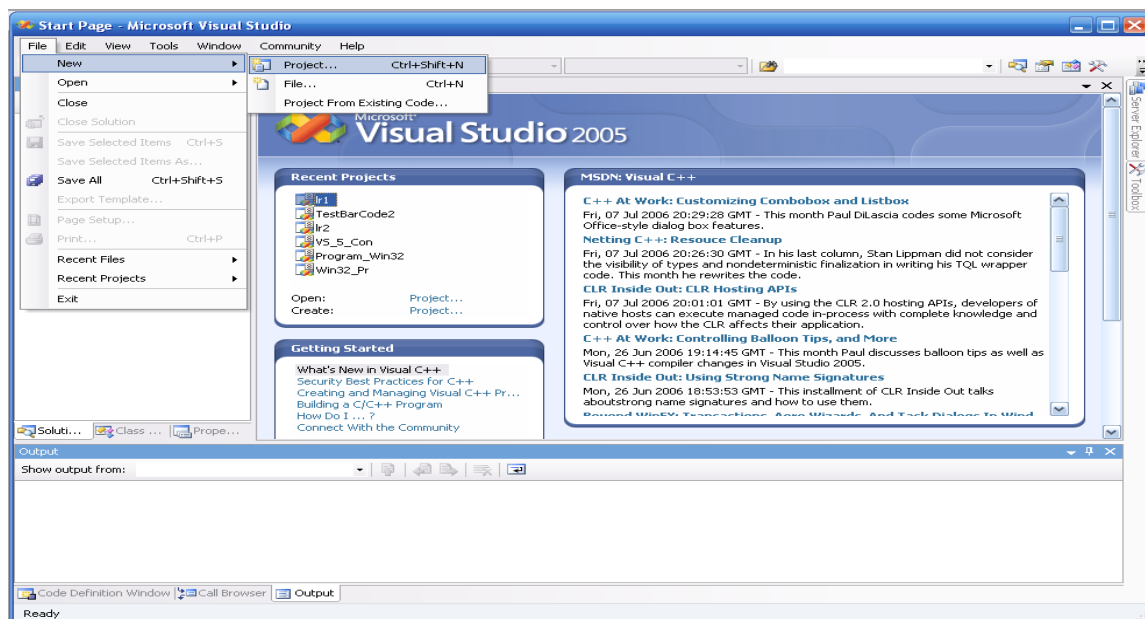


Рис. 1.2. Команда Project

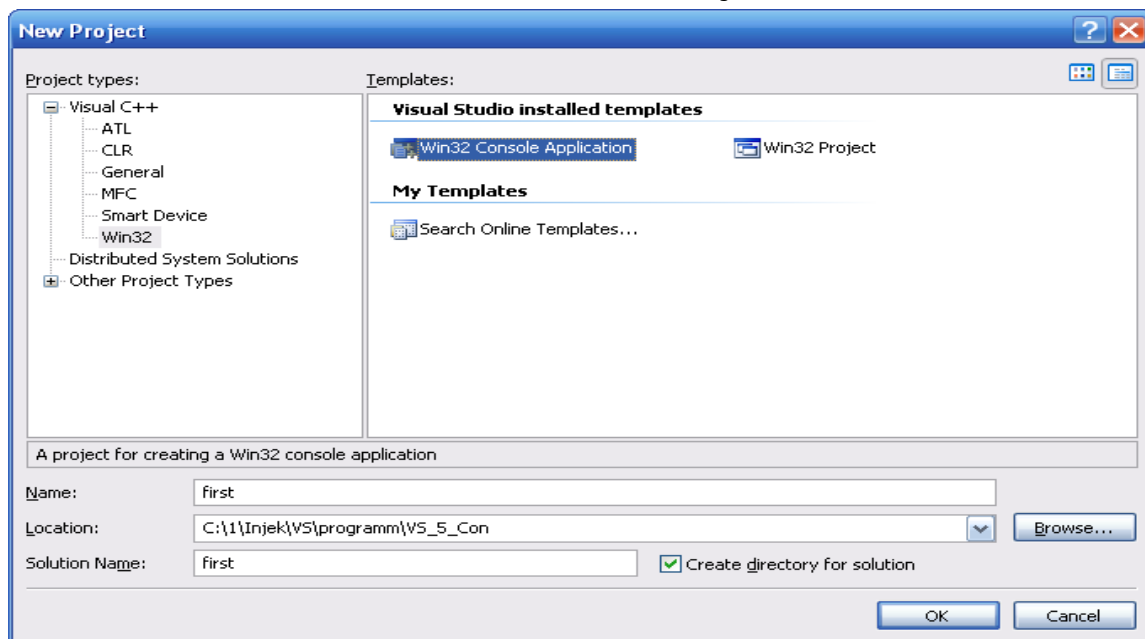


Рис. 1.3. Діалогове вікно New Project

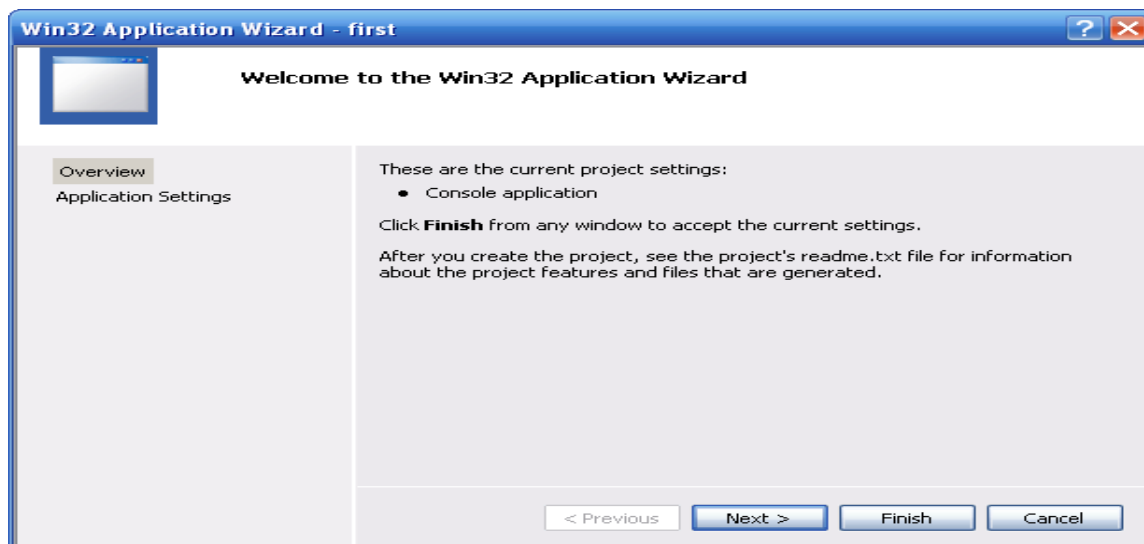


Рис. 1.4. Діалогове вікно Win 32 Application Wizard

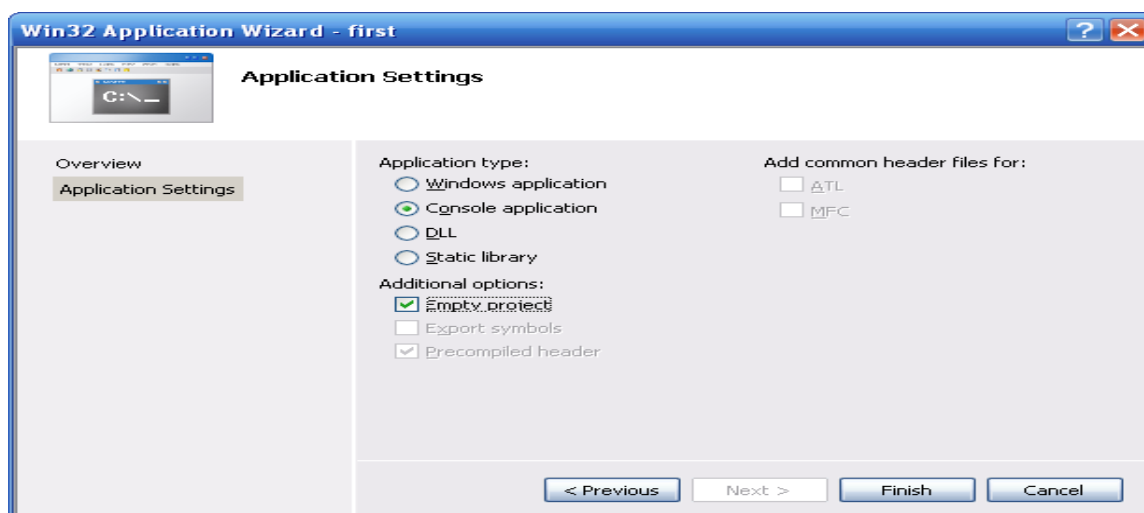


Рис. 1.5. Вікно – Application Settings (Установки додатка)

У діалоговому вікні Win 32 Application Wizard натисніть кнопку Next і на екрані з'явиться наступна сторінка цього вікна – Application Settings (Установки додатка) (рис. 5). У групі перемикачів Application type (Тип додатка) виберіть Console Application (Консольний додаток), а в групі Additional options (Додаткові опції) виберіть Empty project (Порожній проект) (рис. 1.5). Після цього натисніть кнопку Finish.

На екрані знову з'явиться головне вікно Інтегрованого Середовища Розробки Microsoft Visual Studio 2005, в лівій частині якого, у вікні провідника рішення (Solution Explorer) з'являться папки нашого проекту (рис. 1.6)



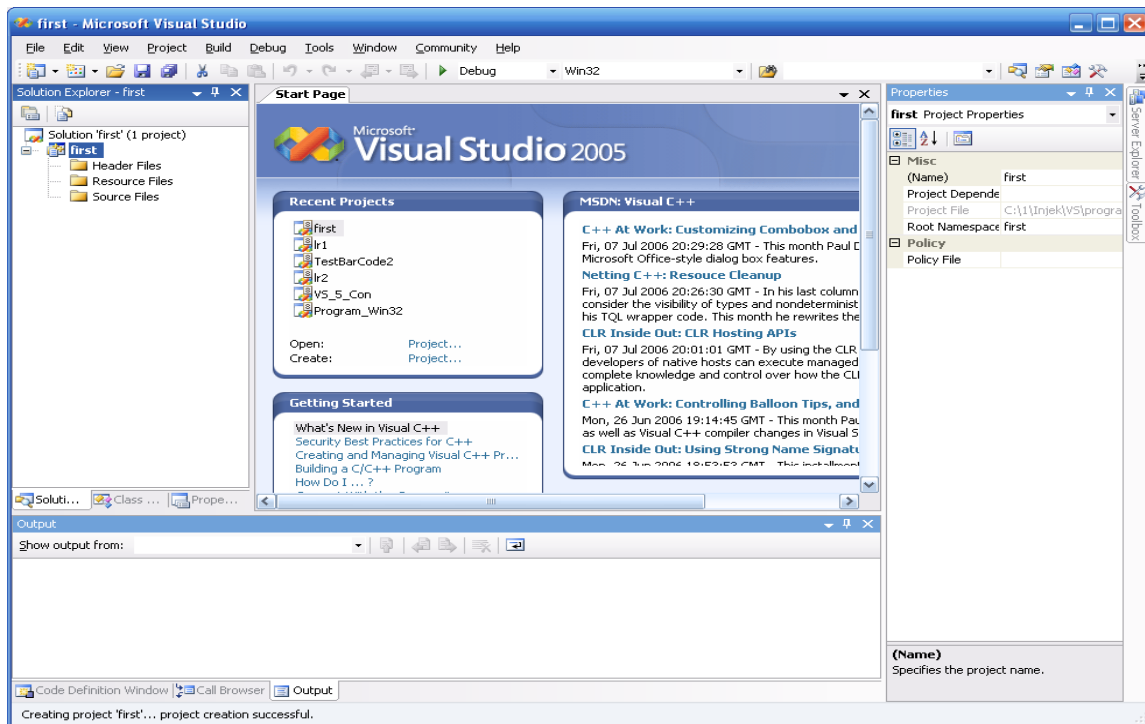


Рис. 1.6. Вікно провідника рішення (Solution Explorer)

Клацніть правою кнопкою миші на папці Source Files у вікні провідника рішення (Solution Explorer). У контекстному меню вікна, що з'явилося, виберіть команду Add і далі в додатковому меню, що з'явилося, виберіть команду Add New Item. У результаті на екрані з'явиться діалогове вікно Add New Item (рис. 1.7).

У цьому вікні виберіть у розділі Categories пункт Code, в розділі Templates – пункт C++File(.cpp) і вкажіть у полі Name ім'я файлу, наприклад, prog1 (рис. 1.7). Буде створений порожній файл prog1.cpp і відкриється редактор коду програми для введення тексту програми (рис. 1.8).

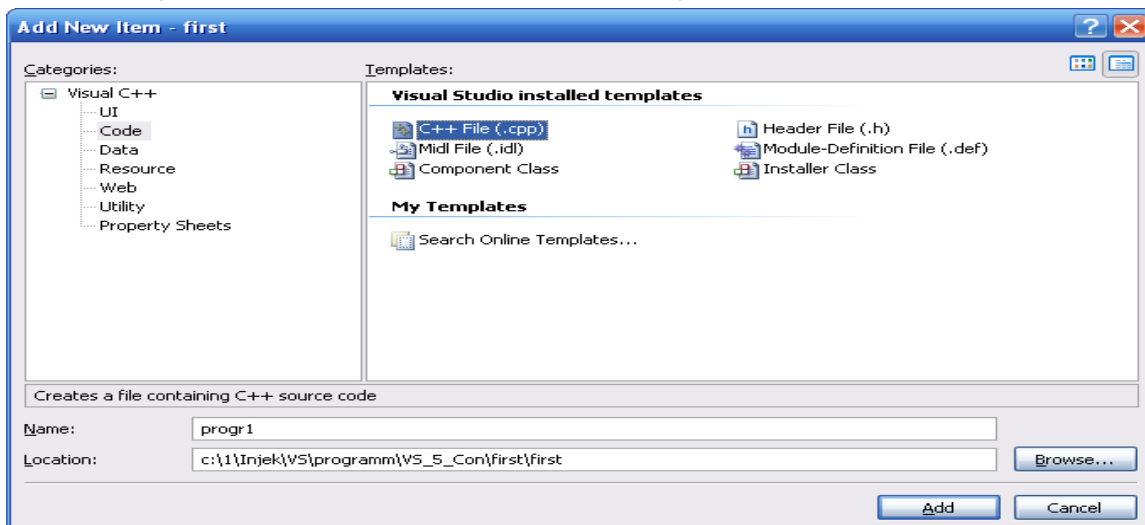


Рис. 1.7. Діалогове вікно Add New Item

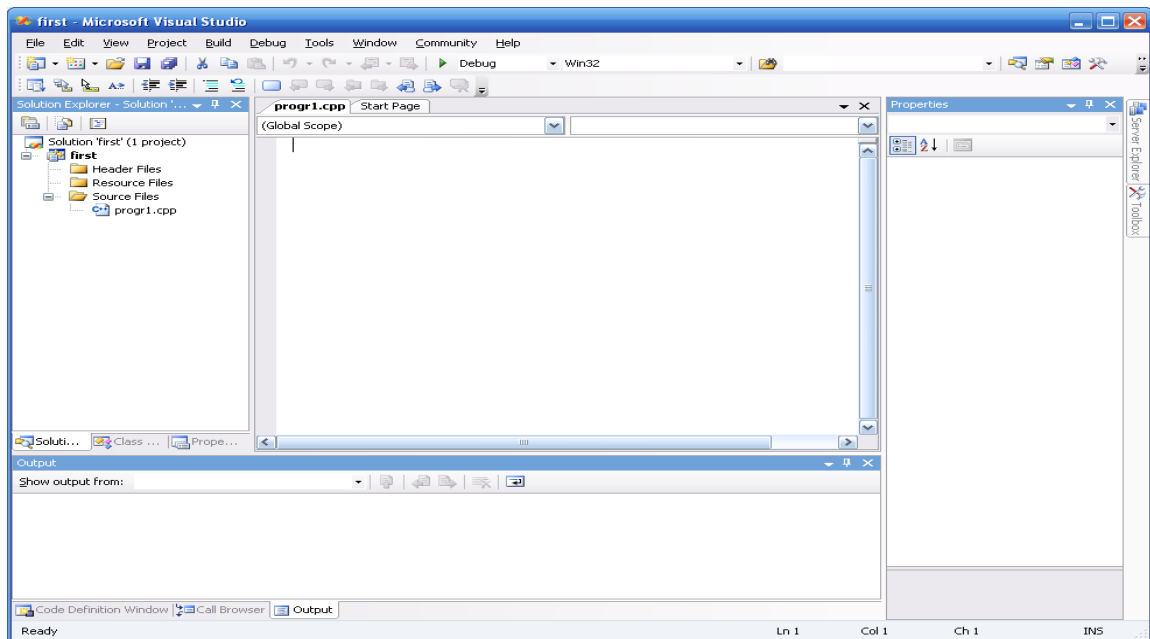


Рис. 1.8. Редактор коду програми

Введіть у редакторі коди наступний текст:

```
#include <iostream>
int main()
{
    std::cout << "Hello!";
    getchar();
    return 0;
}
```

Вікно повинне придбати такий вигляд (рис. 1.9):

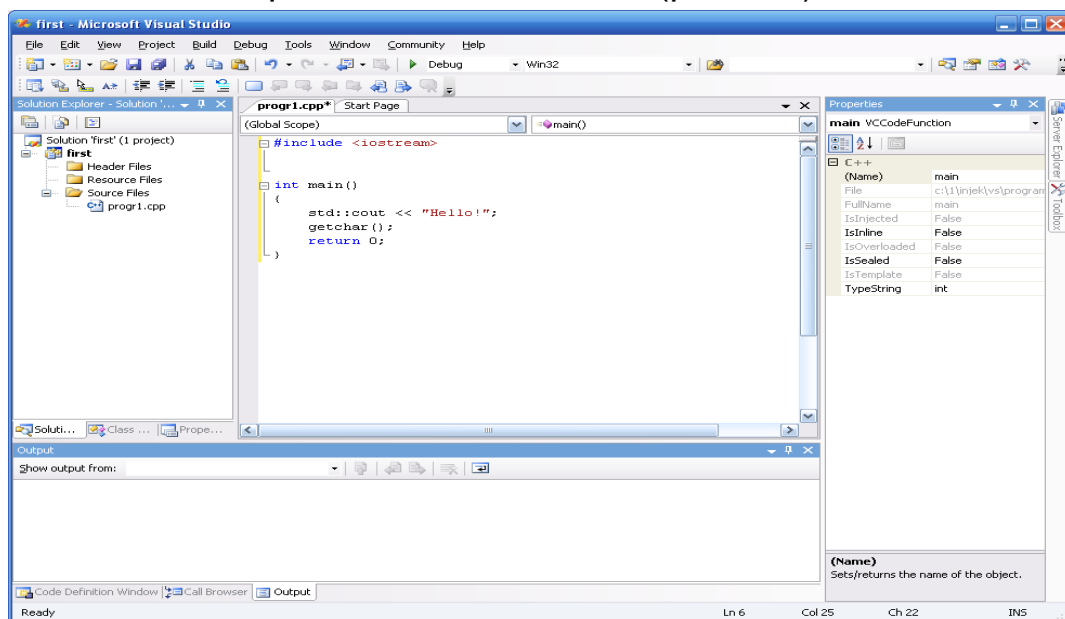


Рис. 1.9. Вікно з кодом програми

Збережіть початковий текст програми на диску, вибравши пункт меню File, а в ньому – команду Save All.

Відкомпілюйте програму й створіть виконуваний файл. Для створення виконуваного файлу рекомендується використовувати команду меню Build> Rebuild Solution.

Натиснувши функціональну клавішу F5 або кнопку Start Debugging на панелі інструментів, запустіть додаток на виконання. У результаті ви побачите консольне вікно з виведеним рядком вітання Hello! (рис. 1.10).



Рис. 1.10. Консольне вікно

Натисніть клавішу Enter, щоб повернутися в головне вікно Інтегрованого Середовища Розробки Microsoft Visual Studio 2005.

Виберіть пункт меню File, а в ньому – команду Exit, щоб завершити роботу з програмою Microsoft Visual Studio 2005.

***Порядок виконання роботи й методичні рекомендації до її виконання:***

створити на диску S теку для проектів (ім'я папки повинне містити вичерпну інформацію про її користувача);

запустити додаток Visual Studio.Net;

ознайомитися з елементами вікна Visual Studio.Net;

ознайомитися з командами кожного пункту головного меню;

ознайомитися з елементами управління вікном Visual Studio.Net (звернути увагу на роботу з ковзаючими вікнами);

отримати у викладача навчально-демонстраційну програму або скористатися програмою, яка використовується в контексті опису середовища;

виконати дії з введення, редагування, відладки й побудови виконуваного модуля навчально-демонстраційної програми;

знайти свою теку проекту й ознайомитися з її вмістом (за допомогою Блокнота відкрити файл ReadMe.txt і перекласти текст, який у ньому записаний);

підсумковий запуск виконуваного модуля виконати з командного рядка;

змінити властивості консольного вікна й зробити фон вікна білим, а шрифт чорним;

за допомогою динамічної довідки з'ясувати призначення всіх службових слів у програмі (службові слова в програмі забарвлені блакитним кольором).

### **Контрольні запитання**

1. Назвіть основні етапи розробки програми на ПЕВМ.
2. Назвіть основні елементи вікна додатка Visual Studio.Net.
3. Назвіть основні органи управління вікном додатка Visual Studio.Net.
4. Перерахуйте засоби відладки програм у середовищі Visual Studio.Net.
5. Назвіть основні елементи структури С-програми.
6. Перерахуйте основні операції з редагування тексту програми.
7. Чим відрізняється оператор мови програмування С++ від коментаря?
8. Яке завдання вирішується на етапі компіляції початкового тексту програми?
9. Яке завдання вирішується на етапі побудови виконуваного модуля програми?

## **Частина 2 (2 години)**

### **Підготовка й вирішення на ПЕВМ завдань лінійного характеру**

**Мета лабораторної роботи** – придбання практичних навиків з підготовки, відладки і виконання лінійних програм.

Змістовний сенс практичного завдання – обчислення похідної в точці, обчислення значення виразу.

Перед виконанням лабораторної роботи студент повинен знати:

класифікацію базових типів даних і їх основні характеристики;

лексичні основи мови С++ – поняття: змінна, вираз, операнд, константа, оператор;

пріоритети операцій;

правила перетворення типів;

основні бібліотечні математичні функції мови С++.

Після виконання лабораторної роботи студент повинен уміти:  
складати лінійні програми з використанням стандартних бібліотечних функцій;  
виконувати відладку й покрокове тестування лінійних програм.

## Короткі теоретичні відомості

### 2.1. Алфавіт мови C++, ідентифікатори та ключові слова

Для програмування завдань лінійного характеру (рис. 1.11), в яких операції виконуються в природному порядку, тобто в порядку їх запису в програмі, необхідно знати наступні конструкції мови C++: ідентифікатори, службові слова, описи даних, вирази, оператори, вбудовані функції

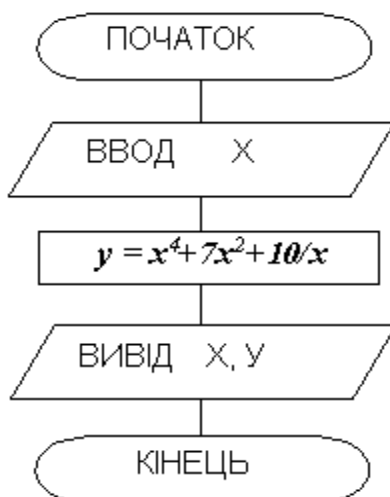


Рис. 1.11. Лінійний алгоритм

Вирази в мові C++ записуються за допомогою 26 рядкових букв англійського алфавіту:

**abcdefghijklmnopqrstuvwxyz;**

26 прописних букв англійського алфавіту:

**ABCDEFGHIJKLMNOPQRSTUVWXYZ;**

десяти цифр: **0123456789;**

наступних спеціальних символів:

**+ - \* / =, . \_ : ; ? \ " ' ~ | ! # \$ & ( ) [ ] { } ^ @.**

До спеціальних символів відноситься також пропуск. Комбінації деяких символів, не розділених пропусками, інтерпретуються як один значущий символ:

**++ -- || << >> >= <= += -= \*= /= .?: :: /\* \*/ //.**

**Ідентифікаторами** називаються імена, що привласнюються змінним, константам, типам даних і функціям, які використовуються в програмах. Після опису ідентифікатора можна посилатися на об'єкт, що позначається ним, у будь-якому місці програми.

Ідентифікатор є послідовністю символів довільної довжини, що містить букви, цифри й символи підкреслення, яка обов'язково повинна починатися з букви або символу підкреслення.

У С++ враховується регістр букв. Компілятор сприймає прописні та рядкові букви, як різні символи. Так, змінні `NM_LEN` і `Nm_Len` розглядаються як два різні ідентифікатори.

Використання символу підкреслення на початку імені ідентифікатора не рекомендується, оскільки даний спосіб запису застосовується в іменах системних підпрограм і змінних. Збіг імені ідентифікатора із зарезервованим ім'ям викличе конфлікт у роботі програми. Два символи підкреслення (`__`) на початку імені ідентифікатора застосовуються в стандартних бібліотеках мови С++.

У процесі формування імен змінних і функцій прийнята угода починати їх з префікса типу даних цього ідентифікатора (наприклад, ідентифікатор `ia` відповідає типу `int`), ідентифікатори з плаваючою комою — буквою `f` (`float`), рядки, що завершуються нульовим символом, — буквами `sz` (`string zero`), покажчики — буквою `p` (`pointer`) і т. д. Це істотно спрощує сприйняття текстів програм.

**Ключові слова є зарезервованими ідентифікаторами, кожному з яких відповідає певна дія.** Змінити призначення ключового слова неможна (директива препроцесора `#define` дозволяє створити "псевдонім" ключового слова, який дублює його дії, можливо, з деякими змінами). Імена ідентифікаторів, що створюються в програмі, не повинні збігатися з ключовими словами мов С++ (табл. 1.1).

Таблиця 1.1

**Ключові слова С/С++**

1	2	3	4
<code>__alignof</code>	<code>__stdcall</code>	<code>else</code>	<code>return</code>
<code>__asm</code>	<code>__super</code>	<code>enum</code>	<code>short</code>
<code>__assume</code>	<code>__try __except</code>	<code>explicit</code>	<code>signed</code>
<code>__based</code>	<code>__try __finally</code>	<code>extern</code>	<code>sizeof</code>
<code>__cdecl</code>	<code>__unaligned</code>	<code>false</code>	<code>static</code>

1	2	3	4
<code>__declspec</code>	<code>__uuidof</code>	<code>float</code>	<code>static_cast</code>
<code>__alignof</code>	<code>__virtual_inheritance</code>	<code>for</code>	<code>struct</code>
<code>__except</code>	<code>auto</code>	<code>friend</code>	<code>switch</code>
<code>__fastcall</code>	<code>__unaligned</code>	<code>goto</code>	<code>template</code>
<code>__finally</code>	<code>bool</code>	<code>extern</code>	<code>this</code>
<code>__forceinline</code>	<code>break</code>	<code>if</code>	<code>throw</code>
<code>__inline</code>	<code>case</code>	<code>inline</code>	<code>true</code>
<code>__int16</code>	<code>catch</code>	<code>int</code>	<code>try</code>
<code>__int32</code>	<code>char</code>	<code>long</code>	<code>typedef</code>
<code>__int64</code>	<code>class</code>	<code>mutable</code>	<code>typeid</code>
<code>__int8</code>	<code>const</code>	<code>namespace</code>	<code>typename</code>
<code>__interface</code>	<code>const_cast</code>	<code>new</code>	<code>union</code>
<code>__leave</code>	<code>continue</code>	<code>operator</code>	<code>using</code>
<code>__multiple_inheritance</code>	<code>default</code>	<code>private</code>	<code>unsigned</code>
<code>__noop</code>	<code>delete</code>	<code>protected</code>	<code>virtual</code>
<code>__pragma</code>	<code>do</code>	<code>public</code>	<code>void</code>
<code>__ptr64</code>	<code>double</code>	<code>register</code>	<code>volatile</code>
<code>__sealed</code>	<code>dynamic_cast</code>	<code>reinterpret_cast</code>	<code>wchar_t</code>
<code>__single_inheritance</code>			<code>while</code>

Ключові слова, що починаються із знаків підкреслення, визначені компанією Microsoft.

## 2.2. Стандартні типи даних, модифікатори, кваліфікатори доступу й перетворення типів даних

Кожна програма обробляє певну інформацію. У C++ дані мають один з восьми базових типів: `char` (текстові дані), `int` (цілі числа), `float` (числа з плаваючою комою одинарної точності), `double` (числа з плаваючою комою подвійної точності), `void` (порожні значення), `bool` (логічні значення), перерахування й покажчики.

Текстом (тип даних `char`) є послідовність символів, які можуть бути розділені пропусками. Зазвичай кожен символ займає 8 біт або один байт з діапазоном значень від 0 до 255.

Цілі числа (тип даних `int`) знаходяться в діапазоні від - 32768 до 32767 і займають 16 біт, тобто два байти або одне слово. У Windows NT і Windows XP і сучасніших ОС Windows використовуються 32-розрядні цілі

числа, що дозволяє розширити діапазон значень від  $-2147483648$  до  $2147483647$ .

У C++ підтримуються три типи цілих чисел. Разом із стандартним типом `int` існують типи `short int` (коротке ціле) і `long int` (довге ціле). Допускається скорочений запис `short` і `long`.

Числа з плаваючою комою одинарної точності (тип даних `float`) можуть бути представлені як у фіксованому форматі, так і в експоненціальному. Діапазон значень — від  $\pm 3.4E-38$  до  $\pm 3.4E+38$ , розмірність — 32 біта, тобто 4 байти або 2 слова.

Числа з плаваючою комою подвійної точності (тип даних `double`) мають діапазон значень від  $\pm 1.7E-308$  до  $\pm 1.7E+308$  і розмірності 64 біта, тобто 8 байтів або 4 слова. Раніше існував тип `long double` з розмірністю 80 біт і діапазоном від  $\pm 1.18E-4932$  до  $\pm 1.18E+4932$ . У нових 32-розрядних версіях компіляторів він еквівалентний типу `double` і підтримується з міркувань зворотної сумісності з написаними раніше додатками.

Перерахування представляються кінцевим набором іменованих констант різних типів.

Тип даних `void`, як правило, застосовується у функціях, що не повертають ніякого значення. Цей тип даних також можна використовувати для створення узагальнених покажчиків.

Покажчики, на відміну від змінних інших типів, містять не дані в звичайному розумінні цього слова, а адреси пам'яті, де зберігаються дані.

Змінні нового логічного типу даних `bool` в C++ можуть містити тільки одну з двох констант: `true` або `false`.

Компілятор мови C++ дозволяє при описі змінних указувати модифікатор `unsigned`. Він застосовується з чотирма типами даних: `char`, `short`, `int` і `long`. Наявність даного модифікатора вказує на те, що значення змінної повинне інтерпретуватися як беззнакове число, тобто самий старший біт є бітом даних, а не бітом знаку. Існує модифікатор `signed`, який виконує протилежну `unsigned` дію.

У C++ використовуються два кваліфікатори доступу `const` і `volatile`. Вони застосовуються для позначення незмінних змінних (`const`) і змінних, значення яких можуть змінитися в будь-який момент (`volatile`).

Іноді потрібне, щоб значення змінної залишалось постійним протягом всього часу роботи програми. Такі змінні називаються



константними. Наприклад, якщо в програмі обчислюється довжина кола або площа круга, часто доводиться оперувати числом (3,14159). У бухгалтерських програмах такою величиною є ПДВ.

Застосування кваліфікатора `volatile` може знадобитися в тому випадку, коли змінна оновлюється системними пристроями, наприклад, таймером. При отриманні сигналу від таймера виконання програми переривається і значення змінної змінюється.

Кваліфікатор `volatile` також застосовується при описі об'єктів даних, спільно використовуваних різними процесами в багатозадачному середовищі.

Допускається в деяких випадках одночасне додатка кваліфікаторів `const` і `volatile`.

У C++ застосовуються й інші кваліфікатори. Їх розгляд виходить за рамки методичних рекомендацій.

Часто буває, коли в операції беруть участь змінні різних типів. Такі операції називаються змішаними. Наприклад:

```
int ival;  
float fres;  
ival=ival*fres;
```

У процесі виконання змішаних операцій компілятор автоматично проводить перетворення типів даних. Цілочисельне значення змінної `ival` зчитується з пам'яті, приводиться до типу з плаваючою комою й помножується на початкове значення змінної `fres`. Результат у вигляді значення з плаваючою комою привласнюється змінною цілого типу `ival`. Автоматичні перетворення типів даних при виконанні змішаних операцій здійснюються відповідно до ієрархії перетворень. Суть полягає в тому, що з метою підвищення продуктивності в змішаних операціях значення різних типів тимчасово приводяться до того типу даних, який має більший пріоритет в ієрархії. Нижче перераховані типи даних у порядку зниження пріоритету: `double`, `float`, `long`, `int`, `short`.

Якщо значення перетвориться на тип, що має велику розмірність, немає втрати інформації, унаслідок чого не страждає точність обчислень.

Іноді потрібно змінити тип змінної, не чекаючи автоматичного перетворення. Для цього призначена операція приведення типу. Якщо в програмі необхідно тимчасово змінити тип змінної, потрібно перед її ім'ям ввести в круглих дужках назву відповідного типу даних. Наприклад:

```
fr=fv+(float) iv/iw;  
fr=fv+iv/(float) iw;  
fr=fv+(float) iv/(float) iw;
```

У всіх трьох випадках перед виконанням ділення відбувається явне приведення значення однієї або двох змінних до типу float.

### 2.3. Специфікатори класу пам'яті

У C++ є чотири специфікатори класу пам'яті: auto, register, static, extern.

Специфікатор класу пам'яті може передувати оголошенням змінних і функцій, указуючи компілятору, як слід зберігати змінні в пам'яті і як діставати доступ до змінних або функцій. Змінні, оголошені із специфікаторами auto і register, є локальними, а із специфікаторами static і extern — глобальними. Пам'ять для локальної змінної виділяється кожного разу, досягнувши блоку, в якому оголошена змінна, і звільняється після закінчення виконання блоку. Пам'ять для глобальної змінної виділяється один раз при запуску програми й звільняється, коли робота програми закінчена.

Вказані чотири специфікатори визначають також зону видимості змінних і функцій, тобто частину програми, в межах якої до ідентифікатора можна звернутися по імені. На зону видимості змінної й функції впливає місце її оголошення в програмі. Якщо оголошення розташоване поза функцією, рівень оголошення є зовнішнім, якщо ж воно знаходиться в тілі функції — внутрішнім.

Специфікатори класу пам'яті розрізняються за сенсом в залежності від того, що оголошується — змінна або функція, а також від того, на якому рівні, зовнішньому або внутрішньому, оголошується даний ідентифікатор.

Змінна, оголошена на зовнішньому рівні, є глобальною і за замовчуванням має клас пам'яті extern. Зовнішнє оголошення може включати ініціалізацію (явну або неявну) або просто бути посиланням на змінну, що ініціалізувалася в іншому місці програми. Наприклад:

```
static int iv;      // за замовчуванням неявно привласнюється 0  
static int iv=10;   // явне привласнення  
int ir=20;          // явне привласнення
```

Зона видимості глобальної змінної розповсюджується до кінця програми. Звернення до змінної не може знаходитися вище за той рядок, в якому вона оголошена.

Змінна оголошується на зовнішньому рівні тільки один раз. Якщо в одному з файлів створена змінна з класом пам'яті `static`, то вона може бути оголошена під тим же ім'ям з тим же специфікатором `static` в будь-якому іншому початковому файлі. Оскільки статичні змінні доступні тільки в межах свого файлу, конфліктів імен не виникне.

За допомогою специфікатора `extern` можна оголосити змінну, яка буде доступна з будь-якого місця програми. Це може бути посилання на змінну, описану в іншому файлі або нижче в тому ж файлі. Остання особливість робить можливим розміщення посилань на змінну до її ініціалізації.

## 2.4. Операції

C++ включає побітові операції, операції інкрементування й декрементування, умовну операцію, операцію кома, операції комбінованого привласнення.

Побітові операції працюють із змінними як із наборами бітів, а не як із числами. Ці операції використовуються в тих випадках, коли необхідно дістати доступ до окремих біт даних (при виведенні графічних зображень на екран). Побітові операції застосовуються тільки до цілочисельних значень. На відміну від логічних операцій, з їх допомогою порівнюються не два числа цілком, а окремі їх біти. Основні побітові операції: І (&), АБО (|) і що виключає АБО (^). Сюди можна також віднести унарну операцію побітового заперечення (~), яка інвертує значення бітів числа.

Операція & записує в біт результату одиницю тільки в тому випадку, якщо обидва порівнюваних біта дорівнюють 1, як показано в наступній таблиці:

Біт 0	Біт 1	Результат
0	0	0
0	1	0
1	0	0
1	1	1

Ця операція часто використовується для маскування окремих бітів числа. Наприклад: `0xF1 & 0x35 = 0x31`.

Операція | записує в біт результату одиницю в тому випадку, якщо хоч би один з порівнюваних бітів дорівнює 1, як показано в наступній таблиці:

Біт 0	Біт 1	Результат
0	0	0
0	1	1
1	0	1
1	1	1

Ця операція часто застосовується для установки окремих бітів числа. Наприклад:  $0x\ F1 \mid 0x\ 35 = 0x\ F5$ .

Операція  $\wedge$  записує в біт результату одиницю в тому випадку, якщо порівнювані біти відрізняються один від одного, як показано в наступній таблиці:

Біт 0	Біт 1	Результат
0	0	0
0	1	1
1	0	1
1	1	0

Ця операція часто застосовується при виведенні зображень на екран, коли відбувається накладення декількох графічних шарів.

Наприклад:  $0x\ F1 \wedge 0x\ 35 = 0x\ C4$ .

У C++ існує дві операції зрушення:  $\ll$  – зрушення вліво,  $\gg$  – зрушення вправо. Дія першої операції полягає в зрушенні бітового представлення цілочисельної змінної, вказаної зліва від операції, вліво на кількість бітів, задану праворуч від операції. При цьому звільнені молодші біти заповнюються нулями, а відповідна кількість старших бітів втрачається.

Зрушення беззнакового числа на одну позицію вліво із заповненням молодшого розряду нулем еквівалентне множенню числа на 2. Наприклад:

```
unsigned int iv=65;           // молодший байт: 01000001
iv<<=1;                       // молодший байт: 10000010
cout<<iv;                     // буде виведене 130
```

Зрушення вправо супроводжується аналогічними діями, тільки бітове представлення числа зрушується на вказану кількість бітів управо. Значення молодших бітів втрачаються, а старші біти, що звільнилися, заповнюються нулями, якщо операнд беззнаковий, і значенням знакового біта інакше. Таким чином, зрушення беззнакового числа на одну позицію вправо еквівалентне діленню числа на два:

```
unsigned int iv=10;           // молодший байт: 00001010
iv>>=1;                       // молодший байт: 00000101
cout<<iv;                     // буде виведене 5
```

Збільшення (зменшення) значення змінної на 1 дуже часто зустрічається в програмах, тому розробники мови C++ передбачили для цих цілей спеціальні операції інкрементування (++) і декрементування (--).

Так, замість рядка `iv+1`, можна ввести рядок `iv++` або `++iv`.

За ситуації, коли операція ++ є єдиною у виразі, не має значення місце її розташування: до імені змінної або після нього. Значення змінної в будь-якому випадку збільшиться на одиницю.

У процесі роботи з складними виразами необхідно уважно стежити, коли саме відбувається модифікація змінної. Потрібно розрізняти префіксні й постфіксні операції, які ставляться відповідно до або після імені змінної.

Наприклад, при постфіксному інкрементуванні `i++` спочатку повертається значення змінної, після чого воно збільшується на одиницю. З іншого боку, операція префіксного інкрементування `++i` указує, що спочатку слід збільшити значення змінної, а потім повернути його як результат. Наприклад:

```
k=++i;    //i=4, k=4
```

```
k=i++;    //i=4, k=3
```

```
k=--i;    //i=2, k=2
```

```
k=i--;    //i=2, k=3
```

У C++ представлені всі стандартні арифметичні операції: складання (+), віднімання (-), множення (\*), ділення (/) і ділення по модулю (%). Перші чотири операції не вимагають роз'яснень. Суть операції ділення по модулю:

```
int ia=3,ib=8,id; id=ib % ia; // результат: 2
```

При діленні по модулю повертається залишок від операції цілочисельного ділення.

У C++ операція привласнення (=) може входити до складу інших виразів. У результаті виконання операції привласнення повертається значення, привласнене лівому операнду. Наприклад, наступний вираз цілком коректний:

```
iv=8*(iw=5); //iv=40.
```

У даному випадку спочатку змінній `iw` привласнюється значення 5, після чого це значення помножується на 8, а результат привласнюється змінною `iv`.

## Комбіновані операції привласнення

Початковий оператор	Еквівалент	Коментар
var=var+3;	var+=3;	До змінної додається 3
var=var-10;	var-=10;	Із змінної віднімається 10
var=var*3.14;	var*=3.14;	Змінна помножується на 3.14
var=var/2.5;	var/=2.5;	Змінна ділиться на 2.5
var=var&0xF;	var&=0xF;	У змінній залишаються тільки 4 молодших розряди
var=var 0xF;	var =0xF;	У змінній встановлюються 4 молодших розряди
var=var<<3;	var<<=3;	Змінна зрушується вліво на 3 розряди
var=var>>5;	var>>=5;	Змінна зрушується вправо на 5 розрядів
var=var%2;	var%=2;	Узяття залишку при діленні var на 2
var=var+1;	var++;	Операція інкремента
var=var-1;	var--;	Операція декремента

**Операції порівняння** призначені для перевірки рівності або нерівності порівнюваних операндів. Усі вони повертають **true** у разі встановлення істинності виразу і **false** інакше. Нижче перераховані оператори порівняння, використовувані в мовах C і C++.

Операція	Виконувана перевірка
==	Дорівнює
!=	Не дорівнює
>	Більше
<	Менше
<=	Менше або дорівнює
>=	Більше або дорівнює

Логічні операції **І** (&&), **АБО** (||) і **НЕ** (!) повертають значення true або false залежно від логічного відношення між їх операндами. Так, операція && повертає true, коли істинні (не дорівнюють нулю) обидва його аргументи. Оператор || повертає false тільки в тому випадку, якщо обидва його аргументи помилкові (дорівнюють нулю). Оператор ! інвертує значення свого операнда з false на true і навпаки.

Приклад використання логічних операцій і операцій порівняння наведений нижче:

```
#include "stdafx.h"
using namespace std;
main() {
float fa=2,fb=4;
cout<<"fa<fb  "<<(fa<fb) <<"\n";
cout<<"fa>fb  "<<(fa>fb) <<"\n";
cout<<"fa<=fb "<<(fa<=fb) <<"\n";
cout<<"fa>=fb "<<(fa>=fb) <<"\n";
cout<<"fa==fb "<<(fa==fb) <<"\n";
cout<<"fa!=fb "<<(fa!=fb) <<"\n";
cout<<"fa&&fb "<<(fa&&fb) <<"\n";
cout<<"fa||fb "<<(fa||fb) <<"\n";
getch();
}
```

Результат роботи програми наведений на рис. 1.12.

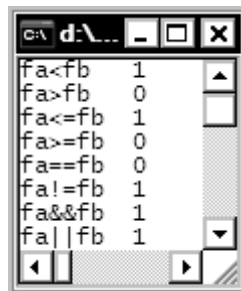


Рис. 1.12. Результат роботи програми

Умовна операція має наступний формат:

Умовний вираз ? вираз 1 : вираз 2;

Якщо умовний вираз true, то виконується вираз 1. Якщо умова false, то виконується вираз 2. Наприклад:

```
var>7 ? x=11:y=7;
```

Часто вирази 1 і 2 використовують одну й ту ж змінну, якою привласнюється або одне, або інше значення. Тоді умовна операція записується трохи інакше:

Змінна = Умовний вираз ? вираз 1 : вираз 2;

Наприклад:

```
char cS=x<=max?'Y':'N';
```

Якщо  $x \leq \max$ , то змінна cS набуває значення 'Y', якщо  $x > \max$ , то cS буде дорівнювати 'N'.

Операція кома (,) дозволяє послідовно виконати два вирази, записані в одному рядку. Результатом є значення виразу, розташованого праворуч від коми. Синтаксис оператора наступний:

лівий\_вираз, правий\_вираз

Найчастіше цей оператор застосовується в циклі for, коли в умову циклу потрібно включити перевірку значень декількох змінних. Наприклад:

```
for(min=0,max=len-1;min<max;min++,max--) { . }
```

Операція sizeof служить для визначення розміру операнда в байтах. Вона може використовуватися як з позначенням змінної, так і з її типом (у останньому випадку операнд слід укласти в круглі дужки). Операція sizeof особливо корисна для визначення розмірів агрегатних змінних – масивів і структур. Наприклад:

```
char cS[]="Операция sizeof";
```

```
int cSLen=sizeof cS;
```

Масив cS займатиме 16 байт пам'яті.

Послідовність виконання різних операцій визначається компілятором.

Якщо не враховувати порядок розбору виразу компілятором, можуть бути отримані неправильні результати.

У табл. 1.3 перераховані всі операції мови C++ в порядку зниження їх пріоритету і вказаний напрям обчислення операндів (асоціативність): зліва направо або справа наліво.

Таблица 1.3

### Пріоритет операцій (від високого до низького)

Операція	Опис	Асоціативність
1	2	3
++	Постфіксний (префіксний) інкремент	Зліва направо
--	Постфіксний (префіксний) декремент	
()	Виклик функції	
[]	Доступ до елемента масиву	
->	Непрямий доступ до члена класу	



## Продовження табл. 1.3

1	2	3
.	Прямий доступ до члена класу	
!	Логічне НЕ	
~	Побітове НЕ	
-	Унарний мінус	
+	Унарний плюс	
&	Узяття адреси	
*	Розкриття покажчика	
sizeof	Отримання розмірності виразу в байтах	
new	Динамічне створення об'єкта	
delete	Динамічне видалення об'єкта	
(тип даних)	Приведення типу	
.*	Прямий доступ до покажчика на член класу (через об'єкт)	Зліва направо
->*	Непрямий доступ до покажчика на член класу (через покажчик на об'єкт)	
*	Множення	Зліва направо
/	Ділення	
%	Ділення по модулю	
+	Складання	Зліва направо
-	Віднімання	
<<	Зрушення вліво	Зліва направо
>>	Зрушення вправо	
<	Менше	Зліва направо
>	Більше	
<=	Менше або дорівнює	
>=	Більше або дорівнює	
==	Дорівнює	Зліва направо
!=	Не дорівнює	
&	Побітове І	Зліва направо
^	Що побітове виключає АБО	Зліва направо
	Побітове АБО	Зліва направо
&&	Логічне І	Зліва направо
	Логічне АБО	Зліва направо
?:	Умовний вираз	Справа наліво
=	Просте привласнення	Справа наліво
*=	Привласнення з множенням	
/=	Привласнення з діленням	
%=	Привласнення з діленням по модулю	

1	2	3
<code>+=</code>	Привласнення зі складанням	
<code>-=</code>	Привласнення з відніманням	
<code>&lt;&lt;=</code>	Привласнення із зрушенням вліво	
<code>&gt;&gt;=</code>	Привласнення із зрушенням управо	
<code>&amp;=</code>	Привласнення з побітовим І	
<code> =</code>	Привласнення з побітовим АБО	
<code>^=</code>	Привласнення з тим, що побітовим виключає АБО	
<code>,</code>	Кома	Зліва направо

## 2.5. Стандартні бібліопакки C++ і бібліотечні математичні функції

Різним бібліотечним функціям потрібні різні заголовні файли. Заголовні файли, які необхідні функції, вказуються в її описі. Наприклад, функції `sqrt()` потрібні оголошення, що містяться в заголовному файлі `math.h`. У Microsoft Visual C++ Run-Time Library Reference перераховані всі бібліотечні функції й відповідні заголовні файли.

Розробниками компілятора C++ передбачені такі категорії бібліотек: класифікації, перетворення, управління каталогами, діагностики, програми, введення/виводу, інтерфейсні, обробки, математичні, управління пам'яттю, управління процесом, стандартні, виведення текстових вікон, для обробки інформації про час і дату.

Стандартні бібліотечні функції:

```
int abs(int x);
```

```
double fabs(double x);
```

Повертає ціле (`abs`) або дробове (`fabs`) абсолютне значення аргументу, який можна використовувати вираз відповідного типу.

```
double acos (double x);
```

```
double asin (double x);
```

```
double atan (double x);
```

```
long double acosl(long double x);
```

```
long double asinl(long double x);
```

```
long double atanl(long double x);
```

Повертає виражену в радіанах величину кута, косинус, синус або тангенс якого переданий відповідній функції як аргумент. Аргумент функції повинен знаходитися в діапазоні від -1 до 1.

```
double cos (double x);  
double sin (double x);  
double tan (double x);  
long double cosl(long double x);  
long double sinl(long double x);  
long double tanl(long double x);
```

Повертає синус, косинус або тангенс кута. Величина кута має бути задана в радіанах.

```
double exp(double x);  
long double exp(long double (x));
```

Повертає значення, яке дорівнює експоненті аргументу (ex, де e — підстава натурального логарифма).

```
double pow (double x, double y);  
long double powl(long double (x), long double (y));
```

Повертає значення, яке дорівнює  $x^y$ .

```
double sqrt(double x);
```

Повертає значення, яке дорівнює квадратному кореню з аргументу.

Заголовний файл: <math.h>

```
int rand(void);
```

Повертає випадкове ціле число в діапазоні від 0 до RAND\_MAX. Перед першим зверненням до функції rand необхідно ініціалізувати генератор випадкових чисел. Для цього треба викликати функцію srand.

```
void srand(unsigned x);
```

Ініціалізував генератор випадкових чисел. Зазвичай як параметр функції використовують змінну, значення якої передбачити заздалегідь неможна, наприклад це може бути поточний час.

Заголовний файл: <stdlib.h>

Наведені нижче функції виконують перетворення рядків у числове значення і чисел у строкове уявлення.

```
double atof(const char* s);
```

Повертає дробове число, значення якого передане функції як аргумент. Функція обробляє рядок до тих пір, поки символи рядка є допустимими. Рядок може бути значенням числа як у форматі з плаваючою точкою, так і в експоненціальному форматі.

```
int atoi(const char* s);  
long atol(const char* s);
```

Повертає ціле відповідного типу, зображення якого передане функції як аргумент. Функція обробляє символи рядка до тих пір, поки не зустрине символ, що не є десятковою цифрою.

```
char *gcvt(double Значення, int Цифр, char* Рядок);
```

Перетворює дробове число в рядок. При перетворенні робиться спроба отримати вказану кількість значущих цифр, а якщо це зробити неможливо, то число зображується у формі з плаваючою точкою.

```
char* itoa (int Значення, char* Рядок, int Підстава);  
char* ltoa (long Значення, char* Рядок, int Підстава);  
char* ultoa(unsigned long Значення, char* Рядок, int Підстава);
```

Відповідно перетворюють ціле, довге ціле і довге беззнакове ціле в рядок. Число зображується у вказаній при виклику функції системі числення.

Рядок — покажчик на рядок, куди буде поміщено зображення числа. Підстава — задає підставу системи числення (від 2 до 36).

Максимальна довжина рядка, формована функцією itoa, — 17 байт, функціями ltoa і ultoa — 33 байти.

Заголовний файл: <stdlib.h>

```
int sprintf(char *Рядок, const char* Формат, Список змінних);
```

Виконує форматований вивід у рядок.

Список змінних — розділені комами імена змінних, задає змінні, значення яких мають бути виведені. Параметр Формат задає спосіб відображення значень змінних.

Дія функції sprintf аналогічно дії функції printf, але вивід виконується в рядок-буфер, а не на екран.

Заголовний файл: <stdio.h>

Бібліотечні функції введення-виводу:

```
int printf(Формат, Список змінних);
```

Виводить на екран значення змінних. Формат виводу задається в рядку форматування, який окрім специфікатора формату може містити текст і символи, що управляють. Значення першої змінної виводиться відповідно до першого специфікатора формату, другий — до другого і т. д.

Специфікатори формату (необов'язковий параметр n задає ширину поля виводу).

Специфікатор	Форма виводу
%ni %nd	Десяткове число із знаком
%nu	Беззнакове ціле десяткове число
%n.mf	Дробове число з десятковою точкою. Необов'язковий параметр m задає кількість цифр дробової частини
%ne	Дробове число з десятковою точкою або, якщо число не може бути представлене у формі з десятковою точкою, в експоненціальній формі
%ns	Рядок символів
%nc	Символ

Символи, що управляють, і спеціальні.

Символ	Дія
\n	Переводить курсор в початок наступного рядка
\t	Переводить курсор в чергову позицію табуляції
\\	Бекслеш
\'	Лапка

`int scanf(const char* Формат, Список адрес змінних);`

Вводить з клавіатури значення змінних відповідно до вказаного специфікатора формату. Перша змінна набуває значення відповідно до першого специфікатора формату, друга — до другого і т. д.

Специфікатор	Вводить
%i %d	Десяткове число із знаком
%u	Беззнакове ціле десяткове число
%e %f	Дробове число
%s	Рядок символів
%c	Символ

`puts(const char* Рядок);`

Виводить на екран рядок символів і переводить курсор в початок наступного рядка екрану. Як параметр функції можна використовувати рядкову константу або строкову змінну.

Заголовний файл: `<stdio.h>`

`int putch(int c);`

Виводить на екран символ.

Заголовний файл: `<conio.h>`

`int getch(void);`

Повертає код символу натиснутої клавіші. Якщо натиснута службова клавіша, то функція `getch` повертає 0. У цьому випадку, для

того, щоб визначити, яка службова клавіша натиснута, потрібно звернутися до функції `getch` ще раз.

Заголовний файл: `<conio.h>`

**Приклад:** Знаходження значення похідної функції в точці.

Постановка завдання:

Задана функція. Знайти її похідну в точці  $x = \pi/2$ .

Для знаходження похідної в точці використовується відомий вираз:

$$\frac{d}{dx} f(x) = \frac{(f(x + dx) - f(x))}{dx}.$$

Крім того, оскільки  $\pi/2 \approx 1,57$ , як значення  $x$  вибираємо 1,57.

Текст програми

```
#include <math.h>
```

```
#include <iostream>
```

```
int main()
```

```
{
```

```
    double f1,f2,pf,x,dx;
```

```
    dx=1.0e-11;    // Вибираємо приріст аргументу
```

```
    x=1.57;        // Вибираємо точку для обчислення похідної
```

```
    f1=sin(x+dx); // Обчислюване значення функції в точці x+dx
```

```
    f2=sin(x);    // Обчислюване значення функції в точці x
```

```
    pf=(f1-f2)/dx; // Знаходимо значення похідної
```

```
    std::cout << "dsin(x)/dx=" << pf << " x= " << x;
```

```
    getch();
```

```
    return 0;
```

```
}
```

Результат роботи програми наведений на рис. 1.13.

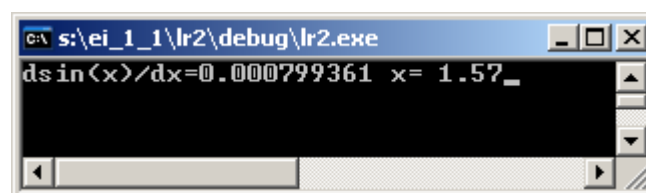


Рис. 1.13. Результат роботи програми

Другий варіант написання програми. Виведення результатів в стилі 3 (за допомогою функції `printf`):

```
#include <math.h>
```

```
#include <conio.h>
```

```
#include <stdio.h>
int main()
{
    double f1,f2,pf,x,dx;
    dx=1.0e-11;
    x=1.57;
    f1=sin(x+dx);
    f2=sin(x);
    pf=(f1-f2)/dx;
    printf("dsin(x)/dx=%6.4f x=%4.2f",pf,x);
    getch();
    return 0;
}
```

### Завдання 1 до ч. 2 лабораторної роботи 1

№	Завдання	№	Завдання
1	$f(x) = \frac{\sqrt{x+1}}{\sqrt{x+1}+1}; f'(0) = ?$	11	$f(x) = \frac{\cos x}{1+\sin x}; f'(\pi/2) = ?$
2	$f(x) = \sin 4x \cos 4x; f'(\pi/3) = ?$	12	$f(x) = \sin^2 x^2; f'(0) = ?$
3	$f(x) = \sin^4 x - \cos^4 x; f'(\pi/12) = ?$	13	$f(x) = \sin^3 \frac{x}{2}; f'(\pi/2) = ?$
4	$f(x) = \frac{\sqrt{x-1} + \sqrt[5]{x-1}}{\sqrt[3]{x-1}}; f'(2) = ?$	14	$f(x) = \sqrt{\frac{1-x}{1+x^2}}; f'(0) = ?$
5	$f(x) = 5(x+1)^2 \sqrt[5]{x-1}; f'(2) = ?$	15	$f(x) = \sqrt{x^2-1} + \sqrt[3]{x}; f'(1) = ?$
6	$f(x) = \frac{1}{2} \sin x \operatorname{tg} 2x; f'(\pi/2) = ?$	16	$f(x) = \frac{2^{2x}}{\sqrt{2-2^{2x}}}; f'(0) = ?$
7	$f(x) = 2^{x-2x^2-1}; f'(0) = ?$	17	$f(x) = (x^2-x)\cos^2 x; f'(0) = ?$
8	$f(x) = \frac{\sin 2x}{\sqrt{x}}; f'(\pi) = ?$	18	$f(x) = \frac{x^2+3}{x-1}; f'(0) = ?$
9	$f(x) = \frac{x}{\sqrt{x^2+3}} + \frac{1}{x+1}; f'(1) = ?$	19	$f(x) = \frac{x-2}{\sin^2 x}; f'(\pi/2) = ?$
10	$f(x) = \frac{x}{3} - \frac{3}{x}; f'(3) = ?$	20	$f(x) = x - \frac{2}{x^2} - \frac{1}{3x^3}; f'(-1) = ?$

**Завдання 2.** Скласти програму для знаходження значень виразів А і В за заданими значеннями початкових даних x, y, z.

Початкові дані (за варіантами) знаходяться в табл. 1.4.

Таблиця 1.4

## Початкові дані

№ вар.	Функції	Початкові дані		
		x	y	z
1	2	3	4	5
1	$A = 2^{-x} \sqrt{x + \sqrt[4]{ y }}, B = \sqrt[3]{e^{x-1/\sin z}}$	3,98	-1,62	0,52
2	$A = y^{\sqrt[3]{ x }} + \sin^3 \left( \sqrt[3]{x} - 3 \right), B = \frac{y \left( \arctg z - \pi/6 \right)}{ x  + 1/\left( \sqrt[3]{y^2} + 1 \right)}$	-0,62	0,82	25
3	$A = 2^{\left( \sqrt[3]{x} \right)} + 3^{\left( \sqrt[3]{y} \right)}, B = \frac{ x-y  \left( 1 + \sin^2 z / \left( \sqrt[3]{x} + y \right) \right)}{e^{ x-y } + 0.5x}$	3,25	0,32	0,4
4	$A = \frac{\sqrt{ x-1 } - \sqrt[3]{ y }}{1 + 0.5x^2 + 0.25y^2}, B = x \left( \arctg z + e^{-\left( \sqrt[3]{x} + 3 \right)} \right)$	-0,62	3,32	5,4
5	$A = \sqrt{y + \sqrt[3]{x-1}}, B =  x-y  \left( \sin z^2 + \tg z \right)$	17,4	10,3	0,82
6	$A = \frac{y^{x+1}}{\sqrt[3]{ y-2 } + 3} + \frac{x + 0.5y}{2 x+y }, B = \left( \sqrt[3]{x} + 1 \right)^{1/\sin z}$	1,62	-15,4	0,25
7	$A = \frac{x^{y+1} + e^{y-1}}{1+x y-\tg z }, B = 1 +  y+x  + \frac{ y-x ^2}{2} - \frac{ y-x ^3}{3}$	2,44	0,86	-0,16
8	$A = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!}, B = x \left( \arctg z + \cos^3 y \right)$	0,33	0,02	32
9	$A = \left( -x \right)^{\frac{x+y/\left( \sqrt[3]{x^2} + 4 \right)}{y^{x-2} + 1/\left( \sqrt[3]{x^2} + 4 \right)}}, B = \frac{1 + \cos \left( \sqrt[3]{x} - 2 \right)}{0.5x + \sin^2 z}$	3,25	4,2	-0,66
10	$A = y + \frac{x}{y + x^2/\left( \sqrt[3]{y} + x^3/y \right)}, B = \left( 1 + \tg^2 z/2 \right)^{\sqrt{ y +6}}$	0,12	-8,75	0,76
11	$A = \lg \left( \sqrt{e^{x-y}} + x^{ y } + z \right), B = x - \frac{x^3}{3!} - \frac{x^5}{5!}$	1,53	-3,26	8,2
12	$A = \frac{2 \cos \left( \sqrt[3]{x} - \pi/6 \right)}{0.5 + \sin^2 y}, B = 1 + \frac{z^2}{3 + z^2/5}$	1,42	-1,22	3,52
13	$A = \frac{\sqrt[3]{8 +  x-y ^2} + 1}{x^2 + y^2 + 2}, B = \cos^2 \left( \arctg 0.5 \right)$	3,74	-0,82	0,16
14	$A = \frac{1 + \sin^2 \left( \sqrt[3]{x} + y \right)}{ x - 2y/\left( \sqrt[3]{x} + x^2 y^2 \right) } x^{ y }, B = e^{ x-y } + \left( g^2 z + 1 \right)^x$	3,74	-0,82	0,16



1	2	3	4	5
15	$A =  \cos x + \cos y ^{1+2 \sin y}, B = 1 + z + \frac{z^2}{2} + \frac{z^3}{3}$	0,42	-0,87	-0,47
16	$A = \ln \left( \sqrt{y} - \sqrt{ y } \right) - y/2, B = \sin^2 \arctg z$	-15,2	4,64	21,3
17	$A = \sqrt{10 \sqrt{x} + x^{y+2}}, B = \arcsin z^2 +  x + y $	16,5	-2,75	0,15
18	$A = 5 \arctg x - 0.25 \arctg y, B = \frac{x + 3 x - y  + x^2}{ x - y ^z + z^2}$	-17,2	6,33	3,25
19	$A = e^{ x+y } +  x - y ^{x+y}, B = \arctg x + \arctg z$	-2,23	-0,82	15,2
20	$A = \left  x^{y/x} - \sqrt{y/x} \right , B = \sqrt{x - \frac{y - z}{1 + \sqrt{x}}} \sqrt{y - x}$	1,82	18,5	-3,29

### Контрольні запитання

1. Поясніть сенс поняття "оператор".
2. Що розуміється під типом даних?
3. Яка інформація повідомляється компілятору при оголошенні змінних і констант?
4. Дайте визначення виразу.
5. Вкажіть правила обчислення виразів.
6. Наведіть приклади операцій з однаковим пріоритетом.
7. Вкажіть операції з найвищим і найменшим пріоритетом.
8. Перерахуйте ключові слова, використовувані при оголошенні стандартних типів даних.

## Лабораторна робота 2

### Підготовка і вирішення на ПК завдань з розгалуженням

**Мета лабораторної роботи** – придбання практичних навиків з підготовки, відладки і виконання програм, що розгалужуються.

Перед виконанням лабораторної роботи студент повинен знати:

методику розробки програми із загальною лінійною частиною й декількома гілками;

алгоритми виконання й синтаксис операторів if, if/else, switch/case і умовній операції ?..

Після виконання лабораторної роботи студент повинен уміти розробляти і відладжувати програми з розгалуженнями.

### **Короткі теоретичні відомості**

У багатьох випадках подальше виконання програми від деякої точки повинне йти різними шляхами залежно від певних умов. Наприклад, при клацанні мишею по одній кнопці програма повинна приступити до виконання гілки 1, а при клацанні по іншій кнопці – до виконання гілки 2. Такі умовні переходи зовсім не обов'язково пов'язані з командами користувача; програма може сама вибрати подальший шлях, наприклад, за наслідками деякої математичної операції: при нульовому результаті виконати один фрагмент, при негативному – другий, при позитивному – третій. У C++ існує три базових оператора вибору: `if`, `if/else`, `switch/case` і умовній операції `?:`.

#### **Оператор `if`**

Оператор `if` призначений для виконання команди або блоку команд залежно від того, істинно задана умова чи ні. Формат оператора `if`:

`if (умова) вираз;`

Якщо в результаті перевірки умови повертається значення `true`, виконується вираз, після чого управління передається наступному рядку програми. Якщо ж результатом перевірки умови є значення `false`, вираз пропускається.

Оператор `if/else` дозволяє вибірково виконувати одну з двох дій залежно від умови. Формат даної інструкції має вигляд:

`if (умова) вираз 1; else вираз 2;`

Якщо в результаті перевірки умови повертається значення `true`, виконується вираз 1, інакше — вираз 2.

Якщо операторна частина гілки `if` або `else` містить не один вираз, а декілька, необхідно укласти їх у фігурні дужки. Після закриваючої фігурної дужки крапка з комою не ставиться.

Оператор `if` першої та другої форми реалізують алгоритми представлені на рис. 2.1.



Рис. 2.1. Алгоритми роботи операторів if

Як аналізований вираз в операторові if найчастіше використовується одна з операцій відношення.

Разом з операціями відношення в інструкції if широко використовуються логічні операції. Об'єднуючи їх з операціями відношення, можна створювати комбіновані конструкції перевірки даних.

### Оператор switch/case

Оператор switch/case дозволяє залежно від значення деякого виразу вибрати один з багатьох варіантів продовження програми. Оператор має наступний формат:

```

switch(вираз){
case значення 1: оператор 1;break;
case значення 2: оператор 2;break;
...
case значення N: оператор N;break;
default: операторN+1;
}

```

Оператор switch/case реалізує алгоритм, наведений на рис. 2.2.

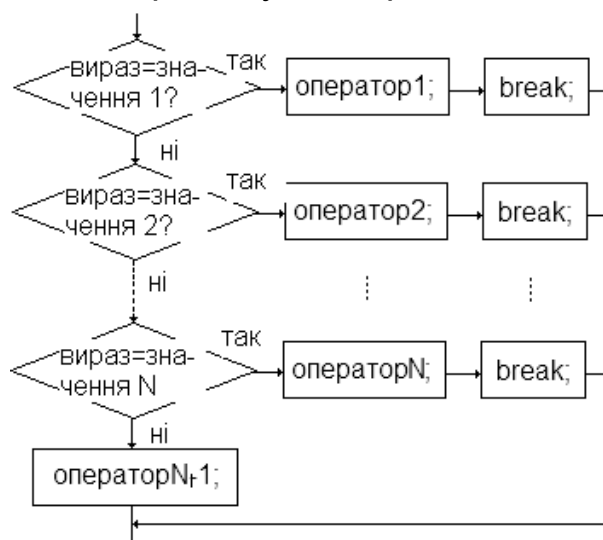


Рис. 2.2. Алгоритми роботи оператора switch/case

Оператор switch/case може бути використаний у варіанті без оператора N+1.

Як вираз при операторові switch зазвичай використовується змінна типу int або char, хоча можна використовувати й складніші вирази, в які входять, наприклад, арифметичні або логічні операції над декількома змінними та константами.

Як значення при операторові case зазвичай використовуються просто константи (у числовій формі або в символьній, якщо вони були заздалегідь визначені за допомогою оператора препроцесора #define), проте можуть використовуватися і вирази над константами.

Виконання оператора switch починається з обчислення виразу в дужках, який повинен давати цілочисельний результат. Цей результат послідовно порівнюється із значеннями при операторах case, і, якщо буде виявлено рівність результатів, то виконується оператор відповідного case. Якщо збіги результатів не виявлено, виконується оператор при операторові default, якщо оператор default відсутній, то починають виконуватися оператори, наступні за всією конструкцією switch/case.

### **Оператори break, continue і goto**

Оператор break використовується для виходу з оператора while, do.while, for і switch, що безпосередньо його містить. Управління передається на оператора, наступного за оператором, з якого здійснюється вихід. Приклад додатка оператора break наведений вище.

Оператор continue використовується для пропускання частини виконуваної ітерації циклу, який безпосередньо його містить, що залишилася. Якщо умовами циклу допускається нова ітерація, то вона виконується, інакше цикл завершується.

Оператора goto реалізує безумовний перехід, тобто дозволяє перейти в будь-яку точку програми, як вперед по тексту програми, так і назад. Точка переходу позначається за допомогою мітки, яка є довільним ідентифікатором з двоточкою в кінці.

## **Завдання до лабораторної роботи 2**

**Задача 1.** Знайти все раціональне коріння полінома  $n$ -й ступеня з цілими коефіцієнтами.

Змістовний сенс практичного завдання – обчислення дійсного коріння многочленів.

При вирішенні таких завдань використовується теорема.

**Теорема.** Для того, щоб нескоротний дріб  $p/q$  був коренем рівняння  $q(x)$  з цілими коефіцієнтами, необхідно, щоб число  $p$  було дільником вільного члена  $a_0$ , а число  $q$  – дільником старшого коефіцієнта  $a_n$ .

Якщо рівняння має цілі коефіцієнти, а старший коефіцієнт дорівнює одиниці (тобто  $a_n=1$ ), то раціональним корінням цього рівняння можуть бути тільки цілі числа, які є дільниками вільного члена  $a_0$ .

### Приклад

$$f(x) = 6x^4 - x^3 - 7x^2 + x + 1.$$

Вільний член цього рівняння має дільників  $\pm 1$ . Старший коефіцієнт рівняння має дільників  $\pm 1, \pm 2, \pm 3, \pm 6$ . Складемо всі можливі дроби  $p/q$ . Отримаємо наступні числа  $1, -1, 1/2, -1/2, 1/3, -1/3, 1/6, -1/6$ .

Усе раціональне коріння початкового рівняння належить цій безлічі чисел.

Текст програми:

```
#include "stdafx.h"
#include <math.h>
#include <conio.h>
#include <stdio.h>
int _tmain(int argc, _TCHAR* argv[])
{
    double x,f;
    x=1.;
    f=6.*pow(x,4.)-pow(x,3.)-7.*pow(x,2.)+x+1.;
    if(f==0.)    printf("f(x) =%6.2f x=%6.2f\n",f,x);
    x=-1.;
    f=6.*pow(x,4.)-pow(x,3.)-7.*pow(x,2.)+x+1.;
    if(f==0.)    printf("f(x) =%6.2f x=%6.2f\n",f,x);
    x=1./6.;
    f=6.*pow(x,4.)-pow(x,3.)-7.*pow(x,2.)+x+1.;
    if(f==0.)    printf("f(x) =%6.2f x=%6.2f\n",f,x);
    x=-1./6.;
    f=6.*pow(x,4.)-pow(x,3.)-7.*pow(x,2.)+x+1.;
```

```

if(f==0.)    printf("f(x) =%6.2f x=%6.2f\n",f,x);
             x=1./3.;
f=6.*pow(x,4.)-pow(x,3.)-7.*pow(x,2.)+x+1.;
if(f==0.)    printf("f(x) =%6.2f x=%6.2f\n",f,x);
             x=-1./3.;
f=6.*pow(x,4.)-pow(x,3.)-7.*pow(x,2.)+x+1.;
if(f==0.)    printf("f(x) =%6.2f x=%6.2f\n",f,x);
             x=1./2.;
f=6.*pow(x,4.)-pow(x,3.)-7.*pow(x,2.)+x+1.;
if(f==0.)    printf("f(x) =%6.2f x=%6.2f\n",f,x);
             x=-1./2.;
f=6.*pow(x,4.)-pow(x,3.)-7.*pow(x,2.)+x+1.;
if(f==0.)    printf("f(x) =%6.2f x=%6.2f\n",f,x);
getch();
return 0; }

```

Результат роботи програми наведений на рис. 2.3.

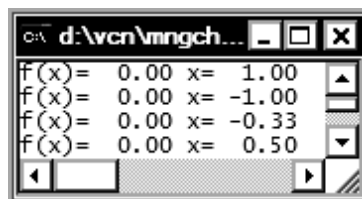


Рис. 2.3. Результат роботи програми

Таблиця 2.1

### Завдання до задачі 1.

№	Завдання
1	2
1	$f(x) = x^3 - 4x^2 - x + 4$
2	$f(x) = 6x^4 + 7x^3 - 36x^2 - 7x + 6$
3	$f(x) = 2x^4 + 7x^3 - 12x^2 - 38x + 21$
4	$f(x) = x^3 + 2x^2 - x - 2$
5	$f(x) = 2x^3 - 4x^2 - x - 15$

1	2
6	$f(x) = 6x^4 + 19x^3 - 7x^2 - 26x + 12$
7	$f(x) = 2x^5 + 5x^4 + 11x^3 + 14x^2 + 11x + 5$
8	$f(x) = x^4 + 4x^3 - 2x^2 - 12x + 9$
9	$f(x) = 2x^3 + 3x^2 + 3x + 2$
10	$f(x) = x^5 + 3x^4 - x^3 + 2x^2 - 24x - 32$
11	$f(x) = 3x^3 + 2x^2 + 2x + 3$
12	$f(x) = x^4 + 3x^3 - 44x^2 + 15x + 25$
13	$f(x) = 2x^3 - 4x^2 - x - 15$
14	$f(x) = 6x^4 + 19x^3 - 7x^2 - 26x + 12$
15	$f(x) = 24x^5 + 10x^4 - x^3 - 19x^2 - 5x + 6$
16	$f(x) = 12x^5 + 18x^4 - 45x^3 - 45x^2 + 18x + 12$
17	$f(x) = 2x^5 + 5x^4 + 11x^3 + 14x^2 + 11x + 5$
18	$f(x) = x^4 + 2x^3 - 2x^2 - 6x + 5$
19	$f(x) = 6x^4 + 7x^3 - 36x^2 - 7x + 6$
20	$f(x) = 6x^4 + 5x^3 - 38x^2 + 5x + 6$

**Задача 2.** Обчислити значення функції  $Y(x)$  при різних значеннях початкових даних  $x$  і  $a$ .

Вид функції і значення початкових даних наведені в табл. 2.2.

Таблиця 2.2

### Вид функції і значення початкових даних

№ вар.	Функція	Початкові дані	
		$x$	$a$
1	2	3	4
1	$Y = \frac{1+a^x}{1-a^x} + \begin{cases} 1 + \operatorname{tg} \left( \frac{\pi}{2} \right) & x > a \\ 1 + \operatorname{tg} \left( \frac{\pi}{4} \right) & x \leq a \end{cases}$	3,1 1,3	2,3

1	2	3	4
2	$Y = \frac{\sqrt{1+e^x}}{\sqrt{1-e^x}} + \begin{cases} \sqrt{a+x/2}, & x > a \\ 1/\sqrt{a+x/2}, & x \leq a \end{cases}$	7,3 2,8	5,8
3	$Y = \frac{\sqrt{1+a^x}}{\sqrt{1-a^x}} + \begin{cases} \operatorname{tg} \sqrt{a+x}, & x > a/2 \\ \operatorname{tg} \sqrt{a/x+1}, & x \leq a/2 \end{cases}$	4,2 1,25	3,7
4	$Y = \operatorname{tg} \frac{1-x}{1+x} + \begin{cases} a + e^{-x}, & x \geq a \\ 1 - e^{-x}, & x < a \end{cases}$	0,5 0,34	0,45
5	$Y = \frac{25 + \sqrt{ x }}{25a} + \begin{cases} a/\sqrt{ x }, & x < a \\ \sqrt{ x } + a, & x \leq a \end{cases}$	3,7 7	6,2
6	$Y = \frac{\sin 5x}{1+5x} + \begin{cases} a + \sin^2 x, & x > a \\ 1/\sqrt{a + \sin^2 x}, & x \leq a \end{cases}$	2,5 0,17	$\pi/20$
7	$Y = \frac{ax}{1+\sqrt{ x }} + \begin{cases} \sqrt{a+1}, & x > a \\ 1 + \sqrt{a+1}/2x, & x \leq a \end{cases}$	2,8 1,73	2,1
8	$Y = \frac{x^3}{1+e^{-3x}} + \begin{cases} a^{3x}/\sqrt{1+x}, & a/x \geq \pi/6 \\ a^{3x}/\sqrt{1-x}, & a/x < \pi/6 \end{cases}$	2,75 12,3	5
9	$Y = \sqrt{a+1} + \begin{cases} a + \sqrt{1-e^{-x}}, & x \geq a \\ a - \sqrt{1+e^x}, & x < a \end{cases}$	2,8 1,1	1,25
10	$Y = x^4 e^{\sqrt{a+1}} + \begin{cases} ax/\sqrt{a+\ln \sqrt{a+1}}, & x > a \\ a\sqrt{a+1}/\sqrt{a+\ln \sqrt{a+1}}, & x \leq a \end{cases}$	2,75 1,89	2,3
11	$Y = \frac{\operatorname{tg} x}{1+x} + \begin{cases} a/\sqrt{a+x^3}, & x > \operatorname{tg} \sqrt{a} \\ ax/\sqrt{a+x^3}, & x \leq \operatorname{tg} \sqrt{a} \end{cases}$	4,6 1,27	1,3
12	$Y = \frac{x}{1+x} + \begin{cases} a \sin \sqrt{a+x}, & x > a \\ a \cos \sqrt{a+x}, & x \leq a \end{cases}$	5,85 2,13	4,3
13	$Y = \frac{\ln \sqrt{a+x^2}}{1+x^2} + \begin{cases} 1+a^x, & x > a \\ 1-a^{-x}, & x \leq a \end{cases}$	7,85 5,2	8,5
14	$Y = \frac{1+\sqrt{x}}{\sqrt{a+\sqrt{x}}} + \begin{cases} 1+\sqrt{x^a}, & a \geq x \\ 1+\sqrt{x-a}, & a < x \end{cases}$	4,3 6,75	5,8



1	2	3	4
15	$Y = \frac{\ln(1 + \sqrt{1 + x^2})}{1 + e^{(+x*x)}} + \begin{cases} a + e^{(+x*x)}, & x > a \\ a - e^{(+x*x)}, & x \leq a \end{cases}$	6 2,7	4,5
16	$Y = \frac{4 + 5 \operatorname{tg} \left( \frac{+3x}{2} \right)}{5.5a} + \begin{cases} e^{2x-1} + a, & x \leq a \\ e^{2x-1} - a, & x > a \end{cases}$	0,75 3,4	1,2
17	$Y = \frac{a \cos^2 x}{1 + \operatorname{tg}^2 \left( \frac{x}{4} \right)} + \begin{cases} (x^2 + 1) e^x, & e^x > a \\ (x^2 - 1) e^x, & e^x \leq a \end{cases}$	1,7 0,6	3,7
18	$Y = \frac{(x + 10)a}{x - 10} + \begin{cases} (60 - x)x / (60 + x^2), & x > a \\ (x - 60)x / (60 + x^2), & x \leq a \end{cases}$	10,2 5,4	8,7
19	$Y = \frac{1 + x^2 + x^3}{1 + \sqrt{ x } + \sqrt[3]{ x }} + \begin{cases} (e^x - 1)a, & x > a \\ (+e^x)a, & x \leq a \end{cases}$	7,3 4,75	5,1
20	$Y = \frac{(x + 10)a}{x - 10} + \begin{cases} (60 - x)x / (60 + x^2), & x > a \\ (x - 60)x / (60 + x^2), & x \leq a \end{cases}$	1,2 8,4	5,7

**Задача 3 (на оцінку 12).** У східному календарі прийнятий 60-річний цикл, що складається з 12-літніх підциклів, що позначаються назвами кольору: зелений, червоний, жовтий, білий і чорний. У кожному підциклі роки носять назви тварин: щура, корови, тигра, зайця, дракона, змії, коня, вівці, мавпи, курки, собаки і свині. За номером року вивести його назву, якщо 1984 рік був початком циклу — роком зеленого щура. Використувати оператор switch/case.

#### Контрольні запитання

1. Що таке обчислювальний процес, що розгалужується?
2. Які форми запису має умовний оператор if?
3. Назвіть відмінні особливості умовного виразу порівняно з умовним оператором.
4. Напишіть програму пошуку мінімального числа з трьох заданих чисел.
5. Назвіть все раціональне коріння рівняння типу:

$$f(x) = x^3 - 4x^2 - x + 4.$$

## Лабораторна робота 3

### Підготовка й вирішення на ПК завдань з використанням циклів

**Мета лабораторної роботи** – придбання практичних навиків з підготовки, відладки й виконання циклічних програм.

Перед виконанням лабораторної роботи студент повинен знати: основи додатка стандартних операторів циклу: `while`, `do while`, `for`.

Після виконання лабораторної роботи студент повинен уміти розробляти типові циклічні програми на мові C++.

### Короткі теоретичні відомості

Оператори циклу служать для виконання деякого фрагмента програми кілька разів. У окремих випадках фрагмент виконується в кожному послідовному кроці циклу без змін; частіше кожен крок циклу декілька відрізняється від попереднього. Цикл може виконуватися задане заздалегідь число кроків, а може завершуватися при настанні деякої умови.

Існує три види циклів: `while`, `for` і `do`.

Оператор циклу `while` називається циклом з передумовою та має наступний формат:

`while (вираз) тіло циклу;`

Оператора `while` реалізує алгоритм, представлений на рис. 3.1.

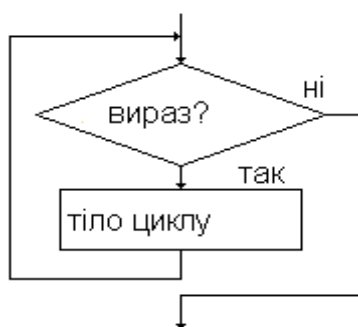


Рис. 3.1. Алгоритми роботи оператора `while`

Як вираз допускається використовувати будь-який вираз мови C++, а як тіло будь-який оператор, зокрема порожній або складений. Схема виконання оператора `while` наступна:

1. Обчислюється вираз.

2. Якщо вираз false, то виконання оператора while закінчується й виконується наступний за порядком оператор. Якщо вираз true, то виконується тіло циклу.

3. Процес повторюється з пункту 1.

Тіло циклу виконується до тих пір, поки значення виразу рівне true. Вираз обчислюється перед кожним виконанням оператора.

Цикл for має наступну формат:

for (вираз 1; вираз 2; вираз 3;) тіло циклу;

Оператора for реалізує алгоритм, представлений на рис. 3.2.

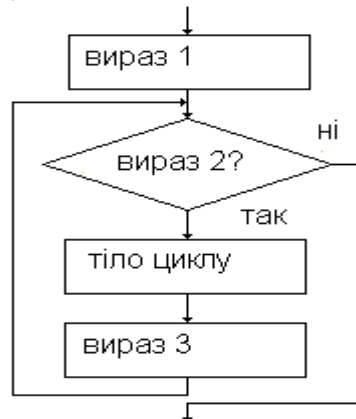


Рис. 3.2. Алгоритми роботи оператора for

Вираз 1 зазвичай використовується для встановлення початкового значення змінних, керівників циклом. Вираз 2 – це вираз, що визначає умову, при якій тіло циклу виконуватиметься. Вираз 3 визначає зміну змінних, керівників циклом після кожного виконання тіла циклу.

Схема виконання оператора for:

1. Обчислюється вираз 1.

2. Обчислюється вираз 2.

3. Якщо значення виразу 2 відмінно від нуля (true), виконується тіло циклу, обчислюється вираз 3 і здійснюється перехід до пункту 2, якщо вираз 2 дорівнює нулю (false), то управління передається до оператора, наступного за оператором for.

Істотне те, що перевірка умови завжди виконується на початку циклу. Це означає, що тіло циклу може жодного разу не виконатися, якщо умова виконання відразу буде помилковою.

Цикл for є зручним скороченим записом для циклу while вигляду

вираз 1;

while (вираз 2) {

тіло циклу;

вираз 3;

}

Вираз 1 служить для завдання початкових умов виконання циклу, вираз 2 забезпечує перевірку умови виходу з циклу, а вираз 3 модифікує умови, задані виразом 1. Будь-який з виразів може бути опущений. Якщо опущено вираз 2, то за замовчуванням замість нього підставляється значення true. Наприклад, цикл for:

for (;вираз 2; ) тіло циклу;

з опущеними вираз 1 і вираз 3 еквівалентний циклу

while (выраз2) тіло циклу;

Цикл for:

for (;;) тіло циклу;

зі всіма опущеними виразами еквівалентний циклу

while (true) тіло циклу;

тобто еквівалентний нескінченному циклу. Такий цикл може бути перерваний тільки явним виходом з нього за допомогою операторів break, goto або return, що містяться в тілі циклу.

Оператор циклу do while називається оператором циклу з умовою поста і використовується в тих випадках, коли необхідно виконати тіло циклу хоч би один раз. Формат оператора має наступний формат:

do тіло циклу while (вираз);

Схема виконання оператора do while:

1. Виконується тіло циклу (яке може бути складеним оператором).

2. Обчислюється вираз.

3. Якщо вираз false, то виконання оператора do while закінчується й виконується наступний за порядком оператор. Якщо вираз true, то виконання оператора продовжується з пункту 1.

Щоб перервати виконання циклу до того, як умова стане помилковою, можна використовувати оператора break.

Оператора do while реалізує алгоритм, наведений на рис. 3.3.

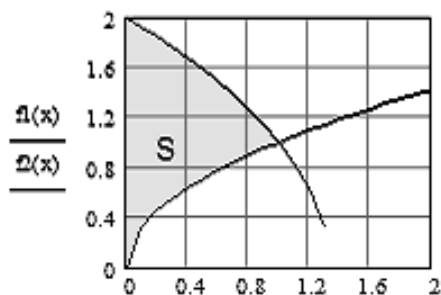


Рис. 3.3. Алгоритм роботи оператора do while

На відміну від циклу `while`, в якому перевірка умови закінчення циклу робиться до виконання тіла циклу, в циклі `do while` така перевірка має місце після виконання тіла циклу. Отже, тіло циклу `do while` буде виконано хоч би один раз, навіть якщо вираз має значення `false` із самого початку.

### Завдання 1

Обчислити площу фігури, обмеженої лініями.



Для обчислення площі фігури обмеженою лініями можна використовувати ітераційний вираз  $s_{i+1} = s_i + (f_2(x_i) - f_1(x_i)) dx$ , де  $s_{i+1} = s_i + (f_2(x_i) - f_1(x_i)) dx$ ,  $s_{i+1} = s_i + (f_2(x_i) - f_1(x_i)) dx$  ;  $i = 0, 1, 2 \dots n$ .

Текст програми:

```
#include "stdafx.h"
#include <math.h>
#include <stdio.h>
#include <windows.h>
int _tmain(int argc, _TCHAR* argv[])
{
    char buf[80];
    double x,dx,f1,f2,s;
    s=0.;
    x=0.;
    dx=1.0e-5;
    f1=sqrt(x);
    f2=sqrt(4.-3.*x);
    for(;f1<f2;x=x+dx)
    {
        f1=sqrt(x);
        f2=sqrt(4.-3.*x);
        s=s+(f2-f1)*dx;
    }
}
```

```

    sprintf(buf", s=%6.2f\n x=%6.2f\n f1=%6.2f\n
f2=%6.2f\n",s,x,f1,f2);
    return 0;
}

```

Результат роботи програми наведений на рис. 3.4.

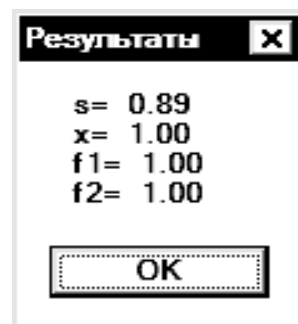
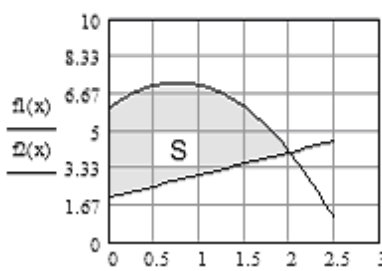
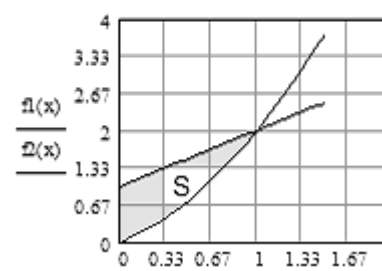
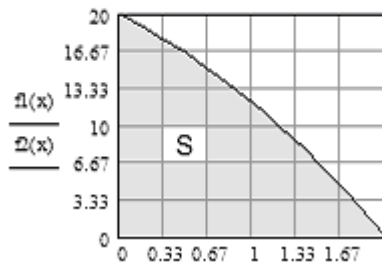
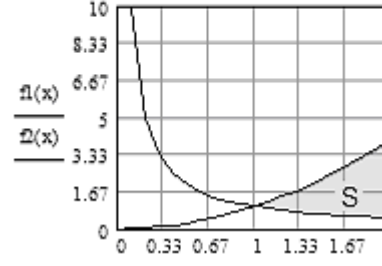
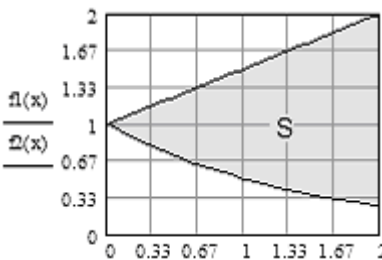
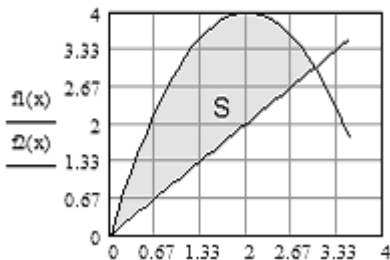
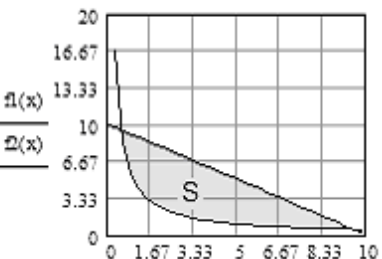
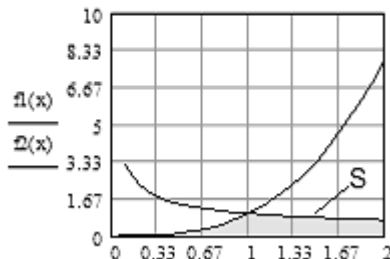
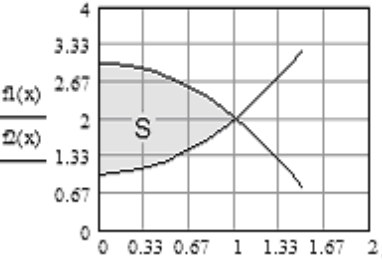
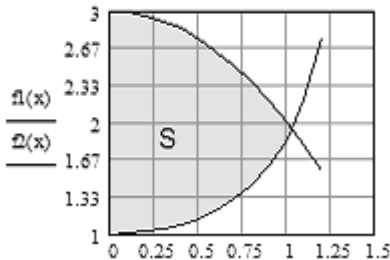
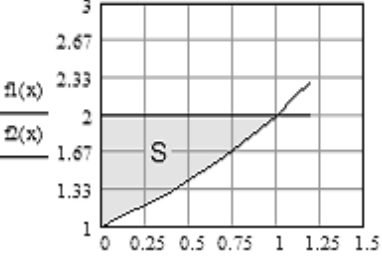
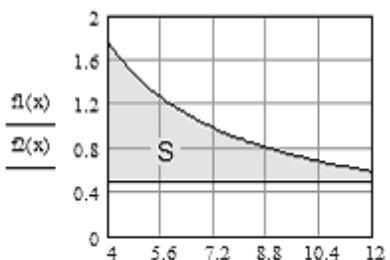
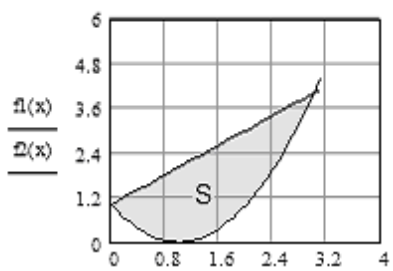
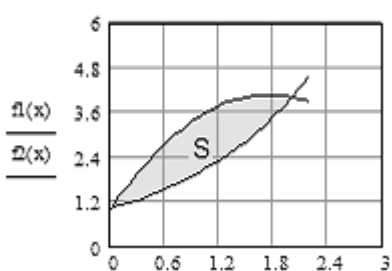
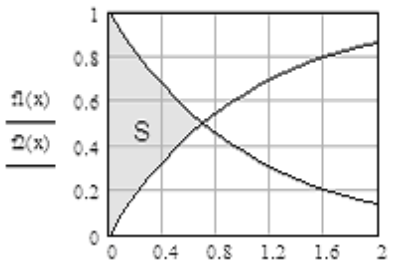
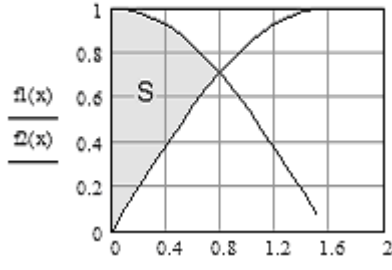
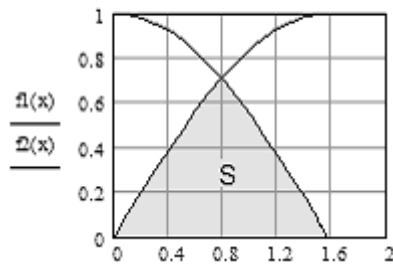
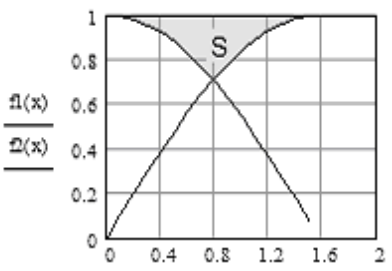
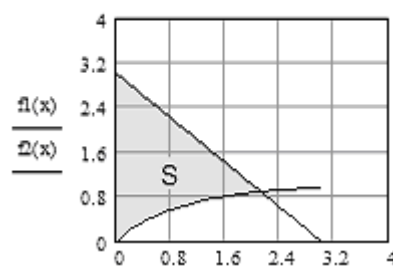
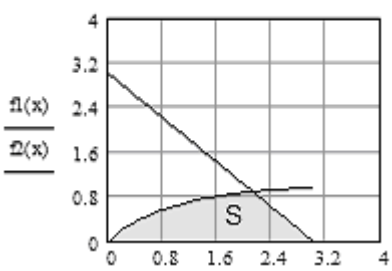


Рис. 3.4. Результат роботи програми

### Завдання до задачі 1

№	Завдання	Цикл	№	Завдання	Цикл
1	2	3	4	5	6
1	$f1(x) = -2 \cdot x^2 + 3 \cdot x + 6$ $f2(x) = x + 2$ 	for	11	$f1(x) = x^2 + x$ $f2(x) = x + 1$ 	do while
2	$f1(x) = 20 - 2 \cdot x^2 - 6 \cdot x$ $f2(x) = 0$ 	while	12	$f1(x) = x^2$ $f2(x) = \frac{1}{x}$ 	while

3	$f1(x) = \left(\frac{1}{2}\right)^x$ $f2(x) = 1 + \frac{x}{2}$ 	do while	13	$f1(x) = 4 \cdot x - x^2$ $f2(x) = x$ 	for
4	$f1(x) = \frac{5}{x}$ $f2(x) = 10 - x$ 	for	14	$f1(x) = x^3$ $f2(x) = \frac{1}{x^{0.5}}$ 	for
5	$f1(x) = x^2 + 1$ $f2(x) = 3 - x^2$ 	while	15	$f1(x) = \frac{1}{\cos(x)}$ $f2(x) = 3 - x^2$ 	while
6	$f1(x) = 2^x$ $f2(x) = 2$ 	do while	16	$f1(x) = \frac{7}{x}$ $f2(x) = 0.5$ 	do while

7	$f1(x) = (x - 1)^2$ $f2(x) = x + 1$	while	17	$f1(x) = 1 - x^2 + \frac{7 \cdot x}{2}$ $f2(x) = 2^x$	while
					
8	$f1(x) = e^{-x}$ $f2(x) = 1 - e^{-x}$	do while	18	$f1(x) = \sin(x)$ $f2(x) = \cos(x)$	do while
					
9	$f1(x) = \sin(x)$ $f2(x) = \cos(x)$	for	19	$f1(x) = \sin(x)$ $f2(x) = \cos(x)$	for
					
10	$f1(x) = 1 - e^{-x}$ $f2(x) = -x + 3$	while	20	$f1(x) = 1 - e^{-x}$ $f2(x) = -x + 3$	while
					

## Задача 2. Обчислення функції за допомогою розкладання в ряд.

Обчислити й вивести на екран у вигляді таблиці значення функції, заданої за допомогою ряду Тейлора, на інтервалі від  $x_{\text{нач}}$  до  $x_{\text{кон}}$  з кроком  $dx$  з точністю  $\varepsilon$ . Таблицю забезпечити заголовком і шапкою. Кожен рядок таблиці повинен містити значення аргументу, значення функції та кількість підсумованих членів ряду.



### Варіант 1

$$\ln\left(\frac{x+1}{x-1}\right) = 2 \cdot \sum_{n=0}^{\infty} \frac{1}{(2 \cdot n + 1) \cdot x^{2 \cdot n + 1}} = 2 \cdot \left( \frac{1}{x} + \frac{1}{3 \cdot x^3} + \frac{1}{5 \cdot x^5} + \dots \right)$$

$|x| > 1$

### Варіант 2

$$e^{-x} = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot x^n}{n!} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x^4}{4!} - \dots$$

$|x| < \infty$

### Варіант 3

$$\ln(x) = 2 \cdot \sum_{n=0}^{\infty} \frac{(x-1)^{2 \cdot n + 1}}{(2 \cdot n + 1) \cdot (x+1)^{2 \cdot n + 1}} = 2 \cdot \left[ \frac{x-1}{x+1} + \frac{(x-1)^3}{3 \cdot (x+1)^3} + \frac{(x-1)^5}{5 \cdot (x+1)^5} + \dots \right]$$

$x > 0$

### Варіант 4

$$\ln(x+1) = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot x^{n+1}}{n+1} = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \dots$$

$-1 < x \leq 1$

### Варіант 5

$$\ln\left(\frac{1+x}{1-x}\right) = 2 \cdot \sum_{n=0}^{\infty} \frac{x^{2 \cdot n + 1}}{2 \cdot n + 1} = 2 \cdot \left( x + \frac{x^3}{3} + \frac{x^5}{5} + \dots \right)$$

$|x| < 1$

### Варіант 6

$$\ln(1-x) = - \sum_{n=1}^{\infty} \frac{x^n}{n} = - \left( x + \frac{x^2}{2} + \frac{x^4}{4} + \dots \right)$$

$-1 \leq x < 1$

### Варіант 7

$$\operatorname{acot}(x) = \frac{\pi}{2} + \sum_{n=0}^{\infty} \frac{(-1)^{n+1} \cdot x^{2 \cdot n + 1}}{2 \cdot n + 1} = \frac{\pi}{2} - x + \frac{x^3}{3} - \frac{x^5}{5} + \dots$$

$|x| \leq 1$

### Варіант 8

$$\operatorname{atan}(x) = \frac{\pi}{2} + \sum_{n=0}^{\infty} \frac{(-1)^{n+1}}{(2 \cdot n + 1) \cdot x^{2 \cdot n + 1}} = \frac{\pi}{2} - \frac{1}{x} + \frac{1}{3 \cdot x^3} - \frac{1}{5 \cdot x^5} + \dots$$

$x > 1$

### Варіант 9

$$\operatorname{atan}(x) = \sum_{n=0}^{\infty} \frac{(-1) \cdot x^{2 \cdot n + 1}}{2 \cdot n + 1} = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \dots$$

$|x| \leq 1$

### Варіант 10

$$\operatorname{atanh}(x) = \sum_{n=0}^{\infty} \frac{x^{2 \cdot n + 1}}{2 \cdot n + 1} = x + \frac{x^3}{3} + \frac{x^5}{5} + \frac{x^7}{7} + \dots$$

$|x| < 1$

### Варіант 11

$$\operatorname{atanh}(x) = \sum_{n=0}^{\infty} \frac{1}{(2 \cdot n + 1) \cdot x^{2 \cdot n + 1}} = \frac{1}{x} + \frac{1}{3 \cdot x^3} + \frac{1}{5 \cdot x^5} + \dots$$

$|x| > 1$

### Варіант 12

$$\operatorname{acot}(x) = \frac{-\pi}{2} + \sum_{n=0}^{\infty} \frac{(-1)^{n+1}}{(2 \cdot n + 1) \cdot x^{2 \cdot n + 1}} = \frac{-\pi}{2} - \frac{1}{x} + \frac{1}{3 \cdot x^3} - \frac{1}{5 \cdot x^5} + \dots$$

$x < -1$

### Варіант 13

$$e^{-x^2} = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot x^{2 \cdot n}}{n!} = 1 - x^2 + \frac{x^4}{2!} - \frac{x^6}{3!} + \frac{x^8}{4!} - \dots$$

$|x| < \infty$

### Варіант 14

$$\cos(x) = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot x^{2 \cdot n}}{(2 \cdot n)!} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots$$

$|x| < \infty$

### Варіант 15

$$\frac{\sin(x)}{x} = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot x^{2 \cdot n}}{(2 \cdot n + 1)!} = 1 - \frac{x^2}{3!} + \frac{x^4}{5!} - \frac{x^6}{7!} + \dots \quad |x| < \infty$$

### Варіант 16

$$\ln(x) = 2 \cdot \sum_{n=0}^{\infty} \frac{(x-1)^{2 \cdot n+1}}{(2 \cdot n + 1) \cdot (x+1)^{2 \cdot n+1}} = 2 \cdot \left[ \frac{x-1}{x+1} + \frac{(x-1)^3}{3 \cdot (x+1)^3} + \frac{(x-1)^5}{5 \cdot (x+1)^5} + \dots \right] \quad x > 0$$

### Варіант 17

$$\ln(x) = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot (x-1)^{n+1}}{n+1} = (x-1) - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} - \dots \quad 0 < x \leq 2$$

### Варіант 18

$$\ln(x) = \sum_{n=0}^{\infty} \frac{(x-1)^{n+1}}{(n+1) \cdot (x+1)^{n+1}} = \frac{x-1}{x} + \frac{(x-1)^2}{2 \cdot x^2} + \frac{(x-1)^3}{3 \cdot x^3} + \dots \quad x > \frac{1}{2}$$

### Варіант 19

$$\begin{aligned} \operatorname{asin}(x) &= x + \sum_{n=1}^{\infty} 1 \cdot 3 \cdot \dots \cdot \frac{2 \cdot n - 1}{2 \cdot 4 \cdot \dots \cdot 2 \cdot n \cdot (2 \cdot n)} \cdot x^{2 \cdot n+1} = x + \\ &+ \frac{x^3}{2 \cdot 3} + \frac{1 \cdot 3 \cdot x^5}{2 \cdot 4 \cdot 5} + \frac{1 \cdot 3 \cdot 5 \cdot x^7}{2 \cdot 4 \cdot 6 \cdot 7} + \frac{1 \cdot 3 \cdot 5 \cdot 7 \cdot x^9}{2 \cdot 4 \cdot 6 \cdot 8 \cdot 9} + \dots \end{aligned} \quad |x| < 1$$

### Варіант 20

$$\begin{aligned} \operatorname{acos}(x) &= \frac{\pi}{2} - \left[ x + \sum_{n=1}^{\infty} \frac{1}{2 \cdot 4 \cdot \dots \cdot 2 \cdot n \cdot (2 \cdot n + 1)} \cdot 3 \cdot \dots \cdot (2 \cdot n - 1) \cdot x^{2 \cdot n+1} \right] = \\ &= \frac{\pi}{2} - \left( x + \frac{x^3}{2 \cdot 3} + \frac{1 \cdot 3 \cdot x}{2 \cdot 4 \cdot 5} + \frac{1 \cdot 3 \cdot 5 \cdot x^7}{2 \cdot 4 \cdot 6 \cdot 8} + \frac{1 \cdot 3 \cdot 5 \cdot 7 \cdot x^9}{2 \cdot 4 \cdot 6 \cdot 8 \cdot 9} + \dots \right) \end{aligned} \quad |x| < 1$$

**Здача 3 (на оцінку 12).** Скласти програму обчислення коріння рівняння  $f(x)=0$  з точністю  $EPS = 0,0001$ .

Інтервал локалізації кореня  $[a, b]$  відомий.

Використовувати:

метод простої ітерації;

метод половинного ділення відрізка  $[a, b]$  локалізації кореня;

метод Ньютона.

При складанні програми використовувати операторів циклу `while` і `do while`.

Примітка:

1. Метод простої ітерації полягає в тому, що за  $i$ -м наближенням кореня знаходиться  $i+1$  наближення за формулою:

$$x_{i+1} = f^{-1}(f(x_i)) \quad i = 0, 1, 2, \dots$$

Процес продовжується до тих пір, поки відносна помилка для двох послідовних наближень не стане менша  $EPS$ :

$$\left| \frac{x_{i+1} - x_i}{x_i} \right| < EPS.$$

Процес ітерації сходиться на  $[a, b]$ , якщо  $|f'(x)| < 1$  при всіх  $x \in [a, b]$ .

Для початкового значення кореня  $x_0$  прийняти  $x_0 = \frac{a+b}{2}$ .

2. Метод половинного ділення.

Завжди локалізований корінь знаходиться в тому інтервалі, на кінцях якого значення функції мають протилежні знаки. Якщо потрібно визначити корінь з точністю  $EPS$ , то ділення інтервалу навпіл продовжують, поки його довжина не стане менша  $2 \cdot EPS$ . У цьому випадку середина останнього інтервалу дає значення кореня з потрібною точністю.

3. Метод Ньютона полягає в послідовному обчисленні за формулою:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \quad i = 0, 1, 2, \dots$$

Оцінка помилки  $k$ -го наближення кореня виконується так:

$$|f(x_k)| < EPS.$$

Початкове наближення кореня можна узяти дорівнюючим  $x_0 = \frac{a+b}{2}$ .

Знайти корінь рівняння за допомогою всіх трьох методів і порівняти отримані результати, для чого підставити набуте значення в рівняння. Обчислити кількість ітерацій, необхідних для знаходження кореня для кожного методу.

1.  $f(x) = x - \sin x - 0,25 = 0$  (0; 2)
2.  $x - 2,89 \sin \frac{x}{3} - 0,126 = 0$  (1; 2)
3.  $5x - 8 \ln x - 8 = 0$  (3; 4)
4.  $0,03125 x^5 - 0,75 x^2 - x + 4,005 = 0$  (1,5; 2,7)
5.  $e^{-x} - x = 0$  (0,1; 0,6)
6.  $x^3 + x - 1 = 0$  (0,5; 0,9)
7.  $\sin x + x - 1 = 0$  (0; 1)
8.  $\operatorname{tg} x - e^{-x} + x - 1 = 0$  (0,5; 1)
9.  $0,5 \ln \frac{1+x}{1-x} + x - 1 = 0$  (0,1; 0,8)
10.  $\cos 2x + 0,165 x^3 - x = 0$  (0,1; 0,8)
11.  $\operatorname{tg} \frac{x+1}{2} + x - 2 = 0$  (0,2; 1)
12.  $0,5e^x + 0,5e^{-x} + x - 5 = 0$  (1,5; 2)
13.  $\operatorname{arctg} x - x^2 = 0$  (0,5; 1)
14.  $\operatorname{arctg} x + x - 1 = 0$  (0; 1)
15.  $e^{-x^2} - x = 0$  (0,5; 1)
16.  $x^6 + 0,008 x^3 + 720 x^2 + 720 x - 720 = 0$  (1; 2)
17.  $e^{-x} - x^2 + x = 0$  (0,5; 1,5)
18.  $\ln \left( x + \sqrt{x^2 + 1} \right) + x - 2 = 0$  (0; 2)
19.  $e^{-x} - x^2 + x = 0$  (1; 2)
20.  $0,5e^x - 0,5e^{-x} + x - 2,3 = 0$  (0; 2)

### **Контрольні запитання**

1. Що таке циклічний обчислювальний процес?
2. Які оператори використовуються для організації циклів?
3. Опишіть ситуації, коли слід використовувати кожен з трьох операторів циклу.
4. Які три операції повинні проводитися в програмі при організації будь-якого циклу?
5. Охарактеризуйте переваги, які дає цикл for.

### **Лабораторна робота 4**

#### **Підготовка й розв'язання на ПЕОМ завдань з використанням функцій**

**Мета лабораторної роботи** – вивчення особливостей застосування функцій при вирішенні типових економічних завдань.

Після виконання лабораторної роботи студент повинен знати:

структури оголошення (прототипи) і визначення функцій;

механізми передачі аргументів у функцію: за значенням, по посиланню, по покажчику;

механізми повернення функцією декількох значень.

Після виконання лабораторної роботи студент повинен уміти:

застосовувати правила розробки й використання функцій різних типів при вирішенні типових економічних завдань;

створювати власні заголовні файли з визначенням призначених для користувача функцій.

### **Короткі теоретичні відомості**

*Функція – це іменована послідовність описів і операторів, що виконує яку-небудь закінчену дію. Функція може приймати параметри і повертати значення.*

Будь-яка програма на C++ складається з функцій, одна з яких повинна мати ім'я main (з неї починається виконання програми). Функція починає виконуватися в момент виклику. Будь-яка функція має бути оголошена та визначена. Як і для інших величин, оголошень може бути декілька, а визначення тільки одне.

Оголошення функції повинне знаходитися в тексті раніше її виклику для того, щоб компілятор міг здійснити перевірку правильності виклику.

*Оголошення функції (прототип, заголовок, сигнатура) задає її ім'я, тип повернутого значення і список параметрів.*

*Визначення функції містить, окрім оголошення, тіло функції, що є послідовністю операторів і описів у фігурних дужках:*

```
[ клас ] тип ім'я ( [ список_параметрів ] )  
{  
    // тіло функції  
}
```

Розглянемо складові частини визначення.

За допомогою необов'язкового модифікатора клас можна явно задавати зону видимості функції, використовуючи ключові слова `extern` і `static`:

`extern` – глобальна видимість у всіх модулях програми (за замовчуванням);

`static` – видимість тільки в межах модуля, в якому визначена функція.

Тип зворотного функцією значення може бути будь-яким, окрім масиву й функції (але може бути покажчиком на масив або функцію). Якщо функція не повинна повертати значення, вказується тип `void`.

Список параметрів визначає величини, які потрібно передати у функцію при її виклику. Елементи списку параметрів розділяються комами. Для кожного параметра, який передається у функцію, вказується його тип і ім'я (у оголошенні імена можна опускати).

*У визначенні, в оголошенні і при виклику однієї і тієї ж функції типи і порядок проходження параметрів повинні збігатися.*

На імена параметрів обмеження за відповідністю не накладаються, оскільки функцію можна викликати з різними аргументами, а в прототипах імена ігноруються компілятором (вони служать тільки для поліпшення читаності програми).

Функцію можна визначити, як вбудовану за допомогою модифікатора `inline`, який рекомендує компілятору замість звернення до функції поміщати її код безпосередньо в кожену точку виклику. Модифікатор `inline` ставиться перед типом функції. Він застосовується для коротких функцій, щоб понизити накладні витрати на виклик (збереження і відновлення регістрів, передача управління). Директива

inline носить рекомендаційний характер і виконується компілятором у міру можливості. Використання inline-функцій може збільшити об'єм виконуваної програми. Визначення функції повинне передувати її викликам, інакше замість inline-розширення компілятор згенерує звичайний виклик.

*Тип зворотного значення й типи параметрів спільно визначають тип функції.*

Для виклику функції в простому випадку потрібно вказати її ім'я, за яким у круглих дужках через кому перераховуються імена аргументів, які передаються. Виклик функції може знаходитися в будь-якому місці програми, де за синтаксисом допустимий вираз того типу, який формує функція. Якщо тип зворотного значення функцією не void, то вона може входити до складу виразу або, в окремому випадку, розташовуватися в правій частині оператора привласнення.

Нижче наведений приклад програми, що дозволяє знаходити максимальне число з трьох введених чисел, при цьому виклик функції здійснюється за її іменем (max3).

```
// Програма 4.1
// Застосування призначеної для користувача функції max3 для
пошуку
// максимального з трьох чисел
// Місце визначення функції – функція визначається
// безпосередньо у програмі (проте, можливий інший варіант //
визначення – у заголовному файлі)
// Виклик функції - за іменем
// Зворотне значення – використовується
// Метод передачі параметрів у функцію -- за значенням
#include <iostream.h>
#include <conio.h>
using namespace std;
double max3 (double, double, double);    // Оголошення прототипу
функції

int main ()
{
    clrscr();
```



```

double x, y, z;
cout << "\n ПОСЛІДОВНО ВВЕДІТЬ ТРИ ЧИСЛА: x,y,z:  "
      << endl;
cin >> x; cin >> y; cin >> z;
cout << "\n ВИ ВВЕЛИ: x = " << x << "\t y = " << y << "\t z = " << z ;
cout << "\n ТОМУ  max (x,y,z) = " << max3(x,y,z);
//                                     |      |-> фактичні параметри
//                                     |-> ім'я функції

return 0;
}
//    Визначення функції max3
double  max3 (double a, double b, double z)
//      -----
//      |-> формальні параметри
{
    double max = a;
    if ( b > max )
        max = b;
    if (z > max )
        max = z;
return max;
}

```

Один з можливих результатів виконання програми може мати наступний вигляд:

**ПОСЛІДОВНО ВВЕДІТЬ ТРИ ЧИСЛА: x,y,z:**

**0.23**

**4**

**-5.7**

**ВИ ВВЕЛИ: x = 0.23      y = 4      z = -5.7**

**ТОМУ      max (x,y,z) = 4**

*Особливості виконання функції*

Усі величини, описані всередині функції, а також її параметри, є локальними. Зоною їх дії є функція. При виклику функції, як і при вході в будь-який блок, в стеку виділяється пам'ять під локальні автоматичні змінні. Крім того, в стеку зберігається вміст регістрів процесора на момент, передуючий виклику функції, і адреса повернення з функції для

того, щоб при виході з неї можна було продовжити виконання зухвалої функції.

При виході з функції відповідна ділянка стека звільняється, тому значення локальних змінних між викликами однієї і тієї ж функції не зберігаються. Якщо цього потрібно уникнути, при оголошенні локальних змінних використовується модифікатор `static`. Наступна програма ілюструє застосування статичної локальної змінної в тілі функції.

// Програма 4.2

// Застосування статичної локальної змінної в тілі функції

```
#include <iostream.h>
using namespace std;
void f (int a) // визначення функції, що формує рядки з трьох
               // колонок з іменами  n, m, p
{
    int m=0;
    cout << "\nn m p\n\n"; // виведення найменувань колонок
    while (a--)           // формування рядків
    {
        static int n=0;
        int p=0;
        cout << n++ << ' ' << m++ << ' ' << p++ << "\n";
    }
}
int main()
{
    f(3);
    f(2);
    return 0;
}
```

Статична змінна `n` розміщується в сегменті даних і ініціалізувалася один раз при першому виконанні оператора, що містить її визначення. Автоматична змінна `m` ініціалізувалася при кожному вході у функцію. Автоматична змінна `p` ініціалізувалася при кожному вході в блок циклу.

Програма виведе на екран:

**n m p**

**0 0 0**

**1 1 0**

**2 2 0**

**n m p**

**3 0 0**

**4 1 0**

### *Зворотне функцією значення*

Механізм повернення з функції у функцію, що викликала її, реалізується оператором

`return [ вираз ];`

Якщо функція описана як `void`, вираз не указується. Форма

`return ;`

використовується для дострокового виходу з функції.

Оператора `return` можна опускати для функції типу `void`, якщо повернення з неї відбувається перед закриваючою фігурною дужкою, і для функції `main`.

Вираз, вказаний після `return`, неявно перетвориться до типу зворотного функцією значення і передається в точку виклику функції.

Функція може містити декілька операторів `return` (це визначається потребами алгоритму). Приклад використання декількох операторів `return` наведений в програмі 4.3.

`// Програма 4.3`

`// Використання декількох операторів return в тілі функції`

`#include <iostream.h>`

`using namespace std;`

`int Doubler(int) // оголошення функції, що подвоює значення`  
`// аргументу`

`int main()`

`{`

`int result = 0;`

```

int input;
cout << "Введіть число між 0 і 10 : ";
cin >> input;
cout << "\n Перед викликом функції Doubler... ";
cout << "\n Ваше число: " << input << " Результат: " << result << "\n";
result = Doubler (input);
cout << "\n Після виклику функції Doubler... ";
cout << "\n Ваше число: " << input << " Результат: " << result << "\n";
return 0;
}
int Doubler ( int x ) // Визначення функції Doubler
{
    if ( x <= 10 )
        return x * 2;
    else
        return -1;
}

```

Результат роботи програми:

**Введіть число між 0 і 10: 3**

**Перед викликом функції Doubler...**

**Ваше число: 3 Результат: 0**

**Після виклику функції Doubler...**

**Ваше число: 3 Результат: 6**

**Введіть число між 0 і 10: 37**

**Перед викликом функції Doubler...**

**Ваше число: 37 Результат: 0**

**Після виклику функції Doubler...**

**Ваше число: 37 Результат: -1**

*Обмін інформацією між функціями*

При спільній роботі функції повинні обмінюватися інформацією. Це можна здійснити:

за допомогою глобальних змінних;

через параметри функції;

через зворотні функцією значення.

*Обмін інформацією за допомогою глобальних змінних*

Глобальні змінні видно в усіх функціях, де не описані локальні змінні з тими ж іменами, тому використовувати їх для передачі даних між функціями дуже легко. Проте, це не рекомендується, оскільки утрудняє відладку програми і перешкоджає переміщенню функцій в бібліотеки загального користування. Потрібно прагнути до того, щоб функції були максимально незалежні, а їх інтерфейс повністю визначався прототипом функції.

### **Використання параметрів функції для обміну інформацією між функціями**

Механізм параметрів є основним способом обміну інформацією між функціями, що викликаються і викликають. Параметри, перераховані в заголовку опису функції, називаються формальними, а записані в операторі виклику функції — фактичними.

При виклику функції насамперед обчислюються вирази, що стоять на місці фактичних параметрів; потім в стеку виділяється пам'ять під формальні параметри функції відповідно до їх типу, і кожному з них привласнюється значення відповідного фактичного параметра. При цьому перевіряється відповідність типів і при необхідності виконуються їх перетворення. При невідповідності типів видається діагностичне повідомлення.

Існує два способи передачі параметрів у функцію: за значенням і за адресою.

*При передачі за значенням в стек заносяться копії значень фактичних параметрів і оператори функції працюють з цими копіями. Доступу до початкових значень параметрів у функції немає, а, отже, немає і можливості їх змінити.*

*При передачі за адресою в стек заносяться копії адрес параметрів, а функція здійснює доступ до елементів пам'яті за цими адресами і може змінити початкові значення параметрів.*

Розглянемо приклад програми, що ілюструє перераховані вище способи передачі параметрів у функцію:

```
// Програма 4.4
// Способи передачі параметрів у функцію
#include <iostream.h>
using namespace std;
void f(int i, int* j, int& до);
int main()
```

```

{
    int i = 1, j = 2, до = 3;
    cout << "i j k\n";
    cout << i << " " << j << " " << до << endl;
    f(i &j, до);
    cout << i << " " << j << " " << до;
    return 0;
}
void f (int i, int* j, int& до)
{
    i++; (*j)++;    k++;
}

```

Результат роботи програми:

```

i j k
1 2 3
1 3 4

```

Перший параметр (i) передається за значенням. Його зміна у функції не впливає на початкове значення. Другий параметр (j) передається за адресою за допомогою покажчика, при цьому для передачі у функцію адреси фактичного параметра використовується операція узяття адреси, а для набуття його значення у функції потрібна операція розіменування. Третій параметр (k) передається за адресою за допомогою посилання.

При передачі за посиланням у функцію передається адреса вказаного при виклику параметра, а всередині функції всі звернення до параметра неявно "розіменовуються". Тому використання посилань замість покажчиків покращує читаність програми, позбавляючи від необхідності застосовувати операції отримання адреси й розіменування. Використання посилань замість передачі за значенням ефективніше, оскільки не вимагає копіювання параметрів, що має значення при передачі структур даних великого об'єму.

Якщо потрібно заборонити зміну параметра усередині функції, використовується модифікатор `const`, наприклад:

```
int f(const char*);
```

Рекомендується указувати `const` перед усіма параметрами, зміна яких у функції не передбачена. Це полегшує відладку великих програм, оскільки за заголовком функції можна зробити висновок про те, які величини в ній змінюються, а які – ні. Крім того, на місце параметра типу `const&` може передаватися константа, а для змінної при необхідності виконуються перетворення типу.

Таким чином, початкові дані, які не повинні змінюватися у функції, переважно передаються їй за допомогою константних посилань. За замовчуванням параметри будь-якого типу, окрім масиву і функції (наприклад, речового, структурного, перерахування, об'єднання, покажчик), передаються у функцію за значенням.

### **Перевантаження функцій**

У мові C++ передбачена можливість створення декількох функцій з однаковим ім'ям. Це називається перевантаженням функцій. Переобтяжені функції повинні відрізнятися один від одного списками параметрів: або типом одного або декількох параметрів, або різною кількістю параметрів, або і тим і іншим одночасно. Розглянемо наступний приклад:

```
int myFunction (int, int);  
int myFunction (long, long);  
int myFunction (long);
```

Функція `myFunction ()` переобтяжена за трьома різними списками параметрів. Перша і друга версії відрізняються типами параметрів, а третя — їх кількістю.

Типи зворотних значень переобтяжених функцій можуть бути однаковими або різними. Слід мати на увазі, що при створенні двох функцій з однаковим ім'ям і однаковим списком параметрів, але з різними типами зворотних значень, згенерує помилка компіляції.

Перевантаження функцій також називається поліморфізмом функцій. Полі (гр. *poly*) означає багато, морфе (гр. *morphe*) — форма, тобто поліморфічна функція — це функція, що відрізняється різноманіттям форм.

Під поліморфізмом функції розуміють існування в програмі декількох переобтяжених версій функції, що мають різні призначення. Змінюючи кількість або тип параметрів, можна привласнити двом або декільком функціям одне і те ж ім'я. При цьому ніякої плутанини при виклику функцій не буде, оскільки потрібна функція визначається за

збігом використовуваних параметрів. Це дозволяє створити функцію, яка зможе, наприклад, усереднювати цілочисельні значення, значення типу `double` або значення інших типів без необхідності створювати окремі імена для кожної функції — `Averageints()`, `AverageDoubles()` і т. д.

Припустимо, потрібно створити функцію, яка подвоює будь-яке передаване нею значення. При цьому хотілося б мати можливість передавати їй значення типу `int`, `long`, `float` або `double`. Без перевантаження функцій довелося б створювати чотири різні функції:

```
int Doubleint(int);
long DoubleLong(long);
float DoubleFloat(float);
double DoubleDouble(double);
```

За допомогою перевантаження функцій можна використовувати наступні оголошення:

```
int Double(int);
long Double(long);
float Double(float);
double Double(double);
```

Завдяки використанню переобтяжених функцій не потрібно турбуватися про виклик у програмі потрібної функції, що відповідає типу передаваних змінних. При виклику переобтяженої функції компілятор автоматично визначить, який саме варіант функції слід використовувати. Приклад використання переобтяжених функцій показаний нижче.

```
// Програма 4.5
// Приклад поліморфізму функцій
// Чотири однойменні функції подвоюють значення аргументу
//кожного з чотирьох типів
#include <iostream.h>
using namespace std;
int Double (int);      // Прототипи однойменних функцій
long Double (long);
float Double (float);
double Double (double);

int main()
{
```



```

int myInt    = 6500;  // Початкові дані різних типів
long myLong  = 6500;
float myFloat = 6.5F;
double myDouble = 6.5e20;
int doubledInt;      // Оголошення змінних для результату
long doubledLong;
float doubledFloat;
double doubledDouble;

cout << "ПОЧАТКОВІ Дані:\n";
cout << "myInt: "  << myInt  << "\n";  // Виведення початкових
                                         //даних
cout << "myLong: "  << myLong  << "\n";
cout << "myFloat: " << myFloat << "\n";
cout << "myDouble: " << myDouble << "\n\n";

cout << Функції:\n, що "ВИКЛИКАЮТЬСЯ";
doubledInt  = Double (myInt);  // Виклики однойменних функцій
doubledLong = Double (myLong);
doubledFloat = Double (myFloat);
doubledDouble = Double (myDouble);

    cout << "ВИВЕДЕННЯ Результатів:\n";
cout << "doubledInt: "  << doubledInt  << "\n";      // Виведення
//результатів
    cout << "doubledLong: "  << doubledLong  << "\n";
    cout << "doubledFloat: " << doubledFloat << "\n";
    cout << "doubledDouble: " << doubledDouble << "\n";

    return 0;
}

int Double (int original)
{
    cout << "ФУНКЦІЯ Double(int)\n";
    return 2 * original;
}

```

```

long Double (long original)
{
    cout << "ФУНКЦІЯ Double(long)\n";
    return 2 * original;
}
float Double (float original)
{
    cout << "ФУНКЦІЯ Double(float)\n";
    return 2 * original;
}
double Double (double original)
{
    cout << "ФУНКЦІЯ Double(double)\n\n";
    return 2 * original;
}

```

Результат роботи програми:

**Вихідні дані:**

**myInt: 6500**

**myLong: 6500**

**myFloat: 6.5**

**myDouble: 6.5e+20**

**Функції, що викликаються:**

**ФУНКЦІЯ Double(int)**

**ФУНКЦІЯ Double(long)**

**ФУНКЦІЯ Double(float)**

**ФУНКЦІЯ Double(double)**

**Виведення результатів:**

**doubledInt: 13000**

**doubledLong:13000**

**doubledFloat: 13**

**doubledDouble: 1.3e+21**

Функція Double() перевантажується для прийому чотирьох типів: int, long, float і double. За виглядом виклику функції Double() нічим не відрізняються один від одного. Проте, компілятор визначає тип

переданого аргументу, на підставі якого вибирає відповідний варіант функції. Результат роботи цієї програми підтверджує очікувану черговість виклику варіантів цієї переобтяженої функції.

### **Параметри функції, використовувані за замовчуванням**

Для кожного параметра, що оголошується в прототипі і визначенні функції, має бути передане відповідне значення у виклику функції. Значення, яке передається, повинне мати оголошений тип. Отже, якщо деяка функція оголошена як

```
long myFunction(int);
```

то вона дійсно повинна набувати цілочисельного значення. Якщо тип оголошеного параметра не збіжиться з типом передаваного аргументу, компілятор повідомить про помилку.

З цього правила існує одне виключення, яке набуває чинності, якщо в прототипі функції для параметра оголошується стандартне значення. Це значення, яке використовується в тому випадку, якщо при виклику функції для цього параметра не встановлено ніякого значення. Декілька змінимо попереднє оголошення:

```
long myFunction(int x = 50);
```

Цей прототип потрібно розуміти таким чином. Функція `myFunction(int)` повертає значення типу `long` і приймає параметр типу `int`. Але якщо при виклику цієї функції аргумент наданий не буде, використовуйте замість нього число 50. А оскільки в прототипах функцій імена параметрів не обов'язкові, то останній варіант оголошення можна переписати по-іншому:

```
long myFunction (int = 50);
```

Визначення функції не змінюється при оголошенні значення параметра, що задається за замовчуванням. Тому заголовок визначення цієї функції виглядатиме як і раніше:

```
long myFunction (int x);
```

Якщо при виклику цієї функції аргумент не встановлюється, то компілятор привласнить змінною `x` значення 50. Ім'я параметра, для якого в прототипі встановлюється значення за замовчуванням, може не збігатися з ім'ям параметра, що вказується в заголовку функції: значення, задане за замовчуванням, привласнюється за позицією, а не за ім'ям.

Установку значень за замовчуванням можна призначити будь-яким або всім параметрам функції. Але одне обмеження все ж таки діє: якщо якийсь параметр не має стандартного значення, то жоден з попередніх по відношенню до нього параметрів також не може мати стандартного значення.

Припустимо, прототип функції має вигляд:

```
long rnyFunction (int Param1, int Param2, int Param3);
```

тоді параметру Param2 можна призначити стандартне значення тільки в тому випадку, якщо призначено стандартне значення і параметру Param3. Параметру Param1 можна призначити стандартне значення тільки в тому випадку, якщо призначені стандартні значення як параметру Param2, так і параметру Param3, Приклад використання значень, що задаються параметрам функцій за замовчуванням, показаний в програмі 4.6.

```
// Програма 4.6
```

```
// Використання стандартних значень параметрів (за замовчуванням)
```

```
#include <iostream.h>
```

```
using namespace std;
```

```
int VolumeCube(int length, int width = 25, int height = 1);
```

```
int main()
```

```
{
```

```
    int length = 100;
```

```
    int width  = 50;
```

```
    int height = 2;
```

```
    int volume;
```

```
    volume = VolumeCube ( length, width, height );
```

```
    cout << "Перший об'єм дорівнює: " << volume << "\n";
```

```
    volume = VolumeCube ( length, width );
```

```
    cout << "Другий об'єм дорівнює: " << volume << "\n";
```

```
    volume = VolumeCube ( length );
```

```
    cout << "Третій об'єм дорівнює:" << volume << "\n";
```

```
    return 0;
```

```
}
```

```
VolumeCube (int length, int width, int height )  
{  
    return (length * width * height);  
}
```

Результат роботи програми:

**Перший об'єм дорівнює: 10000**

**Другий об'єм дорівнює: 5000**

**Третій об'єм дорівнює: 2500**

У прототипі функції VolumeCube() оголошується, що функція приймає три параметри, причому останні два мають значення, що встановлюються за замовчуванням. Ця функція обчислює об'єм паралелепіпеда на підставі переданих розмірів. Якщо значення ширини не передане, то ширина встановлюється рівною 25, а висота — 1. Якщо значення ширини передане, а значення висоти немає, то за замовчуванням встановлюється тільки значення висоти. Але не можна передати у функцію значення висоти без передачі значення ширини.

У подальших рядках ініціалізувалися змінні, призначені для зберігання розмірів паралелепіпеда по довжині, ширині і висоті. Ці значення передаються функції VolumeCube(). Після обчислення об'єму паралелепіпеда результат виводиться у відповідному рядку. Далі функція VolumeCube() викликається знову, але без передачі значення для висоти. У цьому випадку для обчислення об'єму паралелепіпеда використовується значення висоти, задане за замовчуванням, і отриманий результат виводиться в іншому рядку.

При третьому виклику функції VolumeCube() не передається ні значення ширини, ні значення висоти. Тому замість них використовуються значення, задані за замовчуванням, і отриманий результат також виводиться на екран.

### **Створення власних заголовних файлів**

Програміст може сам створювати потрібні йому заголовні файли з визначенням власних функцій.

Заголовні файли містять оголошення й визначення, загальні для різних програмних файлів, і тому часто створюються і включаються у файли програм директивою компілятора #include. Як такі оголошення і визначення виступають класи, структури, об'єднання, типи, що перераховують, і прототипи функцій або їх визначення.

Директива `#include` використовується для включення копії вказаного в директиві файлу в те місце, де знаходиться ця директива. Найчастіше вона застосовується для включення в текст копії заголовного файлу.

Існують три форми директиви:

```
#include <ім'я_заголовного_файлу>
```

```
#include "ім'я_заголовного_файлу"
```

```
#include ідентифікатор_макросу .
```

Відмінність між першими двома формами директиви полягає в методі пошуку препроцесором файлу, що включається.

1. Якщо ім'я файлу поміщене в кутові дужки, то послідовність пошуку препроцесором заданого файлу в каталогах визначається заданими каталогами включення (`include directories`).

2. Якщо ж ім'я файлу поміщене в лапки, препроцесор шукає файл, проглядаючи каталоги в наступній послідовності:  
каталог того файлу, який містить директиву `#include`;

поточний каталог;

каталоги, вказані опцією компілятора.

Якщо ім'я файлу вказане з шляхом, то препроцесор ніде більше цей файл не шукає.

3. Третя форма директиви припускає наявність макросу, що визначає файл, що включається (у даній роботі ця форма не розглядається).

Як приклад, що ілюструє другу форму директиви `#include`, розглянемо програму, де оголошення призначених для користувача функцій `max3( )` і `fakt()` – обчислення факторіалу, винесено в окремо створений заголовний файл з ім'ям `u_head.h`, а визначення призначених для користувача функцій `max3( )` і `fakt()` – обчислення факторіалу, винесено в окремо створений файл з ім'ям `u_head.cpp`.

```
// Програма 4.7
```

```
// Створення призначеного для користувача заголовного файлу  
//u_head.cpp для визначення функції max3() – пошуку максимального з  
//трьох чисел і функції fakt() – розрахунок факторіалу аргументу
```

```
// Місце оголошення функцій – призначений для користувача  
//заголовний файл
```

```
// u_head.h
```

```

// Місце визначення функцій – призначений для користувача
//заголовний файл
// u_head.cpp
// Виклик функції – за ім'ям
// Зворотне значення – використовується
// Метод передачі параметрів у функцію – за значенням
// Призначений для користувача заголовний файл (u_head.h)
// Оголошення функції max3()
double max3(double x, double y, double z);
// Оголошення функції fakt()
int fakt(int x);

#include <iostream.h>
#include <conio.h>
#include "u_head.h" // Підключення призначене для користувача
                    // заголовного файлу u_head.h
                    // Повний шлях указувати
                    // не обов'язково (відлік ведеться з поточної директорії bin)
using namespace std;
int main ()
{
    clrscr();
    double a, b, z;
    int d;
    cout << "\n ПОСЛІДОВНО ВВЕДІТЬ ТРИ ЧИСЛА: a,b,c:  " << endl;
    cin >> a; cin >> b; cin >> z;
    cout << "\n ВИ ВВЕЛИ: a = " << a << "\t b = " << b << "\t z = " << z ;
    cout << "\n ТОМУ max (a,b,c) = " << max3(a,b,c);
    cout << "\n ВВЕДІТЬ ОДНЕ ЦІЛЕ ЧИСЛО: " << endl;
    cin >> d;
    cout << " \n ВИ ВВЕЛИ ЧИСЛО  " << d;
    cout << "\n\n ОБЧИСЛЕННЯ ФАКТОРІАЛУ ЧИСЛА: fakt(" << d
    << ") = " << fakt(d);

    return 0;
}

```

```
// Призначений для користувача файл (u_head.cpp)
// Визначення функції max3()
double max3(double x, double y, double z)
{
    double max=x;
    if (y > max)
        max = y;
    if (z > max)
        max = z;
    return max;
}
// Визначення функції fakt()
int fakt(int x)
{
    int f = 1;
    for (int i=1; i<=x; i++)
        f *= i;          // або f = f * i;
    return f;
}
```

Варіант результату роботи програми:

ПОСЛІДОВНО ВВЕДІТЬ ТРИ ЧИСЛА: a,b,c:

-1.37

8.3

0

ВИ ВВЕЛИ: a = -1.37 b = 8.3 c = 0

ТОМУ max (a,b,c) = 8.3

ВВЕДІТЬ ОДНЕ ЦІЛЕ ЧИСЛО 4

Ви ввели число:4

ОБЧИСЛЕННЯ ФАКТОРІАЛУ ЧИСЛА: fact(4) = 24

### **Завдання 2.1 (за варіантами).**

Використовуючи механізм перевантаження функції, розробити і відладити програму обчислення значення  $= f(x, y, z)$  для різних типів параметрів. Конкретний варіант функції узяти із завдання 2 Лр1. Передбачити завдання параметрів функції "за замовчуванням" і з клавіатури.

Як зразки використовувати фрагменти програм 4.1, 4.5, 4.6.



### **Завдання 2.2 (за варіантами).**

Розробити і відладати програму обчислення виразу  $y = f(x)$ , приведеного в додатку Б.

Кожній з трьох гілок обчислення значення у повинна відповідати окрема функція (fun\_1, fun\_2 і fun\_3). На 12 балів: Передача параметрів у функцію fun\_1 повинна здійснюватися за значенням, у функцію fun\_2 – використовуючи покажчики, а у функцію fun\_3 – на базі посилань.

Як зразки використовувати фрагменти програм 4.3 і 4.4.

### **Завдання 2.3 (за варіантами).**

Аналогічно завданню 2.2, проте всі оголошення функцій помістити в окремий заголовний файл з ім'ям `***.h`, а визначення функцій помістити в окремий файл з ім'ям `***.cpp`.

Як зразки використовувати фрагменти програм 4.3, 4.4 і 4.7.

### **Контрольні запитання**

1. Назвіть три умови, необхідні для використання функції.
2. Що таке прототип функції, для чого він потрібний?
3. Чим відрізняється оголошення функції від визначення функції?
4. Коли доцільно використовувати статичну змінну всередині функції?
5. Назвіть три способи передачі параметрів у функцію. Опишіть ситуації, коли слід використовувати конкретний чин. Наведіть приклади.
6. Що таке перевантаження функції, коли її доцільно використовувати?
7. Назвіть правила завдання параметрів функції за замовчуванням.
8. Як і в яких випадках створюються призначені для користувача заголовні файли?

## **Лабораторна робота 5**

### **Підготовка й розв'язання на ПЕОМ завдань обробки одновимірних масивів**

**Мета лабораторної роботи** – засвоєння методики обробки інформації в одновимірних масивах.

Перед виконанням лабораторної роботи студент повинен знати:  
визначення й правила опису одновимірних масивів різних типів;  
способи доступу до елементів масиву даних;  
способи ініціалізації елементів масиву;

способи роботи з одновимірними масивами;  
способи алгоритмізації пошуку мінімальних і максимальних значень в одновимірних масивах;  
узагальнені способи формування таблиць різної конфігурації за допомогою псевдосимволів.

Після виконання лабораторної роботи студент повинен уміти:  
виконувати ініціалізацію масивів різних типів і розмірності;  
використовувати засоби мови C++ для перетворення типів даних;  
розробляти і виконувати програми пошуку мінімальних і максимальних значень в одновимірних масивах.

### Короткі теоретичні відомості

Масив – це сукупність логічно взаємозв'язаних даних одного типу, об'єднаних загальним ім'ям. Масив в C++ — це набір однотипних даних (об'єктів), що мають загальне ім'я і що розрізняються місцеположенням в цьому наборі (або індексом, привласненим кожному елементу масиву).

У програмі можуть бути масиви цілих чисел або чисел з плаваючою точкою, символьні масиви і т. д.

#### Опис одновимірного масиву

Опис складається із специфікації типу, ідентифікатора й розмірності масиву (рис.5.1).

Звернення до елементів масиву виконується за їх індексами, які завжди починаються від нуля. Якщо, наприклад, оголосити в програмі масив на ім'я `imas` з 100 цілих чисел

```
int imas[100];
```

то найперший елемент масиву отримує позначення `imas[0]`, наступний – `imas[1]` і так далі до останнього елементу `imas[99]`.



Рис. 5.1. Ініціалізація масиву і доступ до елементів масиву

Так само можна оголосити масиви даних будь-яких інших типів:

```
char cSy[20];           //Массив з 20 символів  
float fWe[16];          //Массив з 16 чисел з плаваючою точкою.
```

Після оголошення масиву його можна заповнити числами, використовуючи індексного оператора [ ], наприклад:

```
int  myArray [ 3 ];      // Оголошення масиву  
    myArray [ 0 ] = -12; // Ініціалізація першого елемента (його  
//індекс дорівнює нулю) значенням -12  
    myArray [ 1 ] = 5;  
    myArray [ 2 ] = 37;
```

Доступ до окремих елементів масиву здійснюється також за допомогою індексного оператора:

```
int result = myArray [ 0 ] + myArray [ 1 ] + myArray [ 2 ];
```

Масив можна оголосити і заповнити одночасно, наприклад:

```
int  myArray [ 3 ] = { -12, 5, 37 } ;
```

Якщо розмірність задана, то число значень в списку ініціалізації не повинне її перевищувати.

Якщо розмірність масиву більше числа значень в списку, то елементи масиву, що не ініціалізували явно, будуть встановлені в 0, наприклад:

```
int myArr[ 5 ]= { 0, 1, 2 }; // myArr буде дорівнювати { 0, 1, 2, 0, 0 }
```

Значення розмірності має бути константним виразом, тобто розмірність має бути відома на етапі трансляції. Це означає, що змінна не може використовуватися для завдання розмірності масиву.

Якщо в масиві не дуже багато елементів і їх значення відомі заздалегідь, масив можна ініціалізувати разом з його оголошенням, уклавши перелік значень у фігурні дужки:

```
int iNs[10]={0,1,2,3,4,5,6,7,8,9};
```

Якщо ж масив великий, заповнити його доведеться програмно в циклі. Хай ми хочемо утворити масив з 100 елементів, заповнених натуральним рядом чисел. Це робиться таким чином:

```
int imas[100];  
for (int i=0;i<100;i++) imas[i]=i;
```

До елементів масиву можна звертатися й вибірково, наприклад:

```
test[50]=0x7FFA;
```

### Збереження одновимірних масивів

У C++ одновимірний масив логічно зберігається як послідовність впорядкованих елементів у пам'яті. Кожен елемент відноситься до одного й того ж типу даних. Наприклад, масив *A* розташовується в пам'яті таким чином (рис. 5.2):



Рис. 5.2. Послідовне розташування елементів масиву в пам'яті

У C++ ім'я масиву є константою і розглядається, як адреса першого елемента цього масиву. Так, в оголошенні

```
type A[arraySize];
```

ім'я масиву *A* є константою і визначається місцеположенням у пам'яті першого елемента  $A[0]$ . Елементи  $A[1]$ ,  $A[2]$  далі слідує за ним послідовно.

Допустимо, що  $\text{sizeof}(\text{type})=M$ , тоді весь масив *A* займає  $M \cdot \text{arraySize}$  байтів (рис. 5.3).

Компілятор задає таблицю, яка називається дескриптор масиву (dope vector), для ведення запису характеристик масиву.

Таблиця включає інформацію про розмір кожного елемента масиву, початкову адресу масиву й кількість елементів у цьому масиві:

Початкова адреса:	<i>A</i>
Кількість елементів масиву:	<i>arraySize</i>
Розмір типу:	$M = \text{sizeof}(\text{type})$

Ця таблиця використовується компілятором також для реалізації функції доступу (access function), яка визначає адресу елемента в пам'яті.

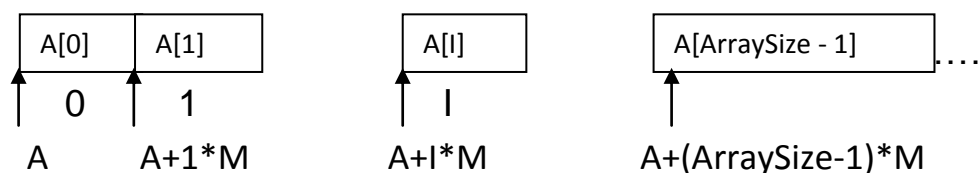


Рис. 5.3. Визначення розміру пам'яті, яку займає масив

### Приклад

Припустимо, що змінна типу `float` зберігається з використанням 4 байт (тобто `sizeof(float)= 4`) і масив `Macizo[ ]` починається в пам'яті за адресою 20000. Хай

```
float Macizo[18];
```

Тоді елемент масиву `Macizo[18]` розміщується за адресою:  
 $20000 + 18 \cdot 4 = 20072$ .

### Символьні масиви

Символьні масиви можуть становити набір окремих символів або зв'язний символьний рядок.

У першому випадку масив можна ініціалізувати так само, як і числовий, перерахуванням всіх його елементів:

```
char cSz[17]={'C','и','м','в','о', 'л','ь','н','и', 'й', ' ','м','а','с','и','в'};
```

Такий масив займає в пам'яті 17 байт.

У другому випадку масив ініціалізувався символьним рядком:

```
char cSz[20]="Символьний масив";
```

При такій ініціалізації компілятор записує в пам'ять в кінці рядка двійковий нуль, який служить символом-обмежувачем. Багато функцій для роботи з рядками (наприклад, копіювання рядка або виведення рядка на екран) за цим символом визначають, де закінчується рядок. Це позбавляє нас від необхідності визначати й указувати при виклику функції фактичну довжину рядка. Таким чином, масив займає в пам'яті на 1 байт більше, ніж у ньому є значущих символів. Цю обставину необхідно враховувати при завданні довжини масиву в його оголошенні, яка повинна вибиратися на одиницю більше максимально можливої довжини рядка.

Оголошення довжини символьного масиву із запасом доцільно лише в тих випадках, коли він заповнюватиметься програмно рядками різної довжини. Якщо ж даний текстовий рядок змінюватися не буде, зручніше оголосити її без зазначення довжини, залишивши в оголошенні пару квадратних дужок, які скажуть компілятору, що змінна `cSz` є не одиночним символом, а символьним масивом:

```
char cSz[]="Символьний масив";
```

Компілятор, виділяючи пам'ять під цей масив, сам визначить його довжину, яка в даному випадку дорівнюватиме 18 байтам.

Типові дії над масивами можна представити наступними фрагментами програм:

1. Ініціалізація масиву нулями:

```
for (i = 0; i < n; i++)  
m [ i ] = 0;
```

2. Друк масиву в рядок:

```
for (i = 0; i < n; i++)  
    cout << setw(7) << m[ i ] << endl;
```

3. Обчислення суми елементів масиву:

```
int sum = 0;  
for (i = 0; i < n; i++)  
    sum + = m[ i ];
```

4. Виведення елементів масиву у вигляді гістограми:

```
        . . . . .  
cout << "Елемент" << setw(13)  
    << "Значення" << setw(17)  
    << "Гістограма" << endl;  
for (i = 0; i < n; i++)  
{  
    cout << setw(7) << i << setw(13) << m[ i ] << "...";  
    for ( int j = 0; j <= m[ i ]; j++ )  
        cout << " * ";  
    cout << endl;  
}
```

5. Сортування масиву методом "бульбашки":

```
int temp;  
cout << "Початковий порядок елементів" << endl;  
for (i = 0; i < n; i++)  
    cout << setw(7) << m[ i ];  
// сортування  
for ( int p = 1; p < n; p++ ) // к-ть проходів по масиву  
for ( i = 0; i < n - 1; i++ )  
    if ( m[ i ] > m[ i + 1 ] )  
        { temp = m[ i ];  
          m[ i ] = m[ i + 1 ];  
          m[ i + 1 ] = temp; }  
cout << "Масив після сортування" << endl;  
for (i = 0; i < n; i++)  
    cout << setw(7) << m[ i ];  
cout << endl;
```

6. Лінійний пошук у масиві:

```
.....  
int key, flag = n;  
cout << "Введіть ключ пошуку" << endl;  
cin >> key;  
for (i = 0; i < n; i ++)  
    if ( m[ i ]== key ) flag = i;  
if ( flag != n )  
    cout << "Шуканий елемент має індекс" << flag << endl;  
    else cout << ""Елемент не знайдений" << endl;
```

7. Пошук максимального і мінімального елементів масиву:

```
int max, min;  
max = min = m[ 0 ];  
for (i = 1; i < n; i ++)  
{  
    if ( m[ i ]> max ) max = m[ i ];  
    if ( m[ i ]< min ) min = m[ i ];  
}  
cout << "Максимальний елемент =" << max << endl;  
cout << "Мінімальний елемент =" << min << endl;
```

8. Перестановка елементів масиву в зворотному порядку:

```
int j, temp;  
//перший спосіб перестановки  
for ( i = 0, j = n - 1; i < j; ++ i -- j )  
{  
    temp = m[ i ];  
    m[ i ]= m[ j ];  
    m[ j ]= temp;  
}  
//друк  
//другий спосіб перестановки  
for ( i = 0; i < (n / 2); i ++ )  
{  
    temp = m[ i ];  
    m[ i ]= m[ n - 1 - i];  
    m[ n - 1 - i ]= temp;  
}  
// друк.
```

9. Двійковий пошук у відсортованому масиві:

```
int low, high, middle, key, flag = 0;
low = 0;
high = n - 1;
cout << "Введіть ключ" << endl;
cin >> key;
// пошук
while ( low <= high )
{
    middle = ( low + high ) / 2;
    if ( key == m[ middle ] ) flag = 1;
    else if ( key < m[ middle ]
            high = middle - 1;
    else low = middle + 1;
} // while
if ( flag ) cout << "Шуканий елемент має індекс" << middle << endl;
else cout << "Елемент не знайдений" << endl;
```

10. Заповнення масиву даними, вибраними випадковим чином.

При відладці програм часто буває зручним ініціювати масив даними (числами або символами), вибраними випадковим чином. Для цього можна використовувати спеціальні функції з бібліотеки `stdlib.h`. Розглянемо їх докладніше.

Функція `rand()` генерує псевдовипадкові числа в діапазоні 0 .. `RAND_MAX`, де `RAND_MAX` – константа, зазвичай співпадаюча з максимальним цілим значенням для даної реалізації.

Функція `random (int num)` генерує випадкові числа в діапазоні 0 .. `num - 1`.

Функція `randomize()` ініціює генератор випадкових чисел від таймера.

Функція `srand (unsigned seed)` ініціює генератор випадкових чисел так, що послідовність, що генерується, залежить від аргументу; якщо аргумент цієї функції не змінювати, то завжди генеруватиметься одна й та ж послідовність.



Приклади:

1. `int i = 1 + rand() % 6;` //генериуються вип. числа в діапазоні 1.. 6
2. `i = rand() & 2;` // генеруються вип. числа в діапазоні 0..1

### **Завдання** (Обробку масиву виконати у функції)

**Завдання 1.** Скласти програму, що виконує з одновимірним масивом наступні дії:

#### Варіант 1

В одновимірному масиві, що складається з  $n$  дійсних елементів, обчислити:

- 1) суму негативних елементів масиву;
- 2) множення елементів масиву, розташованих між максимальним і мінімальним елементами.

Упорядкувати елементи масиву за збільшенням.

#### Варіант 2

В одновимірному масиві, що складається з  $n$  речових елементів, обчислити:

- 1) суму позитивних елементів масиву;
- 2) множення елементів масиву, розташованих між максимальним за модулем і мінімальним за модулем елементами.

Упорядкувати елементи масиву за убаванням.

#### Варіант 3

В одновимірному масиві, що складається з цілих елементів, обчислити:

- 1) множення елементів масиву з парними номерами;
- 2) суму елементів масиву, розташованих між першим і останнім нульовими елементами.

Перетворити масив так, щоб спочатку розташовувалися всі позитивні елементи, а потім — всі негативні (елементи, які дорівнюють 0, вважати позитивними).

#### Варіант 4

В одновимірному масиві, що складається з  $n$  речових елементів, обчислити:

1) суму елементів масиву з непарними номерами;

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

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

#### Варіант 5

В одновимірному масиві, що складається з  $n$  дійсних елементів, обчислити:

1) максимальний елемент масиву;

2) суму елементів масиву, розташованих до останнього позитивного елементу.

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

#### Варіант 6

В одновимірному масиві, що складається з  $n$  речових елементів, обчислити:

1) мінімальний елемент масиву;

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

Перетворити масив так, щоб спочатку розташовувалися всі елементи, які дорівнюють нулю, а потім — всі останні.

#### Варіант 7

В одновимірному масиві, що складається з  $n$  цілих елементів, обчислити:

1) номер максимального елементу масиву;

2) множення елементів масиву, розташованих між першим і другим нульовими елементами.

Перетворити масив так, щоб в першій його половині розташовувалися елементи, що стояли в непарних позиціях, а в другій половині — елементи, що стояли в парних позиціях.

#### Варіант 8

В одновимірному масиві, що складається з  $n$  речових елементів, обчислити:

- 1) номер мінімального елементу масиву;
- 2) суму елементів масиву, розташованих між першим і другим негативними елементами.

Перетворити масив так, щоб спочатку розташовувалися всі елементи, модуль яких не перевищує 1, а потім — всі останні.

#### Варіант 9

В одновимірному масиві, що складається з  $n$  дійсних елементів, обчислити:

- 1) максимальний за модулем елемент масиву;
- 2) суму елементів масиву, розташованих між першим і другим позитивними елементами.

Перетворити масив так, щоб елементи, які дорівнюють нулю, розташовувалися після всіх останніх.

#### Варіант 10

В одновимірному масиві, що складається з  $n$  цілих елементів, обчислити:

- 1) мінімальний за модулем елемент масиву;
- 2) суму модулів елементів масиву, розташованих після першого елементу, який дорівнює нулю.

Перетворити масив так, щоб у першій його половині розташовувалися елементи, що стояли в парних позиціях, а в другій половині — елементи, що стояли в непарних позиціях.

#### Варіант 11

В одновимірному масиві, що складається з  $n$  речових елементів, обчислити:

- 1) номер мінімального за модулем елементу масиву;
- 2) суму модулів елементів масиву, розташованих після першого негативного елементу.

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

#### Варіант 12

В одновимірному масиві, що складається з  $n$  дійсних елементів, обчислити:

- 1) номер максимального за модулем елементу масиву;
- 2) суму елементів масиву, розташованих після першого позитивного елементу.

Перетворити масив так, щоб спочатку розташовувалися всі елементи, ціла частина яких лежить в інтервалі  $[a, b]$ , а потім — всі останні.

#### Варіант 13

В одновимірному масиві, що складається з  $n$  дійсних елементів, обчислити:

- 1) кількість елементів масиву, розташованих в діапазоні від  $A$  до  $B$ ;
- 2) суму елементів масиву, розташованих після максимального елементу.

Упорядкувати елементи масиву за убаванням модулів елементів.

#### Варіант 14

В одновимірному масиві, що складається з  $n$  дійсних елементів, обчислити:

- 1) кількість елементів масиву, які дорівнюють 0;
- 2) суму елементів масиву, розташованих після мінімального елементу.

Упорядкувати елементи масиву за збільшенням модулів елементів.

#### Варіант 15

В одновимірному масиві, що складається з  $n$  дійсних елементів, обчислити:

- 1) кількість елементів масиву, які більше 3;
- 2) множення елементів масиву, розташованих після максимального за модулем елементу.

Перетворити масив так, щоб спочатку розташовувалися всі негативні елементи, а потім — всі позитивні (елементи, які дорівнюють 0, вважати позитивними).

#### Варіант 16

В одновимірному масиві, що складається з  $n$  дійсних елементів, обчислити:

- 1) кількість негативних елементів масиву;
- 2) суму модулів елементів масиву, розташованих після мінімального за модулем елементу.

Замінити всі негативні елементи масиву їх квадратами й упорядкувати елементи масиву за збільшенням.

#### Варіант 17

В одновимірному масиві, що складається з  $n$  цілих елементів, обчислити:

- 1) кількість позитивних елементів масиву;
- 2) суму елементів масиву, розташованих після останнього елементу, який дорівнює нулю.

Перетворити масив так, щоб спочатку розташовувалися всі елементи, ціла частина яких не перевищує 1, а потім — всі останні.

#### Варіант 18

В одновимірному масиві, що складається з  $n$  речових елементів, обчислити:

- 1) кількість елементів масиву, які менше 3;
- 2) суму цілих частин елементів масиву, розташованих після останнього негативного елементу.

Перетворити масив так, щоб спочатку розташовувалися всі елементи, що відрізняються від максимального не більше ніж на 20%, а потім — всі останні.

#### Варіант 19

В одновимірному масиві, що складається з  $n$  речових елементів, обчислити:

- 1) множення негативних елементів масиву;
- 2) суму позитивних елементів масиву, розташованих до максимального елементу.

Змінити порядок проходження елементів у масиві на зворотний.

## Варіант 20

В одновимірному масиві, що складається з  $n$  речових елементів, обчислити:

1) множення позитивних елементів масиву;

2) суму елементів масиву, розташованих до мінімального елемента.

Упорядкувати за збільшенням окремо елементи, що стоять на парних місцях, і елементи, що стоять на непарних місцях.

### **Завдання 2 (підвищеної складності на оцінку 11 – 12 балів)**

Варіант 1. Послідовність з десяти цілих чисел (як позитивних, так і негативних) представлена у вигляді одновимірною масиву. Знайти і вивести на екран підпослідовність підряд чисел, що йдуть, сума яких максимальна. Наприклад, для послідовності : 1 -8 3 2 -1 4 -6 2 1 -5 відповіддю буде наступна підпослідовність: 3 2 -1 4.

Варіант 2. Дана цілочисельна послідовність (одновимірний масив цілих чисел). Написати програму знаходження кількості різних елементів цієї послідовності. Вивести всі елементи, що зустрічаються більше одного разу.

Варіант 3. У заданій послідовності чисел довжиною  $n$  ( $n < 100$ ) визначити довжину найбільшої впорядкованої за збільшенням підпослідовності сусідніх елементів.

### **Контрольні запитання**

1. Дайте визначення масиву.
2. Як розташовується масив у пам'яті?
3. Вкажіть приклади завдання розмірності масиву.
4. Як можна ініціалізувати елементи масиву?
5. Яке інформаційне навантаження несе ім'я одновимірною масиву?
6. Яка операція використовується для визначення адреси довільного елемента масиву?

## **Лабораторна робота 6**

### **Підготовка й розв'язання на ПЕОМ завдань обробки двовимірних масивів**

**Мета лабораторної роботи** – засвоєння методики обробки інформації в двовимірних масивах.

Перед виконанням лабораторної роботи студент повинен знати:  
 визначення й правила опису двовимірних масивів різних типів;  
 способи доступу до елементів масиву даних;  
 способи ініціалізації елементів масиву;  
 способи алгоритмізації пошуку мінімальних і максимальних значень у двовимірних масивах.

Після виконання лабораторної роботи студент повинен уміти:  
 виконувати ініціалізацію масивів різних типів і розмірності;  
 використовувати засоби мови C++ для перетворення типів даних;  
 виконувати алгоритмізацію пошуку мінімальних і максимальних значень у двовимірних масивах.

## Короткі теоретичні відомості

### Двовимірні масиви

Двовимірний масив є структурованим типом даних, який створюється шляхом вкладення одновимірних масивів.

Доступ до елементів виконується за індексами рядків і стовпців.

У C++ оголошення двовимірного масиву визначає кількість рядків, кількість стовпців і тип даних елементів масиву. Наприклад, масив `T` `type T [ RowCount ] [ ColCount ] ;` може бути представлений таким чином:

	Стовпці										
	0	1	2	3	4	5	6	7	8	9	10
0											
1				10							
2									-3		
3											
Рядки											

Посилання на елементи масиву `T` проводиться за допомогою індексів рядка і стовпця:

`T[i][j]`  $0 \leq i \leq \text{RowCount} - 1$   $0 \leq j \leq \text{ColCount} - 1$

Наприклад, матриця `T` – це масив цілих розміром 4 x 8:

`int T[4][8];`

Причому значення `T[1][2] = 10` і `T[2][6] = -3`.

Концепція двовимірного масиву може бути розширена для обхвату основних багатовимірних масивів, елементи в яких доступні за допомогою трьох або більше індексів.

Двовимірний масив можна представити, як список одновимірних масивів.

Наприклад, `T[0]` – це рядок 0, який складається з `ColumnCount` окремих елементів.

Запис

```
int T[ ][8];
```

указує, що `T` є списком з 8-елементних масивів.

Двовимірний масив може ініціалізуватися привласненням елементам по одному рядку кожного разу. Наприклад, масив `T` задає таблицю розміром 3 x 4:

```
int T [3] [4] = { {20, 5 -30, 0}, {-40, 15, 100, 80}, {3, 0, 0 -1} };
```

Двовимірні таблиці є найбільш поширеною формою представлення даних економічного характеру. Будь-яка таблиця – документ, як правило, містить:

- найменування, так звану "шапку" з іменами колонок (рядків);

- рядки – записи таблиці, що мають однакову довжину і містять декілька різнотипних за характером, але зв'язаних між собою полів – реквізитів;

- підсумкову частину.

Багатовимірний масив відповідно до синтаксису мови є масив масивів, тобто масив, елементами якого служать масиви. Визначення багатовимірного масиву в загальному випадку повинне вміщувати відомості про тип, розмірність і кількість елементів кожної розмірності:

```
type ім'я_масиву[K1][K2]...[KN];
```

Тут `type` – допустимий тип (основний або похідний), `ім'я масиву` - ідентифікатор, `N` – розмірність масиву, `K1` – кількість в масиві елементів розмірності `N - 1` кожен і т. д. Наприклад:

```
int ARRAY[4][3][6];
```

Тривимірний масив `array` складається з чотирьох елементів, кожен з яких – двовимірний масив з розмірами 3 x 6. У пам'яті масив `array` розміщується в порядку зростання найправішого індексу (рис. 5.3), тобто сама молодша адреса має елемент `array[0][0][0]`, потім йде `array[0][0][1]` і т. д.



З урахуванням порядку розташування в пам'яті елементів багатовимірного масиву потрібно розміщувати початкові значення його елементів і в списку ініціалізації. (Поправка на правостороннє написання фраз і слів в європейських мовах на відміну від наряду зростання адрес абсолютно природна).

```
int ARRAY[4][3][6]= { 0, 1, 2, 3, 4, 5, 6, 7 };
```

У даному визначенні початкові значення отримали тільки "перші" 8 елементів тривимірного масиву:

```
ARRAY[0][0][0] = 0  
ARRAY[0][0][1] = 1  
ARRAY[0][0][2] = 2  
ARRAY[0][0][3] = 3  
ARRAY[0][0][4] = 4  
ARRAY[0][0][5] = 5  
ARRAY[0][1][0] = 6  
ARRAY[0][1][1] = 7
```

Решта елементів масиву array залишилася не ініціалізованими і набудуть початкових значень відповідно до статусу масиву.

Якщо необхідно ініціалізувати тільки частину елементів багатовимірного масиву, але вони розміщені не на його початку або не підряд, то можна вводити додаткові фігурні дужки, кожна пара яких виділяє послідовність значень, що відносяться до однієї розмірності. (Не можна використовувати дужки без інформації всередині них).

Наступне визначення з ініціалізацією тривимірного масиву

```
int A[4][5][6]= { { {0} },  
{ {100}, {110, 111} },  
{ {200}, {210}, {220, 221, 222} };
```

так задає тільки деякі значення його елементів:

```
A[0][0][0] =0  
A[1][0][0] =100    A[1][1][0]=110    A[1][1][1] = 111  
A[ 2][ 0][ 0]=200    A[2][1][0]= 210  
A[ 2][ 2][ 0]=220    A[ 2][ 2][ 1]= 221    A[2][ 2][ 2] = 222
```

Решта елементів масиву явно не ініціалізувалася.

Якщо багатовимірний масив при визначенні ініціалізувався, то його найлівіша розмірність може в дужках не указуватися. Кількість елементів компілятор визначає за числом членів в ініціалізуючому списку.

Наприклад, визначення

```
float matr [ ] [5] = { {1}, {2}, {3} };
```

формує масив matr з розмірами 3 x 5, але не визначає явно початкових значень усіх його елементів. Оператор

```
cout << "\nsizeof(matr)= " << sizeof (matr);
```

виведе на екран: sizeof (matrix) = 60.

Початкові значення отримують тільки

```
matr[0][0]= 1
```

```
matr[1][0]= 2
```

```
matr[2][0]= 3.
```

Методи обробки даних, розміщених в двовимірних і багатовимірних масивах аналогічні методам роботи з одновимірними масивами.

Розглянемо приклад рішення задачі обробки даних, розміщених у двовимірному масиві.

#### Приклад 1

Написати програму, яка визначає номер рядка квадратної матриці, сума елементів якої максимальна.

```
// Рядок з максимальною сумою елементів
```

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

```
# include <conio.h>
```

```
void main ()
```

```
{
```

```
int N = 3          // розмір квадратної матриці
```

```
int m[N][N+1];    // останній стовпець використовуваний
```

```
                  // для зберігання суми ел-тів рядка
```

```
int max;          // рядок з максимальною сумою
```

```
                  // елементів
```

```
int i,j;          // індекси матриці
```

```
puts ( " \n Визначення рядка з максимальною");
```

```
puts ( " сумою елементів " );
```

```
printf ("Введіть матрицю %ix%i\n", N, N) ;
```

```
for (i = 0; i < N; i++)
```

```
{
```

```
printf ( " Елементи %i-й рядки -> ", i+1);
```

```
for ( j = 0; j < N; j++)
```

```
scanf ( " %i ", m [ i ] [ j ]); }
```

```
// для кожного рядка обчислимо суму елементів
```

```

for ( i = 0; i < N; i++ )
{
m [ i ][N] = 0; for(j = 0; j < N; j++) m[i][N] += m[i][j]; }
// знайдемо рядок з максимальною сумою
max = 0;
for ( i = 1; i < N; i++ )
if ( m [ i ] [ N ] > m [ max ] [ N ] ) max = i;
printf ( " \n В % i-й рядку сума елементів", max+1);
printf ( " максимальна і рівна % l \n ", m [ max ] [ N ] );
printf ( " \n Для завершення натисніть <Enter> \n " );
getch();
}

```

**Завдання** (Обробку масиву виконати у функції)

**Завдання 1.** Скласти програму обробки двовимірних масивів згідно з наступним завданням:

Варіант 1

Дана цілочисельна прямокутна матриця. Визначити:

кількість рядків, що не містять жодного нульового елементу;

максимальне з чисел, що зустрічаються в заданій матриці більше одного разу.

Варіант 2

Дана цілочисельна прямокутна матриця. Визначити кількість стовпців, що не містять жодного нульового елементу.

Характеристикою рядка цілочисельної матриці назвемо суму її позитивних парних елементів. Переставляючи рядки заданої матриці, розташувати їх відповідно до зростання характеристик.

Варіант 3

Дана цілочисельна прямокутна матриця. Визначити:

кількість стовпців, що містять хоч би один нульовий елемент;

номер рядка, в якому знаходиться щонайдовша серія однакових елементів.

#### Варіант 4

Дана цілочисельна квадратна матриця. Визначити:

множення елементів в тих рядках, які не містять негативних елементів;

максимум серед сум елементів діагоналей, паралельних головній діагоналі матриці.

#### Варіант 5

Дана цілочисельна квадратна матриця. Визначити:

суму елементів у тих стовпцях, які не містять негативних елементів;

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

#### Варіант 6

Дана цілочисельна прямокутна матриця. Визначити:

суму елементів у тих рядках, які містять хоч би один негативний елемент;

номери рядків і стовпців усіх сідлових точок матриці.

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

#### Варіант 7

Для заданої матриці розміром  $8 \times 8$  знайти такі, що  $k$ -ий рядок матриці збігається з  $k$ -им стовпцем.

Знайти суму елементів у тих рядках, які містять хоч би один негативний елемент.

#### Варіант 8

Характеристикою стовпця цілочисельної матриці назвемо суму модулів його негативних непарних елементів. Переставляючи стовпці заданої матриці, розташувати їх відповідно до зростання характеристик.

Знайти суму елементів у тих стовпцях, які містять хоч би один негативний елемент.

#### Варіант 9

Сусідами елементу  $A_{ij}$  в матриці назвемо елементи  $A_{kl}$  з  $i - 1 < k < i + 1$ ,  $j - 1 < l < j + 1$  (до,  $l \neq i, j$ ). Операція згладжування матриці дає

нову матрицю того ж розміру, кожен елемент якої виходить як середнє арифметичне наявних сусідів відповідного елемента початкової матриці. Побудувати результат згладжування заданої речової матриці розміром  $10 \times 10$ .

У згладженій матриці знайти суму модулів елементів, розташованих нижче за головну діагональ.

#### Варіант 10

Елемент матриці називається локальним мінімумом, якщо він менше всіх сусідніх елементів. Підрахувати кількість локальних мінімумів заданої матриці розміром  $10 \times 10$ . Знайти суму модулів елементів, розташованих вище за головну діагональ.

#### Варіант 11

Коефіцієнти системи лінійних рівнянь задані у вигляді прямокутної матриці. За допомогою допустимих перетворень привести систему до трикутного вигляду. Знайти кількість рядків, середнє арифметичне елементів яких менше заданої величини.

#### Варіант 12

Ущільнити задану матрицю, видаляючи з неї рядки і стовпці, заповнені нулями. Знайти номер першого з рядків, що містять хоч би один позитивний елемент.

#### Варіант 13

Здійснити циклічне зрушення елементів прямокутної матриці  $n$  елементів управо або вниз (залежно від введеного режиму),  $n$  може бути більше кількості елементів в рядку або стовпці.

#### Варіант 14

Здійснити циклічне зрушення елементів квадратної матриці розмірності  $M \times N$  управо до елементів таким чином: елементи 1-го рядка зрушуються в останній стовпець зверху вниз, з нього — в останній рядок справа наліво, з неї — в перший стовпець від низу до верху, з нього — в перший рядок; для решти елементів — аналогічно.

#### Варіант 15

Дана цілочисельна прямокутна матриця. Визначити номер першого із стовпців, що містять хоч би один нульовий елемент.

Характеристикою рядка цілочисельної матриці назвемо суму її негативних парних елементів. Переставляючи рядки заданої матриці, розташувати їх відповідно до убування характеристик.

#### Варіант 16

Упорядкувати рядки цілочисельної прямокутної матриці за збільшенням кількості однакових елементів у кожному рядку.

Знайти номер першого із стовпців, що не містять жодного негативного елементу.

#### Варіант 17

Шляхом перестановки елементів квадратної речової матриці добитися того, щоб її максимальний елемент знаходився в лівому верхньому кутку, наступний за величиною — в позиції (2,2), наступний за величиною — в позиції (3,3) і т. д., заповнивши таким чином всю головну діагональ.

Знайти номер першого з рядків, що не містять жодного позитивного елементу.

#### Варіант 18

Дана цілочисельна прямокутна матриця. Визначити:  
кількість рядків, що містять хоч би один нульовий елемент;  
номер стовпця, в якому знаходиться щонайдовша серія однакових елементів.

#### Варіант 19

Дана цілочисельна квадратна матриця. Визначити:  
суму елементів у тих рядках, які не містять негативних елементів;  
мінімум серед сум елементів діагоналей, паралельних головній діагоналі матриці.

#### Варіант 20

Дана цілочисельна прямокутна матриця. Визначити:  
кількість негативних елементів у тих рядках, які містять хоч би один нульовий елемент;

номери рядків і стовпців усіх сідлових точок матриці.

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

### **Завдання 2 (підвищеної складності на оцінку 11 – 12 балів)**

Варіант 1. Перестановкою рядків і стовпців упорядкувати за збільшенням елементи головної діагоналі квадратної матриці.

Варіант 2. Написати програму циклічного зрушення елементів квадратної матриці на задану кількість позицій, вважаючи, що матриця зв'язана за рядками.

Варіант 3. Написати програму, яка для квадратної матриці визначає максимум серед сум елементів, розташованих на лініях, паралельних головній діагоналі, і вище.

Варіант 4. Дана цілочисельна прямокутна матриця. Переходом назвемо переміщення до сусіднього елементу по горизонталі або по вертикалі. Знайти кількість різних шляхів проходу матриці з лівого верхнього кута в правий нижній, якщо допустимими є тільки переходи "вправо на один елемент" і "вниз на один елемент". Визначити шлях, сума елементів якого максимальна.

### **Контрольні запитання**

1. Дайте визначення масиву.
2. Що розуміється під двовимірним масивом?
3. Як розташовується двомірний масив у пам'яті?
4. Вкажіть приклади завдання розмірності масиву.
5. Як можна ініціалізувати елементи масиву?
6. Наведіть приклад визначення двовимірного масиву.
7. Яке інформаційне навантаження несе ім'я одновимірного масиву?
8. Яке інформаційне навантаження несе ім'я двовимірного масиву?
9. Яка операція використовується для визначення адреси довільного елементу масиву?

### **Рекомендована література**

1. Павловская Т. А. С/C++. Программирование на языке высокого уровня. – СПб.: Питер, 2006. — 462 с.
2. Павловская Т. А. С++. Объектно-ориентированное программирование: Практикум. – СПб.: Питер, 2006. — 266 с.
3. Пирогов В. Ю. Программирование на Visual C++.NET. – СПб.: БХВ-Петербург, 2003. – 800 с.
4. Подбельский В. В. Язык С++: Учебное пособие. – 4-е изд. – М.: Финансы и статистика, 1999, – 560 с.



## **Додатки**

## Середовище візуальної розробки програм Microsoft Visual Studio .NET

### 1.1. Середовище MDE

Середовище MDE (Microsoft Development Environment – Середовище Розробки Мікрософтвер), зване також IDE (Integrated Development Environment – Інтегроване Середовище Розробки) призначене для створення, відкриття, перегляду, редагування, збереження, компіляції, побудови й відладки додатків, написаних на С або С++.

Запуск Microsoft Visual Studio .NET виконується за допомогою меню Пуск за стандартною схемою (рис. А.1).



Рис. А.1. Запуск Microsoft Visual Studio .NET

Після цього з'явиться головне вікно MDE із стартовою сторінкою (рис. А.2).

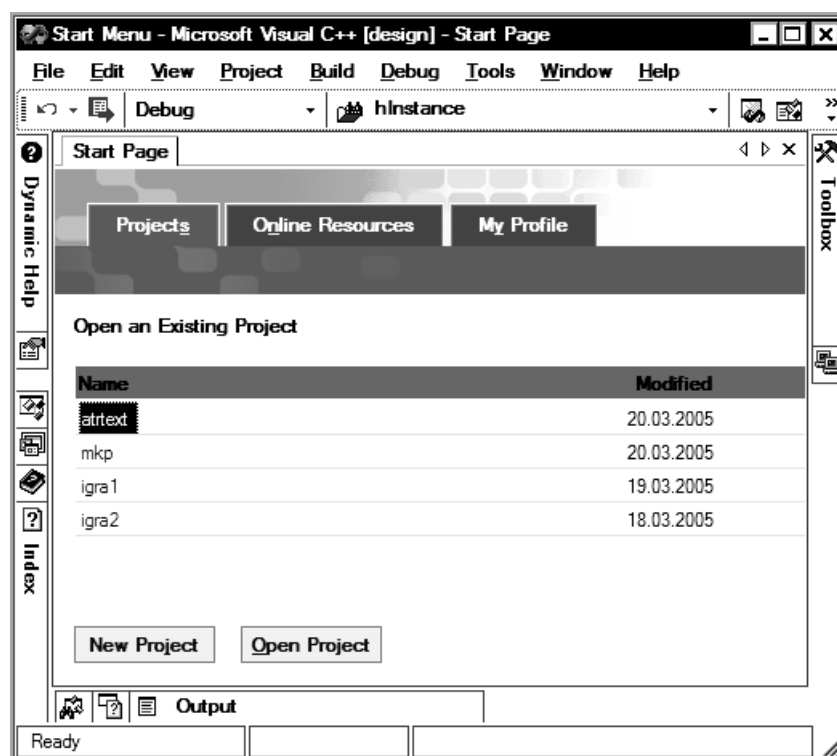


Рис. А.2. Стартова сторінка

При певному налаштуванні середовища може з'явитися головне вікно MDE (рис. А.3).

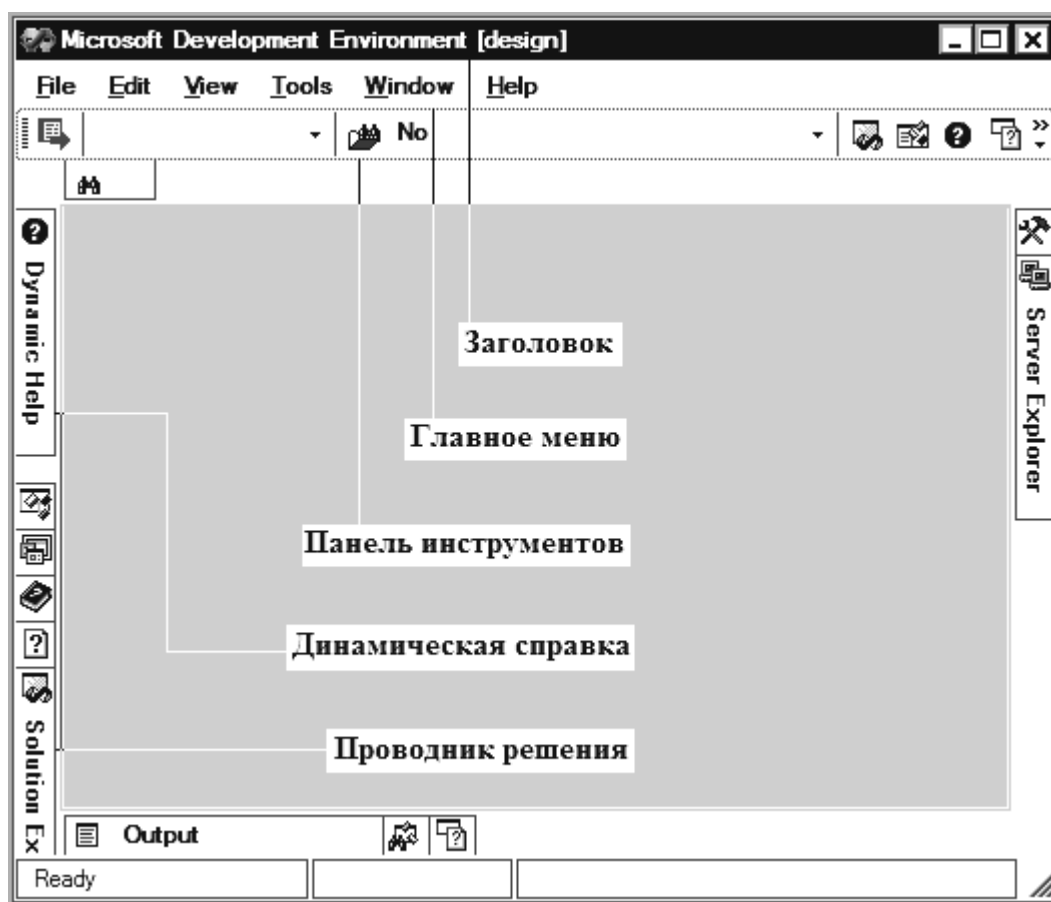


Рис. А.3. Головне вікно MDE

## 1.2. Команди головного меню

Виклик команд меню виконується трьома способами:









- за допомогою гарячих клавіш (вони вказуються в меню праворуч від відповідного пункту);
- за допомогою мишки (вказати й клацнути);
- за допомогою відповідної кнопки (піктограми) на панелі інструментів (вони вказуються зліва від відповідної команди).

Активізація меню здійснюється за допомогою мишки (вказати й клацнути) або клавіш швидкого виклику (Alt + підкреслена буква в пункті меню). У випадних меню можна зустріти команди, виділені блідо-сірим кольором. Такі команди в нинішній момент часу недоступні через відсутність умов, необхідних для їх виконання.

Якщо за назвою команди меню виникає три крапки, це означає, що в результаті вибору даної команди буде відкрито діалогове вікно, якщо темний трикутник, то підменю.

Головне меню включає наступні пункти:

1. File містить стандартний для багатьох Windows-додатків набір команд, призначених для маніпулювання файлами:

<u>N</u> ew		▶ Команда призначена для створення нового проекту.
<u>O</u> pen		▶ Команда приводить до завантаження діалогового вікна Open File, за допомогою якого відкривається будь-який раніше збережений файл, або діалогового вікна Open Project, призначеного для відкриття проектів.
<u>C</u> lose		Команда призначена для закриття завантажених файлів.
 Add <u>N</u> ew Item...	Ctrl+Shift+A	Команди дозволяють створити файл нового проекту, відкрити існуючий проект, а також додати новий проект у поточний відповідно.
 Add <u>E</u> xisting Item...	Shift+Alt+A	
Add <u>P</u> roject		
 <u>O</u> pen Solution...		За допомогою цих команд відкривається існуюче рішення або закривається активне рішення.
 <u>C</u> lose Solution		
 <u>S</u> ave fp.cpp	Ctrl+S	Збереження вмісту активного вікна.
<u>S</u> ave fp.cpp <u>A</u> s...		За допомогою цієї команди копія вмісту вікна може бути збережена у файлі під новим ім'ям.
 <u>S</u> ave <u>A</u> ll	Ctrl+Shift+S	Команда дозволяє зберегти всі відкриті на даний момент файли.
<u>S</u> ource Control		▶ Команда приводить до відкриття підменю, яке призначене для порівняння властивостей файлу, для об'єднання окремих компонентів, встановлення зв'язку з базою даних, здійснення доступу до даних і контролю параметрів оновлення.
 <u>P</u> age Setup...		Установка параметрів сторінки.
 <u>P</u> rint...	Ctrl+P	Вивід на друк вмісту активного вікна.
<u>R</u> ecent <u>F</u> iles		▶ Списки є засобом швидкого завантаження файлу або проекту.
<u>R</u> ecent <u>P</u> rojects		
<u>E</u> xit		Вихід з MDE Visual C++.









2. Edit містить команди, які дозволяють редагувати текст і проводити пошук за ключовими словами в програмному коді, що відображається в активному вікні. Ці команди ідентичні тим, що застосовуються в більшості текстових редакторів.








 <u>U</u> ndo	<b>Ctrl+Z</b>	Команда Undo дозволяє відмінити останню операцію редагування.
 <u>R</u> edo	<b>Ctrl+Y</b>	Відновлює зміни, скасовані за допомогою команди Undo.
 <u>C</u> ut	<b>Ctrl+X</b>	Команда дозволяє скопіювати виділений в активному вікні блок тексту в буфер обміну Clipboard і видалити його з вікна.
 <u>C</u> opy	<b>Ctrl+C</b>	Команда копіює і поміщає виділений блок тексту в буфер обміну.
 <u>P</u> aste	<b>Ctrl+V</b>	Команда здійснює вставку вмісту буфера в поточній позиції курсора.
<b>Cycle Clipboard Ring</b>	<b>Ctrl+Shift+V</b>	Команда дозволяє вибрати необхідний фрагмент з буфера обміну і вставляє його в код.
 <u>D</u> elete	<b>Del</b>	Команда застосовується для видалення символів або виділених фрагментів тексту в активному вікні.
<b>Select <u>A</u>ll</b>	<b>Ctrl+A</b>	Команда дозволяє виділити вміст активного вікна з метою подальшого вирізування, копіювання або видалення.
<b><u>F</u>ind and Replace</b>	►	Команда за дією ідентична однойменному засобу пошуку, який надається в текстових редакторах.
<b><u>G</u>o To...</b>	<b>Ctrl+G</b>	Команда є засобом швидкого переходу до певного місця активного вікна.
<b><u>I</u>nsert File As Text...</b>		У результаті активізації команди відкривається вікно File Manager, що дозволяє вибрати текстовий файл, який можна скопіювати в активний документ.
<b><u>A</u>dvanced</b>	►	Команди підменю дозволяють управляти символами табуляції і стовпців, переносити слова, що не поміщаються, на новий рядок, знаходити і видаляти пропуски, символи табуляції та порожні рядки.

<b>Bookmarks</b>	► Команда викликає підменю, яке дозволяє розмістити закладки в тих місцях програми, до яких ви звертаєтеся найчастіше.
<b>Outlining</b>	► Команда викликає підменю, яке дозволяє приховати вибраний фрагмент коди.
<b>IntelliSense</b>	► Команда викликає підменю, яке дозволяє встановити режим контекстної довідки.

3. View містить команди управління зовнішнім виглядом рішення, проекту, класів, коди, шаблонів і т. д.








За допомогою меню View можна отримати доступ до списку високого рівня, що містить назви всіх вікон і панелей, за допомогою яких на екран виводиться широкий спектр даних.

 <b>Open</b>	Команда дозволяє відкрити файл у вибраному редакторі (наприклад, в двійковому редакторі Binary, редакторі ресурсів Resource і т. д.).
<b>Open With...</b>	
 <b>Solution Explorer</b> <b>Ctrl+Alt+L</b>	Команда надає можливість виконувати різні операції з компонентами рішення: переносити файли з одного проекту в інший, виводити на екран властивості файлів і проектів, а також додавати в рішення як нові, так і існуючі файли або проекти.
 <b>Class View</b> <b>Ctrl+Shift+C</b>	Команда дозволяє виводити інформацію про вибраний клас.
 <b>Server Explorer</b> <b>Ctrl+Alt+S</b>	У вікні Server Explorer демонструються ресурси серверів, до яких можливий доступ по мережі з даного комп'ютера (наприклад, SQL-серверу).
 <b>Resource View</b> <b>Ctrl+Shift+E</b>	Команда виводить на екран список усіх ресурсів активного додатка.
 <b>Properties Window</b> <b>Alt+Enter</b>	Команда надає детальну інформацію про властивості, які розподілені за категоріями і представлені в алфавітному порядку.
 <b>Toolbox</b> <b>Ctrl+Alt+X</b>	Вікно включає елементи управління періоду проектування, елементи управління ACTIVEX.
 <b>Pending Checkins</b>	Команда дозволяє сумісне використання оновленої коди початкових файлів декількома користувачами.
<b>Web Browser</b>	► Команда дозволяє вивести Web-сторінку на екран безпосередньо в IDE.




 Other Windows	► Команда є засобом проглядання макросів, об'єктів, схем документів, списків завдань і результатів пошуку. Вона також забезпечує проглядання вмісту вікна Command Window.
 Show Tasks	► У результаті вибору однієї з команд цього підменю на екран виводиться список зареєстрованих завдань.
 Toolbars	► Дана команда служить для включення/відключення всіх доступних панелей інструментів.
 Full Screen      Shift+Alt+Enter	Активізація цієї команди приводить до збільшення сторінки з кодом до розмірів екрану, що спрощує його перегляд і редагування.
 Navigate Backward      Ctrl+-	Команди забезпечують переміщення вперед і назад по відкритих вікнах програми, а також повернення до змінених місць програми.
 Navigate Forward      Ctrl+Shift+.	
 Property Pages	У вікні публікується детальна інформація про властивості, які розподілені за категоріями і представлені в алфавітному порядку.

#### 4. Project містить команди для управління відкритими проектами.


Даний пункт головного меню з'явиться після створення проекту.

 Add Class...	Чотири перші команди меню використовуються для додавання в поточний проект компонентів, вказаних в їх назвах.
 Add Resource...	
 Add New Item...      Ctrl+Shift+A	
 Add Existing Item...      Shift+Alt+A	
 New Folder	Команда дозволяє створювати нові підкаталоги для розміщення ресурсів проекту.
 Add Web Reference...	Команда завантажує браузер, з якого інтерактивні Web-сервіси можна додати в додаток середовища Visual Studio .NET.
 Set as StartUp Project	Команда використовується при розробці складних проектів.






5. Build забезпечує доступ до елементів MDE, призначених для генерації коду, відладки і запуску створеної програми. Найбільш важливою в даному меню є команда Rebuild Solution. Даний пункт головного меню з'явиться після створення проекту.

	<b>B</b> uild Solution	<b>F7</b>	Команда досліджує всі файли проекту, після чого компілюються і компонуються тільки ті залежні файли, які були створені пізніше, ніж виконуваний файл проекту.
	<b>R</b> ebuild Solution		Відмінність між командами Build Solution і Rebuild Solution полягає в тому, що остання не враховує дату створення файлів і компілює й компонує всі файли проекту.
	<b>C</b> lean Solution		Команда дозволяє видалити всі файли з проміжних каталогів у будь-якій конфігурації робочого простору проекту.
	<b>B</b> uild fp		Цей блок команд використовується для побудови вирішення файлу, а не проекту.
	<b>R</b> ebuild fp		
	<b>C</b> lean fp		
	<b>B</b> atch Build...		
	<b>C</b> onfiguration Manager...		Дана команда аналогічна команді Build Solution, але з її допомогою можна обробити відразу декілька конфігурацій одного проекту.
	<b>C</b> ompile	<b>Ctrl+F7</b>	Команда дозволяє змінити деякі параметри конфігурації скомпонованного блоку.
			Вибір даної команди служить вказівкою відкомпілювати вміст поточного вікна.



6. Debug містить команди для налаштування відладчика. Даний пункт головного меню з'явиться після створення проекту.

	<b>W</b> indows	►	Команда дозволяє проглянути розставлені точки зупинки.
►	<b>S</b> tart	<b>F5</b>	При виклику даної команди програма виконується в цілому або до точки зупинки.
!	<b>S</b> tart Without Debugging	<b>Ctrl+F5</b>	Команда дозволяє задати безупинне виконання алгоритму без використання яких-небудь засобів відладки.
	<b>P</b> rocesses...		Команда приводить до завантаження однойменного діалогового вікна, де можна проглядати програми проектів Visual Studio і маніпулювати ними.



	<b>E</b> xceptions...	Ctrl+Alt+E	Команда викликає діалогове вікно Exceptions, яке містить ієрархічний список виключень, згрупованих за категоріями.
	Step <b>I</b> nto	F11	Команди встановлюють покроковий режим роботи. Різниця між ними виявляється, коли відладчик викликає функцію або метод.
	Step <b>O</b> ver	F10	
	New <b>B</b> reakpoint...	Ctrl+B	Команда дозволяє задати позицію точки зупинки відладки, в якій відладчик зупиниться.
	Clear All Breakpoints	Ctrl+Shift+F9	Команда видаляє всі точки зупинки за один крок.

7. Tools містить команди, які дозволяють активізувати утиліти, необхідні для управління надзвичайно складними додатками, написаними на мові C++.

	Debug <b>P</b> rocesses...	Ctrl+Alt+P	Дана команда подібна до команди Processes з меню Debug.
	Connect to <b>D</b> atabase...		Команда дозволяє визначити спосіб підключення до Microsoft SQL-серверу з середовища Visual Studio .NET для отримання даних.
	Add/Remove <b>T</b> oolbox Items...		Команда застосовується для додавання та видалення елементів управління, об'єктів і інших компонентів вікна Toolbox.
	<b>A</b> dd-in Manager...		Команда дозволяє відобразити список вбудованих додаткових пристроїв.
	<b>B</b> uild Comment Web Pages...		Команда викликає діалогове вікно Build Comment Web Pages, на яких створюються сторінки htm із структурою коду, збереженої у файлах проектів, а також тексти коментарів.
	<b>M</b> acros		<ul style="list-style-type: none"> <li>За допомогою команд підменю Macros забезпечується повне управління макросами в MDE системи Visual C++. При записі макросу фіксуються вироблювані дії. На основі цієї інформації і створюється код макросу. Проте не всі дії, що виконуються за допомогою елементів призначеного для користувача інтерфейсу, можна записати в макросах.</li> </ul>
	<b>E</b> xternal <b>T</b> ools...		Команда використовується для додавання спеціальних команд в меню Tools. За допомогою цих команд з середовища розробки можна запускати зовнішні інструменти.


Customize...

Дана стандартна команда дозволяє налаштовувати всі панелі інструментів MDE, додавати або видаляти кнопки виклику команд, а також проводити налаштування їх роботи.


Options...

Команда використовується для зміни налаштувань MDE, заданих за замовчуванням.

8. Window містить стандартний набір команд, характерний для всіх Windows-додатків. Команди меню View надають можливість визначити, в якому вигляді вікна мають бути виведені на екран.

 New Window

Після активізації команди New Window створюється копія файлу в новому вікні.

 Split

Команда Split поверх вікна Edit виводить чотири квадратні підвікна, в яких ви можете власноруч визначити положення горизонтальних і вертикальних ліній розділу.

Dockable


Hide

Floating

Панель інструментів може бути закріплена в будь-якому краю батьківського вікна (команда Dockable), а може знаходитися у власному маленькому вікні і займати будь-яку позицію на екрані (команда Floating).

 Close All Documents

Команда дозволяє закрити всі вікна, відкриті в середовищі MDE.

 1 fp.cpp

Щоб потрапити в одне з відкритих вікон, досить вибрати його назву в динамічному списку меню Window.

Windows...

Ця команда відкриває список активних на даний момент вікон.

9. Help включає команди, які дозволяють звернутися до довідкових засобів середовища Visual Studio .NET і Dynamic Help.

 Dynamic Help      **Ctrl+F1**



Команда забезпечує виклик динамічної довідки.

 Contents...      **Ctrl+Alt+F1**


 Index...      **Ctrl+Alt+F2**

 Search...      **Ctrl+Alt+F3**

Ці команди забезпечують доступ і управління інтерактивною довідковою системою, розробленою фірмою Microsoft.

 **Index results...**    **Shift+Alt+F2**  
 **Search results...**    **Shift+Alt+F3**

↑ Previous topic  
 ↓ Next topic  
 ↔ Sync Contents

 **Show Start Page**

**Check for Updates**

 **Technical Support**

**Help on Help**

**About Microsoft Visual C++...**

Команди дозволяють виводити назви розділів, знайдених в наочному покажчику, і результати пошуку певних довідкових розділів.

Перераховані в заголовку команди доступні, якщо у вікні Edit міститься довідкова інформація.

Команда виводить на екран зміст початкової стартової сторінки середовища Visual Studio .NET, на якій можна дістати доступ до новинок, пов'язаних з Visual Studio .NET.

<sup>x</sup> Команда виводить інформацію про останні розробки компанії Microsoft для середовища Visual Studio .NET.

Команда забезпечує зв'язок з компанією Microsoft з питань, що стосуються середовища Visual Studio.

Команда дозволяє отримати інформацію про способи використання складних команд контекстно-залежної довідки середовища Visual Studio .NET.

У стандартному вікні About відображається інформація про версію програми, її ідентифікаційний номер і встановлені компоненти.

### 1.3. Створення проекту

Створення нового проекту здійснюється за допомогою пунктів системи меню (рис. А.4) середовища MDE або за допомогою гарячих клавіш Ctrl+Shift+N.



Рис. А.4. Виклик діалогового вікна New Project.

Після виконання вказаних дій з'явиться діалогове вікно New Project (рис. А.5).

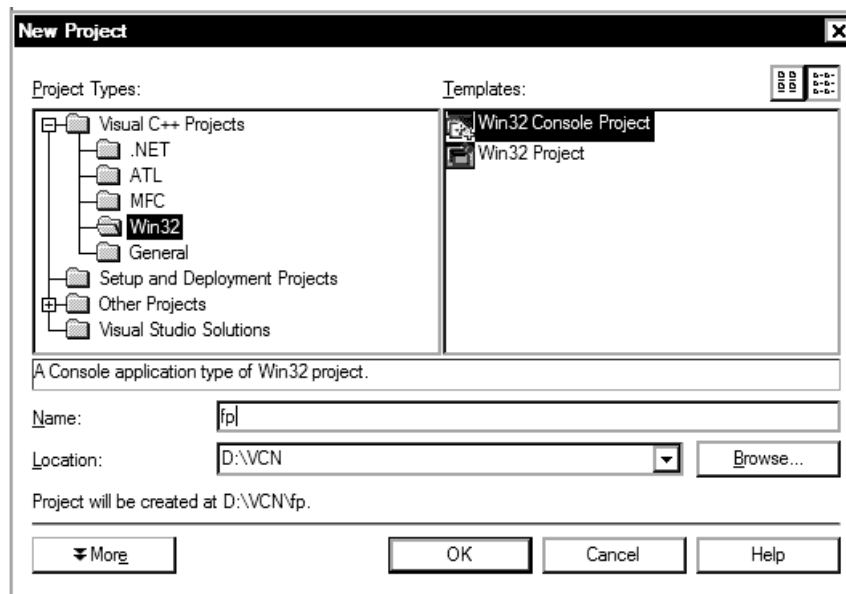


Рис. А.5. Діалогове вікно New Project

У цьому вікні потрібно вибрати:  
 тип проекту (Project Types);  
 шаблон (Templates);  
 теку, в яку буде поміщений новий проект;  
 ім'я проекту.

Після клацання на кнопці ОК на екрані відобразиться вікно майстра (рис. А.6).

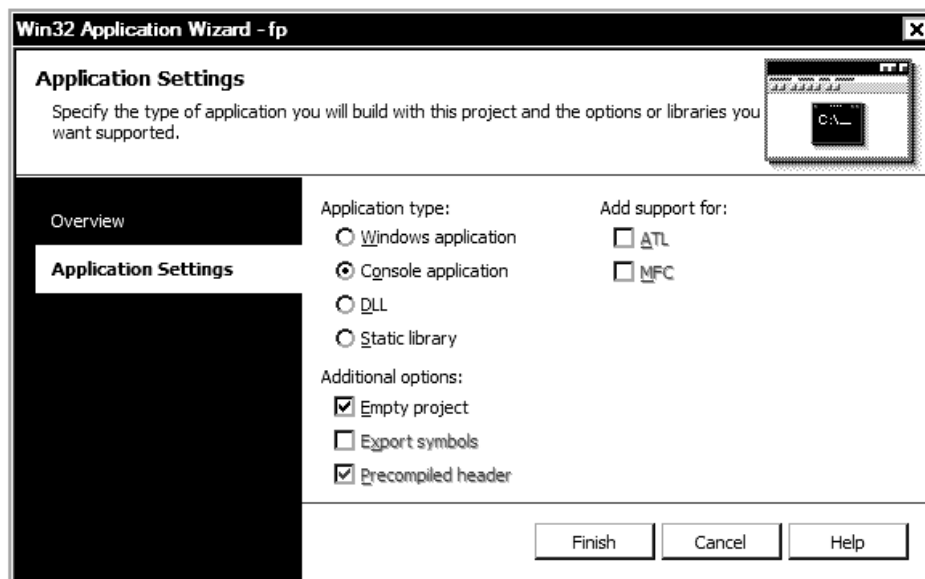


Рис. А.6. Вікно майстра створення додатка Win32

Для створення простого Windows-додатка необхідно вибрати закладку Application Settings, включити перемикач Console Application і встановити прапорець Empty project.

Після натиснення кнопки Finish в зовнішній пам'яті D:\VCN\fp буде створена тека проекту fp і на екрані дисплея з'явиться головне вікно MDE з темною робочою областю. У рядку головного меню з'явиться три додаткові пункти (Project, Build, Debug).

Далі необхідно створити файл початкового тексту програми. Для цього необхідно виділити теку Source Files у вікні провідника рішення (Solution Explorer). У контекстному меню вікна треба вибрати команду Add і далі в додатковому меню, що з'явилося, вибрати команду Add New Item (рис. А.7).

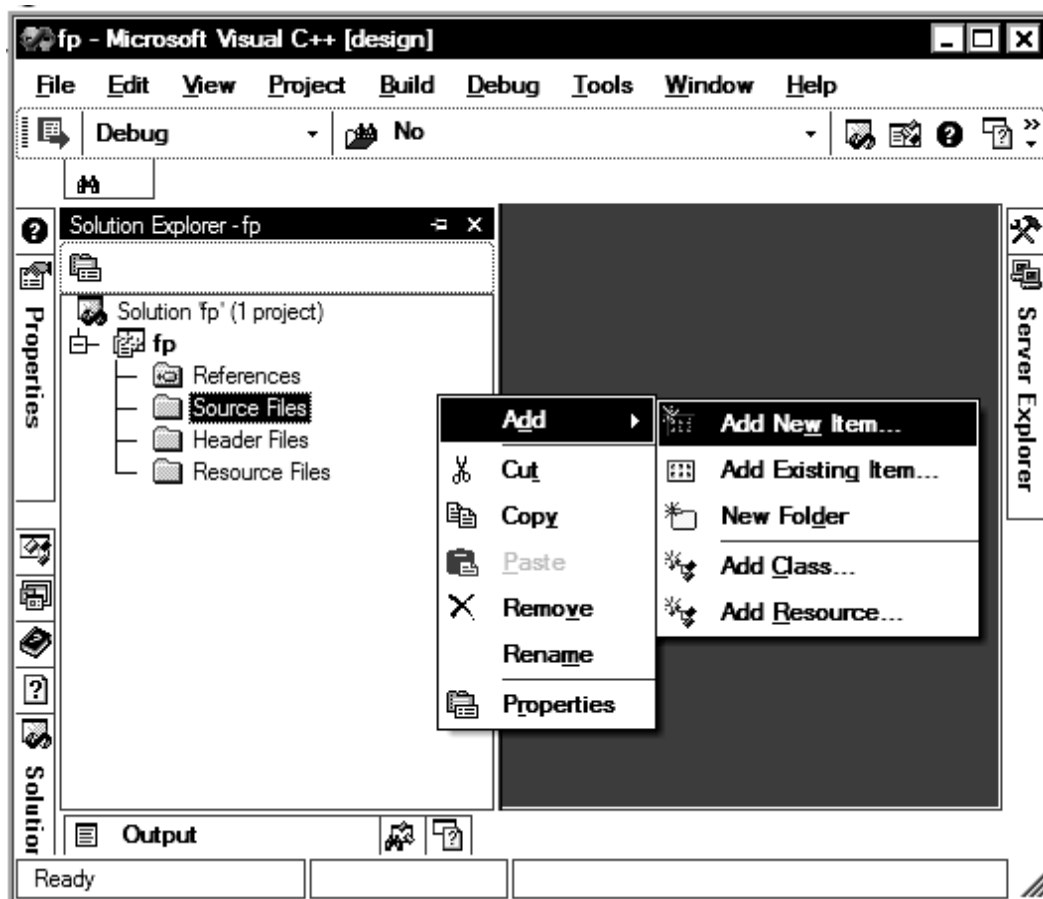


Рис. А.7. Процес створення файлу початкового тексту програми

У результаті виконаних дій з'явиться діалогове вікно Add New Item – fp (рис. А.8). У вікні треба вказати ім'я файлу і вибрати шаблон (Templates).

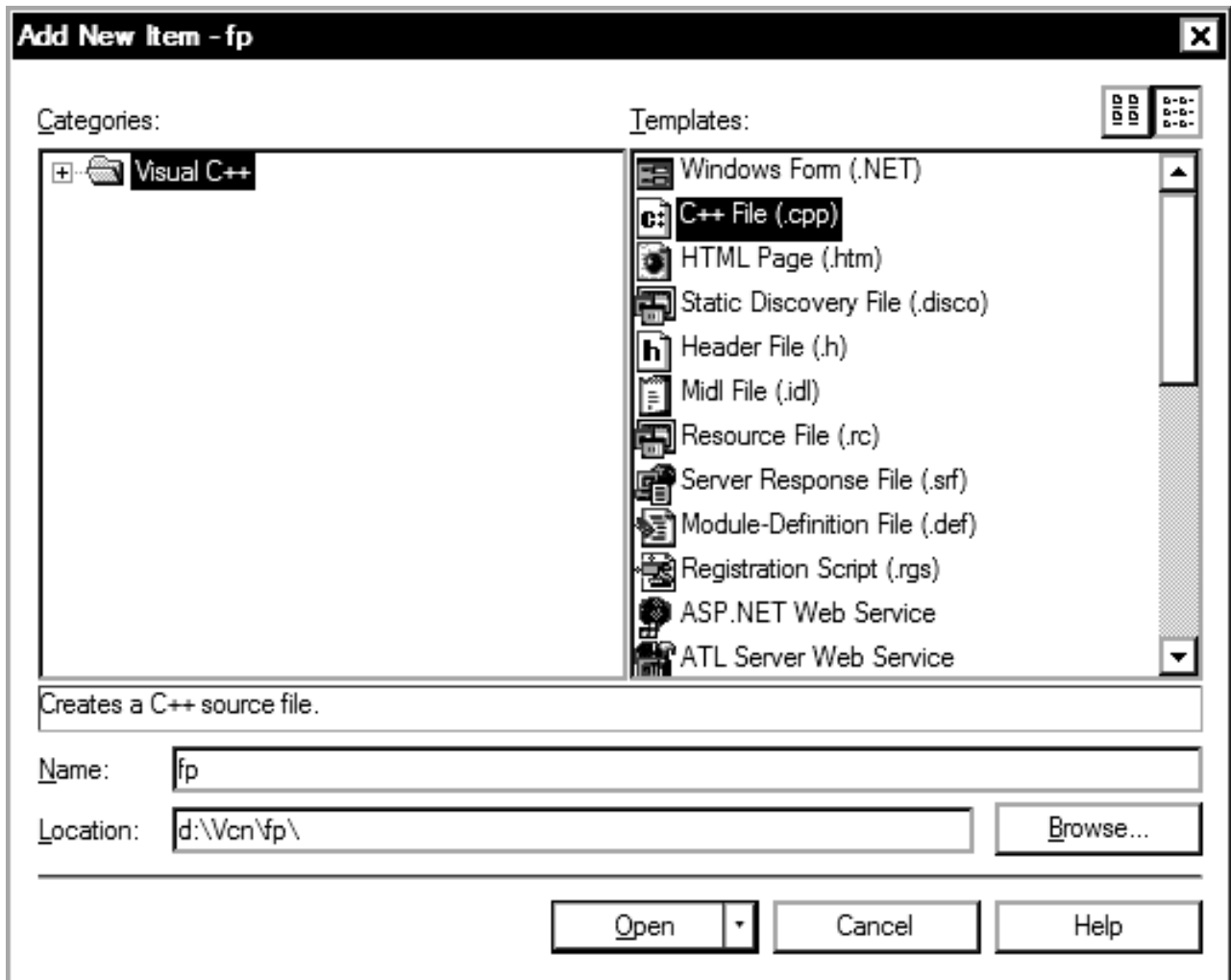


Рис. А.8. Діалогове вікно Add New Item-fp

Після натиснення кнопки Open в папці проекту буде створений файл початкового тексту програми fp.cpp і активізується редактор початкового тексту програми. У заголовку головного вікна MDE з'явиться ім'я файлу.

#### **1.4. Введення й редагування початкового тексту програми**

Введення й редагування початкового тексту програми виконується за правилами, які використовуються в текстових редакторах.

##### **Внесення виправлень до тексту**

Невеликі зміни вносяться до тексту за допомогою клавіш [Backspace] або [Del] або введенням виправленого тексту в режимі заміни. У разі "великомасштабних" змін використовуються команди меню Edit: Cut, Copy і Paste (для цих операцій передбачені поєднання клавіш і кнопки в інструментальній панелі Standard і контекстному меню).

Друкарські помилки в процесі введення тексту виправляються за допомогою клавіші [Backspace]. При цьому віддаляються символи, що стоять зліва від курсора. [Ctrl]+[Backspace] дозволяють видалити ціле слово зліва від курсора.

Натиснення клавіші [Ins] або подвійне клацання на полі INS в рядку стану активізує режим заміни. У рядку стану з'являється індикація поля OVR. Зазвичай редактор за замовчуванням працює в режимі вставки. Це означає, що символ вставляється в існуючий текст, а текст, розташований праворуч від курсора, зрушується вправо.

Для вставки тексту слід перевести курсор у потрібну позицію, використовуючи навігаційні клавіші або мишу, і набрати текст, що вставляється.

Для відміни введення або видалення необхідно натиснути [Alt]+[Backspace] або вибрати команду меню Edit>Undo.

##### **Переміщення курсора, виділення тексту**

Курсор введення (мерехтливий вертикальний штрих) позначає поточну позицію введення тексту.

За допомогою клацання миші в деякій позиції документа можна помістити курсор введення в цю позицію. При роботі з клавіатурою окрім навігаційних клавіш (клавіш управління курсором) для переміщення курсора в розпорядження користувача надані спеціальні поєднання клавіш (табл. А.1).

Таблиця А.1

**Клавіші для переміщення курсора**

Клавіша	переводить курсор...
[стрілка вліво], [стрілка управо]	...на один символ управо або вліво
[стрілка вгору], [стрілка вниз]	...на один рядок вгору або вниз
[Ctrl]+[стрілка управо]	...у позицію перед першим символом наступного слова
[Ctrl]+[стрілка вліво]	...у позицію перед першим символом попереднього слова
[Home]	...у початок поточного рядка
[End]	...у кінець поточного рядка
[Ctrl]+[Home]	...у початок програми
[Ctrl]+[End]	...у кінець програми

**Прокручування зображення**

Прокручування тексту у вікні за допомогою смуг прокрутки дозволяє переміщати вікно над документом, залишаючи незмінним положення курсора. Позицію курсора можна визначити за виведеною в рядку стану інформацією (Ln ... Col ...).

Якщо клацнути на кнопці-стрілці смуги прокрутки, то вміст вікна документа зміщується по рядкам вгору або вниз.

Якщо клацнути над бігунком (під бігунком), вікно переміщується над документом на один екран вгору (вниз).

Переміщати екран над текстом програми за бігунок.

Горизонтальна прокрутка виконується аналогічно вертикальній, з тією відмінністю, що вміст екрану зміщується управо або вліво.

Горизонтальна прокрутка можлива (і необхідна) лише в тому випадку, якщо довжина текстових рядків перевищує ширину екрану.

**Виділення тексту**

Для виконання більшості команд редагування або внесення змін у текст програми необхідно заздалегідь відповідним чином виділити фрагмент, що підлягає обробці.

Виділення найпростіше виконується за допомогою миші, проте і за допомогою клавіатури можна виділити фрагмент тексту посимвольно, послівно, позиція за позицією.



Виділення тексту (розширення виділення) можна виконати, натискаючи навігаційну клавішу спільно з клавішею [Shift] або [Shift]+[Ctrl]. Так, наприклад, [Shift]+[Ctrl]+[стрілка управо]/[стрілка вліво] забезпечують розширення виділення тексту послівно [Shift]+[Home] — до початку рядка і т. д.

### **Виділення тексту за допомогою миші**

Щоб виділити текст програми за допомогою миші, треба помістити курсор миші на перший символ виділення, натиснути ліву кнопку миші і, утримуючи її натиснутою, протягнути мишу в потрібному напрямі. По досягненню кінця фрагмента, що виділяється, відпустити кнопку миші.

Для виділення слова потрібно двічі клацнути на ньому лівою кнопкою миші.

Для виділення рядка потрібно клацнути лівою кнопкою миші напроти рядка, що виділяється.

### **Копіювання, перенесення й видалення**

Ефективність процесу редагування тексту програми можна підвищити, якщо скористатися вбудованим засобами копіювання, видалення й вставки.

Буфер обміну є постійно доступною під час робочого сеансу областю пам'яті, до якої можуть звертатися всі програми, що працюють в середовищі Windows. У буфер обміну переносяться, копіюються, а потім витягаються з нього для вставки фрагменти програми.

Остання вирізана або скопійована порція тексту зберігається в буфері обміну.

Вміст буфера обміну можна вставити в текст програми в декількох позиціях довільне число разів.

### **Швидке переміщення й копіювання фрагмента програми**

Фрагменти тексту можуть бути перенесені і без використання буфера обміну. Для цього потрібно виділити переміщуваний текст. Помістити покажчик миші у виділений текст, натиснути й утримувати натиснутою ліву кнопку миші. Помістити покажчик миші в ту позицію, куди слід перемістити виділений текст, і лише після цього відпустити кнопку миші.

Для копіювання фрагмента тексту програми потрібно його виділити. Помістити покажчик миші у виділений текст, натиснути й утримувати натиснутими клавішу [Ctrl] і ліву кнопку миші. Покажчик миші змінює свій зовнішній вигляд, і він доповнюється прямокутником і знаком "+". Перемістити покажчик миші в ту позицію, куди слід скопіювати виділений фрагмент, і лише після цього відпустити кнопку миші.

### Пошук і заміна

У редакторів початкового тексту програми передбачена можливість пошуку символів, слів або позицій, в яких був застосований вказаний формат. Знайдені елементи можна автоматично замінити іншими.

Для пошуку й заміни потрібно виконати команду Edit>Find and Replace або натиснути [Ctrl]+[F]. Відкривається діалогове вікно Find (рис. А.9).

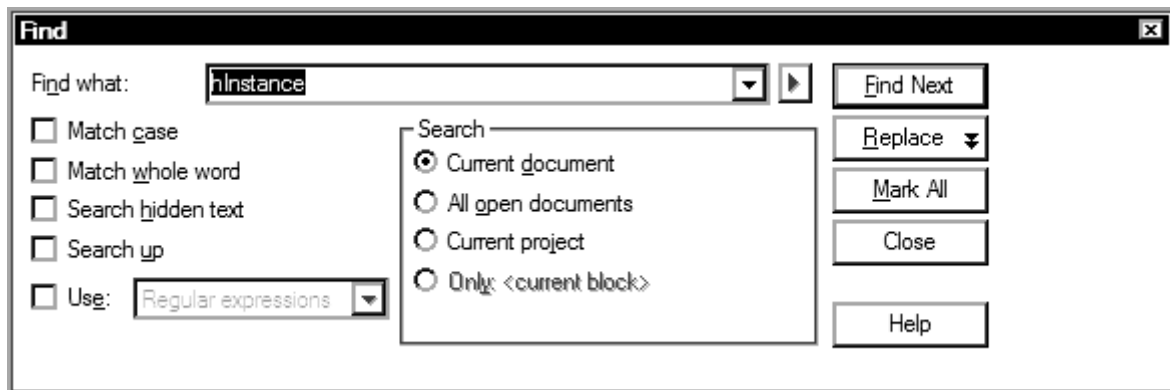


Рис. А.9. Діалогове вікно Find

У полі Find what потрібно ввести текстовий фрагмент, який має бути знайдений (зразок). Його довжина не повинна перевищувати 255 символів.

Пошук можна виконувати:

- у поточному документі;
- у всіх відкритих документах;
- у поточному проекті;
- у виділеному блоці поточного документа.

Вказані можливості пошуку й заміни важливі при роботі з багатофайловими проектами.

Для збереження тільки що введеного початкового тексту програми потрібно скористатися командою меню File>Save або натиснути поєднання клавіш Ctrl+S.

На рис. А.10 показано вікно редагування до збереження файлу. За замовчуванням файли автоматично зберігаються при виконанні команди Build Solution або Rebuild Solution. При виході з редактора, він запропонує зберегти результати роботи.

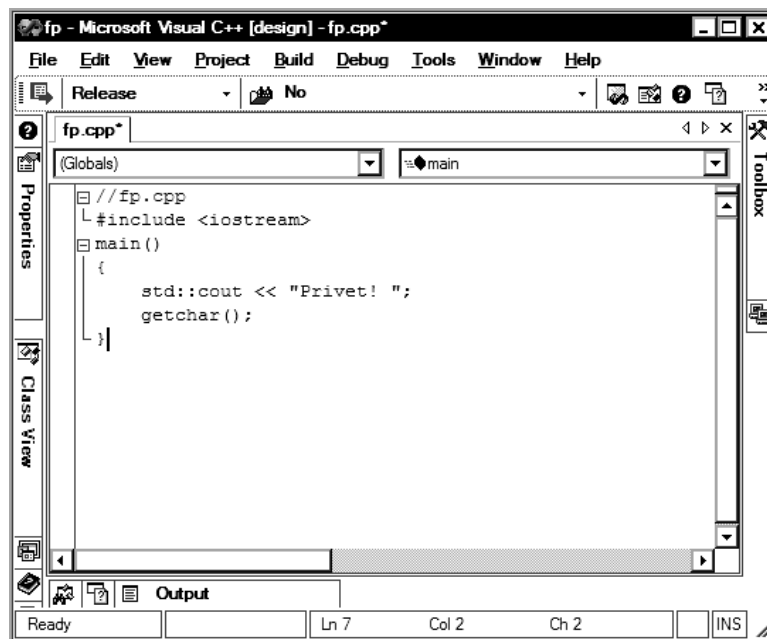


Рис. А.10. Вікно MDE з текстом програми

### 1.5. Створення виконуваного файлу

Для створення виконуваного файлу рекомендується використувати команду меню Build> Rebuild Solution.

Якщо в програмі були допущені синтаксичні помилки, то повідомлення про них відображатимуться на вкладці Build вікна Output (рис. А.11).

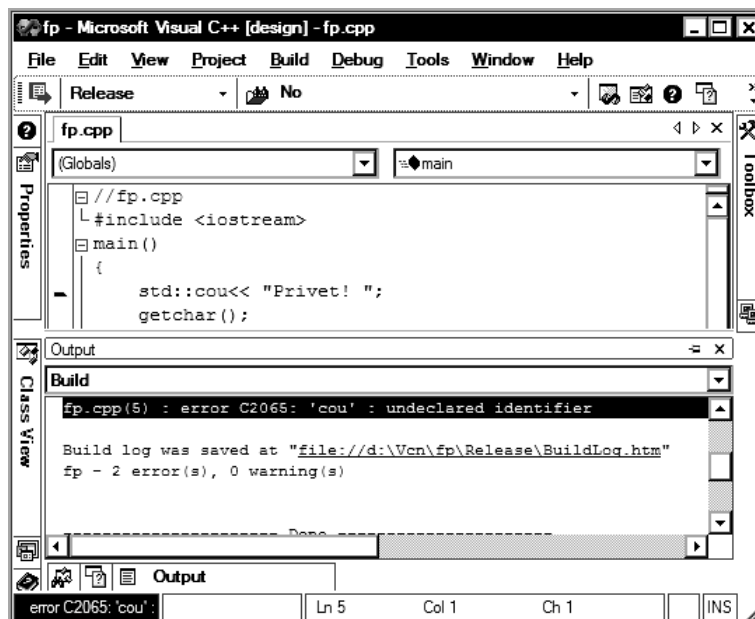


Рис. А.11. Вікно Output з повідомленням про помилки

Повідомлення про помилку включає:

- номер помилки;
- категорію помилки (warning — попередження, error — помилка);
- короткий опис;
- ім'я файлу;
- номер терміну, в якому міститься помилка або попередження.

У даному прикладі суть помилки, що знаходиться в п'ятому рядку, полягає в тому, що ідентифікатор `con` не визначений.

Застережливі повідомлення можуть з'являтися в тих випадках, коли компілятор автоматично виконує деякі стандартні перетворення і повідомляє про це програміста. Наприклад, якщо змінною типу `int` (ціле число) привласнюється дробове значення, то автоматично відбувається округлення. Це не означає, що в програмі допущена помилка, але оскільки перетворення типів даних виконується непомітно для програміста, компілятор вважає своїм обов'язком повідомити про це. Більшість функцій, оголошених у файлі `math.h`, приймають аргументи і повертають значення типу `double` (дійсне число подвійної точності). Якщо програма передасть одній з таких функцій аргумент типу `float` (дійсне число поодинокі точності), компілятор, перш ніж направити дані в стек аргументів функції, виведе застережливе повідомлення про те, що тип даних `float` перетворений в `double`.

Для виявлення рядка в початковому тексті програми, яка викликала помилку, необхідно встановити курсор на рядок повідомлення і натиснути клавішу `Enter` або виконати подвійне клацання мишею. Курсор у вікні редагування буде автоматично поміщений в рядок програми, що викликав появу повідомлення про помилку, а зліва від рядка з'явиться покажчик (рис. А.11).

Після усунення помилки можна повторно використовувати команду меню `Build> Rebuild Solution` (рис. А.12).

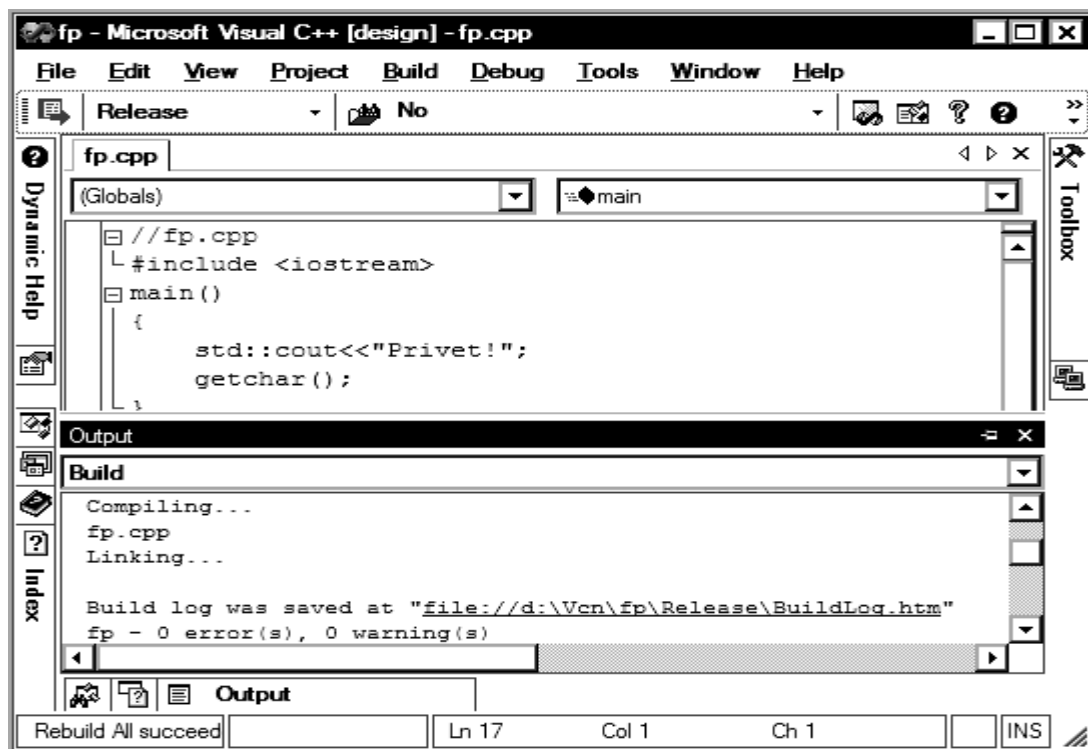


Рис. А.12. Вікно Output після повторного сеансу побудови проекту

Якщо проаналізувати вміст вікна Output, то можна побачити, що процес побудови виконавчого модуля програми включає трансляцію (Compiling) початкового тексту програми і скріплення (Linking) всіх складових проекту у виконуваний модуль. Результатом роботи транслятора і компоувальника є файли, що мають розширення \*.obj і \*.exe відповідно. Помилки можуть виникати як на етапі трансляції, так і на етапі побудови виконуваного модуля.

## 1.6. Запуск і відладка програми

Після успішної побудови виконуваного модуля, його можна запустити на виконання. Для цього слід активізувати в меню Debug команду Start Without Debugging або натиснути Ctrl+F5. Якщо на етапі виконання не виникло помилок, то на екрані дисплея з'явиться консольне вікно з результатом (рис. А.13).



Рис. А.13. Результат роботи програми

Помилки, що виникають на етапі виконання, усуваються засобами відладки середовища MDE. Вони включають точки зупинки, вікна попереднього перегляду стану змінних і покрокове виконання програми.

Для демонстрації засобів відладки, внесемо невелику зміну до початкового тексту програми і створимо точку зупинки на сьомій строчці програми (рис. А.14).

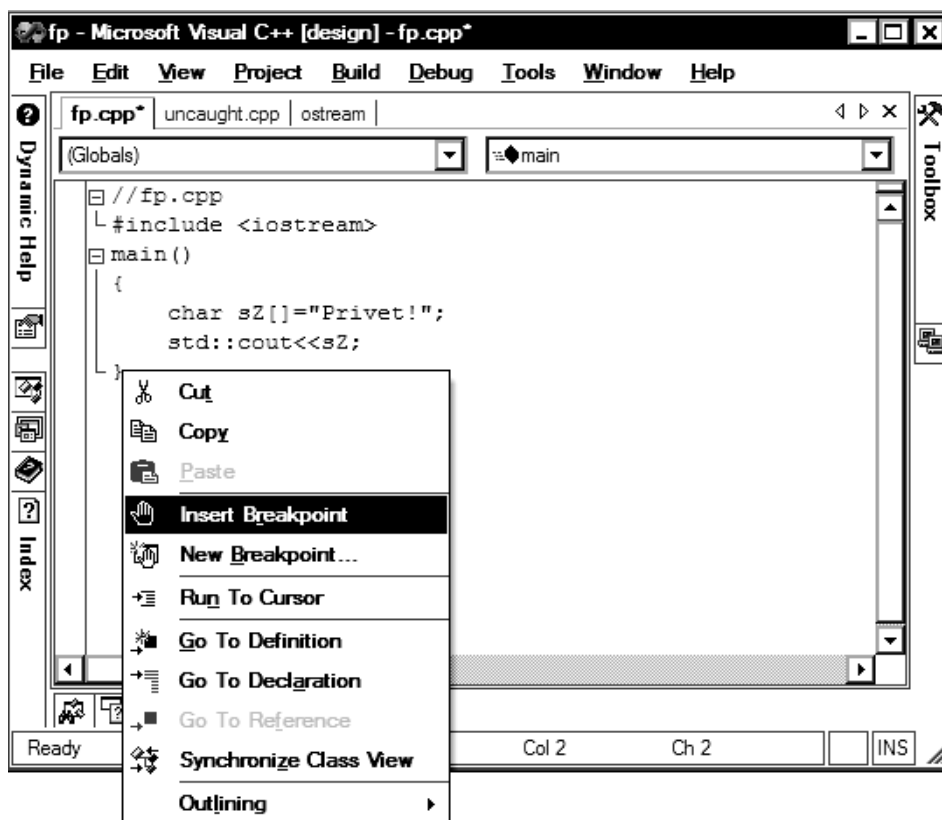


Рис. А.14. Створення точки зупинки

Точки зупинки створюються за допомогою діалогового вікна New Breakpoints, що викликається командою New Breakpoints меню Debug або контекстного меню. Наявність кольорового кружка на початку рядка говорить про наявність точки зупинки в цьому місці (рис. А.15.).

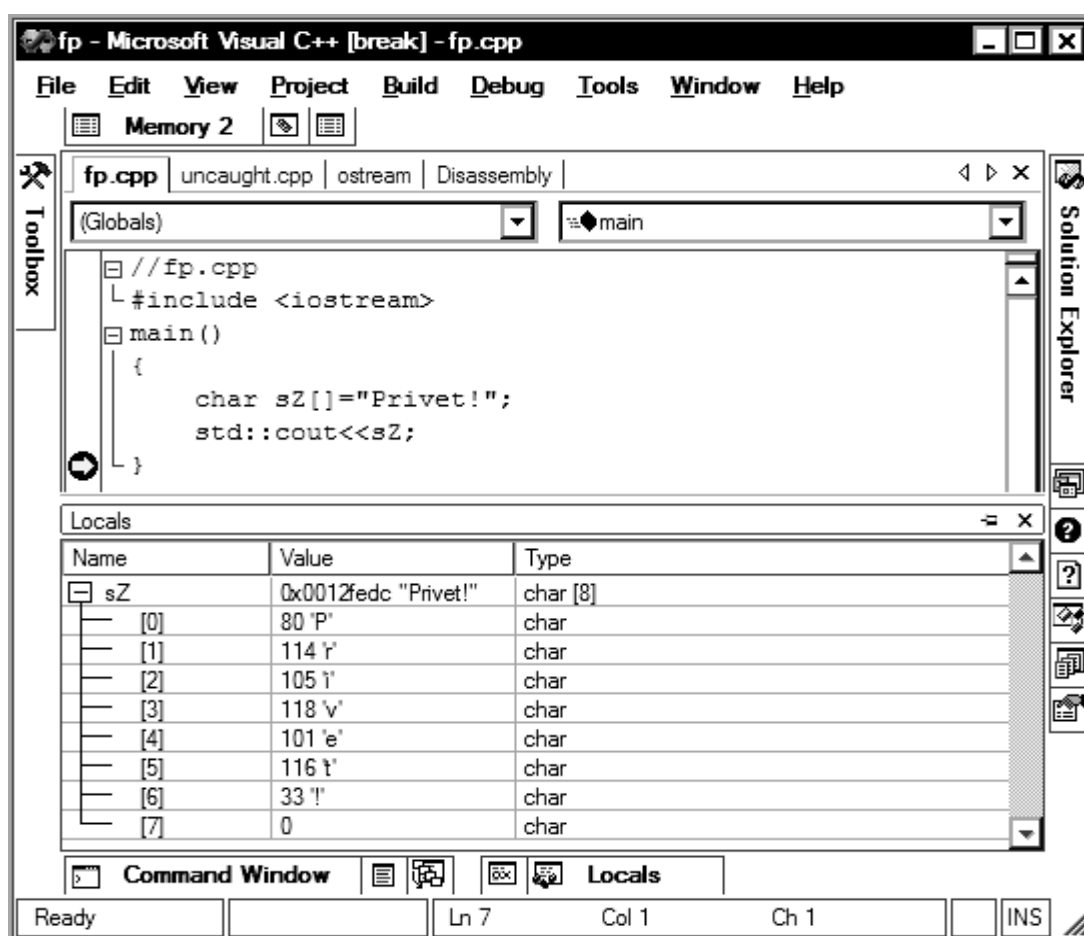


Рис. А.15. Вікно перегляду стану змінної sZ

Після розміщення точок зупинки можна активізувати команду Start меню Debug або натиснути F5. На рядку, де розміщена точка зупинки, виконання програми перерветься. Далі за допомогою команди Windows меню Debug можна викликати вікно Locals і проглянути вміст змінної sZ (рис. А.15). У даному випадку можна переконатися, що транслятор в кінець строкової змінної вставляє нуль.

Далі виконання програми можна продовжувати в покроковому режимі. Покроковий режим виконання програми супроводжується покажчиком трасування (контурна стрілка на початку рядка) і активізується командами Step Into (F11) і Step Over (F10) меню Debug. Відмінності між командами з'являються тільки тоді, коли в програмі зустрічається виклик функцій. Якщо вибрати команду Step Into, то у функції відладчик починає виконувати всіх операторів крок за кроком. При виборі команди Step Over відладчик виконує функцію як єдине ціле.

**Завдання**

Розробити програму для обчислення і виводу на екран у вигляді таблиці значень функції  $F(a,b,c)$ , де  $a, b, c$  — дійсні числа на інтервалі від  $X_{\text{нач.}}$  до  $X_{\text{кон.}}$  з кроком  $dX$ .

**Варіант 1**

$$F = \begin{cases} a \cdot x^2 + b & \text{при } x < 0 \text{ і } b \neq 0 \\ (x - a) / (x - z) & \text{при } x > 0 \text{ і } b = 0 \\ x / z & \text{у решті випадків} \end{cases}$$

**Варіант 2**

$$F = \begin{cases} 1 / a \cdot x - b & \text{при } x + 5 < 0 \text{ і } c = 0 \\ (x - a) / x & \text{при } x + 5 > 0 \text{ і } z \neq 0 \\ 10 \cdot x / (c - 4) & \text{у решті випадків} \end{cases}$$

**Варіант 3**

$$F = \begin{cases} a \cdot x^2 + b \cdot x + z & \text{при } a < 0 \text{ і } z \neq 0 \\ -a / (x - c) & \text{при } a > 0 \text{ і } z = 0 \\ a \cdot (x + z) & \text{у решті випадків} \end{cases}$$

**Варіант 4**

$$F = \begin{cases} -a \cdot x - z & \text{при } z < 0 \text{ і } x \neq 0 \\ (x - a) / -c & \text{при } z > 0 \text{ і } x = 0 \\ b \cdot x / (c - a) & \text{у решті випадків} \end{cases}$$

**Варіант 5**

$$F = \begin{cases} a - x / (10 + b) & \text{при } x < 0 \text{ і } b \neq 0 \\ (x - a) / (x - c) & \text{при } x > 0 \text{ і } b = 0 \\ 3 \cdot x + 2 / z & \text{у решті випадків} \end{cases}$$

**Варіант 6**

$$F = \begin{cases} a \cdot x^2 + b^2 \cdot x & \text{при } z < 0 \text{ і } b \neq 0 \\ (x + a) / (x + c) & \text{при } z > 0 \text{ і } b = 0 \\ x / z & \text{у решті випадків} \end{cases}$$



**Варіант 7**

$$F = \begin{cases} -a \cdot x^2 - b \\ (x+a) / x \\ -x / z \end{cases} \quad \begin{array}{l} \text{при } x < 5 \text{ і } z \neq 0 \\ \text{при } x > 5 \text{ і } z = 0 \\ \text{у решті випадків} \end{array}$$

**Варіант 8**

$$F = \begin{cases} -a \cdot x^2 + b \\ (a - x) / c \cdot x \\ -x / z \end{cases} \quad \begin{array}{l} \text{при } z < 0 \text{ і } a \neq 0 \\ \text{при } z > 0 \text{ і } a = 0 \\ \text{у решті випадків} \end{array}$$

**Варіант 9**

$$F = \begin{cases} a \cdot x^2 + b \cdot x \\ x - a / (x - z) \\ -1 + x / z \end{cases} \quad \begin{array}{l} \text{при } a < 0 \text{ і } x \neq 0 \\ \text{при } a > 0 \text{ і } x = 0 \\ \text{у решті випадків} \end{array}$$

**Варіант 10**

$$F = \begin{cases} a \cdot x^2 + b \cdot x + z \\ (x - a) / (x - z) \\ x / z \end{cases} \quad \begin{array}{l} \text{при } x < 3 \text{ і } b \neq 0 \\ \text{при } x > 3 \text{ і } b = 0 \\ \text{у решті випадків} \end{array}$$

**Варіант 11**

$$F = \begin{cases} a \cdot x^2 + b / z \\ (x - a) / (x - z)^2 \\ x^2 / c^2 \end{cases} \quad \begin{array}{l} \text{при } x < 1 \text{ і } z \neq 0 \\ \text{при } x > 1,5 \text{ і } z = 0 \\ \text{у решті випадків} \end{array}$$

**Варіант 12**

$$F = \begin{cases} a \cdot x^2 + b \cdot x + z \\ (x - a) / (x - z) \\ x / z + x / a \end{cases} \quad \begin{array}{l} \text{при } x < 0,6 \text{ і } b + z \neq 0 \\ \text{при } x > 0,6 \text{ і } b + z = 0 \\ \text{у решті} \end{array}$$

випадків

**Варіант 13**

$$F = \begin{cases} a \cdot x^2 + b \\ (x - a) / x \\ x / z \end{cases} \quad \begin{array}{l} \text{при } x - 1 < 0 \text{ і } b - x \neq 0 \\ \text{при } x - 1 > 0 \text{ і } b + x = 0 \\ \text{у решті випадків} \end{array}$$

**Варіант 14**

$$F = \begin{cases} -a \cdot x^3 - b & \text{при } x + z < 0 \text{ і } a \neq 0 \\ (x - a) / (x - z) & \text{при } x + z > 0 \text{ і } a = 0 \\ x / z + x / z & \text{у решті випадків} \end{cases}$$

**Варіант 15**

$$F = \begin{cases} -a \cdot x^2 + b & \text{при } x < 0 \text{ і } b \neq 0 \\ x / (x - z) & \text{при } x > 0 \text{ і } b = 0 \\ x / -c & \text{у решті випадків} \end{cases}$$

**Варіант 16**

$$F = \begin{cases} a \cdot (x + z)^2 - b & \text{при } x = 0 \text{ і } b \neq 0 \\ (x - a) / -c & \text{при } x \neq 0 \text{ і } b = 0 \\ a + x / -c & \text{у решті випадків} \end{cases}$$

**Варіант 17**

$$F = \begin{cases} a \cdot x^2 - c \cdot x + b & \text{при } x + 10 < 0 \text{ і } b \neq 0 \\ (x - a) / (x - z) & \text{при } x + 10 > 0 \text{ і } b = 0 \\ -x / (a - z) & \text{у решті випадків} \end{cases}$$

**Варіант 18**

$$F = \begin{cases} a \cdot x^3 + b \cdot x^2 & \text{при } x < 0 \text{ і } b \neq 0 \\ (x - a) / (x - z) & \text{при } x > 0 \text{ і } b = 0 \\ (x + 5) / (c \cdot (x - 10)) & \text{у решті випадків} \end{cases}$$

**Варіант 19**

$$F = \begin{cases} a \cdot (x + 7)^2 - b & \text{при } x < 5 \text{ і } b \neq 0 \\ (x - c \cdot d) / a \cdot x & \text{при } x > 0 \text{ і } b = 0 \\ x / -c & \text{у решті випадків} \end{cases}$$

**Варіант 20**

$$F = \begin{cases} -(2 \cdot x - z) / (c \cdot x - a) & \text{при } x < 0 \text{ і } b \neq 0 \\ (x - a) / (x - z) & \text{при } x > 0 \text{ і } b = 0 \\ (x / c + -c / 2) \cdot x & \text{у решті випадків} \end{cases}$$

Міністерство освіти і науки України  
Харківський національний економічний університет  
Кафедра інформаційних систем

Звіт  
про лабораторну роботу №1  
з дисципліни "Основи програмування та алгоритмічні мови"  
на тему: "Вирішення на комп'ютері завдань лінійного характеру"

Виконав:  
студент 1 курсу 1 групи  
факультету ЕІ  
Никамунінужний  
Варфоломей Полуєктовіч

Перевірив:  
доцент кафедри ІС  
Преподаватель А. В.

м. Харків  
2008

**Завдання 1. Варіант 5**

**Завдання:** Розробити програму для знаходження суми двох цілих чисел.

**Специфікація програмних вимог****Формулювання**

Задано два довільні цілі числа. Обчислити суму цих чисел і вивести її на екран.

**Вхідні дані**

Два цілі числа, кожне за модулем не перевищує 109. Прочитуються з клавіатури. Завжди коректні.

**Обробка**

Введені числа складаються.

**Вихідні дані**

Ціле число виводиться на екран.

**Призначений для користувача інтерфейс**

Введіть перше число

10

Введіть друге число

20

Сума дорівнює 30.

**План тестування**

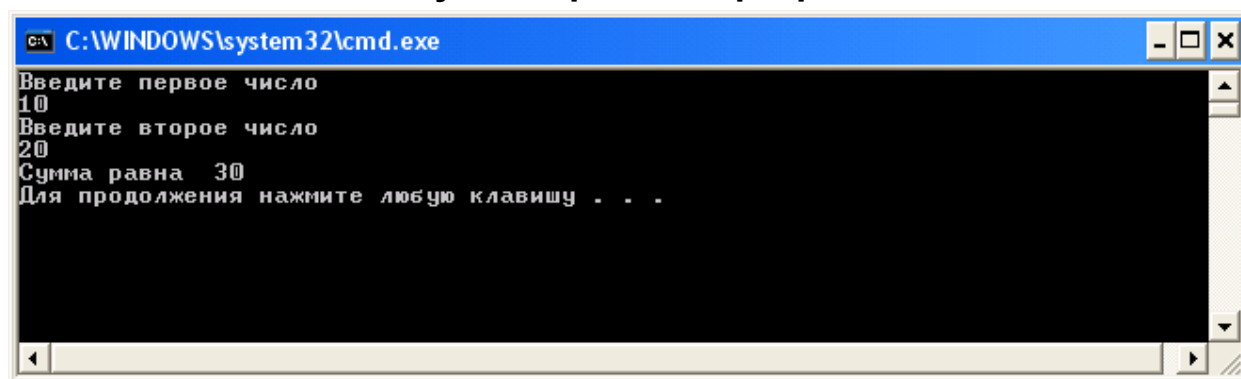
№	Вхідні дані		Очікуваний результат	Отриманий результат	Відмітка про проходження тесту
	1-е число	2-е число			
1	2	2	4		
2	1234	4321	5555		

**Текст програми**

```
#include <math.h>
```

```
#include <iostream>
int main()
{
    int a,b;
    std::cout << "Введіть перше число" << endl;
    std::cin >> a;
    std::cout << "Введіть друге число" << endl;
    std::cin >> b;
    std::cout << "\nСума дорівнює" << abs(a+b);
    getchar();
    return 0;
}
```

### Результат роботи програми



### Висновки за роботою

У ході виконання лабораторної роботи 1 я вивчив методику вирішення на комп'ютері завдань лінійного характеру, вивчив можливості мови C++ з введення інформації з клавіатури і виводу на екран, вивчив порядок роботи з математичними функціями (бібліотека Math), отримав практичні навички роботи в середовищі Microsoft Visual Studio 2005.

## НАВЧАЛЬНЕ ВИДАННЯ

**Методичні рекомендації  
до виконання лабораторних робіт  
з навчальної дисципліни  
"ОСНОВИ ПРОГРАМУВАННЯ  
ТА АЛГОРИТМІЧНІ МОВИ"**

**для студентів напряму підготовки  
"Комп'ютерні науки"  
всіх форм навчання**

## Частина 1

Укладачі: Парфьонов Юрій Едуардович  
Федорченко Володимир Миколайович  
Лосєв Михайло Юрійович  
Щербаков Олександр Всеволодович

Відповідальний за випуск **Пономаренко В. С.**

Редактор Грицай І. М.

Корректор Грицай І. М.

План 2008 р. Поз. №216.

Підп. до друку Формат 60 × 90 1/16. Папір MultiCopy. Друк Riso.

Ум.-друк. арк. 8,0. Обл.-вид. арк. 10,0. Тираж                      прим. Зам. №

Видавець і виготівник — видавництво ХНЕУ, 61001, м. Харків, пр. Леніна, 9а

Свідоцтво про внесення до Державного реєстру суб'єктів видавничої справи  
Дк №481 від 13.06.2001 р.

**Методичні рекомендації  
до виконання лабораторних робіт  
з навчальної дисципліни  
"ОСНОВИ ПРОГРАМУВАННЯ  
ТА АЛГОРИТМІЧНІ МОВИ"  
для студентів напряму підготовки  
"Комп'ютерні науки"  
всіх форм навчання  
Частина 1**