

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

В.А. Данілова

ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ. ПРАКТИКУМ

*Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів ступеня бакалавра
за спеціальністю 163 «Біомедична інженерія»*

Київ
КПІ ім. Ігоря Сікорського
2021

Рецензент *Богомолів М.Ф.*, к.т.н., доцент кафедри БМІ КПІ ім. Ігоря Сікорського,
Дубко А.Г., к.т.н., доцент, наук. співроб. відд. зварювання та споріднених технологій в медицині та екології Інституту електрозварювання ім.Є.О.Патона

Відповідальний редактор *Зубчук В.І.*, к.т.н., доц., доцент кафедри біомедичної інженерії КПІ ім. Ігоря Сікорського

*Гриф надано Методичною радою КПІ ім. Ігоря Сікорського
(протокол № 1 від 16.09.2021 р.)
за поданням Вченої ради факультету біомедичної інженерії
(протокол № 16 від 30.08.2021 р.)*

Данілова Валентина Анатоліївна, старший викладач

ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ. ПРАКТИКУМ

«Об'єктно-орієнтоване програмування. Практикум»: навч. посіб. для студентів спеціальності 163 «Біомедична інженерія» / КПІ ім. Ігоря Сікорського; уклад. В.А. Данілова.– КПІ ім. Ігоря Сікорського.- 2021. – 121 с.

Навчальний посібник розроблено для отримання студентами практичних навичок з програмування мовою C++. Навчальне видання призначене для студентів, які навчаються за спеціальністю 163 – «Біомедична інженерія» факультету біомедичної інженерії КПІ ім. Ігоря Сікорського.

© В.А. Данілова, 2021

© КПІ ім. Ігоря Сікорського, 2021

ЗМІСТ

ВСТУП	4
ПРАКТИЧНА РОБОТА №1	
Тема роботи: Робота в інтегрованому середовищі розробки Microsoft Visual Studio	5
ПРАКТИЧНА РОБОТА №2	
Тема роботи: Ввід та вивід інформації в C++. Потокові операції мови C++	14
ПРАКТИЧНА РОБОТА №3	
Тема роботи: Лінійні програми. Обчислення арифметичних виразів та математичних функцій	23
ПРАКТИЧНА РОБОТА №4	
Тема роботи: Оператори розгалуження	33
ПРАКТИЧНА РОБОТА №5	
Тема роботи: Прості цикли із відомим числом повторів	44
ПРАКТИЧНА РОБОТА №6	
Тема роботи: Прості цикли із невідомим числом повторів	51
ПРАКТИЧНА РОБОТА №7	
Тема роботи: Одновимірні масиви	57
ПРАКТИЧНА РОБОТА № 8	
Тема: Розробка програм з використанням двовимірних масивів	65
ПРАКТИЧНА РОБОТА №9	
Тема: Функції.....	74
ПРАКТИЧНА РОБОТА №10	
Тема: Рядки	81
ПРАКТИЧНА РОБОТА № 11-12	
Тема: Структури. Масиви структур.....	87
ПРАКТИЧНА РОБОТА №13	
Тема: Класи	98
ПРАКТИЧНА РОБОТА № 14-15	
Тема: Поліморфізм. Перевантаження функцій, операторів і методів класу	107
ПЕРЕЛІК РЕКОМЕНДОВАНОЇ ТА ВИКОРИСТАНОЇ ЛІТЕРАТУРИ.....	115
ДОДАТКИ.....	117

ВСТУП

Основне завдання цього навчального посібника – навчити студентів розробляти програми мовою C++ з використанням технології структурного програмування. Не секрет, що протягом декількох останніх десятиліть на ринку програмного забезпечення відбулися значні зміни, які є очевидними навіть для не професіонала. Зокрема, мова C++ стала універсальною мовою програмування високого рівня з підтримкою декількох сучасних парадигм програмування: об'єктно-орієнтованої, узагальненої та процедурної.

Дисципліна «Об'єктно-орієнтоване програмування» призначена для оволодіння технікою об'єктно-орієнтованого програмування на основі мови програмування C++.

Дисципліна знайомить студентів із сучасним методами та принципами розробки програмного забезпечення ПК: операційною системою Windows, інтегрованим середовищем розробки Visual Studio .NET, об'єктно-орієнтованим програмуванням та мовою C++.

У цьому посібнику вивчення технології програмування починається з найпростіших прикладів розроблення програм, поступово ускладнюючи їх, і завершуючи повноцінними структурними програмами.

Методичні вказівки включають низку практичних робіт, під час виконання яких студенти мають можливість отримати досвід роботи з інтегрованим середовищем розробки Visual Studio .NET.

Перед виконанням практичної роботи студенти повинні: ознайомитися з методичними вказівками; повторити лекційний матеріал, пов'язаний з практичною роботою; підготувати відповіді на питання, які наведені у методичних вказівках наприкінці кожної практичної роботи.

Виконавши ці завдання, студент повинен продемонструвати викладачеві роботу на комп'ютері, оформити звіт за результатами даної практичної роботи, захистити його та здати викладачеві.

ПРАКТИЧНА РОБОТА №1

Тема роботи: Робота в інтегрованому середовищі розробки Microsoft Visual Studio

Мета роботи: формування навичок організації роботи в середовищі Microsoft Visual Studio при написанні та відлагодженні програм на C++.

Теоретичні відомості

Microsoft Visual Studio являє собою інтегровані в єдину оболонку засоби, що дозволяють створювати, відкривати, переглядати, редагувати, зберігати, компілювати, відлагоджувати та виконувати програми на C++. Будь-яка програма, що створюється в середовищі Microsoft Visual Studio, оформляється у вигляді окремого проекту.

Проект – це набір взаємопов’язаних вихідних файлів, призначених для розв’язку певної задачі. Їх компіляція та компоновка дозволяє отримати готову до виконання програму. До проекту входять як файли, що безпосередньо створюються програмістом, так і файли, які створюються та редагуються самим середовищем. Після запуску Microsoft Visual Studio відкривається головне вікно програми, приведене на рис.1.1 (в залежності від налаштувань його вигляд може дещо відрізнятися від приведенного).

Робочий стіл Microsoft Visual Studio складається із трьох вікон:

1. вікно робочої області (*Project Workspace*), де відображається список файлів текучого проекту;
2. вікно редактора (*Editor*) для вводу та редагування вихідного коду на мові C++;
3. вікно виводу (*Output*), куди виводяться повідомлень про хід компіляції та компоновки програми. Зокрема в цьому вікні виводяться всі повідомлення про помилки на етапах компіляції та компоновки програми.

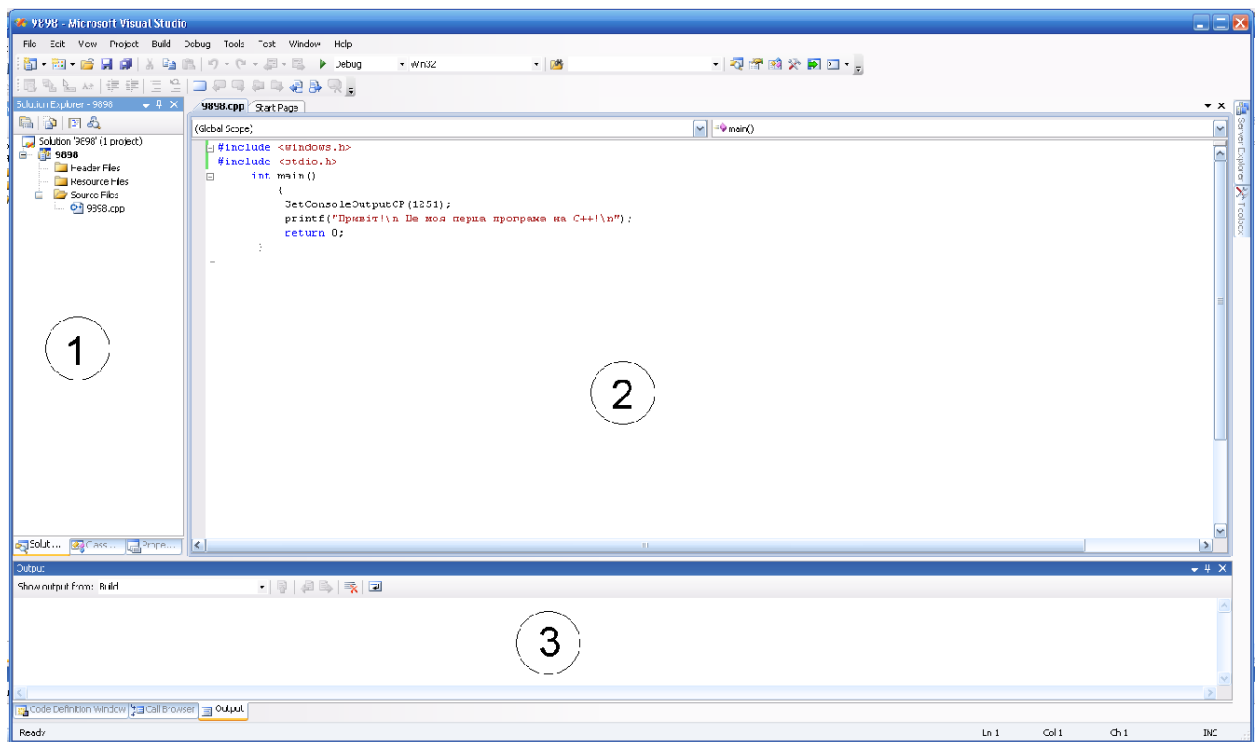


Рис.1.1 Вигляд вікна середовища Microsoft Visual Studio після створення нового проекту.

Середовище дозволяє будувати як проекти – віконні *Windows* – *додатки* із графічним інтерфейсом, так і проекти – *консольні додатки*.

Консоль – це інтерфейс, що використовується програмою, яка працює в текстовому режимі. Програма має вхідний (зв'язаний із клавіатурою) та вихідний (зв'язаний з екраном монітора) буфери і взаємодіє з Windows через консоль, для роботи якої створюється вікно, властивості якого аналогічні звичайному вікну Windows. Проект – консольний додаток є найзручнішим для вивчення мови програмування C++.

Для створення нового проекту типу *консольний додаток* необхідно виконати наступні дії:

- у рядку меню головного вікна Microsoft Visual Studio вибрати команду *File-New- Projects*;
- у діалоговому вікні *New Projects* вибрати тип проекту: *Visual C++- Win32*;
- задати тип створюваного проекту - *Win32 Console Application*;
- в полі *Project Name* ввести ім'я проекту (наприклад *Lab1*);

- в полі *Locations* вибрати каталог, де він буде зберігатися (діалогове вікно *Browse*);
- клікнути мишкою по кнопці *Ok*;
- у вікні майстра додатків *Win32 Console Application* вибрати пункт меню *Application Settings* та задати порожній проект (*Empty project*);
- клікнути мишкою по кнопці *Finish*.

В результаті буде створено порожній консольний проект. До нього необхідно додати новий, або вже існуючий файл з текстом програми на C++. Для того, щоби додати новий файлу до проекту, можна скористатися командою *Project - Add New Item* : в полі *Name* ввести ім'я файлу (для зручності бажано задавати ім'я, що співпадає із іменем самого проекту) та клікнути мишкою по кнопці *Add*.

Для того, щоби додати вже існуючий файл із програмою на C++, необхідно попередньо скопіювати його в робочу папку проекту, скористатися командою *Project - Add Existing Item* та вибрати відповідний файл. У папці проекту можуть знаходитися декілька файлів та дві вкладені папки.

Файли та папки призначені для:

- файл із розширенням *.dsw* – файл проекту, що об'єднує всі файли, які входять до проекту (його може і не бути);
- файл із розширенням *.dsp* – для побудови окремого проекту або підпроекту;
- файл із розширенням *.opt* – містить всі налаштування даного проекту;
- файл із розширенням *.ncb* – службовий файл;
- *Debug* – папка, в якій знаходяться файли, створені на етапі компіляції та компоновки програми; виконуваний файл із розширенням *.exe* (якщо компіляція та компоновка програми пройшла успішно);
- папка із іменем програми на C++ включає в себе файл із розширенням *.cpp* – файл тексту програми на C++.

Компіляцію та компоновку проекту (програми на C++) можна виконати за допомогою меню головного вікна *Build* або за допомогою відповідних кнопок панелі інструментів.

Основними командами меню *Build* є:

- *Compile (Ctrl+F7)* – компіляція файлу із програмою C++, що є складовою частиною проекту. Повідомлення про помилки на етапі компіляції виводиться у вікно *Output*. Якщо помилок на етапі компіляції не виявлено, то буде створено об'єктний файлу із розширенням *.obj*;

- *Build Solution (F7)* – компоновка проекту. Компілюються всі файли, в яких відбулися зміни з моменту останньої компоновки. Після компіляції проходить компоновка (*link*) всіх об'єктних модулів, в тому числі бібліотечних, та створюється виконуваний файл. Повідомлення про помилки на етапі компоновки виводяться у вікно *Output*. Якщо обидва етапи завершилися без помилок, то буде створено виконуваний файл із розширенням *.exe*, який можна запустити на виконання. Запустити проект на виконання можна за допомогою меню головного вікна *Debug* (або за допомогою відповідних кнопок панелі інструментів):

- *Start Without Debugging (Ctrl+F5)* – старт виконуваного файлу, створеного в результаті компоновки проекту. В результаті відкривається консольне вікно для роботи з програмою. Для відкриття проекту, над яким ви працювали раніше, виберіть пункт меню *File – Open – Project/Solution*, знайдіть папку із вашим проектом, а в ній виберіть файл із розширенням *.dsw*.

Алфавіт, лексеми, синтаксис мови

У природній мові спілкування виділяють чотири основні елементи: символ, слово, словосполучення та речення. Подібні елементи існують і в алгоритмічній мові, тільки слова мають назву лексеми, словосполучення — вирази, а речення — оператори. Лексеми створюються із символів, вирази — із лексем та символів, оператори — з символів, виразів і лексем.

Алфавіт мови C++ включає:

- великі (A-Z) і малі (a—z) літери латинського алфавіту та символ підкреслення (_);
- арабські цифри від 0 до 9;
- знаки арифметичних дій +, -, *, /, %, ++, —;
- знаки побітових операцій <<, >>, &, |, ~, ^;
- знаки відношень <, <=, ==, !=, >, >=;
- знаки логічних операцій &&, ||, !;
- розділові знаки , ; : пропуск;
- спеціальні знаки ., =, ->, ?, \, \$, #, ‘, “;
- символи дужок (,), [,], {, }.

Інші символи, а також літери кирилиці не використовуються для побудови базових елементів мови або для їх розділу, але вони можуть застосовуватись у символічних константах та коментарях.

Лексеми, тобто базові елементи мови з певним самостійним значенням, складаються із символів алфавіту. До них відносять ідентифікатори, ключові слова, знаки операцій, константи, роздільники (дужки, крапка, кома, символи пропуску). Межі лексем визначаються іншими лексемами-роздільниками або знаками операцій.

Ідентифікатором, тобто ім'ям програмного об'єкта, називається будь-яка послідовність літер латинського алфавіту, цифр і символу підкреслення за умови, що першою стоїть літера або символ підкреслення, а не цифра.

Існує два різновиди ідентифікаторів:

- *стандартні*, наприклад, імена всіх вбудованих у мову функцій;
- *користувальницькі*.

Характерно, що мова C++ чуттєва до регістру літер, тому компілятор розпізнає великі і малі літери латинського алфавіту як різні символи. Це дає можливість створювати ідентифікатори, що однаково читаються, але

відрізняються написом одного або декількох символів. Наприклад, ідентифікатори «Sigma», «sigma» і «sigMa» вважаються різними.

Ідентифікатори можуть мати будь-яку довжину, але значимими є не більше 31 символу від початку ідентифікатора, а в деяких компіляторах це обмеження ще більш суворе (не більше 8 символів). Імена програмних об'єктів створюються на етапі оголошення даних, після цього їх можна використовувати в різних операторах програми.

Надамо декілька порад щодо вибору ідентифікатора:

- рекомендується не жалкувати часу на створення ідентифікаторів переважно за їх змістовим призначенням;
- ім'я програмного об'єкта повинно легко розпізнаватись і, бажано, не мати символів, які можна переплутати між собою;
- ідентифікатор не повинен збігатися з ключовими словами, а також з іменами стандартних об'єктів мови C++;
- не слід починати ідентифікатори із символу підкреслення, бо вони можуть співпадати з іменами системних функцій або змінних;
- для розділу частин імені можна використовувати символ підкреслення;
- всередині ідентифікатора не можна розміщати символи пропуску.

Ключовими (службовими) словами називають ряд зарезервованих ідентифікаторів, що вживаються для побудови конструкцій мови і мають фіксоване значення. За смисловим навантаженням службові слова поділяються на такі основні групи:

- специфікатори типів — char, int, long, typedef, short, float, double, enum, struct, union, signed, unsigned, void;
- кваліфікатори типів — const і volatile;
- класи пам'яті — auto, extern, register, static;
- для побудови операторів — for, while, do, if, else, switch, case, continue, goto, break, return, default, sizeof.

У табл. 1.1 наведено список основних ключових слів C++.

Ключові слова C++

asm	delete	goto	register	throw
auto	do	if	return	try
break	double	inline	short	typedef
case	else	int	signed	typename
catch	enum	long	sizeof	union
char	explicit	new	static	unsigned
class	extern	operator	struct	virtual
const	float	private	switch	void
continue	for	protected	template	volatile
default	friend	public	this	while

Як роздільники лексем застосовуються такі символи: пропуск, табуляція, символ нового рядка, коментар. Між будь-якими двома лексемами допускається довільна кількість символів-роздільників. Крім того, деякі лексеми («*», «+», «,», « », «(», «->» тощо) самі є роздільниками і відділяти їх від інших лексем символами-роздільниками необов'язково.

Завдання для виконання

1. Запустіть інтегроване середовище розробки Microsoft Visual Studio та створіть новий консольний додаток. Проект зберігайте на диску у своїй папці. Додайте до проекту новий файл та введіть програму на C++ :

```
#include <iostream>

int main(){
    cout<< "Hello";
    return 0;
}
```

```
#include <iostream>

int main(){
    cout<< "Hello";
    cout<<endl;
    return 0;
}
```

2. Скомпілюйте введену програму, задавши відповідну команду. Якщо програма набрана правильно, то у вікні виводу результатів компіляції буде вказано, що помилок (*errors(s)*) та попереджень (*warnings(s)*) немає (стоять нулі) та що створено виконуваний файл (*Build: 0 succeeded, 0 failed, 1 up-to-date, 0 skipped*). Якщо при наборі програми були допущені помилки, то будуть видані відповідні повідомлення. Якщо у вікні виводу (*Output*) вибрати таке повідомлення та натиснути *Enter* (або два рази клікнути лівою клавішею миші), то у вікні програми рядок із помилкою буде відмічено маркером, а курсор буде поміщено у даний рядок. За повідомленням про помилку слідує номер помилки та її короткий опис. Компілятор не завжди може точно локалізує положення помилки, особливо якщо пропущено дужки чи крапки з комою – помилка може знаходитись дещо вище від вказаного місця. Попередження (*warnings(s)*), як правило, не є критичними, а виконуваний файл може бути створений та виконаний (бажано, щоби їх не було).

3. Модифікуймо нашу програму до вигляду:

```
#include <iostream.h>

#include <conio.h>

int main()
{
    char name[20];
    cout<< "What is your name:\n";
    cin >> name;
    cout<< "Hello: " <<name << endl;
    getch();
}
```

```
}
```

`#include <conio.h>` - директива для роботи з екраном

`getch()` -функція для затримки екрана до натиснення будь-якої клавіші.

Після запуску вона повинна вивести на екрані питання *What is your name:*, ми відповідно повинні ввести ім'я, наприклад *Ivan* та натиснути клавішу Enter, на це програма повинна вивести: *Hello:Ivan*.

4. Запишіть в звіті модифіковану програму з коментарем біля кожного рядка програми.

Наприклад:

```
#include <iostream.h> //підключення бібліотеки для введення-виведення інформації
```

Контрольні питання:

1. З чого складається алфавіт мови C++?
2. Які службові слова використовує мова C++?

ПРАКТИЧНА РОБОТА №2

Тема роботи: Ввід та вивід інформації в C++. Поточкові операції мови C++

Мета роботи: Формування навичок та умінь організації операцій форматного вводу-виводу інформації засобами C++.

Теоретичні відомості

Основними частинами типової структури програми на C++ є такі:

- директиви препроцесорної обробки;
- опис зовнішніх змінних (вихідних даних і результатів) та функцій;
- функції програми;
- головна функція — програми **main()**, що має вигляд:

```
main()
{
    опис змінних;
    виконавчі оператори;
}
```

Програма завжди починається директивами препроцесору (**#**), які обробляється ним до початку компіляції. Задачею препроцесора є доповнення тексту програми бібліотечними функціями чи об'єктами C++, описаними у відповідних заготовочних (*header*) файлах. Найбільш часто використовувані заголовочні файли та їх призначення приведено в додатку №1.

Програма на мові C++ складається із набору окремих функцій. Обов'язковою є функція **main()** – вхідна точка програми (місце її розміщення у програмі значення немає):

директиви препроцесора

...

```

# директиви препроцесора

функція a()
{
    тіло функції a
}

функція b()
{
    тіло функції b
}

...

void main()

/* функція, з якої починається
виконання програми. */

// void – функція не повертає ніяких значень
{
    тіло функції
}

```

Текст, що знаходиться між дужками `/*` та `*/` є коментарем до програми та на етапі компіляції ігнорується. Текст, що слідує за `//` є однорядковим коментарем (до кінця даного рядка). Такі коментарі можуть використовуватися в будь-яких місцях програми для її «документування».

Тіло функції – послідовність описів змінних та виконуваних операторів, взятих у фігурні дужки. Кожний опис чи оператор повинен завершуватися крапкою з комою. Всі змінні, що використовуються в програмі, повинні бути оголошені до їх першого використання (див. додаток №2). Для іменування змінних використовують ідентифікатори. При їх формуванні можуть використовуватися великі та малі букви англійського алфавіту (*A* та *a* сприймаються як різні символи), цифри та символи підкреслення. Ідентифікатор не може починатися із цифри чи містити в собі пробіли.

Ідентифікатори також не можуть співпадати із ключовими словами мови C++.

Синтаксис оголошення змінних є наступним:

<тип> <імя> = <значення>

де:

<тип> – тип змінної (один із базових);

<імя> – ідентифікатор змінної;

<значення> – значення, яке буде присвоєно змінній після її ідентифікації

(не є обов'язковим).

Наприклад:

int a;

unsigned int b=2300;

double x, y;

double N_Avag =6.022045e23;

char c1='a', c2 = 'A';

*int c3 =5+5*2;*

const float pi = 3.1415926;

Ключове слово **const** вказує на величину, яка не міняє свого значення в ході виконання програми. Кожна змінна має певну область видимості. Якщо змінна оголошена за межами будь-якої функції, то вона називається глобальною і може використовуватися (є видима) в будь-якому місці програми. Якщо змінна оголошена в програмному блоці (чи деякій функції), то вона є локальною і може використовуватися (видима) тільки в межах даного блоку (функції).

У C++ немає вбудованих засобів для організації вводу-виводу. Вони здійснюються за допомогою функцій вводу-виводу, опис яких містяться міститься у заголовочних файлах **stdio.h** та **iostream**, які підключаються на початку програми за допомогою директиви препроцесора **#include**.

При використанні файлу **stdio.h** (ввід-вивід у стилі C) для вводу використовується функція

scanf(<список форматів>,<список вводу>)

де:

scanf – ім'я функції форматного вводу;

<список форматів> – специфікатори, що задають формат вводу даних того чи іншого типу (див додаток №3);

<список вводу> – через кому адреси змінних (***&<список вводу>***), в які вводяться дані із клавіатури. Кількість специфікаторів форматів та їх тип повинні відповідати кількості та типу адрес змінних, значення яких необхідно ввести. Символ ***&*** означає унарну операцію одержання адреси змінної. Для вводу значень рядкових змінних ***&*** не використовується.

Ввід символічних даних із клавіатури здійснюється за допомогою наступних функцій:

getchar() – із відображенням введеного символу на екрані;

getch() – без відображення введеного символу на екрані.

Наприклад:

```
#include <stdio.h> //підключення файлу
```

```
void main()
```

```
{
```

```
int k,m; //опис змінних
```

```
scanf(" %d%d",&k,&m); //ввід даних
```

```
}
```

Вивід даних здійснюється за допомогою функції ***printf***:

printf(<список форматів>,<список виводу>)

Функція ***printf*** в якості ***<список форматів>*** використовує ті ж формати, що і функція вводу ***scanf*** (див додаток №3). Додатково можна використати наступні керуючі символи (escape-символи):

|n – перехід на новий рядок;

|r – повернення на початок рядка;

|v – вертикальна табуляція;

|t –горизонтальна табуляція.

В *<список виводу>* через кому задаються ідентифікатори змінних, значення яких виводяться. Для виводу на екран символу служить функція *putchar()*.

Приклад програми, що ілюструє використання операцій вводу-виводу:

```
#include<stdio.h>
#include <windows.h>
void main()
{
    char ch='I';
    int k,m; //опис змінних
    float a; //опис змінних
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    putchar(ch); //вивід символу на екран
    printf("\nвведіть дані\n"); //вивід даних на екран
    scanf("%i%i%f",&k,&m,&a ); //ввід даних
    printf("\nk=%d\tm=%d\ta=%f\n",k,m,a); /*вивід даних на екран*/
}
```

Результат роботи програми:

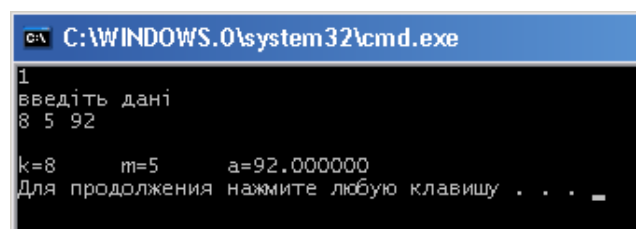


Рис. 2.1 – Результат роботи програми

Операції вводу - виводу можна також реалізувати із використанням бібліотеки потокового вводу - виводу *iostream* (функції вводу – виводу з використанням класів C++):

- ввід даних із клавіатури: *cin>><ідентифікатор змінної>* ;
- вивід даних на екран: *cout<< <вираз>*.

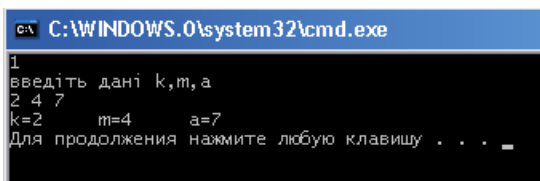
тут *<вираз>* - ідентифікатор змінної, рядок символів або арифметичний вираз.

Для включення у потік символу нового рядка (еквівалентний *\n*) служить *endl*.

Приклад програми, що ілюструє використання операцій потокового вводу - виводу :

```
#include<iostream>
#include <windows.h>
using namespace std;
void main()
{
    char ch='l';
    int k,m; //опис змінних
    float a; //опис змінних
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    cout << ch; //вивід символу на екран
    //ввід підказки для вводу даних
    //та перехід на новий рядок
    cout <<endl<< "введіть дані k,m,a"<<endl;
    cin >> k >> m >> a ; //ввід даних
    //вивід даних та перехід на новий рядок
    cout << "k=" << k << "\tm=" << m << "\ta=" << a
    <<endl;
}
```

Результат роботи програми:



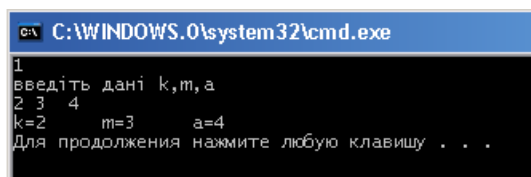
```
cmd C:\WINDOWS.0\system32\cmd.exe
1
введіть дані k,m,a
2 4 7
k=2      m=4      a=7
Для продолжения нажмите любую клавишу . . .
```

Рис. 2.2 – Результат роботи програми

Примітка: директива **using namespace std** вказує, що ми будемо працювати із іменами зі стандартної бібліотеки. При відсутності такої директиви замість **cin** необхідно писати **std::cin**, а замість **cout** - **std::cout**. Попередній приклад при відсутності такої директиви матиме вигляд (тут також замість потокового символ нового рядка **endl** використано керуючий символ функції **printf** переходу на новий рядок **\n**):

```
#include<iostream>
#include <windows.h>
void main()
{
    char ch='I';
    int k,m; //опис змінних
    float a; //опис змінних
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    std::cout << ch; //вивід символу на екран
    std::cout << "\nвведіть дані k,m,a\n";
    //вивід даних на екран
    std::cin >> k >> m >> a ; //ввід даних
    std::cout <<"k="<< k << "\tm="<<m<< "\ta="<< a << "\n";
    //вивід даних на екран
}
```

Результат роботи програми:



```
C:\WINDOWS.0\system32\cmd.exe
1
введіть дані k,m,a
2 3 4
k=2    m=3    a=4
Для продолжения нажмите любую клавишу . . .
```

Рис. 2.3 – Результат роботи програми

Потокові операції вводу - виводу можуть бути пов'язані не тільки із клавіатурою та екраном, а і із зовнішніми файлами. Для цього в командному рядку виклику програми (*exe* – файл) необхідно задати імена відповідних файлів. Наприклад:

- *lab2.exe < dani.in* – при виконанні програми *lab2.exe* дані вводяться не з клавіатури, а читаються з файлу *dani.in*;

- *lab2.exe > dani.out* – при виконанні програми *lab2.exe* дані виводяться не на екран, а в файл *dani.out*;

- *lab2.exe < dani.in > dani.out* – при виконанні програми *lab2.exe* вхідні дані читаються із файлу *dani.in*, а результат виводиться у файл *dani.out*.

При створенні програми враховують такі основні вимоги:

- усі використані константи, змінні, функції та нестандартні типи повинні бути оголошеними (описаними) до їхнього першого використання, і ці оголошення можна розміщати в будь-якому місці програми;
- кожний оператор мови закінчується символом «;»;
- фігурні дужки (« { » та « } ») виділяють складений оператор і все, що подано між такими дужками, синтаксично сприймається як один оператор;
- вкладені блоки повинні мати відступ у 3-4 символи, при цьому блоки одного рівня вкладеності слід вирівняти за вертикаллю.

Завдання для виконання

Написати програми, що вводять-виводять змінні всіх стандартних типів (див. додаток №2). Перша програма – із використанням директиви препроцесору *#include <stdio.h>*, а друга – із використанням директиви *#include <iostream>*. Використати також операції форматного виводу даних та перенаправлення потоків для роботи із файлами.

Примітка: якщо на виконання запускається **exe** - файл, то для можливості перегляду результатів виконання програми необхідно зупинити закриття консольного вікна командою **system("pause")**, добавивши її в кінці програми!!!

Контрольні питання:

1. Що являє собою структура програми?
2. Наведіть основні вимоги, які слід враховувати при створенні програм мовою C++.

ПРАКТИЧНА РОБОТА №3

Тема роботи: Лінійні програми. Обчислення арифметичних виразів та математичних функцій

Мета роботи: Формування навичок та умінь програмування арифметичних виразів, обчислення математичних функцій та написання простих лінійних програм.

Теоретичні відомості

Оператор присвоєння = дозволяє замінити значення операнду, що стоїть зліва від знаку рівності, значенням виразу, обчисленим справа від нього.

Синтаксис оператора присвоєння:

<ідентифікатор змінної>=<вираз>

Допустимим є запис виду:

<ідентифікатор змінної1>=<ідентифікатор змінної2>=<вираз>

Основні математичні операції C++ приведено в додатку №4. У складних виразах порядок виконання операцій визначається дужками та пріоритетом операцій. Можна використовувати декілька рівнів вкладення дужок: обчислення проходить від внутрішніх дужок до зовнішніх. Порядок виконання операцій у виразі має значення при наявності декількох операцій із різним пріоритетом: операції із однаковим пріоритетом виконуються раніше операцій із нижчим пріоритетом незалежно від того, де вони знаходяться у виразі.

Операції виконуються зліва направо в порядку їх пріоритету від найвищого до найнижчого:

1. виклик математичних функцій;
2. ++, --;
3. *, /, %;
4. +, -;
5. операції присвоєння: =, +=, -=, *=, /=, %=.

В операторі присвоєння змінні, що входять до складу виразу, повинні бути одного типу. Якщо вони різного типу, то необхідно виконати перетворення змінних до одного типу. При цьому перетворення проводиться так, щоб змінні, що займають менший об'єм оперативної пам'яті були перетворені до типу змінних, що займають більший об'єм.

Наприклад:

```
int a;
```

```
float y,b;
```

```
y=a+b;
```

У операторі присвоєння $y=a+b$ необхідно писати:

```
y=float(a)+b.
```

Операція ***float(a)*** називається операцією приведення типів: вона перетворює змінну цілого типу до дійсного типу. При виконанні операції приведення типів може відбуватися втрата інформації.

Наприклад, в результаті виконання приведенного нижче фрагменту програми змінна ***k*** прийме значення 4.

```
double a=4.24;
```

```
int k;
```

```
k=int(a);
```

Для «зв'язування» декількох виразів використовується операція слідування, (кома). Вирази, розділені між собою комами, обчислюються зліва направо.

Наприклад:

```
a=4, b=b+a+5, c=b/5;
```

Математичні функції знаходяться в бібліотеці ***math***, яка підключається на початку програми директивою препроцесора ***#include<math.h>*** (див. додаток №5). Необхідно пам'ятати, що всі тригонометричні функції в C++ працюють із кутовим величинами, заданими в радіанах.

Приклади виконання завдання

Приклад №1

Написати програму обчислення арифметичного виразу та форматного виводу результату на екран. Сталі величини, що використовуються при обчисленні арифметичного виразу, задати як константи. На екран вивести a, v, x, z .

$$v = \sin(\sqrt[3]{2z+1}) - 2.34, z = \left| 2 - x \cdot \cos(a^2) \right| \text{ при } a = 2, x = 0.5.$$

```
#include<stdio.h>
#include <math.h>
#include <windows.h>
// Задання значення константи a
#define a 2.0
// Задання значення константи x
#define x 0.5
void main()
{
    double v, z;
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    //обчислення значення z
    z=fabs(2-x*cos(a*a));
    //обчислення значення v
    v=sin(pow(2*z+1,1/3.0))-2.34;
    printf("z=%f\tv=%e\n",z,v);
    printf("a=%f\tx=%e\n",a,x);
}
```

Результат роботи програми:

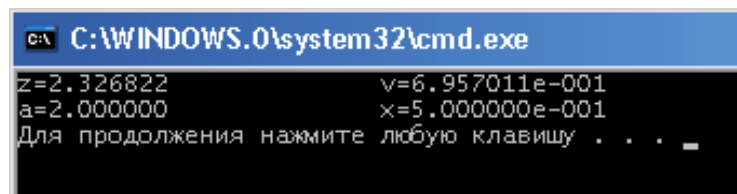


Рис. 3.1 – Результат роботи програми

Зверніть увагу, що для присвоєння константам a та x початкового значення використано директиву препроцесора *#define*. Ця директива використовується для заміни всіх наступних входжень констант (ключових слів, операторів, виразів) заданим значенням. Тип константи визначається компілятором за типом значення, заданого директивою. Аналогічний результат отримується в при виконанні наступної програми:

```
#include<stdio.h>
#include <math.h>
#include <windows.h>
void main()
{
    double v, z;
    // Задання значення константи a
    const double a=2.0;
    // Задання значення константи x
    const double x=0.5;
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    //обчислення значення z
    z=fabs(2-x*cos(a*a));
    //обчислення значення v
    v=sin(pow(2*z-1,2/3.0));
    printf("z=%f\tv=%e\n",z,v);
    printf("a=%f\tx=%e\n",a,x);
}
```

Приклад №2

Написати програму обчислення арифметичного виразу та виводу результату на екран за допомогою потокових операцій вводу-виводу.

$$h = \frac{x^{2y} + e^{y-1}}{1 + x|y - \operatorname{tg}(z)|} + 10 \cdot \sqrt[3]{x} - \ln(z)$$

При $x = 2.45$, $y = -0.423 \cdot 10^{-2}$, $z = 1.232 \cdot 10^3$ результат обчислення рівний $h = 6.9465$.

```
#include <iostream>
#include <math.h>
#include <windows.h>
using namespace std;
int main()
{
    double x,y,z,a,b,c,h;
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    cout << "Введіть x:";
    cin >> x;
    cout << "Введіть y:";
    cin >> y;
    cout << "Введіть z:";
    cin >> z;
    a=pow(x,2*y)+exp(y-1);
    b=1+x*fabs(y-tan(z));
    c=10*pow(x,1/3.0)-log(z);
    h=a/b+c;
    cout << "Результат обчислення h= "<<h<<endl;
    return 0;
}
```

Примітка: для простоти та усунення можливих помилок обчислення h розбито на два етапи: вихідний вираз розбивається на три підвирази (a , b , c), а потім обчислюється результуюче значення $h=a/b+c$.

Результат роботи програми:

```

C:\WINDOWS.0\system32\cmd.exe
Введіть x:2.45
Введіть y:-0.423e-2
Введіть z:1.232e3
Результат обчислення h= 6.9465
Для продовження натисніть будь-яку клавішу . . .
  
```

Рис. 3.2 – Результат роботи програми

Завдання для виконання

Завдання №1

При написанні програми використати форматний вивід та пояснювальну текстову інформацію для зручного візуального сприйняття результатів.

1. Обчислити $b = \sin^3(x) - a$, $z = |1 - \sqrt{a} \cdot \cos(b)|$ при $a = 2$, $x = 0.5$. На екран вивести a, x, b, z .
2. Обчислити $y = x^4 - \sqrt[3]{c}$, $d = 2y + \cos(c)$ при $x = 3$, $c = 2.5$. На екран вивести x, c, y, d .
3. Обчислити $f = \ln(3x) - h$, $e = \sqrt[5]{f^3}$ при $x = 1.6$, $h = 1.6$. На екран вивести x, h, f, e .
4. Обчислити $d = \frac{1}{(z - 2a)^2} - z^2$, $f = \sqrt[3]{d^2}$ при $z = 3.6$, $a = 2.6$. На екран вивести z, a, d, f .
5. Обчислити $h = \frac{1}{n} - e^n \sin(p)$, $y = \frac{h}{|h+1|}$ при $n = 2$, $p = 0.5$. На екран вивести n, p, h, y .

6. Обчислити $r = \sin(x) + \ln(s)$, $r_1 = \sqrt[2]{r^3}$ при $x = 0.6$, $s = 5.6$. На экран вывести x, s, r, r_1 .
7. Обчислити $m = \sin(x) + \cos(x)$, $n = \sqrt[2]{m} + \sin(x)$ при $x = 0.3$. На экран вывести x, m, n .
8. Обчислити $p = 3.62 \ln(x + 2.3)$, $k = p^3$ при $x = 1.6$. На экран вывести x, p, k .
9. Обчислити $t = \frac{\sqrt{x} + \sqrt{y}}{e^x}$, $z = 1 + t$ при $x = 0.5$, $y = 4.4$. На экран вывести x, y, t, z .
10. Обчислити $w = |1 - \sin(2x) + n|$, $t = 2w + \frac{1}{7^3}$ при $x = 0.9$, $n = 1.6$. На экран вывести n, x, w, t .
11. Обчислити $y = \sqrt{t^3 - 1} + 3.31$, $k = \sin(2y + 1.21)$ при $t = 2.7$. На экран вывести t, y, k .
12. Обчислити $t = \sqrt[3]{x} - e^{(x+1)} - 10.4$, $r = 3t - \frac{1}{2}$ при $x = 1.24$. На экран вывести x, t, r .
13. Обчислити $n = \frac{1}{(x^3 - e^{1/x})}$, $m = n^{\frac{3}{2}}$ при $x = 0.8$. На экран вывести x, n, m .
14. Обчислити $f = 0.45z^5 + \frac{1}{z}$, $z = 1 + \frac{1}{x^2}$ при $x = 1.1$. На экран вывести x, f, z .
15. Обчислити $d = \frac{3 \sin(x) - 1.1w}{2 \cos(x + 2.14)}$, $l = 1 + \sin(d)$ при $x = 2.6$, $w = \frac{\pi}{2}$, $\pi = 3.1415$. На экран вывести x, w, d, l .

Завдання №2

Обчислити значення виразу при заданих вхідних даних, які в програмі задати як константи. Отриманий результат порівняти із вказаним правильним.

При написанні програми використати потокові операції вводу-виводу та пояснювальну текстову інформацію для зручного візуального сприйняття результатів.

$$1. \quad s = \frac{2 \cos\left(x - \frac{2}{3}\right)}{\frac{1}{2} + \sin^2(y)} \left(1 + \frac{z^2}{3 - \frac{z^2}{5}} \right) \quad \text{при} \quad x = 14.26, y = -1.22, z = 3.5 \cdot 10^{-2}.$$

Результат обчислення $s = 0.749155$.

$$2. \quad x = \frac{\sqrt{\sin(wt + \varepsilon)} - e^{-wt}}{\sqrt[3]{\ln(2k + d) + d^{3k}}} \quad \text{при} \quad t = 1.6, w = -3.4, \varepsilon = -1.5 \cdot 10^{-3}, k = 2.1,$$

$d = 3.2$. Результат обчислення $x = -229.579$.

$$3. \quad s = \frac{\sqrt[3]{9 + (x - y)^2}}{x^2 + y^2 + 2} - e^{|x - y|} \lg^3(z) \quad \text{при} \quad x = -4.5, y = 0.75 \cdot 10^{-4},$$

$z = -0.845 \cdot 10^2$. Результат обчислення $s = -3.23765$.

$$4. \quad y = \frac{(\arctg x^3 + \cos \sqrt{x})^{2x}}{h^x + \ln|2.4x^3|} - a \quad \text{при} \quad x = 1.45, a = -5.89, h = 4.1. \quad \text{Результат}$$

обчислення $y = 6.30082$.

5. $q = \frac{1 + \sin^2(x+y)}{\left|x - \frac{2y}{1+x^2y^2}\right|} x^{|y|} + \cos^2\left(\arctg\left(\frac{1}{z}\right)\right)$ при $x = 3.74 \cdot 10^{-2}$, $y = -0.825$,
 $z = 0.16 \cdot 10^2$. Результат обчисления $q = 1.05534$.
6. $y = \left(\sqrt{\frac{ax+b}{c+dx}} + \sqrt{\arctg x}\right)^{2/3} - e^{2x}$ при $x = 1.6$, $a = 3.3$, $b = 7.245$, $c = 6.4$,
 $d = -1.45 \cdot 10^{-1}$. Результат обчисления $y = -22.7215$.
7. $w = |\cos(x) - \cos(y)|^{1+2\sin^2(y)} \left(1 + z + \frac{z^2}{2} + \frac{z^3}{3} + \frac{z^4}{4}\right)$ при $x = 0.4 \cdot 10^4$,
 $y = -0.875$, $z = -0.475 \cdot 10^{-3}$. Результат обчисления $w = 1.98727$.
8. $y = \sqrt[3]{\left(\frac{ax}{b+cx} + \tg x\right)^2} - e^{x^2}$ при $x = 4.22 \cdot 10^{-1}$, $a = 1.1$, $b = 2.45$, $c = 1.3$.
Результат обчисления $y = -2.80621$.
9. $k = \ln\left(y^{-\sqrt{|x|}}\right)\left(x - \frac{y}{2}\right) + \sin^2(\arctg(z))$ при $x = -15.246$, $y = 4.642 \cdot 10^{-2}$,
 $z = 21$. Результат обчисления $k = -182.038$.
10. $z = \frac{1}{1 - \frac{1 + e^{tx}}{\frac{r}{x}(j + x^{x^2})}}$ при $x = 3.004$, $t = 0.89$, $r = 2.05$, $j = 3.0$. Результат
обчисления $z = 1.00111$.
11. $v = \sqrt{10\left(\sqrt[3]{x} + x^{y+2}\right)}\left(\arcsin^2(z) - |x - y|\right)$ при $x = 16.55 \cdot 10^{-3}$, $y = -2.75$,
 $z = 0.15$. Результат обчисления $v = -40.6307$.
12. $f = \frac{\sqrt{8 + |x-6|^2 + b}}{\ln x + 2} + e^{x^k} (\ln x + 2)$ при $x = 1.7$, $b = 0.5$, $k = 2.0$. Результат
обчисления $f = 47.5873$.
13. $r = 5\arctg(x) - \frac{1}{4}\arccos(x) \frac{x + 3|x-y| + x^2}{|x-y|z} + x^2$ при $x = 0.1722$, $y = 6.33$,
 $z = 3.25 \cdot 10^{-4}$. Результат обчисления $r = -205.306$.

14. $y = \frac{1}{2} \cdot \ln \left| \frac{a + \sin x^2}{b - \cos^2 x} \right|$ при $x = 3.7, a = 33.01, b = 1.25 \cdot 10^2$. Результат обчисления $y = -0.649392$.
15. $h = \frac{e^{|x-y|} |x-y|^{x+y}}{\arctg(x) - \arctg(z)} + \sqrt[3]{x^6 + \ln^2(y)}$ при $x = -2.235 \cdot 10^{-2}, y = 2.23, z = 15.221$. Результат обчисления $h = 39.3741$.
16. $f = \frac{e^a + \ln |bx| + 5}{1 + \ln \sqrt{|dx|} + e^{x^2}}$ при $x = 1.34, a = 1.1, b = -2.55 \cdot 10^1, d = 3.98$. Результат обчисления $f = 1.46761$.
17. $q = \left| x^{\frac{y}{x}} - 3\sqrt{\frac{y}{x}} \right| + (y-x) \frac{\cos(y) - \frac{z}{(y-x)}}{1 + (y-x)^2}$ при $x = 1.825 \cdot 10^2, y = 18.225, z = -3.298 \cdot 10^{-2}$. Результат обчисления $q = 1.21308$.
18. $b = x + \frac{\sin^2 |y|}{3/5 + \cos^2 z^2}$ при $x = 8.5 \cdot 10^{-1}, y = -15.2, z = -1.4 \cdot 10^{-1}$. Результат обчисления $b = 0.9979$.
19. $s = 2^{-x} \sqrt{x + 4\sqrt{|y|}} \sqrt[3]{e^{x-1/\sin(z)}}$ при $x = 3.981 \cdot 10^{-2}, y = -1.625 \cdot 10^3, z = 0.512$. Результат обчисления $s = 1.26185$.
20. $y = \frac{3x^2 + 25e^{x^2}}{|x^7| + \sqrt{ax^2 + 2}} + \ln |x + k|$ при $x = 2.2, a = 1.2 \cdot 10^1, k = 1.76$. Результат обчисления $y = 13.7262$.
21. $k = y^{\sqrt[3]{|x|}} + \cos^3(y) \frac{|x-y| \left(1 + \frac{\sin^2(z)}{\sqrt{x+y}} \right)}{e^{|x-y|} + \frac{x}{2}}$ при $x = 6.251, y = 0.827, z = 25.001$. Результат обчисления $k = 0.712122$.
22. $y = \frac{\sqrt{a_0 + a_1 x + a_2 x^2}}{\sqrt[3]{a_2 |\sin x|}}$ при $x = 5, a_0 = 2, a_1 = 5.0, a_2 = 3$. Результат обчисления $y = 1.1012$.

ПРАКТИЧНА РОБОТА №4

Тема роботи: Оператори розгалуження

Мета роботи: Формування навичок та умінь розробки розгалужених програм для організації альтернативних обчислень.

Теоретичні відомості

Оператор умовного переходу служить для спрямування ходу обчислення по одній із можливих віток обчислення в залежності від істинності або хибності деякої умови. Він має дві форми запису.

Синтаксис повного умовного оператора переходу:

if(<умова>) <оператор1>

else <оператор2>

Якщо <умова> істина (*true*), то виконується <оператор1>, якщо хибна (*false*), виконується <оператор2>.

В якості <умови> можуть використовуватися операції відношення або логічні операції (див. додаток №6), арифметичний вираз чи просто ідентифікатор. Якщо як умова використовується арифметичний вираз, то коли він не рівний нулю (*true*) виконується <оператор1>, а коли рівний нулю (*false*) - <оператор2> .

Неповна форма запису умовного оператора, коли альтернативна вітка *else* відсутня:

if(<умова>) <оператор>

Якщо <умова> істинна, то виконується <оператор>, а якщо хибна, то <оператор> пропускається і виконується наступний оператор. В якості <оператора> може бути як простий, так і складений оператори.

Складеним називають два чи більше простих операторів, які виконуються як єдине ціле. Для створення складеного оператора використовують операторні

дужки $\{...\}$. Транслятор сприймає складений оператор як єдине ціле.
Наприклад:

```
{  
    pi=3.1415;  
    y=fabs(log(b-2*x*cos(pi/3.0)+x*x*log(d+x)));  
    y=a*sqrt(y);  
    cout << "Результат h= "<<y;  
}
```

При необхідності перевірки виконання декількох умов можна скористатися вкладеними логічними операторами. Наприклад для перевірки одночасного виконання двох умов ($a < b$ та $b < c$) можна скористатися наступним вкладенням логічних операторів:

if(a<b) if(b<c)

Для об'єднання двох і більше логічних виразів простіше використовувати логічні операції (див. додаток №6), оскільки вкладення логічних операторів може приводити до логічних помилок та поганої «читабельності» програми.

Наприклад:

$(a < b \ \&\& \ b < c)$ – одночасне виконання двох умов;

$(a < b \ || \ b < c)$ – виконання хоча б однієї із умов;

$(a < b \ ! \ b < c)$ – виконання першої та невиконання другої умови;

$(a < b \ \&\& \ b < c \ ! \ b < c)$ – одночасне виконання двох перших умов та невиконання третьої.

У випадку, коли варіантів вибору є декілька, оптимальніше використовувати оператор вибору **switch**. Він має наступний синтаксис запису:

```
switch (<вираз>)  
{  
    case n1: <оператор 1>;  
    break;  
    case n2: <оператор 2>;  
    break;
```

....

```
case nn: <оператор n>;  
break;  
default: <оператор n+1>;  
}
```

<Вираз> обчислює значення цілого типу і може приймати одне із значень міток **case**. Якщо отримане значення не співпадає із жодним значенням міток, то виконується мітка **default**. Щоб в кожному конкретному випадку виконувався тільки оператор відповідної мітки обов'язковим є використання команди **break**, яка передає управління в кінець оператора **switch**.

Для реалізації альтернативних обчислень в залежності від виконання певної умови може також використовуватися умовна операція ? :

<Умова> ? <Вираз1> : <Вираз2>

Спочатку обчислюється значення виразу **<Умова>**. Якщо вона істина, то обчислюється значення **<Виразу1>**, а якщо ні - **<Виразу2>**.

Наступний приклад ілюструє використання умовної операції для знаходження більшого із двох чисел **x** та **y**:

max=(x>y)?x:y;

Приклади виконання завдання

Приклад №1 (використання оператора умовного переходу)

Програма обчислення функції, значення якої рівне:

$$y(x) = \begin{cases} -1 & \text{при } x < 0 \\ 0 & \text{при } x = 0 \\ +1 & \text{при } x > 0 \end{cases}$$

```
#include "pch.h"
```

```
#include <iostream>
```

```
#include <windows.h>
```

```
using namespace std;
```

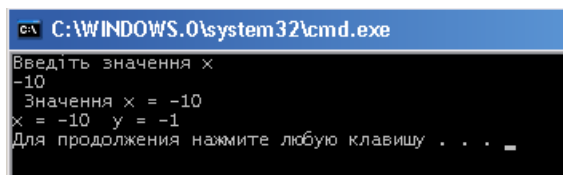
```

int main()
{
    int x, y;
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    cout << "Введіть значення x \n";
    cin >> x;
    cout << "Значення x = " << x;
    if (x < 0) y = -1; else if (x == 0) y = 0; else y = 1;
    cout << "\nx = " << x << "\t " << "y = " << y << "\n";
}

```

Результати роботи програми:

Введено значення $x=-10$



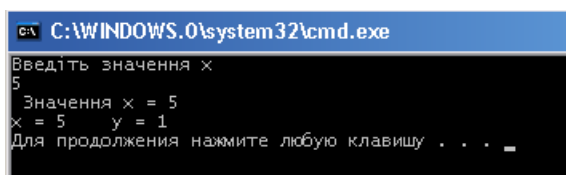
```

C:\WINDOWS.0\system32\cmd.exe
Введіть значення x
-10
Значення x = -10
x = -10 y = -1
Для продовження натисніть будь-яку клавішу . . . _

```

Рис. 4.1 – Результат роботи програми

Введено значення $x=5$



```

C:\WINDOWS.0\system32\cmd.exe
Введіть значення x
5
Значення x = 5
x = 5 y = 1
Для продовження натисніть будь-яку клавішу . . . _

```

Рис. 4.2 – Результат роботи програми

Приклад №2 (використання оператора вибору *switch*)

За введеним із клавіатури номером дня тижня програма виводить його назву. Якщо введеному номеру не відповідає жоден із днів тижня, то видається відповідне повідомлення.

```

#include "pch.h"
#include <iostream>
#include <windows.h>
using namespace std;
int main()
{
    int i;
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    cout << "Введіть номер дня тижня N \n";
    cin >> i;
    cout << "Номеру дня тижня N = " << i << " відповідає ";
    switch (i)
    {
        case 1: cout << "\tПонеділок \n";
                break;
        case 2: cout << "\tВівторок \n";
                break;
        case 3: cout << "\tСереда \n";
                break;
        case 4: cout << "\tЧетвер \n";
                break;
        case 5: cout << "\tП'ятниця \n";
                break;
        case 6: cout << "\tСубота \n";
                break;
        case 7: cout << "\tНеділя \n";
                break;
        default: cout << "\tТакого дня в тижні немає \n";
    }
}

```

}

Результати роботи програми:

Введено значення $N=4$

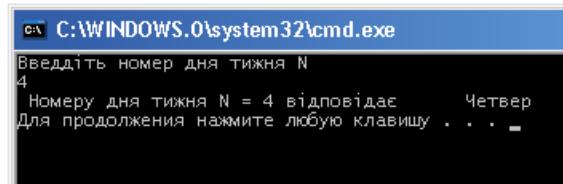


Рис. 4.3 – Результат роботи програми

Введено значення $N=13$

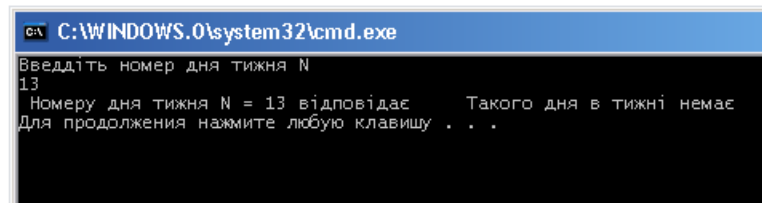


Рис. 4.4 – Результат роботи програми

Завдання для виконання

Завдання №1

Написати програму обчислення значення функції. Значення аргументу вводиться із клавіатури у діалоговому режимі. Значення інших змінних задати у вигляді констант. Передбачити вивід на друк вихідних даних та результатів у зручній для сприйняття формі з використанням форматного виводу.

$$1. \ y(x) = \begin{cases} \cos^2(bx), & x < a \\ \sqrt{(x-a)^3}, & a \leq x \leq b \\ \left| \frac{ax-2b}{b^2+a^2} \right|, & x > b \end{cases}$$

$$2. \quad y(x) = \begin{cases} e^{|x+a|} \cdot \sin(x), & x < a \\ x^{a+b}, & a < x < b^2 \\ (x-1)^2 \cos(x)^2, & x \geq a^2 \end{cases}$$

$$3. \quad y(x) = \begin{cases} \cos(ax), & x < a \\ \sqrt{(x^4 + a)}, & a < x < b \\ \left| \frac{1}{b^2 + a^2} \right|, & x > b \end{cases}$$

$$4. \quad d(\alpha) = \begin{cases} a^2 \sin(\alpha + \pi/3), & \alpha < b \\ 1, & b \leq \alpha \leq a+1 \\ b \cos^2(\alpha^2), & \alpha > a+1 \end{cases}$$

$$5. \quad y(x) = \begin{cases} \sqrt[5]{\cos^4(x-a)}, & x \leq a \\ x^2 \operatorname{tg}(bx), & a < x \leq b \\ \left| \frac{\sin(a+b)x}{x^2 - b^2} \right|, & x > b \end{cases}$$

$$6. \quad f(k) = \begin{cases} \frac{k+b}{2 + \sin^2(a)}, & k < 2a \\ k + \sin(k^2), & 2a \leq k < 2a+1 \\ \sqrt{a+4} \cdot \sin(a+b), & k \geq 2a+1 \end{cases}$$

$$7. \quad y(x) = \begin{cases} \sqrt[3]{e^{ax}}, & x \leq a \\ a \cos(bx), & a < x < b \\ \frac{a+bx}{x^2 - b^2}, & x \geq b \end{cases}$$

$$8. \quad g(h) = \begin{cases} 2 \cos^2(h+a)^2, & b+1 < h < a \\ \operatorname{tg}(a), & a \leq h \leq a+b \\ \ln(b+3 \sin(h^2)), & h > a+b \end{cases}$$

$$9. \ y(x) = \begin{cases} a \sin(x^3), & x < a \\ e^{a+b}(x+1), & a < x \leq b \\ \sqrt{(x^2+b^2)}, & x > b \end{cases}$$

$$10. \ o(q) = \begin{cases} \ln^2(a^2+2q), & q > a+b \\ \ln(q+2a), & q = a+b \\ \frac{b^2+2}{2} + \sin^3(q), & q < a+b \end{cases}$$

$$11. \ y(x) = \begin{cases} \cos^2(bx), & x < a \\ \sqrt{(x-a)}, & a \leq x \leq b \\ \left| \frac{ax-2b}{(x^2+b^2)} \right|, & x > b \end{cases}$$

$$12. \ p(x) = \begin{cases} e^{\frac{x}{2} + \sqrt{x}}, & a \leq x \\ e^{\frac{x}{2}}, & a < x \leq a+2b \\ \ln^2(b+x) + \operatorname{tg}\left(x + \frac{\pi}{3}\right), & x > a+2b \end{cases}$$

$$13. \ s(p) = \begin{cases} \frac{a^2-p}{\log_2(b+p)}, & p < 3 \\ \sqrt{p}, & 3 < p \leq 5 \\ \frac{\sqrt{p^3}}{a-1}, & p > 5 \end{cases}$$

$$14. \ t(r) = \begin{cases} \frac{a + \sqrt[3]{r}}{\ln^2(r+b)}, & r \geq a \\ \sqrt{ae^r}, & b < r < a \\ \frac{\sqrt{a+e^r}}{r}, & r \leq b \end{cases}$$

$$15. k(t) = \begin{cases} \sqrt[5]{b \sin(t + \pi/6)}, & t < a \\ \cos\left(t + \frac{\pi}{2}\right), & a \leq t < b \\ \sqrt{\frac{a}{2} \cos\left(t + \frac{\pi}{4}\right)}, & t \geq b \end{cases}$$

$$16. \varphi(c) = \begin{cases} \lg^2(a), & b \geq c > a \\ \lg\left(a^{|c-b|}\right), & b < c \\ \ln(b + c^2 a), & c \leq a \end{cases}$$

$$17. \gamma(x) = \begin{cases} \sqrt[4]{|x-2a|} + b, & x \leq 0 \\ |x-a|, & 0 < x \leq |ab| \\ \frac{|x-a|^3}{3}, & x > |ab| \end{cases}$$

$$18. c(\varphi) = \begin{cases} atg(\varphi - b/4), & \varphi > a \\ 0, & a \geq \varphi \geq a+b \\ btg\left(\frac{\varphi+b}{3}\right), & \varphi < a+b \end{cases}$$

$$19. d(a) = \begin{cases} e^{\sqrt{a+b}} + b\sqrt{a}, & \alpha > 2b \\ b\sqrt{a} + a \ln(b), & a \leq \alpha \leq 2b \\ e^{\sqrt{a+b}} + a \ln(b), & \alpha < a \end{cases}$$

$$20. y(i) = \begin{cases} b + \ln^2|i + \pi|, & i < -b \\ \sin(i), & 0 > i \geq -b \\ i^2 + \ln^2|i + 2\pi|, & 0 \leq i \end{cases}$$

Завдання №2

Розв'язати приведену нижче задачу. Вхідні дані вводяться із клавіатури у діалоговому режимі. Інші дані, необхідні для розв'язку задачі, задаються у вигляді констант. Передбачити вивід на друк результатів у зручній для сприйняття формі, використовуючи форматний вивід.

1. Написати програму, яка вводить вік користувача і, якщо йому більше 18 років, повідомляє, що він має право голосу. В іншому разі вона обчислює, через скільки років користувач буде мати право голосу.

2. Дано координати двох точок. Визначити, чи розміщені вони на одному колі із центром в початку координат.

3. Із клавіатури вводиться номер місяця в році. Виводиться повідомлення про назву місяця та пору року. Якщо місяця із таким номером не існує – виводиться відповідне повідомлення.

4. Дано радіус кола та довжина сторони квадрата. Перевірити, чи пройде квадрат крізь коло.

5. Ввести із клавіатури три числа та знайти найменше та найбільше серед цих чисел.

6. Дано радіус кола та довжина сторони квадрата. Визначити, чи пройде круг через квадрат.

7. На площині задано коло радіусу R із центром в початку координат. Ввести із клавіатури координати точки та визначити, чи знаходиться вона на колі, чи в середині кола, чи за його межами.

8. Відома два кути трикутника γ_1 та γ_2 . Визначити тип трикутника: гострокутний, прямокутний, тупокутний.

9. Дано довжини трьох відрізків. Визначити, чи можна із цих відрізків побудувати трикутник.

10. Дано довжини трьох сторін трикутника. Визначити, якого типу це є трикутник (прямокутний, гострокутний, рівносторонній, рівнобедрений ...).

11. Знайти найбільше із трьох значень $f(x)$, $f(2x)$ та $f(3x)$, де $f(x) = \sin(x)$.
Значення аргументу вводиться із клавіатури у градусах, тому перед обчисленням функції перевести їх у радіани.

12. Прямокутник задано координатами його вершин. Визначити, чи належить введена точка області прямокутника.

13. Прямокутник задано координатами його вершин. Визначити, чи коло із заданим радіусом та координатами центру повністю належить області прямокутника.

14. Трикутник задано координатами вершин. Визначити, в яких квадрантах координатної площини він знаходиться.

15. Визначити, чи перетинаються два відрізки, задані координатами своїх кінців.

16. Два кола задано їхніми радіусами та координатами центрів. Визначити кількість спільних точок у цих кіл (одна, дві, чи жодної).

17. Із клавіатури вводиться порядковий номер місяця. Програма виводить назву пори року. Передбачити перевірку коректності вводу номера місяці.

18. Дано координати точки $M(x, y)$. Визначити, якому квадранту системи координат належить дана точка.

19. Дано параболу виду $y = ax^2 + bx + c$ та координати двох точок. Визначити, як знаходяться ці точки відносно віток параболи (обидві між вітками, обидві зовні віток чи інші варіанти).

Контрольні питання:

1. Для чого використовують умовний оператор?
2. Назвіть дві форми умовного оператора?
3. Наведіть приклад з використанням розгалуження?
4. Назвіть операції порівняння в C++?
5. Для чого використовують ключове слово «else»?

ПРАКТИЧНА РОБОТА №5

Тема роботи: Прості цикли із відомим числом повторів

Мета роботи: Формування навиків в розробці простих циклічних програм із відомим числом повторень для реалізації типових алгоритмів обробки даних.

Теоретичні відомості

Оператор циклу *for* забезпечує циклічне (повторне) виконання деякого простого або складеного оператора певне число разів.

Синтаксис оператора *for* є наступним:

for(*<ініціалізація>*; *<умова>*; *<вираз>*)
<оператор>

де:

<ініціалізація> – змінна або вираз, що управляє виконанням циклу;

<умова> – перевірка умови досягнення кінця циклічних обчислень;

<вираз> – міняє значення лічильника циклу і може бути будь яким арифметичним виразом допустимого типу;

<оператор> – тіло циклу – будь який простий чи складений оператор.

У заголовку циклу *for* одна чи декілька компонент може бути відсутня, але при цьому обов'язково мусять бути збережені всі крапки з комою (;). В ініціалізації початкового значення може стояти або один вираз, а декілька, розділених комою. При перевірці умови на закінчення циклічного обчислення можна використати декілька умов, об'єднаних логічними операціями. *<Вираз>* також може містити не одну змінну, а декілька, розділені між собою за допомогою коми.

Алгоритм роботи оператора *for* є наступний:

1. виконується початкова ініціалізація;
2. перевіряється умова, і якщо вона *false*, то виконання циклу завершується, інакше управління передається пункту 3;

3. виконується тіло циклу - *<оператор>*;

4. обчислюється *<вираз>* та управління передається пункту 2.

For може мати і наступну форму запису (нескінчений цикл):

for (; ;) < оператор >

Для виходу із такого нескінченного циклу обов'язковим є перевірка в тілі циклу виконання певної умови та завершення циклічного обчислення за допомогою ***break***. Він виконує негайний вихід із циклу та передачу управління наступному оператору.

Наприклад:

```
/* Вивід парних чисел на проміжку від 0 до 100 */
```

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

```
int main()
```

```
{
```

```
int i=0;
```

```
for(;;)
```

```
{
```

```
i++;
```

```
if (i%2 == 1)
```

```
continue;
```

```
else
```

```
printf("\n %ld", i );           // %ld - long decimal.
```

```
/* Перевірка умови завершення циклічного  
обчислення */
```

```
if (i == 100) break;
```

```
}
```

```
printf("\n");
```

```
}
```

Для передачі управління (виконання циклу) на наступну ітерацію в межах циклічного обчислення служить оператор ***continue***. Він перериває виконання даної ітерації циклічного обчислення та передає управління наступній ітерації.

Наприклад:

```
/* Вивід парних чисел на проміжку від 0 до100 */  
#include <stdio.h>  
int main()  
{  
int i;  
for(i=1;i<=100;i++)  
if (i%2 == 1)  
continue;  
else  
printf("\n %ld", i ); // %ld - long decimal.  
printf("\n");  
}
```

Для безумовної передачі управління використовується оператор переходу **goto**:

goto <мітка>

Керування передається оператору з даною міткою <мітка>:оператор, а сама мітка в програмі не декларується. В якості мітки може виступати будь-який звичайний ідентифікатор. Область дії мітки обмежується функцією, в якій вона визначена, тому неможливо передати управління у іншу функцію.

Приклади виконання завдання

Приклад №1 (використання оператора циклу)

Написати програму обчислення n чисел Фобіначі, що визначаються рекурентним співвідношенням:

$$a_1 = 1, a_2 = 1, a_i = a_{i-1} + a_{i-2}.$$

```
#include <iostream>  
#include <windows.h>  
using namespace std;  
int main ( )
```

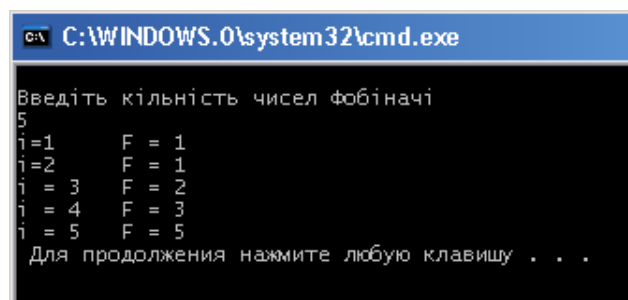
```

{ int n,a1,a2,a;
SetConsoleOutputCP(1251);
SetConsoleCP(1251);
a1=1;
a2=1;
cout << "\nВведіть кількість чисел Фобіначі \n";
cin >>n;
cout << "i=1" << "" << "\t" << "F = " << a1;
cout << "\ni=2" << "" << "\t" << "F = " << a2;
for(int i=3;i<=n;i++)
{
a=a2+a1;
a2=a1;
a1=a;
cout << "\ni = " << i << "\t" << "F = " << a;
}
cout << "\n ";
}

```

Результат роботи програми:

Введено значення $N=5$



```

C:\WINDOWS.0\system32\cmd.exe
Введіть кількість чисел фобіначі
5
i=1      F = 1
i=2      F = 1
i = 3    F = 2
i = 4    F = 3
i = 5    F = 5
Для продовження натисніть будь-яку клавішу . . .

```

Рис. 5.1 – Результат роботи програми

Приклад №2 (розв’язок попередньої задачі з використанням оператора умови та безумовного переходу)

```
#include <iostream>
```

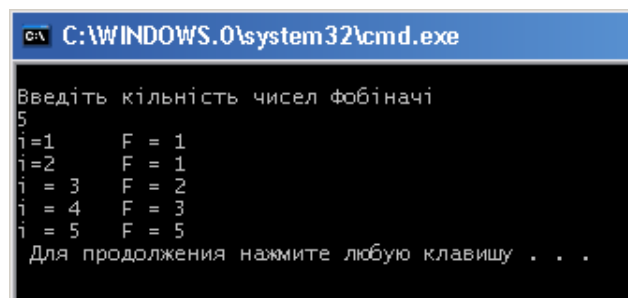
```

#include <windows.h>
using namespace std;
int main ( )
{
    int i,n,a1,a2,a;
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    i=2;
    a1=1;
    a2=1;
    cout << "\nВведіть кількість чисел Фобіначі \n";
    cin >>n;
    cout << "i=1" << "" << "\t" << "F = " << a1;
    cout << "\ni=2" << "" << "\t" << "F = " << a2;
    q:i=i+1;
    a=a2+a1;
    a2=a1;
    a1=a;
    cout << "\ni = " << i << "\t" << "F = " << a;
    if (i<n) goto q;
    cout << "\n ";
}

```

Результат роботи програми:

Введено значення $N=5$



The screenshot shows a Windows command prompt window with the title bar "C:\WINDOWS.0\system32\cmd.exe". The output of the program is displayed in the console, showing the prompt "Введіть кількість чисел Фобіначі" followed by the input "5". The program then outputs the Fibonacci sequence for i=1 to i=5, with the corresponding Fibonacci number F. The output is as follows:

```

Введіть кількість чисел Фобіначі
5
i=1      F = 1
i=2      F = 1
i = 3    F = 2
i = 4    F = 3
i = 5    F = 5
Для продовження натисніть будь-яку клавішу . . .

```

Рис. 5.2 – Результат роботи програми

Завдання для виконання

Написати програму для розв'язку приведеної нижче задачі. В програмі передбачити використання форматного виводу та пояснювальної текстової інформації. Розв'язок реалізувати двома способами: у першому використати оператор циклу **for**, а у другому організувати циклічне обчислення за допомогою оператора **if**.

1. Обчислити суму $S = \sum_{i=1}^m \frac{1}{i^3}$. Значення m вводиться із клавіатури.
2. Обчислити суму $S = \sum_{n=1}^m (-1)^n n^2$, де m – ціле число, що вводиться із клавіатури.
3. Обчислити суму $S = \sum_{n=1}^k 2^n$, де k – ціле число, що вводиться із клавіатури.
4. Обчислити приблизно площу фігури, обмеженої графіками функцій $y(x) = x^2$, $x = 2$ та $y = 0$, розбивши інтервал зміни x на 100 частин та сумуючи площі прямокутників із основою, рівною інтервалу зміни x та висотою, що визначається значенням функції в середині основи.
5. Для введеного з клавіатури натурального числа n обчислити добуток $1 \cdot 3 \cdot \dots \cdot n$, якщо n – непарне та $2 \cdot 4 \cdot \dots \cdot n$, якщо n – парне.
6. Обчислити суму $S = \sum_{k=1}^m \frac{1}{k!}$. Значення m вводиться із клавіатури.
7. Обчислити n – ний член ряду, елементи якого обчислюються за формулами: $a_1 = 2$; $a_2 = 1$; $a_i = a_{i-1} + 2a_{i-2}$ для $i > 2$.
8. Обчислити кількість розміщень із n по m $A_n^m = n(n-1)\dots(n-m+1)$.
9. Написати програму обчислення для заданого n суми $1^2 + 2^2 + 3^2 + \dots + n^2$ та перевірити правильність отриманого результату (сума рівна $n(n+1)(2n+1)/6$).

10. Обчислити суму $S = \sum_{i=1}^n (-1)^i \cdot 2^i$. Значення n вводиться із клавіатури.
11. Обчислити суму $S = \sum_{i=1}^n (-1)^i \cdot (2i)$. Значення i вводиться із клавіатури.
12. Для заданого n обчислити добуток $P = 1 \cdot 3 \cdot 5 \cdot 7 \cdot \dots \cdot (2n+1)$.
13. Обчислити суму $S = 1 \cdot 3 + 3 \cdot 5 + 5 \cdot 7 + \dots + (2n-1)(2n+1)$ для заданого n .
14. Обчислити добуток $P = (1+3) \cdot (5+7) \cdot \dots \cdot ((2n-1)+(2n+1))$ для заданого n .
15. Обчислити суму ряду $S = 1 + (1+2) + (2+3) + (3+4) + \dots + ((n-1)+n)$ для введенного із клавіатури значення n .
16. Протабулювати функцію $y = x^3$ при зміні x на проміжку від a до b (вводяться із клавіатури), розбивши його на 25 частин.
17. Обчислити значення інтегралу $\int_0^{\pi/2} \sin(x) dx$ за формулою трапецій із заданим числом розбиттів n .
18. Протабулювати функцію $y = \sin^2(x)$ на проміжку від a до b , розбивши його на 30 частин.
19. Обчислити значення інтегралу $\int_0^5 \sqrt{x^2+1} dx$ за формулою трапецій із числом розбиттів $n=100$.
20. Обчислити суму $S = \sum_{i=1}^n \frac{3^i}{i!}$. Значення n вводиться із клавіатури.

Контрольні питання

1. Що таке цикл?
2. Які оператори називають операторами повторення (циклу)?
3. Назвіть оператори повторення (циклу) C++.
4. Наведіть загальний синтаксис циклу for.
5. Яку змінну називають лічильником циклу?

ПРАКТИЧНА РОБОТА №6

Тема роботи: Прості цикли із невідомим числом повторів

Мета роботи: Формування навиків в розробці простих циклічних програм із невідомим числом повторень для реалізації основних алгоритмів обробки даних.

Теоретичні відомості

Оператор циклу *while* використовується для організації циклічних обчислень, якщо кількість повторів не відома наперед. Він виконується, поки виконується певна умова. Його синтаксис є наступним:

while (<логічний вираз>) <оператор>;

<Логічний вираз> - арифметичний або логічний вираз, що управляє роботою циклу. Поки вираз істинний – робота циклу продовжується. Якщо значення стає хибним – виконання циклу припиняється і управління передається наступному після циклу оператору. Цикл *while* не виконається жодного разу, якщо на початку його виконання <Логічний вираз> буде хибним.

<Оператор> - простий або складений оператор, сформований з використанням операторних дужок *{}*. Вони становлять тіло циклу. Для формування тіла циклу можна також використати операцію слідування , (кома).

Оператора циклу *do ... while* служить для виконання циклічних обчислень (тіла циклу) до тих пір, поки логічний вираз не стане хибним. Його синтаксис є наступним:

do <оператор> *while* (<логічний вираз>;

Тіло циклу - <оператор> - може бути або простим, або складеним оператором. Тіло циклу виконується до тих пір, поки <логічний вираз> істинний. Коли значення логічного виразу стає хибним, проходить завершення циклічного обчислення та передача управління наступному оператору. Якщо на початку виконання циклу *while* <Логічний вираз> є хибним, то тіло циклу виконається один раз. Для негайного виходу із циклу можна використати

оператора *break*, а для передачі управління на наступну ітерацію циклу – оператор *continue* (див. попередню роботу).

Приклади виконання завдання

Приклад №1 (використання оператора циклу *while* та складеного оператора) Написати програму знаходження значення функції $y = \sin(x)$ із точністю ε (задати як константу) за допомогою її розкладу в ряд Маклорена:

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

Примітка: сумування проводиться до тих пір, поки модуль чергового члену ряду не стане меншим заданої точності ε .

```
#include <iostream>
#include <math.h>
#include <windows.h>
#define eps 1e-9
using namespace std;
int main ( )
{
    int i;
    double x,ai,s;
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    cout << "\nВведіть значення x \n";
    cin >>x;
    ai=x;
    s=0;
    i=1;
    while (fabs(ai)>eps)
    // тіло циклу
    { // початок складеного оператора
```

```

s=s+ai;
i=i+1;
ai=ai*(-x*x)/((2.0*i-1.0)*(2.0*i-2.0));
} // кінець складеного оператора
cout << "\nСума ряду рівна\t\t" << s;
cout<< "\nПросумовано \t " <<i<< "\t членів ряду ";
s=sin(x);
cout << "\nТочна сума ряду рівна\t" << s;
cout << "\n ";
}

```

Результат роботи програми:

Введено значення $x=1.5$

```

C:\WINDOWS.0\system32\cmd.exe
Введіть значення x
1.5
Сума ряду рівна      0.997495
Просумовано      8      членів ряду
Точна сума ряду рівна 0.997495
Для продовження натисніть будь-яку клавішу . . . _

```

Рис. 6.1 – Результат роботи програми

Приклад №2 (використання оператора циклу *do...while* та операції слідування) Написати програму знаходження суми ряду, i -тий член якого визначається формулою:

$$a_i = \left(\frac{1}{2i}\right)^i$$

із точністю ϵ (вводиться із клавіатури) з використанням оператора циклу *do...while*.

```

#include <iostream>
#include <math.h>
#include <windows.h>
using namespace std;
int main ( )

```

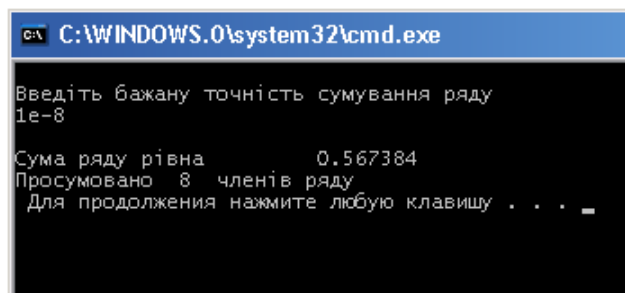
```

{
int i;
SetConsoleOutputCP(1251);
SetConsoleCP(1251);
double ai,s;
double eps;
cout << "\nВведіть бажану точність сумування
ряду \n";
cin >>eps;
s=0;
i=1;
do
// тіло циклу з використанням оператора слідування
ai=pow(1.0/(2*i),i);
s=s+ai;
i=i+1;
while(ai>eps);
cout << "\nСума ряду рівна\t\t" << s;
cout<< "\nПросумовано " <<i<< " членів ряду ";
cout << "\n ";
}

```

Результат роботи програми:

Введено значення бажаної точності сумування ряду $eps=1e-8$



```

C:\WINDOWS.0\system32\cmd.exe
Введіть бажану точність сумування ряду
1e-8
Сума ряду рівна 0.567384
Просумовано 8 членів ряду
Для продовження натисніть будь-яку клавішу . . . _

```

Рис. 6.2 – Результат роботи програми

Завдання для виконання

Написати програму розв'язку завдання двома способами – із використанням оператора циклу **while** (перший спосіб) та за допомогою оператора циклу **do...while** (другий спосіб). Перевірити правильність виконання програми, використавши контрольні приклади.

1. Обчислити значення квадратного кореня $y = \sqrt{x}$ із точністю ε , використавши ітераційну формулу Ньютона: $y_0 = 1$, $y_i = 0.5(y_{i-1} + x/y_{i-1})$ ($i = 1, 2, 3, \dots$). Підрахувати кількість ітерацій, за які досягається бажана точність. Точність обчислення ε визначається як модуль різниці двох послідовних ітерацій.
2. Обчислювати суму членів ряду, поки черговий член не стане меншим ε .

$$S = 1 + 1/2 + 1/4 + 1/8 + \dots$$

3. Обчислити значення кореня кубічного $y = \sqrt[3]{x}$ з точністю ε , використавши ітераційну формулу Ньютона $y_0 = 1$, $y_i = \frac{1}{3}(2y_{i-1} + x/y_{i-1}^2)$ ($i = 1, 2, 3, \dots$). Підрахувати кількість ітерацій, за які досягається бажана точність. Точність обчислення ε визначається як модуль різниці двох послідовних ітерацій.
4. Значення функції $\ln(x)$ можна обчислити її розкладом в ряд Маклорена:

$$\ln(x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \dots$$

Обчислити $\ln(x)$ з точністю ε (обчислення продовжується до моменту, поки черговий член ряду по модулю не буде меншим ε). Підрахувати кількість членів ряду, які просумовано для досягнення заданої точності.

5. Для двох натуральних чисел m та n ($1 < m < n$) знайти найменше значення k , при якому $m^k > n$.
6. Значення функції $\cos(x)$ можна обчислити її розкладом в ряд Маклорена

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \dots$$

Обчислити $\cos(x)$ з точністю ε (обчислення продовжується до моменту, поки абсолютне значення чергового члену ряду не буде меншим ε). Підрахувати кількість членів ряду, які необхідні просумувати для досягнення заданої точності.

8. Знайти наближено з точністю 0.01 найбільше значення функції $y = (ax^2 + bx + c)/(d \cdot x + e)$ на відрізку $[x_1, x_2]$ (спосіб вибрати самостійно). Значення a, b, c, d, e, x_1, x_2 вводяться з клавіатури.
9. Для цілого числа $m > 10$ отримати найбільше ціле k , для якого $4^k < m$.
10. Для натурального числа n знайти найменше натуральне число виду m^2 , що перевищує n .
11. Значення функції $\sin^2(x)$ можна обчислити за допомогою її розкладу в ряд Маклорена

$$\sin^2(x) = x^2 - 2x^4/3 - \dots + (-1)^{n-1} 2^{2n-1} x^{2n} / (2n)!$$

Обчислити $\sin^2(x)$ з точністю ε (обчислення продовжується до моменту, поки черговий член ряду по модулю не буде меншим ε). Підрахувати кількість членів ряду, які необхідно просумувати для досягнення заданої точності.

12. Міняючи x від початкового значення a з кроком h визначити, при якому значенні x добуток $x \cdot \sin(x)$ стане більшим $\cos(x)$. Вхідні дані вводяться із клавіатури.
13. Знайти корінь рівняння $e^x - 10x = 0$ з точністю ε методом простої ітерації.
14. Знайти корінь рівняння $\lg(1.5773x) - 2.3041x = 0$ з точністю ε методом простої ітерації.
15. Знайти корінь рівняння $\ln(7.622x) - 8.59x + 0.5 = 0$ з точністю ε методом простої ітерації.

Контрольні питання

1. Який цикл називають циклом з передумовою? Поясніть.
2. Наведіть загальний синтаксис циклу while.
3. Який цикл називають циклом з постумовою? Поясніть.
4. Наведіть загальний синтаксис циклу do while.
5. Яка різниця між циклом, що керується лічильником і циклом, що керується контрольним значенням?

ПРАКТИЧНА РОБОТА №7

Тема роботи: Одновимірні масиви

Мета роботи: закріплення навичок роботи з масивами, освоєння найпростіших алгоритмів сортування.

Теоретичні відомості

Розглянемо одновимірні масиви, оголошення яких допускає одну з таких форм запису:

```
<тип><ім'я> [n];  
<тип><ім'я> [n] = {значення};  
<тип><ім'я> [ ] = {значення};
```

При оголошенні одновимірного масиву, коли масив відразу ініціюється, можна не вказувати його розмір. Якщо ж ініціювання не здійснюється під час оголошення масиву, то кількість індексів слід задати обов'язково константним виразом.

Наприклад:

```
float m [6];  
float m [6] = {3.4, 4.5, 5.6, 6.7, 8.9, 10.3};  
float m [ ] = {3.45, 4.56, 5.67, 6.78};
```

Приклад 1. Обчислити функцію $y = ax_i^2 - \sin x_i$ аргументи якої x_j – елементи одновимірного масиву, що мають значення:

$x_0 = -0,81; x_1 = -0,58; x_2 = -0,11; x_3 = 0,2; x_4 = 0,91; x_5 = 1,83.$

```
#include <iostream.h>  
#include <math.h>  
#include <conio.h>  
main ( )  
{ const int n = 7;
```

```

float x[n], y, a(10.5);
int i;
for (i = 0; i < n; i++)
{
    cin >> x[i];
    y = a * x[i] * x[i] - sin(x[i]);
    cout << " x[" << i << "] = " << x[i] << " y = " << y << endl;
}
getch ();
}

```

Результати обчислень:

-0.81 -0.58 -0.11 0.2 0.91 1.83

x[0] = -0.81 y = 7.61334

x[1] = -0.58 y = 4.08022

x[2] = -0.11 y = 0.236828

x[3] = 0.2 y = 0.221331

x[4] = 0.91 y = 7.90555

x[5] = 1.83 y = 34.1969

У цьому випадку передбачено введення елементів масиву в рядок, і клавішу Enter слід натиснути в кінці процесу введення.

Приклад 2. За один перегляд масиву $c_i (i = 0 \dots n-1)$, $n = 15$ визначити значення, а також положення максимального і мінімального його елементів та поміняти їх місцями.

```

#include <iostream.h>
#include <conio.h>
const n=15;
void main ()
{

```

```
float c[n] = {6.4, 1.5, -5.6, 3.7, 18.9, 10.3, -0.6, 44.5, -0.2, 8.9, 55.3, 6.9, 4.3, 7.7, 10.9};
```

```
float max, min;
```

```
int imax, imin;
```

```
cout<< " ***** massiv c[n] ***** n= " << n << endl;
```

```
for (int i = 0; i < n; i++)
```

```
cout << c[i] << " ";
```

```
max = min = c[0];
```

```
imax = imin = 0;
```

```
for (int i = 1; i < n; i++)
```

```
{
```

```
    if (c[i] > max)
```

```
{ max = c[i];
```

```
    imax = i; }
```

```
else
```

```
    if (c[i] < min)
```

```
{ min = c[i];
```

```
    imin = i; }
```

```
}
```

```
c[imin] = max;
```

```
c[imax] = min;
```

```
cout<< "\n\t max= " << max << " min= " << min << endl;
```

```
cout<< "\t imax= " << imax+1 << " imin= " << imin+1 << endl;
```

```
cout<< " ***** Rezult massiv *****" << endl;
```

```
for (int i = 0; i < n; i++)
```

```
cout<< c[i] << " ";
```

```
getch ();
```

```
}
```

Результати обчислень:

***** massiv c[n] ***** n= 15

6.4 1.5 -5.6 3.7 18.9 10.3 -0.6 44.5 -0.2 8.9 55.3 6.9 4.3 7.7 2.1

max= 55.3 min= -6

imax= 11 imin= 3

**** Rezult massiv ****

6.5 1.5 55.3 3.7 18.9 10.3 -0.6 44.5 -0.2 8.9 -5.6 6.9 4.3 7.7 2.1

Алгоритми сортування

Алгоритмом сортування називається алгоритм для впорядкування деяких елементів множини. Зазвичай алгоритмом сортування вважають алгоритм упорядкування елементів за зростанням або спаданням.

Практично кожен алгоритм сортування можна розбити на 3 етапи:

- порівняння, що визначає впорядкованість пари елементів;
- перестановку, що міняє місцями пару елементів;
- власне алгоритм що сортує, який здійснює порівняння і перестановку елементів до тих пір, поки всі елементи множини не будуть впорядковані.

Алгоритми сортування мають велике практичне застосування. Їх можна зустріти там, де мова йде про обробку та зберігання великих обсягів інформації. Деякі завдання обробки даних вирішуються простіше, якщо дані заздалегідь упорядкувати. Існує величезна кількість різних алгоритмів сортування. Універсального, найкращого алгоритму сортування на даний момент не існує. Однак, маючи приблизні характеристики вхідних даних, можна підібрати метод, який працює оптимальним чином при даних умовах. Для цього необхідно знати параметри, за якими буде проводитися оцінка алгоритмів.

Основні параметри алгоритмів сортування

Час сортування - основний параметр, що характеризує швидкодію алгоритму.

Необхідна пам'ять - один з параметрів, який характеризується тим, що ряд алгоритмів сортування вимагають виділення додаткової пам'яті для

тимчасового зберігання даних. При оцінці пам'яті що використовується не буде враховуватися місце, яке займає вихідний масив даних та витрати які не залежать від вхідної послідовності, наприклад, на зберігання коду програми.

Стійкість - це параметр, який відповідає за те, що сортування не змінює взаємного розташування рівних елементів.

Природність поведінки - параметр, який вказує на ефективність методу при обробці вже відсортованих, або частково відсортованих даних. Алгоритм поводить себе природно, якщо враховує цю характеристику вхідної послідовності і працює краще.

Сортування вибором

Алгоритм сортування за зростанням:

1. Знайти мінімальне значення в поточному списку.
2. Знайдене мінімальне значення міняється місцем з елементом на першій позиції.
3. Повторюємо сортування, виключивши з розгляду вже відсортований перший елемент, тобто, починаючи з другої позиції.

Алгоритм не використовує додаткової пам'яті: всі операції відбуваються "на місці".

Метод простого обміну (метод бульбашки).

Алгоритм полягає в повторюваних проходах по сортованому масиву. За кожен прохід елементи послідовно порівнюються попарно і, якщо порядок у парі невірний, виконується обмін елементів. Проходи по масиву повторюються до тих пір, поки на черговому проході не виявиться, що обміни більше не потрібні, що означає - масив відсортований. При проході алгоритму, елемент, що стоїть не на своєму місці, «спливає» до потрібної позиції як бульбашка у воді, звідси і назва алгоритму.

Алгоритм сортування методом бульбашки:

1. Порівнюємо перший і другий елементи масиву. Якщо перший елемент більший, ніж другий, то міняємо їх місцями.

2. Порівнюємо другий і третій елементи масиву. Якщо другий елемент більший, ніж третій, то міняємо їх місцями.

3. ...

4. Порівнюємо передостанній ($N-1$) і останній (N) елементи масиву. Якщо передостанній елемент більший, ніж останній, то міняємо їх місцями.

В результаті останнім елементом в масиві у нас виявиться найбільший елемент.

Другий крок сортування методом бульбашки - повторюємо вищевказані дії для частини масиву, починаючи з першої позиції до $N-1$:

1. Порівнюємо перший і другий елементи масиву. Якщо перший елемент більший, ніж другий, то міняємо їх місцями.

2. Порівнюємо другий і третій елементи масиву. Якщо другий елемент більший, ніж третій, то міняємо їх місцями.

3. ...

4. Порівнюємо елемент ($N-2$) і елемент ($N-1$) масиву. Якщо ($N-2$)-й елемент більший, ніж елемент ($N-1$), то міняємо їх місцями.

В результаті передостанній елемент в масиві у нас теж буде на своєму, "відсортованому" місці.

На наступних кроках сортування методом бульбашки вищевказані дії повторюються для частини масиву, починаючи з 1 позиції до ($N-2$) (крок 3), а потім для діапазону $1 \dots (N-3)$ і так далі до діапазону $1 \dots 2$.

Після завершення останнього кроку масив буде відсортований за зростанням.

Середнє число порівнянь та обмінів мають квадратичний порядок зростання, звідси можна зробити висновок, що алгоритм бульбашки досить повільний, проте він простий і його можна покращити простими засобами.

По-перше, розглянемо ситуацію, коли на якому-небудь з проходів не відбулося жодного обміну. Це означає, що всі пари розташовані в правильному порядку, так що масив вже відсортований. Одна з можливостей поліпшення

алгоритму полягає в запам'ятовуванні, чи проводився на даному проході обмін. Якщо ні - алгоритм закінчує роботу.

Процес поліпшення можна продовжити, якщо запам'ятовувати не тільки сам факт обміну, але і індекс останнього обміну k . Дійсно: всі пари сусідніх елементів з індексами, більшими за k , вже розташовані в потрібному порядку. Подальші проходи можна закінчувати на індексі k , замість того щоб рухатися до встановленої заздалегідь верхньої межі.

Завдання для виконання

Написати програму, в якій:

1. Згенерувати одновимірний масив цілих чисел розмірністю згідно варіанту.
2. Елементи масиву задати випадковим чином в діапазоні 0 ... 50.
3. Виконати друк цього масиву на екран.
4. Виконати обробку масиву відповідно до варіанту. При написанні програми використати вказаний алгоритм сортування.
5. Виконати друк перетвореного масиву на екран.

1	Відсортувати одновимірний масив довжиною $N = 50$ за спаданням методом простого обміну.
2	Відсортувати одновимірний масив довжиною $N = 55$ за зростанням методом вибору.
3	Є одновимірний масив довжиною $N = 45$. Відсортувати за спаданням методом вибору ті елементи масиву, які є парними числами.
4	Є одновимірний масив довжиною $N = 70$. Відсортувати за зростанням за допомогою методу вибору ті елементи масиву, які розташовуються на непарних позиціях.
5	Відсортувати одновимірний масив довжиною $N = 29$ за спаданням методом вибору.
	Відсортувати одновимірний масив довжиною $N = 31$ по зростанню

6	методом простого обміну.
7	Є одномірний масив довжиною $N = 32$. Відсортувати за зростанням за допомогою методу простого обміну ті елементи масиву, які розташовуються на непарних позиціях.
8	Є одномірний масив довжиною $N = 26$. Відсортувати за спаданням за допомогою методу вибору ті елементи масиву, які є непарними числами.
9	Є одномірний масив довжиною $N = 25$. Упорядкувати масив методом вибору таким чином, щоб елементи, що знаходяться на парних позиціях розташовувалися по зростанню, а на непарних позиціях - за спаданням.
10	Є одномірний масив довжиною $N = 27$. Упорядкувати масив методом простого обміну таким чином, щоб елементи, що знаходяться на парних позиціях розташовувалися за спаданням, а на непарних позиціях - по зростанню.

Контрольні питання

1. Що називають масивом?
2. Наведіть синтаксис оголошення одновимірного масиву.
3. Що називають індексом елемента масиву?
4. Як можна отримати доступ до елементів масиву?
5. Наведіть синтаксис звернення до певного елемента масиву.
6. В який спосіб можна задавати початкові значення елементам масиву?

ПРАКТИЧНА РОБОТА № 8

Тема: Розробка програм з використанням двовимірних масивів

Мета роботи: засвоїти техніку використання двовимірних масивів при програмуванні мовою C++, навчитись будувати математичну модель задачі.

Теоретичні відомості

Двовимірний масив (матриця) можна представити як одновимірний масив, кожний елемент якого—масив.

Тривимірний масив — це масив, кожний елемент якого являє собою двовимірну матрицю.

```
char Matrix2D[6][9];          // Двовимірний масив 6х9 елементів
```

```
unsigned long Arr3D[4][2][8];  // Тривимірний
```

Багатовимірні масиви ініціалізуються в порядку якнайшвидшої зміни самого правого індексу: спочатку відбувається присвоєння початкових значень всіх елементів останнього індексу, потім попереднього і т. д.:

```
int Mass[3][2][4] = { 1,2,3,4,5,6,7,8,9,10,11,12,  
                      13,14,15,16,17,18,19,20,21,22,23,24};
```

Рекомендується для наочності групувати дані за допомогою проміжних фігурних дужок:

```
int Mass[3][2][4]={ { {1,2,3,4},{5,6,7,8}},  
                   { {9,10,11,12},{13,14,15,16}},  
                   { {17,18,19,20},{21,22,23,24}}};
```

Для багатовимірних масивів при ініціалізації дозволяється опускати лише величину першої розмірності:

```
int main()  
{  
    char x[][3]={ {9,8,7},{6,5,4},{3,2,1}};
```

```

for(int i=0; i<3; i++)
{
    for(int j=0; j<3; j++)
        printf("%d ", (int)x[i][j]);
    printf("\n");
}
return 0;
}

```

Приклад використання двовимірного статичного масиву

Задача №1. Реалізувати програму, яка обчислює суму елементів, що розташовані на обох діагоналях квадратної матриці.

Формалізація задачі

1. Вхідні дані: N – максимальна розмірність матриці (матриця квадратна, оскільки мова йде про діагоналі матриці; n – реальна розмірність матриці A.
2. Вихідні дані: S – сума елементів обох діагоналей.
3. Здійснити перевірку коректності вхідних даних:
4. Для заповнення елементів матриці використаємо генератор випадкових чисел.
5. Для можливості здійснення перевірки правильності обчислень доцільно вивести матрицю на екран.

Лістинг програми

```

#include "stdafx.h"
#include <iostream>
#include <conio.h>
#include <locale.h>
#define N 10
using namespace std;
int main()
{ setlocale(0, "");

```

```

int a[N][N];
int n=N+1;
int A=0, B=0, S, S1=0, S2=0;
while(n>N) // перевірка правильності введення
{
    printf("\nВведіть розмірність матриці: n = ");
    scanf_s("%d",&n);
}
while (A>=B)
{
    printf("\nВведіть границі проміжку A і B: ");
    scanf_s("%d %d",&A, &B);
}
printf("\n\nМатриця A:");
for (int i=0;i<n;i++)
{
    printf("\n\n");
    for (int j=0;j<n;j++)
    {
        a[i][j]=rand()%(B-A)+A;
        printf("%5d",a[i][j]);
    }
}
for (int i=0;i<n;i++)// підрахунок суми
{
    S1 += a[i][i];
    S2 += a[n-1-i][i];
}
S = S1 + S2;
printf("\n\nСума діагональних елементів S = %7d",S);
return 0;
}

```

Схема роботи програми

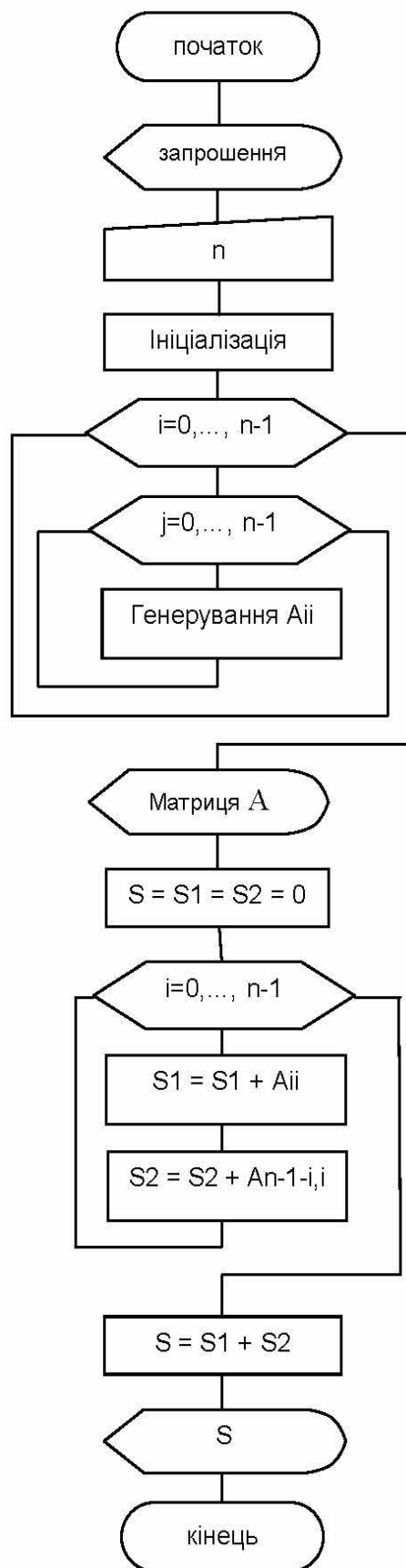


Рис. 8.1 – Схема роботи програми

Результати виконання програми:

```
Введіть розмірність матриці: n = 5
Введіть границі проміжку A і B: 1 10
Матриця A:
  6   9   8   5   9
  2   4   1   8   3
  9   3   8   7   8
  6   8   9   4   1
  1   7   6   1   5
Сума діагональних елементів S =      61
```

Рис. 8.2 – Результат роботи програми

Функція rand

Як зробити випадкові числа у певному діапазоні? Для цього є особлива функція яка міститься у бібліотеці `cstdlib`. Ця функція – `rand`.

```
#include <iostream>
#include <cstdlib>
#include <conio.h>
using namespace std;

int main()
{
    cout << rand() << endl;
    _getch();
    return 0;
}
```

Результат буде 41.

Спробуйте запустити програму знову, результат не зміниться. Що не так? Вся справа в тому, що `rand` не видумує число, а працює за відповідним алгоритмом. Цей алгоритм використовує «зерно» - початкова точка відносно якої визначається `rand`. Все, що нам потрібно це задати якесь число, яке постійно змінюється. Таким постійним зерном буде час. Час змінюється

щосекунди і це те, що нам потрібно. Щоб працювати з часом нам потрібно підключити спеціальну бібліотеку `time.h`. Але ще потрібно знати як задавати це «зерно». Для цього потрібна ще одна функція `srand()`. Саме вона задає початкову точку для функції `rand`. Тепер як ми казали раніше, треба задати час. Будемо використовувати бібліотеку `time.h` і функцію `time()`. Але варто додати параметр `NULL`. Ось так: `time(NULL)`. Ця функція повертає кількість мілісекунд з 1 січня 1970 року.

Тепер поєднаємо всі наші знання у два рядки коду.

```
#include <iostream>
#include <cstdlib>
#include <time.h>
#include <conio.h>
using namespace std;

int main()
{
    srand(time(NULL));
    cout << rand() << endl;
    _getch();
    return 0;
}
```

Загальний діапазон чисел, які можна отримати викликаючи функцію `rand`: 0 – 32767. Якщо $x \% 52 = 0$ (тобто $x = 52$), це максимальне значення, яке ми можемо досягти. Всі інші будуть мати певний залишок і при цьому цей залишок не може бути більшим 51.

```
rand() % 52; // Діапазон від 0 до 51
```

Добре, ми навчилися встановлювати діапазони з 0. Але нам може знадобитися діапазон від x до $(x+15)$ (для прикладу, де $x > 0$).

Допустимо нам потрібен діапазон від 15 до 24. Для цього потрібно зробити декілька кроків:

1. $24 - 15 = 9$

2. Згенерувати числа від 0 до 9. Тобто, $\text{rand()} \% 10$;
3. Тепер посунути це все на 15 одиниць. $\text{rand()} \% 10 + 15$;

Завдання для виконання

Задано двомірний масив дійсних чисел m на n . Елементи масиву ініціалізуються або вводяться з клавіатури.

1. Знайти найбільший елемент масиву. Вивести на екран вихідний масив і новий, в якому елементи стовпчика, що містить найбільший елемент, помножено на цей елемент.

2. Знайти найменший елемент масиву. Вивести на екран вихідний масив і новий, в якому елементи рядка, що містить найменший елемент, помножено на цей елемент.

3. Знайти номери рядка й стовпця, які містять найбільший елемент масиву. Вивести на екран вихідний масив і новий, елементи якого по стовпчиках впорядковано за зростанням.

4. Знайти номери рядка й стовпця, які містять найменший елемент масиву. Вивести на екран вихідний масив і новий, елементи якого по рядках впорядковано за спаданням.

5. Визначити добуток елементів у рядках, що не містять нульових елементів. Вивести на екран вихідний масив і новий, в якому всі додатні елементи замінено відповідними від'ємними.

6. Визначити суму елементів в стовпчиках, що не містять від'ємних елементів. Вивести на екран вихідний масив і новий, в якому всі від'ємні елементи замінено відповідними додатними.

7. Визначити номери рядків без нульових елементів. Вивести на екран вихідний масив і новий, в якому елементи вихідного масиву помножено на задане дійсне число.

8. Визначити номери стовпчиків без нульових елементів. Вивести на екран вихідний масив і новий, в якому до елементів першого рядка додано значення середнього арифметичного його елементів.

9. Визначити номер рядка, в якому знаходиться найбільша кількість заданих елементів. Вивести на екран вихідний масив і новий, елементи якого впорядковано за зростанням.

10. Визначити стовпчик з найбільшою кількістю нульових елементів. Вивести на екран вихідний масив і новий, елементи якого впорядковано за спаданням.

11. Знайти максимальний елемент першого рядка масиву. Вивести на екран вихідний масив і новий, в якому всі додатні елементи помножено на знайдене число.

12. Знайти мінімальний елемент останнього стовпчика масиву. Вивести на екран вихідний масив і новий, в якому всі від'ємні елементи помножено на знайдене дійсне число.

13. Знайти кількість нульових елементів в кожному стовпчику масиву. Вивести на екран вихідний масив і новий, в якому всі нульові елементи замінено заданим дійсним числом.

14. Знайти кількість ненульових елементів в кожному рядку масиву. Вивести на екран вихідний масив і новий, в якому елементи рядка, що містить найбільшу кількість ненульових елементів, помножено на знайдене дійсне число.

15. Знайти кількість від'ємних елементів в кожному стовпчику масиву. Вивести на екран вихідний масив і новий, в якому елементи стовпчика, що містить найбільшу кількість від'ємних елементів, замінено заданим цілим числом.

16. Знайти кількість додатних елементів в кожному рядку масиву. Вивести на екран вихідний масив і новий, в якому елементи рядка, що містить найменшу кількість додатних елементів, помножено на задане ціле число.

17. Знайти кількість елементів, що перевищують значення середнього арифметичного елементів масиву. Вивести на екран вихідний масив і новий, в якому всі елементи, менші за значення середнього арифметичного, замінено нулями.

18. Визначити стовпчик з найбільшою кількістю ненульових елементів. Вивести на екран вихідний масив і новий, в якому до елементів стовпчика з найменшою кількістю ненульових елементів додано число 5.

19. Знайти максимальний елемент головної діагоналі масиву. Вивести на екран вихідний масив і новий, в якому елементи першого стовпчика помножено на знайдене число.

20. Знайти мінімальний елемент головної діагоналі масиву. Вивести на екран вихідний масив і новий, в якому всі елементи головної діагоналі замінено на знайдене число.

Контрольні питання

1. Наведіть синтаксис звернення до певного елемента двовірного масиву.
2. В який спосіб можна задавати початкові значення елементам масиву?
3. Як ініціалізувати двовірний масив за допомогою списку ініціалізації?
4. Як ініціалізувати двовірний масив за допомогою циклу for?
5. Якими значеннями ініціалізуються елементи масиву, якщо початкових значень менше, ніж елементів масиву?
6. Які початкові значення отримають елементи двовірного масиву у наступному випадку ініціалізації: `array[2][3] = {1, 2, 3, 4, 5, 6};`
7. Які початкові значення отримають елементи двовірного масиву у наступному випадку ініціалізації: `array[2][3] = {{1, 2}, {4}};`
8. Чим визначається обсяг пам'яті, що виділяється для багатомірного масиву?

ПРАКТИЧНА РОБОТА №9

Тема: Функції

Мета роботи: розробка програмного забезпечення з реалізації алгоритмів на основі функцій.

Теоретичні відомості

Функції являють собою основу, на якій будується будь-яка C++ програма. Частиною середовища програмування C++ є *функції стандартної бібліотеки ANSI C*. Програміст сам пише функції, щоб визначити певні специфічні задачі, які можна використовувати у різних місцях програми. Ці функції називають *функціями, що визначені користувачем*.

Функція – це програмний блок, що містить одну чи кілька інструкцій та виконує певну задачу.

З використанням в C++ програмі функцій пов'язано три поняття: *оголошення функції (прототип), визначення функції та виклик функції*.

Подібно до того, як не можна використовувати змінну, не повідомивши компілятору інформацію про неї, так не можна звернутися до функції, не вказавши в програмі її необхідні атрибути. Є два способи описати функцію: *оголосити* або *визначити* функцію до її першого виклику.

Оголошення функції

Оголошення функцій називають *прототипами* функцій. Оголошення функції означає, що десь нижче в лістингу програми буде міститися код цієї функції.

У загальному випадку прототип містить інформацію про тип значення, що повертається функцією, а також кількість і тип аргументів функції.

Загальний формат оголошення функції має наступний вигляд:

тип_значення_що_повертається ім'я_функції (список_параметрів);

Наприклад,

```
float myfunc (float);      // прототип функції myfunc()
```

```
int sum_of_squares (int, int); // прототип функції sum_of_squares()
```

Оголошення функції закінчується крапкою з комою (;) і насправді є звичайним оператором.

Якщо специфікатор типу не заданий, то передбачається, що функція повертає значення типу `int`.

Відмінною рисою функцій є круглі дужки, що йдуть слідом за її ім'ям: по них компілятор відрізняє ім'я функції від імені змінної або іншого елемента програми.

Функції можна передати одне чи кілька значень. Значення, що передаються функції, називають *аргументами*.

В круглих дужках вказується кількість і тип аргументів функції – *формальних параметрів*. Імена формальних параметрів при оголошенні функції можна не вказувати, а якщо вони вказані, то їх область дії поширюється лише до кінця оголошення.

Допускається також використання функцій, які не мають аргументів і функцій, що не повертають жодних значень. Дія таких функцій може полягати, наприклад, у зміні значень деяких змінних, виведенні на друк деяких текстів і т.п.

Наприклад,

```
void box (int, int, int); // прототип функції box()
```

```
void starline (); // прототип функції starline()
```

Функція, яка не описана як `void`, повинна повертати деяке (єдине) значення. Це значення, що повертається, і є результатом виконання функції, котрий під час виконання програми підставляється у точку виклику функції. Значення, що повертається, задається оператором **return**.

За **return** може слідувати будь-який вираз.

Наприклад,

```
return x*x + y*y;
```

```
return 0;
```

Визначення функції

Визначення функції містить програмний код, тобто опис дій, які виконує функція. Визначення функції може слідувати як за визначенням функції `main()`, так й перед ним, або навіть знаходитися в іншому файлі.

Визначення функції складається із *заголовка* і *тіла* функції.

Загальний формат визначення функції має наступний вигляд:

```
тип_значення_що_повертається ім'я_функції (список_параметрів)  
{  
    //тіло функції  
}
```

Заголовок функції повинен відповідати її прототипу: ім'я функції і тип значення, що нею повертається, повинні збігатися із позначеними у прототипі. Крім того, аргументи функції (формальні параметри), якщо вони є, повинні мати ті ж типи і слідувати у тому ж порядку, в якому вони вказувалися у прототипі. *Формальні параметри* – це змінні, використовувані усередині тіла функції, які отримують значення під час виклику функції шляхом копіювання в них значень відповідних *фактичних параметрів*, що вказуються при виклику функції. Якщо тип формального параметра не зазначений, то цьому параметру присвоюється тип `int`.

Тіло функції являє собою послідовність операторів, укладених у фігурні дужки.

Наприклад,

```
int sum_of_squares (int x, int y) // заголовок функції  
{  
    return x*x + y*y; // тіло функції  
}
```

Заголовок функції не обмежується крапкою з комою (;).

У мові C++ немає вимоги, щоб визначення функції обов'язково передувало її виклику.

Виклик функції

Функція починає виконувати запроектовану для неї задачу шляхом *виклику* функції.

Під час виклику функції їй за допомогою аргументів (формальних параметрів) передаються певні значення (фактичні параметри), які використовуються під час виконання функції. При цьому тип кожного фактичного параметра звіряється з типом, зазначеним в описі функції.

Для виклику функції необхідно вказати її ім'я та список фактичних параметрів, які й будуть використовуватися під час виконання функції, укладений у круглі дужки.

Наприклад,

```
sum_of_squares (a, b); // виклик функції sum_of_squares()
```

```
void box (7, 6, 9); // виклик функції box()
```

```
starline(); // виклик функції starline()
```

Виклик функції завершується крапкою з комою (;).

Виконання оператора виклику функції ініціює виконання самої функції. Це означає, що управління передається операторам функції, які після свого виконання, у свою чергу, передають управління оператору, що слідує за викликом функції.

Приклад програми

Завдання: Написати програму, яка використовує функцію обчислення суми квадратів двох цілих чисел

```
#include <iostream>
```

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

```
using namespace std;
```

```
int sum_of_squares (int, int); // оголошення функції sum_of_squares()
```

```
void main ()
```

```
{
```

```
    int a, b;
```

```

cout << "Enter a & b: ";
cin >> a >> b;
cout << "Sum of squares " << a << " & " << b << " = "
    << sum_of_squares (a, b);    // виклик функції sum_of_squares()
cout << endl;
getch();
}
// визначення функції sum_of_squares()
int sum_of_squares (int x, int y) // заголовок функції
{
    return x*x + y*y;           // тіло функції sum_of_squares()
}

```

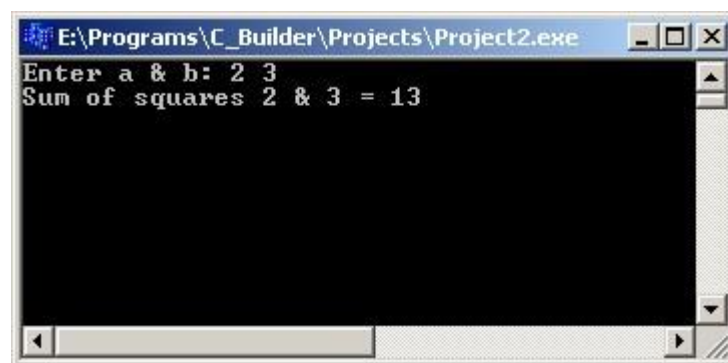


Рис. 9.1 – Результат роботи програми

Завдання для виконання

Написати програму, яка використовує функцію:

1. Обчислення факторіала числа, переданого їй в якості аргумента.

Використати функцію для виведення таблиці факторіалів чисел від 0 до 10.

2. Обчислення значення виразу a^b , де a і b - отримані в якості аргументів будь-які цілі числа.

3. Порівняння двох дійсних чисел, отриманих в якості аргументів. Функція має повертати результат порівняння у вигляді одного зі знаків: = чи !=.

4. Виведення на екран рядка однакових символів. Довжина рядка та символ отримуються функцією в якості аргументів.

5. Перетворення часу, отриманого в якості аргумента у вигляді годин, хвилин і секунд, на час у секундах.

6. Обчислення площі круга, а також функцію обчислення довжини кола, якщо в якості аргумента вони отримують радіус.

7. Порівняння двох цілих чисел, отриманих в якості аргументів. Функція має повертати результат порівняння у вигляді одного зі знаків: $>$, $<$ чи $=$.

8. Обчислення значення виразу $(a + b)^x$, де a , b та x – будь-які цілі числа, отримані в якості аргументів.

9. Знаходження найбільшого спільного дільника трьох цілих чисел, отриманих в якості аргументів.

10. Знаходження мінімального з трьох дійсних чисел, отриманих в якості аргументів.

11. Перетворення часу у секундах, отриманого в якості аргумента, на час у вигляді годин, хвилин і секунд.

12. Знаходження дискримінанта квадратного рівняння $ax^2+bx+c=0$. Коефіцієнти рівняння (a , b c) отримуються функцією в якості аргументів.

13. Обчислення відстані між двома точками площини. Координати точок отримуються функцією в якості аргументів.

14. Перетворення значення температури в градусах Цельсія, отриманої в якості аргумента, на еквівалентну температуру в градусах Фаренгейта, а також функцію перетворення значення температури в градусах Фаренгейта, отриманої в якості аргумента, на еквівалентну температуру в градусах Цельсія. $[F = 9/5 C + 32 = 1.8 C + 32; C = (F - 32)/1.8]$.

15. Обчислення об'єму конуса, а також функцію обчислення площі його основи, якщо в якості аргумента вони отримують радіус основи.

16. Перетворення значення довжини в метрах, отриманої в якості аргумента, на еквівалентну довжину у футах, а також функцію перетворення значення довжини у футах, отриманої в якості аргумента, на еквівалентну довжину в метрах $[1 \text{ фут} \approx 0.3048 \text{ м}]$.

17. Знаходження найменшого спільного кратного трьох цілих чисел, отриманих в якості аргументів.

18. Знаходження більшого з трьох дійсних чисел, отриманих в якості аргументів.

19. Обчислення значення виразу $a^x + b^x$, де a , b та x – будь-які цілі числа, отримані в якості аргументів.

20. Обчислення площі поверхні кулі, а також функцію обчислення її об'єму, якщо в якості аргумента вони отримують радіус кулі.

Контрольні питання

1. Що таке функція?
2. У чому полягає причина побудови програм на основі функцій?
3. Як оголошувати функції?
4. Як визначати функції?
5. Що називають тілом функції?
6. Що називають списком параметрів функції?
7. Які параметри називають фактичними?
8. Які параметри називають формальними?
9. Які змінні називають локальними змінними?
10. Що називають прототипом функції?
11. Що таке тип значення, що повертається?
12. Як повертати значення функції?
13. Як треба оголосити функцію, що не повертає значення?
14. Яке призначення оператора `return`?
15. Яким чином активізується функція?

ПРАКТИЧНА РОБОТА №10

Тема: Рядки

Мета роботи: освоєння прийомів роботи із символами й рядками, застосування бібліотечних функцій для обробки рядків.

Теоретичні відомості

Рядки як одновимірні масиви

У мові C++ рядок подається як одновимірний масив змінних типу `char`, останнім елементом якого автоматично є нульовий символ `'\0'`. Можливі кілька видів оголошень масиву символів. По-перше, це може бути абстрактний рядок без вказівки розміру масиву:

```
char name[] = "Andrey";
```

Тут число елементів масиву визначається автоматично й дорівнює 7 (з урахуванням останнього елемента `'\0'`). По-друге, масив символів може мати більше елементів, чим містить абстрактний рядок:

```
char name[10] = "Andrey";
```

Тут три останніх елементи масиву не використовуються. Нарешті, рядок може бути оголошений за допомогою покажчика. Перед використанням покажчика рядок вже повинен існувати, наприклад:

```
char name[] = "Andrey";
```

```
char *namePtr;
```

```
namePtr = name;
```

Відзначимо, що при присвоєнні покажчику адреси масиву не потрібно використати адресний оператор `&`, тому що ім'я масиву поводить себе подібно адреси.

Передача рядка у функцію

Якщо потрібно передати в деяку функцію рядок, то передається як параметр ім'я рядка-масиву або покажчик на рядок.

Використання імені масиву. Запишемо визначення функції, що друкує рядок, і виклик цієї функції:

```
void print(char client[]){  
    cout<< "Name of client: " <<client<<endl;}  
  
...  
  
int main(){  
    char client1[] = "Serg";  
  
    ...  
  
    print(client1);  
  
    ...
```

Використання покажчика на рядок.

Якщо є покажчик на масив-рядок:

```
char *client1Ptr= client1;
```

то можна передати у функцію покажчик:

```
print(client1Ptr);
```

Результат буде тим же, як і при передачі імені масиву, оскільки масив передається у функцію через посилання, тобто через покажчик.

Введення/виведення символів і рядків

Розглянемо функції, які призначені для введення/виведення символів і рядків і належать до стандартної бібліотеки (заголовний файл `stdio.h`).

Введення по одному символу з потоку введення – здійснюється із клавіатури за допомогою функції `getchar()`, що повертає код уведеного символу:

```
int getchar(void);
```

Виведення одного символу на екран – здійснюється функцією `putchar()`, параметром якої є код виведеного символу:

```
int putchar(int ch);
```

Введення символу без відображення на екрані – здійснюється функцією `getch()` з бібліотеки `conio`. (заголовний файл `conio.h`).

```
char getch(void);
```

У цієї функції, як і в `getchar` немає параметру. Крім того, значення, що повертається функцією, може бути присвоєно символьній змінній типу `char`.

Введення рядка символів із клавіатури – здійснюється функцією `gets()`, параметром якої є покажчик на початок того рядка, у який вводяться дані. Функція повертає той же самий покажчик:

```
char *gets(char *string);
```

Функція `gets` читає все, що набирається, поки не натиснута `Enter`. Ця функція дозволяє вводити пробіли, чим відрізняється від функції введення `scanf`, що читає всі символи, поки не зустрінеться пробіл, табуляція або кінець рядка.

Виведення рядка символів на екран – здійснюється функцією `puts()`, параметром якої є покажчик на початок того рядка, з якого виводяться дані. Функція повертає кількість виведених символів:

```
int puts(char *string);
```

Бібліотечні функції для роботи з символами та рядками

У мовах C/C++ передбачені спеціальні функції, призначені для роботи із символами (табл. 10.1). Частина функцій перевіряють одиночні символи й повертають ненульове значення (`true`) або нуль (`false`). Наприклад, `isdigit(c)` дозволяє перевірити, чи є змінна `c` однією із цифр між 0 і 9. Інша частина функцій у таблиці 10.1 забезпечує конвертування регістра букв. Так, `tolower(c)` повертає версію букви `c` у нижньому регістрі, тобто малу літеру. Наприклад, інструкція `cout<<tolower('F')` виведе на екран символ `f`.

Наступний фрагмент коду використовується як елемент циклу `do-while`, у якому користувач для продовження циклу повинен увести 'Y'. Цикл буде виконуватися незалежно від того, у якому регістрі уведений символ: 'y' або 'Y'.

```
do{ // тіло циклу  
}  
while (toupper(choice)=='Y')
```

Параметром функцій у таблиці 10.1 є змінна `c` типу `char` або `int`, а для звертання до функцій підключається заголовний файл `ctype.h`.

Табл. 10. 1. Функції для роботи із символами

Тип	Функція	Опис
int	isalnum(c)	повертає true, якщо c – буква або цифра
int	isalpha(c)	повертає true, якщо c – буква
int	isblank(c)	повертає true, якщо c – пробіл або символ табуляції
int	isdigit(c)	повертає true, якщо c – цифра
int	islower(c)	повертає true, якщо c – символ у нижньому регістрі
int	isupper(c)	повертає true, якщо c – символ у верхньому регістрі
int	isspace(c)	повертає true, якщо c – пробіл, табуляція, повернення каретки, новий рядок
int	tolower(c)	повертає символ c у нижньому регістрі
int	toupper(c)	повертає символ c у верхньому регістрі

Для обробки рядків призначені функції, прототипи яких утримуються у файлі string.h. Найбільш популярні з них наведені в таблиці 10.2.

Табл. 10.2. Функції обробки рядків

Прототип	Опис
int strlen(char*s)	Довжина рядка, але без нуль-символу кінця рядка
int strcmp(char*s1, char*s2)	Порівнює рядки s1 і s2, повертає 0 (рівні), < 0 (s1 < s2), > 0 (s1 > s2)
char *strcpy(char*s1, char*s2)	Копіювання рядка s2 у рядок s1; повертає s2
char *strcat(char*s1, char*s2)	Додає s2 до рядка s1; перший символ s2 записується поверх нуль-символу s1; повертає s1
char *strchr (char*s, charch)	Пошук у рядку s першого входження символу ch

Завдання для виконання

Скласти програму для роботи з рядками відповідно свого варіанту, використовуючи по можливості бібліотечні функції.

Варіант	Завдання
1	Скласти програму, що у рядку, уведеному користувачем, підраховує число входжень того або іншого символу. Написати функцію count

	обчислення числа входжень у рядок s символу c: <code>int count(char *s, char c);</code>
2	Скласти програму, що у рядку, уведеному користувачем, перетворить першу букву кожного слова в прописну. Написати функцію такого перетворення, що має прототип: <code>void capitalize(char *s);</code>
3	Скласти програму, що дозволяє для рядка, уведеного користувачем, визначити кількість слів, що починаються із цифри
4	Скласти програму, що дозволяє для рядка, уведеного користувачем, визначати: (1) його довжину; (2) кількість цифр; (3) кількість букв. Вибір дії здійснюється за допомогою меню.
5	У масиві рядків, уведених користувачем, знайти перші входження символу 'm'. Використати функцію <code>strchr()</code> .
6	У масиві рядків, уведених користувачем, знайти перші входження символів 's' і 'S'. Використати функції <code>strchr()</code> , <code>tolower()</code> .
7	Скласти програму сортування масиву рядків, уведених користувачем. Використати функцію <code>strcmp()</code> .
8	Скласти програму, що видаляє всі пробіли в рядку, уведеному користувачем.
9	Скласти програму, що дозволяє для двох рядків, уведених користувачем, виконати: (1) порівняння; (2) копіювання; (3) конкатенацію. Вибір дії здійснюється за допомогою меню.
10	Скласти програму, що дозволяє для рядка, уведеного користувачем, визначати: (1) його довжину; (2) кількість слів. Вибір дії здійснюється за допомогою меню.
11	Скласти програму, що визначає, чи є слово, уведене користувачем, паліндромом.
12	Скласти програму, що читає слово, уведене користувачем, і міняє порядок букв у слові на зворотний. Написати функцію обертання слова, що має прототип: <code>char *word_reverse(char *s);</code>
13	Скласти програму, що підраховує число слів у рядку, уведеного користувачем. Для цього написати функцію із прототипом: <code>int num_word(char *s);</code>
14	Скласти програму, що підраховує число малих літер у рядку, уведеної користувачем. Для цього написати функцію із прототипом: <code>int num_lower(char *s);</code>
15	Скласти програму, що підраховує число слів у рядку, уведеного користувачем. Для цього написати функцію із прототипом: <code>int num_word(char *s);</code>

16	Скласти програму, що у рядку, уведеному користувачем, підраховує число входжень того або іншого символу. Написати функцію count обчислення числа входжень у рядок s символу c: <code>int count(char *s, char c);</code>
17	Скласти програму, що дозволяє для рядка, уведеного користувачем, визначати: (1) його довжину; (2) кількість слів. Вибір дії здійснюється за допомогою меню.
18	Скласти програму, що дозволяє для рядка, уведеного користувачем, визначити кількість слів, що починаються із цифри
19	Скласти програму сортування масиву рядків, уведених користувачем. Використати функцію <code>strcmp()</code> .
20	Скласти програму, що підраховує число малих літер у рядку, уведеної користувачем. Для цього написати функцію із прототипом: <code>int num_lower(char *s);</code>

Контрольні питання

1. У чому є специфіка символьних масивів?
2. Що таке рядок, який закінчується нулем (null-terminated string)?

ПРАКТИЧНА РОБОТА № 11-12

Тема: Структури. Масиви структур

Мета роботи: оволодіти практичними навичками зі створення та використання структур, передавання структур у функції в якості параметрів; навчитись використовувати масиви структур та здійснювати операції з полями структур; навчитися складати програми з використанням структур.

Теоретичні відомості

Структури дозволяють об'єднувати в єдиному об'єкті сукупність значень, які можуть мати різні типи. Оголошення структури здійснюється за допомогою ключового слова *struct*.

Синтаксис опису структури:

```
struct [ім'я_структури] {  
    тип1 елемент1;  
    .....  
    типN елементN;  
} [список описів];
```

Приклад структури – представлення поняття "дата", що складається з декількох частин: число (день, місяць, рік), назва тижня та місяця:

```
struct date {  
    int day ;  
    int month ;  
    int year;  
    char day_name[15];  
    char mon_name[14];  
} arr[100], *pd, data, new_data;
```

Тут *data*, *new_data* – змінні типу *date*; *pd* – покажчик на тип *date*; *arr* – масив зі 100 елементів, кожний з яких має тип *date*.

Доступ до полів структури

Доступ до полів структури забезпечується операторами вибору: *.* (прямий селектор) та *->* (непрямий селектор), наприклад,

```
struct mystruct {  
    int i;  
    char str[21];  
    double d;  
} s, *sptr=&s;  
s.i =3;  
sptr->d = 1.23;
```

Для змінних одного і того ж самого структурного типу визначена операція присвоювання, при цьому здійснюється поелементне копіювання значень полів.

```
data=new_data;
```

Для порівняння структур необхідно перевіряти рівність відповідних полів цих структур.

```
struct point {  
    float x,y;  
    char c;  
} point1,point2;  
if (point1.x==point2.x&&point1.y==point2.y&&point1.c==point2.c){  
    /* ... */  
};
```

Масиви структур

Масивами структур, елементи якого мають структурований тип, можна оперувати, як звичайними масивами простих типів. Приклад:

```
typedef struct Date  
{
```



```

    int d; /* день */
    int m; /* місяць */
    int y; /* рік */
} Date;
Date arr[100];

```

Доступ до полів структури аналогічний доступу до звичайних змінних, при цьому у квадратних дужках вказують індекс номеру елемента:

```
arr[25].d=24; arr[12].m=12;
```

Об'єднання (union)

Об'єднання дозволяють в різні моменти часу зберігати в одному об'єкті значення різного типу. В кожний момент часу об'єднання може зберігати значення тільки одного типу з набору. Контроль за тим, значення якого типу зберігається в даний момент, покладається на програміста.

<i>union sign</i>	<i>union</i>
{ int svar;	{ char *a,b;
unsigned uvar;	float f[20];
} number;	} var;

Бітові поля

Бітові поля (*bit fields*) – особливий вид полів структури. Вони дають можливість задавати кількість бітів, в яких зберігаються елементи цілих типів. При оголошенні бітового поля вслід за типом елемента ставиться двокрапка (:) і вказується цілочисельна константа, яка задає розмір поля (кількість бітів). Розмір поля – константа в діапазоні між 0 і числом бітів, яке використовується для зберігання даного типу даних.

```

struct bit_field {
    int bit_1 : 1;
    int bits2_5 : 4;
    int bit_6 : 1;

```

```
int bits7_16: 10;  
} bit_var;
```

Приклади програм з використанням структур

Приклад 1. Сформувати список про зареєстровані перездачі заборгованостей студентів: ПІБ, предмет, кількість перездач. Роздрукувати список за заданою ознакою і видати його на екран дисплея.

Лістинг програми

```
#include <stdio.h>  
#include <conio.h>  
#define n 3  
void main()  
{ int i,f=0,obrz;  
  struct  
  { char FIO[20];  
    char predmet[20];  
    int kol_peresdach;  
  } dolg[n];  
  for(i=0;i<n;i++)  
  { printf("\nBedite FIO      ");  
    scanf("%s",&dolg[i].FIO);  
    printf("Bedite predmet    ");  
    scanf("%s",&dolg[i].predmet);  
    printf("Bedite kol_peresdach ");  
    scanf("%d",&dolg[i].kol_peresdach);  
  }  
  printf("\nBedite obrz    ");  
  scanf("%d",&obrz);  
  for(i=0;i<n;i++)  
  { if(dolg[i].kol_peresdach==obrz)
```

```

{ f++;
  printf("\n%15s\t%15s\t%3i ",
        dolg[i].FIO,dolg[i].predmet,dolg[i].kol_peresdach);
}
}
if(f==0)printf("\n Error");
getch();
}

```

Завдання для виконання

До практичної роботи № 11:

1.	<p>Описати структуру з іменем STUD, яка містить поля: NAME – прізвище та ініціали; GROUP – група; SES – оцінки з п'яти предметів (масив з п'яти елементів). Написати програму, що реалізує наступні дії окремими функціями:</p> <ul style="list-style-type: none"> – введення з клавіатури даних в масив ST, що складається з N змінних типу STUD; – виведення на екран прізвищ і номерів груп для всіх студентів, середній бал яких більший за 4.0; якщо таких немає, то вивести повідомлення.
2.	<p>Описати структуру з іменем ABIT, яка містить поля: NAME – прізвище, ініціали; GENDER – стать; SPEC – назва спеціальності; EXAM – результати вступних іспити з трьох предметів (масив з трьох елементів). Написати програму, що окремими функціями реалізує наступні дії:</p> <ul style="list-style-type: none"> – введення з клавіатури даних в масив AB, що складається з N змінних типу ABIT; – виведення на екран прізвищ та назв спеціальностей для всіх абітурієнтів, що мають бал нижче, ніж прохідний, який визначається користувачем програми; якщо таких студентів немає, то вивести повідомлення про це.
3.	<p>Описати структуру з іменем NOTE, яка містить поля: NAME – прізвище, ім'я; TEL – номер телефону; BDAY – день народження (масив із трьох чисел). Написати програму, що окремими функціями виконує наступні</p>

	<p>дії:</p> <ul style="list-style-type: none"> – введення з клавіатури даних в масив BLOCKNOTE, що складається з N змінних типу NOTE; – виведення на екран інформації про людей, чиї дні народження припадають на місяць, значення якого введено з клавіатури; якщо таких людей немає, то вивести відповідне повідомлення.
4.	<p>Описати структуру з іменем AEROFLOT, яка містить поля: CITY – назва населеного пункту призначення; NUM – номер рейса; TYPE – тип літака. Написати програму, що окремими функціями реалізує наступні дії:</p> <ul style="list-style-type: none"> – введення з клавіатури даних в масив AIR, що складається з N змінних типу AEROFLOT; – виведення на екран номерів рейсів і типів літаків, що вилетіли в пункт призначення, назва якого співпала з назвою, введеною з клавіатури; якщо таких рейсів немає, то вивести відповідне повідомлення.
5.	<p>Описати структуру з іменем SKLAD, яка містить поля: NAME – назва товару; TYPE – одиниця виміру; COST – ціна одиниці товару; QUANTITY – кількість. Написати програму, що окремими функціями виконує дії:</p> <ul style="list-style-type: none"> – введення з клавіатури даних в масив SHOP, що складається з N змінних типу SKLAD; – виведення на екран інформації про товар, його кількість, ціну одиниці та обчислену загальну суму на складі, назва якого вводиться з клавіатури; якщо такого немає, то вивести відповідне повідомлення.
6.	<p>Описати структуру з іменем WORK, яка містить поля: NAME – прізвище та ініціали працівника; POS – назва посади; YEAR, MONTH – рік і місяць прийняття на роботу. Написати програму, що окремими функціями виконує наступні дії:</p> <ul style="list-style-type: none"> – введення з клавіатури даних в масив TABL, що складається з N змінних типу WORK; – виведення прізвища працівників, стаж роботи яких перевищує значення, введене з клавіатури; якщо таких немає, то вивести повідомлення.
7.	<p>Описати структуру з іменем TRAIN, яка містить поля: NAZV – назва пункту призначення; NUMR – номер потягу; DATE, TIME – дата і час відправлення. Написати програму, що окремими функціями виконує дії:</p> <ul style="list-style-type: none"> – введення з клавіатури даних в масив RASP, що складається з N змінних типу TRAIN;

	<p>– виведення на екран інформації про поїзди, що відправляються після введеного з клавіатури дня та часу; якщо таких поїздів немає, то вивести відповідне повідомлення.</p>
8.	<p>Описати структуру з іменем TABLE, яка містить поля: NAZV – назва пункту призначення; NUMR – номер поїзда; DATE, TIME – дата і час відправлення. Написати програму, що окремими функціями виконує дії:</p> <ul style="list-style-type: none"> – введення з клавіатури даних в масив TRAIN, що складається з N змінних типу TABLE; – виведення на екран інформації про поїзди, що направляються в пункт призначення, назва якого введена з клавіатури; якщо таких немає, то вивести повідомлення про це.
9.	<p>Описати структуру з іменем ZNAK, яка містить поля: NAME – прізвище, ім'я; ZODIAC – знак Зодіаку; BDAY – день народження (масив із трьох чисел). Написати програму, що окремими функціями виконує наступні дії:</p> <ul style="list-style-type: none"> – введення з клавіатури даних в масив BOOK, що містить N змінних типу ZNAK; – виведення на екран інформації про людину, чиє прізвище введене з клавіатури; якщо таких людей немає, то вивести відповідне повідомлення.
10.	<p>Описати структуру з іменем ITNR, яка містить наступні поля: FIRST – назва початкового пункту маршруту; FINAL – назва кінцевого пункту маршруту; NUM – номер маршруту; DISTANCE – відстань у кілометрах. Написати програму, що окремими функціями виконує наступні дії:</p> <ul style="list-style-type: none"> – введення з клавіатури даних в масив RT, що складається з N змінних типу ITNR; – виведення на екран інформації про маршрут, номер якого введений з клавіатури; якщо таких маршрутів немає, то вивести відповідне повідомлення.
11.	<p>Описати структуру з іменем ITIN, яка містить поля: BEG, END – назви початкового і кінцевого пунктів маршруту; NUM – номер маршруту; DIST – відстань у кілометрах. Написати програму, що окремими функціями виконує дії:</p> <ul style="list-style-type: none"> – введення з клавіатури даних в масив RT, що складається з N змінних типу ITIN; – виведення на екран інформацію про маршрути, які починаються або закінчуються в пункті, назва якого введена з клавіатури; якщо таких

	немає, то вивести повідомлення.
12.	<p>Описати структуру з іменем SCHL, яка містить поля: NAME – прізвище та ім'я учня; GROUP – номер групи; SUBJ – успішність з п'яти предметів (масив з п'яти елементів). Написати програму, що окремими функціями виконує дії:</p> <ul style="list-style-type: none"> – введення з клавіатури даних в масив LEARN, що складається з N змінних типу SCHL; – виведення на екран прізвищ і номерів груп для всіх студентів, що мають хоча б одну оцінку "2"; якщо таких немає, то вивести відповідне повідомлення.
13.	<p>Описати структуру з іменем TBL, яка містить поля: NAZV – назва пункту призначення; NUMR – номер поїзда; DATE – дата відправлення; TIME – час відправлення. Написати програму, що окремими функціями виконує наступні дії:</p> <ul style="list-style-type: none"> – введення з клавіатури даних в масив TRAIN, що складається з N структур типу TBL; – виведення на екран інформацію про поїзди, дата відправлення яких введена з клавіатури; якщо таких поїздів немає, то вивести відповідне повідомлення.
14.	<p>Описати структуру з іменем ABIT, яка містить поля: NAME – прізвище, ініціали; GENDER – стать; SPEC – назва спеціальності; EXAM – результати вступних іспитів з трьох предметів (масив з трьох елементів). Написати програму, що окремими функціями виконує наступні дії:</p> <ul style="list-style-type: none"> – введення з клавіатури даних в масив ABIT, що складається з N змінних типу ABIT; – виведення на екран прізвищ та назв спеціальностей для всіх абітурієнтів, що набрали прохідний бал, який визначається користувачем програми; якщо таких студентів немає, то вивести відповідне повідомлення.
15.	<p>Описати структуру з іменем TOVAR, яка містить наступні поля: NAME – назва товару; TYPE – одиниця виміру товару; SORT – сорт товару; QUANTITY – кількість одиниць товару; COST – ціна. Написати програму, що окремими функціями виконує наступні дії:</p> <ul style="list-style-type: none"> – введення з клавіатури даних в масив SHOP, що складається з N змінних типу TOVAR; – виведення на екран інформації про товар, його кількість, ціну одиниці та обчислену загальну суму товару на складі; назва товару

	вводиться з клавіатури, якщо його немає, то вивести відповідне повідомлення.
--	--

До практичної роботи № 12

1.	Інформація по N заводах міста задається рядком такого вигляду: прізвище, середній вік, спеціальність, середній оклад. Ввести інформацію по заводах, порахувати кількість слюсарів та токарів. Надрукувати значення та номери заводів, де середній вік вище 35 років.
2.	Багаж пасажирів характеризується кількістю речей і загальною вагою. Ввести інформацію про N пасажирів і визначити, чи є серед пасажирів такий, у якого найбільший багаж по кількості і за вагою.
3.	Інформація по N заводах міста задається рядком такого вигляду: прізвище, середній вік, спеціальність, середній оклад. Ввести інформацію по заводах, порахувати середній оклад по всіх заводах. Порахувати кількість заводів, де середній оклад по заводу вище середнього по всіх заводах. Надрукувати це значення і вивести інформацію по цих заводах.
4.	Є інформація за підсумками іспитів в інституті, всього в списку N чоловік. По кожному з студентів є такі відомості: прізвище, оцінка з математики, оцінка з інформатики та оцінка з фізики. Ввести інформацію про іспити і надрукувати кількість і прізвища відмінників та кількість і прізвища студентів, які мають хоча б одну двійку.
5.	Є інформація за підсумками іспитів в інституті, всього в списку N чоловік. По кожному із студентів є такі відомості: прізвище, оцінка з математики, оцінка з інформатики та оцінка з фізики. Ввести інформацію про іспити, обчислити і надрукувати середній бал з кожної дисципліни. Надрукувати прізвища студентів, у яких бал по кожному з предметів вище середнього по цьому предмету.
6.	У населеному пункті проживає N чоловік. Про кожного відомі прізвище,

	вік, стать. Ввести інформацію про жителів даного пункту і порахувати кількість жінок і чоловіків, вивести інформацію про тих, кого більше.
7.	У населеному пункті проживає N чоловік. Про кожного відомі прізвище, вік, стать. Ввести інформацію про жителів даного пункту і порахувати кількість жінок і чоловіків, вивести інформацію про середній вік чоловіків, надрукувати прізвища тих чоловіків, чий вік нижче середнього.
8.	Картотека відеотеки організована у вигляді масиву структур з полями: назва фільму, вартість, режисер. Ввести інформацію по відеотеці і вивести інформацію про фільми одного режисера. Вивести інформацію про всі фільми за зростанням вартості.
9.	Картотека відеотеки організована у вигляді масиву структур з полями: назва фільму, вартість, режисер. Ввести інформацію по відеотеці і вивести інформацію про фільми, вартість яких більша, ніж середня, максимальна і мінімальна вартість.
10.	Є інформація про N членів спортивної секції: прізвище, вік, зріст. Ввести інформацію про кожного. Надрукувати прізвище найвищого. Надрукувати інформацію про тих, чий вік нижче середнього, а зріст вище середнього по секції.
11.	Є інформація про N членів спортивної секції: прізвище, вік, зріст. Ввести інформацію про кожного. Вивести інформацію про всіх спортсменів у таблиці. Надрукувати прізвища та вік тих, чий вік вище середнього.
12.	Є інформація за підсумками іспитів в інституті всього в списку N чоловік. По кожному із студентів є такі відомості: прізвище, оцінка з математики, оцінка з інформатики та оцінка з фізики. Ввести інформацію про іспити і надрукувати кількість і прізвища студентів, які отримали на іспиті дві п'ятірки і одну четвірку.
13.	Є інформація про N учасників спортивних змагань з п'ятиборства. Про

	кожного учасника відома наступна інформація: прізвище, місце, зайняте по кожному з видів. Ввести інформацію про учасників змагань і вивести інформацію про переможця у п'ятиборстві.
14.	Є інформація про N учасників спортивних змагань з п'ятиборства. Про кожного учасника відомо: прізвище, місце, зайняте по кожному з видів. Ввести інформацію про учасників змагань і вивести інформацію про переможця у кожному виді спорту.
15.	Є інформація за підсумками іспитів в інституті, всього в списку N чоловік. По кожному із студентів є такі відомості: прізвище, оцінка з математики, оцінка з інформатики та оцінка з фізики. Ввести інформацію про іспити і надрукувати кількість і прізвища студентів, які склали іспити тільки з однією четвіркою. Надрукувати кількість і прізвища студентів, які здали інформатику з оцінкою відмінно.

Контрольні питання

1. Що таке структури? Який синтаксис опису структур?
2. Як можна оголосити статичний та динамічний масив структур?
3. Наведіть приклад оголошення масиву змінних типу структура.
4. Як можна звернутись до полів структури? Наведіть приклади .
5. Як здійснити присвоєння структур та порівняння структур? В чому особливість цих операцій над структурами?
6. Що являють собою об'єднання? Наведіть приклади об'єднань.
7. Що таке бітові множини? Для чого їх використовують?

ПРАКТИЧНА РОБОТА №13

Тема: Класи

Мета роботи: розробка програмного забезпечення з реалізації алгоритмів роботи з класами

Теоретичні відомості

Клас – це базова C++ одиниця інкапсуляції. Класи використовують для створення об'єктів.

Клас можна представити як деякій шаблон, що визначає формат об'єкта. Клас включає як дані, так і функції для виконання дій над цими даними. У C++ специфікацію класу використовують для побудови об'єктів. Об'єкти – це екземпляри класу. Важливо розуміти, що клас – це логічна абстракція, котра реально не існує до тих пір, поки не буде створений об'єкт цього класу.

Функції та змінні, що складають клас, називають його членами. Змінну, оголошену у класі, називають даним-членом, а функцію, оголошену у класі, називають функцією-членом. Члени класу можуть мати три рівні доступу: закритий (private), відкритий (public) чи захищений (protected). Рівні доступу до членів класу визначають спосіб роботи користувачів із класом.

Оголошення класу

Оголошення класу містить оголошення членів даних і функцій-членів класу. Клас оголошується за допомогою ключового слова `class`. Загальний формат оголошення класу має наступний вигляд:

```
class ім'я_класу {  
    private:  
        // закриті дані і функції  
    public:  
        // відкриті дані і функції  
    protected:  
        // захищені дані і функції
```

```
};
```

Наприклад:

```
class MyClass {  
private:          // закриті дані і функції  
    int date;  
public:           // відкриті дані і функції  
    void memfunc(int d)  
    {date = d;}  
};
```

Ім'я класу стає ім'ям нового типу даних, яке можна використовувати для створення об'єктів класу. Всі операції програма виконує над об'єктами. Об'єкти визначаються у функції `main()`. Визначення об'єкта схоже на визначення змінної: воно означає виділення пам'яті, необхідної для зберігання об'єкта.

Доступ до членів класу

Дані-члени класу та функції-члени мають областю дії клас. В області дії дані-члени класу безпосередньо доступні усім функціям-членам цього класу й на них можна посилатися просто за іменем. Поза областю дії до елементів класу можна звертатися або через ім'я об'єкта, або посиланням на об'єкт, або за допомогою вказівника на об'єкт.

Доступ до функцій (методів) класу можливий лише через конкретний об'єкт цього класу. При цьому використовують операцію крапка (`.`), яка пов'язує метод з ім'ям об'єкта.

Конструктор класу

Класи в C++ мають спеціальну функцію, яку називають конструктором.

Конструктор – це функція, яка автоматично викликається при створенні (реалізації) об'єкта класу.

Конструктор використовують для ініціалізації змінних-членів класу, виділення необхідної пам'яті та виконання інших дій, необхідних перед початком використання класу. Ім'я конструктора повинне співпадати з ім'ям

класу. Це слугує характерною ознакою конструктора. Крім того, для конструктора не вказують тип значення, що повертається, адже конструктор не може повертати ніякого значення.

Клас може мати більше одного конструктора. Це можливе завдяки перевантаженню функцій. Наприклад, можна визначити конструктор без аргументів (конструктор за замовчуванням) та конструктор з параметрами, що приймає один чи кілька аргументів для ініціалізації членів-даних.

Деструктор класу

Деструктор – це спеціальна функція-член класу, що спрацьовує під час знищення динамічно розміщеного об'єкту класу і звільняє займану ним пам'ять.

Ім'я деструктора співпадає з ім'ям класу. Перед ім'ям деструктора записується символ тильда «~». Так само як і для конструктора, для деструктора не вказують тип значення, що повертається.

Наприклад:

```
class MyClass {  
    public:  
        MyClass; // конструктор класу  
        ~MyClass; // деструктор класу  
        ...  
};
```

Деструктори необхідні, якщо конструктор або інші функції-члени класу динамічно розподіляють пам'ять, створюючи в ній об'єкти. Тоді деструктор повинен ці об'єкти видалити. В інших випадках зазвичай можна обійтися без деструктора.

Якщо деструктор явним чином в класі не оголошений, компілятор сам генерує необхідні коди звільнення пам'яті.

Приклад програми

Завдання: Написати програму для тестування класу `Int`, що імітує стандартний тип даних `int`. Єдине поле цього класу повинне мати тип `int`. Передбачити конструктори класу та методи, що будуть ініціалізувати його

об'єкти нулем, цілим значенням, виводити значення на екран, а також додавати два цілі числа.

```
#include <iostream>
#include <conio.h>
using namespace std;
class Int      // оголошення класу
{
private:
    int i;
public:
    Int()      // конструктор за замовчуванням
    { i = 0; } // ініціалізація Int нулем
    Int(int i1) // конструктор з одним параметром
    { i = i1; } // ініціалізація Int
    void add(Int i2, Int i3) // додавання двох значень типу Int
    { i = i2.i + i3.i; }
    void display() // виведення Int
    { cout << i; }
};
void main()
{
    Int Int1(5); // створення та ініціалізація першої змінної типу Int
    // значенням 5
    Int Int2(12); // створ. та ініціалізація другої змінної типу Int значенням 12
    Int Int3;      // створення та ініціалізація нулем третьої змінної типу Int
    cout << "\nInt1 = "; Int1.display(); // виведення першої змінної
    cout << endl;
    cout << "\nInt2 = "; Int2.display(); // виведення другої змінної
    cout << endl;
    cout << "\nInt3 = "; Int3.display(); // виведення третьої змінної
```

```

cout <<endl;
Int3.add(Int1,Int2);    //додавання двох змінних типу Int
cout <<"\nInt3 = "; Int3.display(); //виведення рез-ту додавання
cout <<endl;
getch();
}

```

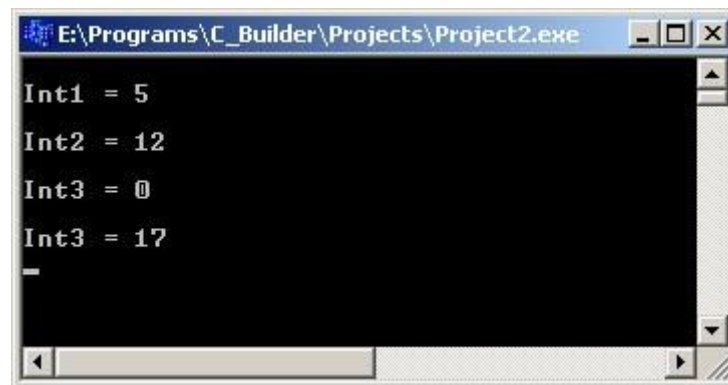


Рис. 13.1 – Результат роботи програми

Завдання для виконання

Написати програму для тестування класу:

1. Клас `Int` імітує стандартний тип даних `int`. Єдине поле цього класу повинне мати тип `int`. Передбачити конструктори класу та методи, що будуть ініціалізувати його об'єкти нулем, цілим значенням, виводити значення на екран, а також додавати, віднімати, множити і ділити два цілих числа.

2. Клас `Float` імітує стандартний тип даних `float`. Єдине поле цього класу повинне мати тип `float`. Передбачити конструктор класу та методи, що будуть ініціалізувати об'єкти класу, виводити значення на екран, а також додавати і віднімати два дійсних числа.

3. Клас `Vector` імітує роботу із векторами у двовимірному просторі. Полями цього класу повинні бути чотири поля типу `int`, призначені для зберігання координат точок початку і кінця вектора. Передбачити конструктор

класу та методи для виведення інформації на екран, а також для знаходження координат вектора та порівняння (= чи ?) двох векторів на рівність.

4. Клас `Date` моделює роботу з календарем. Полями цього класу повинні бути три поля типу `int`: рік, місяць і день. Передбачити конструктор класу, методи ініціалізації та виведення дати у форматі `dd/mm/yy`, а також декремента поточної дати на один день. При цьому необхідно врахувати можливий перехід до попереднього місяця та року.

5. Клас `Time` моделює роботу з годинником. Полями цього класу повинні бути три поля типу `int`, призначені для зберігання годин, хвилин, секунд. Один з конструкторів має ініціалізувати годинник нульовими значеннями, а інший – заданим набором значень. Передбачити методи класу для виведення часу на екран у форматі `23:59:59`, а також інкремента хвилин поточного часу. При цьому необхідно врахувати можливий перехід до наступної години.

6. Клас `Complex` імітує роботу з комплексними числами. Полями цього класу повинні бути два поля типу `int`, призначені для зберігання дійсної й уявної частини комплексного числа. Передбачити конструктор за замовчуванням, методи класу для віднімання та множення двох комплексних чисел, а також виведення комплексного числа на екран.

7. Клас `RatNum` імітує роботу зі звичайними дробами. Полями цього класу повинні бути два поля типу `int`: чисельник і знаменник дроби. Передбачити конструктор класу та методи, що будуть ініціалізувати об'єкти класу, ділити і множити два дроби, а також виводити на екран значення у форматі `a/b`.

8. Клас `Account` моделює роботу з поточним банківським рахунком. Полями цього класу повинні бути прізвище власника рахунку та сума на рахунку. Передбачити конструктор класу та методи для перевірки існування певного рахунку, зняття певної суми з рахунку, за умови наявності необхідної суми, а також виведення інформації на екран.

9. Клас `Vector2D` імітує роботу із векторами у двовимірному просторі. Полями цього класу повинні бути два поля типу `int`, призначені для зберігання координат вектора. Передбачити конструктор класу та методи для виведення

інформації на екран, а також для знаходження довжини вектора та добутку вектора на число.

10. Клас `Complex` імітує роботу з комплексними числами. Полями цього класу повинні бути два поля типу `int`, призначені для зберігання дійсної й уявної частини комплексного числа. Передбачити конструктор та методи класу, що будуть ініціалізувати об'єкти класу, виводити значення на екран, а також порівнювати (`=` чи `?`) два комплексні числа.

11. Клас `Time` моделює роботу з годинником. Полями цього класу повинні бути три поля типу `int`, призначені для зберігання годин, хвилин, секунд. Один з конструкторів має ініціалізувати годинник нульовими значеннями, а інший – заданим набором значень. Передбачити метод класу для виведення часу на екран у форматі `11:59:59`, а також метод додавання значень двох об'єктів типу `Time`, які передаються в якості аргументів.

12. Клас `Float` імітує стандартний тип даних `float`. Єдине поле цього класу повинне мати тип `float`. Передбачити конструктор класу та методи, що будуть ініціалізувати об'єкти класу, виводити значення на екран, а також множити і ділити два дійсних числа.

13. Клас `Vector` імітує роботу із векторами у двовимірному просторі. Полями цього класу повинні бути чотири поля типу `int`, призначені для зберігання координат точок початку і кінця вектора. Передбачити конструктор та методи класу, що будуть ініціалізувати об'єкти класу, виводити значення на екран, а також знаходити координати вектора та перевіряти, чи є два вектори колінеарними.

14. Клас `Date` моделює роботу з календарем. Полями цього класу повинні бути три поля типу `int`: рік, місяць і день. Передбачити конструктор класу, методи ініціалізації та виведення дати у форматі `dd.mm.yuuu`, а також інкремента поточної дати на один день. При цьому необхідно врахувати можливий перехід до наступного місяця та року.

15. Клас `Time` моделює роботу з годинником. Полями цього класу повинні бути три поля типу `int`, призначені для зберігання годин, хвилин, секунд. Один з

конструкторів має ініціалізувати годинник нульовими значеннями, а інший – заданим набором значень. Передбачити методи класу для виведення часу на екран у форматі 11:59:59, а також інкремента секунд поточного часу. При цьому необхідно врахувати можливий перехід до наступної хвилини.

16. Клас `Complex` імітує роботу з комплексними числами. Полями цього класу повинні бути два поля типу `int`, призначені для зберігання дійсної й уявної частини комплексного числа. Передбачити конструктор за замовчуванням, методи класу для додавання та ділення двох комплексних чисел, а також виведення комплексного числа на екран.

17. Клас `RatNum` імітує роботу зі звичайними дробами. Полями цього класу повинні бути два поля типу `int`: чисельник і знаменник дробу. Передбачити конструктор класу та методи, що будуть ініціалізувати об'єкти класу, додавати і віднімати два дроби, а також виводити на екран значення у форматі `a/b`.

18. Клас `Account` моделює роботу з поточним банківським рахунком. Полями цього класу повинні бути прізвище власника рахунку та сума на рахунку. Передбачити конструктор класу та методи для перевірки існування певного рахунку, збільшення рахунку на задану суму, а також виведення інформації на екран.

19. Клас `Vector3D` імітує роботу із векторами у тривимірному просторі. Полями цього класу повинні бути три поля типу `int`, призначені для зберігання координат вектора. Передбачити конструктор класу та методи для виведення інформації на екран, а також додавання та скалярного добутку двох векторів.

20. Клас `Employee` моделює роботу з даними про співробітників фірми. Клас повинен включати поле типу `int` для зберігання ідентифікаційного номера співробітника й поле типу `float` для зберігання розміру його окладу. Передбачити конструктор класу та методи, призначені для введення й відображення інформації про співробітника. Передбачити запит даних на трьох співробітників.

Контрольні питання

1. Що представляє собою клас?
2. Чим клас відрізняється від структури?
3. Як визначити новий клас?
4. Наведіть загальний формат оголошення класу.
5. Поясніть принцип інкапсуляції даних.
6. Які рівні доступу до членів класу Вам відомі?
7. У чому відмінність між відкритими і закритими членами класу?
8. Як визначити об'єкти класу?
9. Який оператор використовують для доступу до членів класу через об'єкт?
10. Що таке конструктор класу?
11. Назвіть особливості конструктора.
12. Як і коли викликається конструктор класу?
13. Чи може клас мати більше одного конструктора?
14. Що таке деструктор?
15. Назвіть особливості деструктора.

ПРАКТИЧНА РОБОТА № 14-15

Тема: Поліморфізм. Перевантаження функцій, операторів і методів класу

Мета роботи: засвоїти правила створення перевантажених функцій; навчитись перевантажувати унарні та бінарні оператори; навчитись на практиці перевантажувати функції і операції у класах.

Теоретичні відомості

Перевантаження функцій

При визначенні функцій в своїх програмах необхідно вказати тип значення, що повертається функцією, а також кількість параметрів і тип кожного з них. Припустимо, є функція з ім'ям *add_values*, яка працює з двома цілими значеннями, а нам треба використовувати подібну функцію для додавання трьох цілих значень. Тоді слід створити функцію з іншим ім'ям, наприклад, *add_two_values* і *add_three_values*. Аналогічно якщо ви хотіли використовувати подібну функцію для складання значень типу *float*, то була б необхідна ще одна функція з ще одним ім'ям.

Для уникнення дублювання функції C++ дозволяє вам визначати декілька функцій з одним і тим же ім'ям. У процесі компіляції C++ бере до уваги кількість аргументів, що використовуються кожною функцією, і потім викликає саме потрібну опцію. *Надання компілятору вибору серед декількох функцій називається перевантаженням функцій.*

Обмеження на перевантажені функції:

1. Будь-які перевантажені функції повинні мати різні списки параметрів (при цьому аргумент даного типу і посилання на цей тип розглядаються як одне те саме).
2. Не допускається перевантаження функцій із співпадаючими списками параметрів лише на основі типу повертаємих значень.

3. Функції-члени не можуть бути перевантажені лише на основі того, що одна з них – статична, а друга – ні.

Typedef - визначення не впливають на механізм перевантаження, оскільки вони не вводять нові типи, а є лише синонімами існуючих. Наприклад, якщо є визначення `typedef char *ptr;` то дві функції `void setval (char *s)` та `void setval (ptr s)` не є різними. Це буде помилкою.

4. Всі `enum`-типи розглядаються як різні і тому можуть використовуватись для перевантаження функцій.

5. Типи “масив (чогось)” і “показчик (на щось)” розглядаються як ідентичні з точки зору перевантаження. Наприклад,

```
void setval(char pz);
```

```
void setval (char *ptr);
```

є помилкою при перевантаженні.

Але це стосується тільки одномірних масивів. Для багатомірних масивів друга, третя,..... розмірності розглядаються як частина типу даних.

Наприклад, не буде помилки:

```
void setval (char sz [ ]);
```

```
void setval (char sz [ ][4]);
```

Приклад 1. Наступна програма перевантажує функцію з ім'ям *add_values*. Перше визначення функції складає два значення типу `int`. Друге визначення функції складає три значення.

```
#include <iostream.h>  
int add_values(int a,int b){  
    return(a + b);  
}  
int add_values (int a, int b, int c){  
    return(a + b + c);  
}  
void main(void){  
    cout<<"200 + 801 = "<<add_values(200,801)<<endl;
```

```
cout<<"10 + 21 + 70 = "<<add_values(10,21,70)<<endl;
}
```

Приклад 2. Наступна програма перевантажує функцію *show_message*. Перша функція виводить стандартне повідомлення, параметри їй не передаються. Друга виводить передане їй повідомлення, а третя виводить два повідомлення.

```
#include <iostream.h>

void show_message(void){ cout << "Стандартное сообщение: "
    << "Учимся программировать на C++" << endl; }

void show_message(char *message){
    cout << message << endl; }

void show_message(char *first, char *second){
    cout << first << endl;
    cout << second << endl; }

void main(void){
    show_message();
    show_message("Учимся программировать на языке C++!");
    show_message("В C++ нет предрассудков!", "Перегрузка-это круто!"); }
```

Перевантаження операторів

Перевантаження оператора полягає у зміні сенсу оператора при використанні його з певним класом.

Перевантаження операторів може спростити найбільш загальні операції класу і поліпшити читабельність програми

Для перевантаження операторів програми використовують ключове слово ***operator***.

Якщо програма перевантажує оператор для певного класу, то сенс цього оператора змінюється тільки для зазначеного класу, решта програми буде використовувати цей оператор для виконання його стандартних операцій.

У загальному випадку програми можуть перевантажити майже всі оператори C ++, окрім таких:

- оператор вибору члена (оператор “.”);
- оператор вибору члена за покажчиком (оператор “.*”);
- оператор розширення області бачення (оператор “::”);
- оператор умови (“? :”);
- препроцесорний оператор (“#”) та препроцесорний символ (“##”).

Приклад 3. Нижче наведено визначення класу *strings*. Цей клас містить один елемент даних, який являє собою власне символьний рядок. Крім того, цей клас містить декілька перевизначених операторів:

```
#include <iostream>
#include <string.h>
#include <locale.h>
using namespace std;
class strings
{
public:
    strings(char *); // Конструктор
    void operator +(char *);
    void operator -(char);
    void show_string(void);
private:
    char data[256] ;
};
strings::strings(char *str){
    strcpy(data, str);
}
void strings::operator +(char *str){
    strcat(data, str);
}
```

```

void strings::operator -(char letter){
    char temp[256] ;
    int i, j;
    for (i = 0, j = 0; data[i]; i++)
        if (data[i]==letter) temp[j++] = data[i];
    temp[j] = NULL;
    strcpy(data, temp);
}

void strings::show_string(void){
    cout << "\n\t"<<data << endl;
}

void main(void){
    setlocale(0,"");
    strings title( "\n\tУчимся программировать на C++");
    strings lesson("\n\tПерезгрузка операторов");
    title.show_string();
    title + " и я учусь тоже!";
    title.show_string() ;
    lesson.show_string();
    lesson – 'p';
    lesson.show_string();
    system("pause");
}

```

Завдання для виконання

Завдання до практичної роботи №14. Визначившись з власними силами, вибрати собі для виконання одну із задач:

(низька складність – оцінка "задовільно", "добре")	
1	Визначте перевантажені функції <i>square</i> для знаходження квадратів цілих чисел, чисел з плаваючою комою одинарної та подвійної точності.
2	Визначте перевантажені функції <i>triple</i> для потроювання значень цілих чисел, чисел з плаваючою комою одинарної та подвійної точності.
3	Визначте перевантажені функції <i>minimum</i> для знаходження найменшого з трьох цілих чисел, чисел з плаваючою комою одинарної та подвійної точності.
4	Визначте перевантажені функції <i>maximum</i> для знаходження найбільшого з трьох цілих чисел, чисел з плаваючою комою одинарної та подвійної точності.
(середня складність – оцінка "відмінно")	
1	Визначте перевантажені функції <i>power</i> для піднесення до цілого невід'ємного степеня цілих чисел, чисел з плаваючою комою одинарної та подвійної точності.
2	Визначте перевантажені функції <i>round_value</i> для округлення чисел з плаваючою комою подвійної точності до заданої точності. Точність може бути задана у вигляді цілого числа, що вимагає необхідне число знаків після коми, та у вигляді числа з плаваючою комою у форматі 0,0...01 (наприклад, 0,01 означає округлення до сотих).

Завдання до практичної роботи №15. Описати клас, що реалізує вказаний тип даних згідно індивідуального завдання.

Клас повинен містити:

- множину конструкторів для створення об'єктів певного типу (конструктор за замочуванням, конструктор з параметрами, конструктор копіювання);
- вказані операції над об'єктами класу з використанням механізму перевантаження операцій.

Написати програму, яка демонструє роботу з об'єктами створеного класу.

№	Предметна область	Операції для перевантаження
1	Матриця	Віднімання, множення на матрицю, додавання цілого числа
2	Комплексні числа	Сума, добуток на комплексне число, добуток на дійсне число інкремент
3	Вектор у просторі	Додавання векторів, векторний добуток двох векторів
4	Множина	Вилучення елемента, об'єднання множин, перетин множин
5	Точка на площині	Додавання двох точок, множення на число, дзеркальне відображення
6	Дроби	Віднімання, множення, порівняння
7	Лінія	Порівняння, зменшення, пересування по горизонталі
8	Цілі числа	Інкремент, декремент, додавання, віднімання, логічні операції
9	Вектор на площині	Скалярний добуток, порівняння векторів, множення вектора на число
10	Матриця	Додавання, множення на число, транспонування
11	Прямокутник	Збільшення, пересування по вертикалі, перевірка на попадання точки в прямокутник
12	Дроби	Додавання, інкремент, пошук оберненого дроби

13	Дійсне число	Модуль числа, сума, добуток на ціле, добуток на дійсне
14	Комплексні числа	Різниця, порівняння, ділення на дійсне число, декремент
15	Комплексні числа	Обчислення оберненого комплексного числа, додавання до комплексного числа дійсного і навпаки

Контрольні питання

1. Для чого використовують перевантаження функцій?
2. Які обмеження на перевантажені функції?
3. Як компілятор діє з перевантаженими функціями?
4. Наведіть приклад перевантаження функцій.
5. В чому сенс перевантаження операторів?
6. Наведіть приклади перевантаження унарних та бінарних операторів.
7. Які оператори не можна перевантажувати?

ПЕРЕЛІК РЕКОМЕНДОВАНОЇ ТА ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Библиотека MSDN [Электронный ресурс]. – Режим доступа: URL <http://msdn.microsoft.com/ru-ru/library/default.aspx> – Назва з екрана.
2. NET Framework [Электронный ресурс]. – Режим доступа: URL [https://msdn.microsoft.com/ru-ru/library/w0x726c2\(v=vs.110\).aspx](https://msdn.microsoft.com/ru-ru/library/w0x726c2(v=vs.110).aspx) – Назва з екрана.
3. Visual Studio 2015 [Электронный ресурс]. – Режим доступа: URL [https://msdn.microsoft.com/library/dd831853\(v=vs.140\).aspx](https://msdn.microsoft.com/library/dd831853(v=vs.140).aspx) – Назва з екрана.
4. Б. Страуструп, Дизайн и эволюция C++, М.: ДМК Пресс; Спб.: Питер, 2006. – 448 с.
5. B. Stroustrup, The C++ Programming Language 4th Edition – 2013, 1281 p.
6. С. Майерс, Наиболее эффективное использование C++, Москва, ДМК Пресс, 2014, 298 с.
7. Г. Саттер, А. Александреску, Стандарты программирования на C++, Москва, Вильямс, 2015, 224 с.
8. Глинський Я.М. C++ і C++ Builder / Я.М. Глинський, В.Є. Анохін, В.А. Ряжська. – Львів : Деол, СПД Глинський, 2003. – 192 с.
9. Шилдт, Герберт. Искусство программирования на C++ : пер. с англ. / Герберт Шилдт. – СПб. : Изд-во БХВ-Петербург, 2005. – 496 с. 27.
10. Шилдт, Герберт. Полный справочник по C++ / Герберт Шилдт. – 4-е изд. : пер. с англ. – М. : Изд. дом "Вильямс", 2010. – 800 с. 28.
11. Шилдт, Герберт. C++: Базовый курс / Герберт Шилдт. – 3-е изд. : пер. с англ. – М. : Изд. дом "Вильямс", 2005. – 624 с. 29.
12. Шилдт, Герберт. Самоучитель C++ / Герберт Шилдт. – 3-е изд. : пер. с англ. – СПб. : Изд-во БХВ-Петербург, 2005. – 688 с. 30. Штерн В. Основы C++. Методы программной инженерии / В. Штерн. – М. : Изд-во "Лори", 2003. – 860 с.

13. Ковалюк Т. В. Алгоритмізація та програмування: підручник з грифом МОН України / Т. В. Ковалюк. – Львів : Магнолія-2006, 2013. – 400 с.
14. Кравець П. Об'єктно-орієнтоване програмування : навч. посібник / П.О. Кравець. – Львів: Видавництво Львівської політехніки, 2012. – 624 с.
15. Лавріщева К. М. Програмна інженерія / К. М. Лавріщева. – К. : Академперіодика, 2008. – 319 с.
16. С++. Теорія та практика : навч. посібник / О. Г. Трофименко, Ю. В. Прокоп, І. Г. Швайко, Л. М. Буката та ін. ; за ред. О. Г. Трофименко. – 587 с.

ДОДАТКИ

Додаток №1. Основні заголовочні файли C++

Заголовочний файл	Короткий опис
<i>ctype.h</i>	Перетворення та обробка символів
<i>math.h</i>	Математична бібліотека
<i>stdio.h</i>	Стандартна бібліотека вводу-виводу
<i>stdlib.h</i>	Функції форматного перетворення даних (рядка в число і навпаки)
<i>string.h</i>	Функції для роботи із рядковими змінними
<i>windows.h</i>	Файл для підтримки роботи Windows-додатків
<i>iostream</i>	Стандартна бібліотека потокового вводу-виводу
<i>fstream</i>	Стандартна бібліотека потокового вводу-виводу при роботі з файлами
<i>conio.h</i>	Функції для організації консольного вводу-виводу при роботі із текстовими даними

Додаток №2. Базові типи даних

Тип даних	Назва	Діапазон значень
char	Цілий довжиною не менше 8 біт	-128 .. 127
unsigned int	Без знаковий цілий	0 .. 65535
short int (short)	Короткий цілий	-32768 .. 32767
unsigned short	Беззнаковий короткий цілий	0 .. 65535
int	Цілий	-32768 .. 32767
unsigned long	Беззнаковий довгий цілий	0 .. 4294967295
long int (long)	Довгий цілий	-2147483648 .. 2147483647
float	Дійсний одинарної точності	3.4E-38 .. 3.4E+38
double	Дійсний подвійної точності	1.7E-308 .. 1.7E+308
long double	Дійсний максимальної точності	3.4E-4932 .. 1.1E+4932

Додаток №3. Основні специфікатори формату функції *printf*

Специфікатор	Формат
%c	Символ
%d	Десяткове ціле число зі знаком
%i	Десяткове ціле число зі знаком
%e	Науковий формат (рядкова буква e)
%E	Науковий формат (прописна буква E)
%f	Десяткове число із плаваючою крапкою
%g	Дійсна величин як f або E в залежності від значення
%s	Рядок символів
%u	Десяткове ціле число без знаку
%%	Знак %

Можна дещо розширити визначення специфікації формату: після % можна задати число позицій, що виділяються для даного формату.

Наприклад:

%10.5f (для десяткового числа із плаваючою комою)

%10i (для десяткового цілого числа зі знаком).|

Додаток №4. Основні операції мови C++

Операція	Короткий опис	Приклад застосування
Бінарні арифметичні операції		
+	Плюс (додавання арифметичних операндів)	$x+y$
-	Мінус (віднімання арифметичних операндів)	$x-y$
*	Множення	$x*y$
/	Ділення (якщо операнди цілочисельні, абсолютне значення результату заокруглюється до цілого, тобто $20/3$ дорівнює 6)	x/y
%	Залишок від ділення цілочисельних операндів (Наприклад $17\%3=1$)	$x\%y$
Унарні арифметичні операції		
+	Унарний плюс (підтвердження знаку)	$+x$
-	Унарний мінус (зміна знаку)	$-x$
++	Інкремент (збільшення на одиницю): префіксна операція ($++x$) збільшує операнд на 1 до його використання; постфіксна операція ($x++$) збільшує операнд на 1 після його використання.	$m++$ $++n$
--	Декремент (зменшення на одиницю): префіксна операція ($--x$) зменшує операнд на 1 до його використання; постфіксна операція ($x--$) зменшує операнд на 1 після його використання.	$--x$ $y--$

Операції присвоєння

Операція	Короткий опис
$x=y$	Змінній x присвоюється значення змінної y
$x+=y$	Означає $x=x+y$
$x-=y$	Означає $x=x-y$
$x*=y$	Означає $x=x*y$
$x/=y$	Означає $x=x/y$
$x\%=y$	Означає $x=x\%y$

Додаток №5. Основні математичні функції бібліотеки *math.h*

№ п/п	Функція бібліотеки	Опис
1.	<i>abs(x)</i>	Обчислення модуля числа (тільки для цілих чисел)
2.	<i>acos(x)</i>	Обчислення арккосинуса <i>x</i> (в радіанах)
3.	<i>asin(x)</i>	Обчислення арксинус <i>x</i> (в радіанах)
4.	<i>atan(x)</i>	Обчислення арктангенса <i>x</i> (в радіанах)
5.	<i>atan2(x/y)</i>	Обчислення арктангенса відношення <i>x</i> та <i>y</i> (в радіанах)
6.	<i>ceil(x)</i>	Повертає дійсне значення, що відповідає найменшому цілому числу, більшому або рівному <i>x</i>
7.	<i>cos(x)</i>	Обчислення косинуса <i>x</i> , заданого в радіанах
8.	<i>cosh(x)</i>	Обчислення гіперболічного косинуса <i>x</i> , заданого в радіанах
9.	<i>exp(x)</i>	Повертає результат піднесення числа <i>e</i> до степені <i>x</i>
10.	<i>fabs(x)</i>	Обчислення абсолютного значення дійсного числа <i>x</i>
11.	<i>floor(x)</i>	Повертає дійсне значення, що відповідає найбільшому цілому числу, меншому або рівному <i>x</i>
12.	<i>fmod(x,y)</i>	Остача від цілочисленого ділення <i>x</i> на <i>y</i> дійсного типу
13.	<i>log(x)</i>	Обчислення натурального логарифма <i>x</i>
14.	<i>log10(x)</i>	Обчислення десяткового логарифма <i>x</i>
15.	<i>pow(x,y)</i>	Піднесення <i>x</i> до степені <i>y</i>
16.	<i>sin(x)</i>	Обчислення синуса <i>x</i> , заданого в радіанах
17.	<i>sinh(x)</i>	Обчислення синуса гіперболічного <i>x</i> , заданого в радіанах
18.	<i>sqrt(x)</i>	Обчислення квадратного кореня <i>x</i>
19.	<i>tan(x)</i>	Обчислення тангенса <i>x</i> , заданого в радіанах
20.	<i>tanh(x)</i>	Обчислення тангенса гіперболічного <i>x</i> , заданого в радіанах

Додаток №6.

Операції порівняння

Операція	Значення
<	Менше
<=	Менше або рівно
==	Рівно
>=	Більше або рівно
>	Більше
!=	Не рівно

Логічні операції

Операція	Значення
&&	Логічне І (AND)
	Логічне АБО (OR)
!	Логічне заперечення (NOT)