Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Інформатика. Частина 2.

Програмування та алгоритмічні мови

Практикум

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів ступеня бакалавра
за освітніми програмами «Електронні компоненти і системи»,
«Електронні прилади та пристрої»
спеціальності G5 Електроніка, електронні комунікації, приладобудування та радіотехніка

Укладачі: К.С. Клен, О.М. Коваленко, О.О. Абакумова

Електронне мережеве навчальне видання

Київ
КПІ ім. Ігоря Сікорського

2026

УДК 621.314
К38

Укладачі: *Клен Катерина Сергіївна*, д-р техн. наук, доц.
Коваленко Олександр Миколайович
Абакумова Олена Олегівна, канд. філос. наук

Рецензент *Найда Сергій Анатолійович*, д-р техн. наук, проф.
КПІ ім. Ігоря Сікорського

Відповідальний редактор *Вербицький Євген Володимирович*, д-р техн. наук, проф.

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського
(протокол № 8 від 04.06.2026 р.)
за поданням вченої ради факультету електроніки
(протокол № 05/2026 від 27.05.2026 р.)

К38

Інформатика. Частина 2. Програмування та алгоритмічні мови [Електронний ресурс] : практикум : навч. посіб. для здобувачів ступеня бакалавра за освіт. програмами «Електронні компоненти і системи», «Електронні прилади та пристрої» спец. G5 Електроніка, електронні комунікації, приладобудування та радіотехніка / КПІ ім. Ігоря Сікорського ; уклад.: К. С. Клен, О. М. Коваленко, О.О. Абакумова. – Електрон. текст. дані (1 файл). – Київ : КПІ ім. Ігоря Сікорського, 2026. – 58 с.

У навчальному посібнику викладено інформацію щодо основ програмування мовою С++.

Практикум містить теоретичні відомості та завдання до 6 комп'ютерних практикумів з контрольними запитаннями та список використаних джерел.

Навчальний посібник призначений для здобувачів ступеня бакалавра за освітніми програмами «Електронні компоненти і системи», «Електронні прилади та пристрої» спеціальності G5 Електроніка, електронні комунікації, приладобудування та радіотехніка.

УДК 621.314

Реєстр. № НП 25/26-441. Обсяг 2,64 авт. арк.
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
проспект Берестейський, 37, м. Київ, 03056
https://kpi.ua
Свідоцтво про внесення до Державного реєстру видавців, виготовлювачів
і розповсюджувачів видавничої продукції ДК № 5354 від 25.05.2017 р.

ЗМІСТ

Вступ

Навчальна дисципліна «Інформатика. Частина 2. Програмування та алгоритмічні мови» належить до циклу професійної та практичної підготовки бакалаврів за спеціальністю G5 Електроніка, електронні комунікації, приладобудування та радіотехніка, освітніми програмами «Електронні компоненти і системи», «Електронні прилади та пристрої». Дисципліна читається протягом одного семестру (2) і є базовою для подальшого навчання.

В процесі вивчення курсу студенти вивчають основи програмування мовою C++, яка призначена для розробки високопродуктивного програмного забезпечення та дуже популярна серед програмістів. Вивчивши C++, студенти отримують фундаментальні знання, які дозволять їм у подальшому опанувати будь-які аспекти сучасного програмування, навички проєктування та відлагодження пристроїв на мікроконтролерах та інших приладів та систем сучасної електроніки.

Метою навчальної дисципліни є передача студентам знання основ розробки прикладного програмного забезпечення сучасними програмними засобами мови C++; формування у студентів досвіду програмування з використанням сучасних можливостей мови C++ у відповідності до професійних потреб, розробка прикладного програмного забезпечення для розв’язку інженерних завдань та для мікропроцесорних систем електроніки.

Лабораторна робота №1. Управляючі структури

Мета: набуття навичок створення С++-програм із використанням управляючих структур розгалуження (вибору) та повторення (циклу). Набуття навичок роботи з потоками вводу/виводу, та методами класу iostream.

Завдання: Обчислити і вивести на екран у табличному вигляді значення функції *f(x)* на заданому інтервалі зміни значень аргумента *x* від $x_{поч}$ до $x_{кін}$ з кроком *h*. Коефіцієнти *a, b, c* – дійсні числа.

Значення *a, b, c,* $x_{поч}$, $x_{кін}$ та *h* вводити з клавіатури. Передбачити перевірку допустимості значень, введених користувачем, та ОДЗ функції.

1. $$f(x)=\begin{cases} -\dfrac{2x+c}{cx+2a}, & x<0 \ i \ b+c>5 \\ \dfrac{x-a}{\sqrt{x}+c}, & x>0 \ i \ b-c<0 \\ \dfrac{3x}{c}+\dfrac{-b}{2x(x^2+1)}, & \text{в інших випадках} \end{cases}$$

2. $$f(x)=\begin{cases} \dfrac{\sqrt{x+bx^2}}{2x}+\dfrac{a}{3x}, & x-b>0 \ i \ c<0 \\ a(x^2b+c)+x, & x-b<0 \ i \ c\geq 0 \\ -\dfrac{\sqrt{x+bc}}{x(\sqrt{x^2+1})+b}, & \text{в інших випадках} \end{cases}$$

3. $$f(x)=\begin{cases} \dfrac{1}{ax}-b, & x+10<0 \ i \ c>2 \\ \dfrac{x-a}{2b+x}, & x+10>0 \ i \ b-c<0 \\ \dfrac{5x}{c+b}, & \text{в інших випадках} \end{cases}$$

4. $$f(x)=\begin{cases}\dfrac{ax^2+bx}{c},\ x-c<0\ i\ b\geq 0\\ \dfrac{\sqrt{x}}{2x-a}+\dfrac{a}{3x+c},\ x-c>0\\ -\dfrac{x\sqrt{x+bc}}{c+b},\ \textit{в інших випадках}\end{cases}$$

5. $$f(x)=\begin{cases}\dfrac{x^2+x}{3x-c},\ x>2\ i\ b-c<0\\ ax^3-bx+c,\ x<2\ i\ b+c\geq 0\\ \dfrac{x}{15c}+\sqrt{ax+b},\ \textit{в інших випадках}\end{cases}$$

6. $$f(x)=\begin{cases}\dfrac{\sqrt{x}-3a}{x^2+c},\ x>0\ i\ c<0\\ -ax-c,\ x<0\ i\ c>0\\ \dfrac{x}{c+a}+\dfrac{b}{x},\ \textit{в інших випадках}\end{cases}$$

7. $$f(x)=\begin{cases}a+\dfrac{2x}{cx+5a},\ x<0\ i\ b+c>5\\ \dfrac{2x+a}{3x-b},\ x>0\ i\ b-c<0\\ \dfrac{3}{x}+\dfrac{-c}{2x^2},\ \textit{в інших випадках}\end{cases}$$

8. $$f(x)=\begin{cases}x+\dfrac{a}{x-c},\ x>0\ i\ b<0\\ ax^2-bx,\ x<0\ i\ b\geq 0\\ \dfrac{x}{5c}-x^2,\ \textit{в інщих випадках}\end{cases}$$

9. $$f(x)=\begin{cases}3x+\dfrac{ax}{x^2+2c},\ x>0\ i\ b<0\\ -ax^2+b,\ x<0\ i\ b\geq 0\\ \dfrac{2x}{c-a}-bx^2,\ \textit{в інших випадках}\end{cases}$$

10. $f(x)=\begin{cases} -ax^3+bcx,\ x<0\ i\ b\geq 0 \\ \sqrt{x^3}+\dfrac{ax}{xc+b},\ x>0\ i\ b<0 \\ x\cdot(x^2-c)+b^2,\ в\ \ інших\ випадках \end{cases}$

11. $f(x)=\begin{cases} a\sqrt{x+c}-bx^2,\ x<4\ i\ b+c\geq 10 \\ x+\dfrac{a}{x-c},\ x>4\ i\ b<10 \\ x(x-b)-x^2c,\ в\ інших\ випадках \end{cases}$

12. $f(x)=\begin{cases} x^2+\dfrac{ax}{x^3-c},\ x\geq 3\ i\ ac<0 \\ \sqrt{x^2-b}+cx,\ x<0\ i\ ac\geq 0 \\ \dfrac{\sqrt{x}}{1+x}-(c-x)^2,\ в\ інших\ випадках \end{cases}$

13. $f(x)=\begin{cases} x^3+\dfrac{ax}{x^2-ca},\ x\geq 0\ i\ ac<0 \\ x\sqrt{x+b}+cx^2,\ x<0\ i\ b+x\geq 0 \\ \dfrac{\sqrt{x}}{1+x}-\dfrac{2}{3x},\ в\ інших\ випадках \end{cases}$

14. $f(x)=\begin{cases} \dfrac{x^2-bx}{2a-x}+\dfrac{a}{x^3},\ x\geq 3\ i\ ac<0 \\ \sqrt{x^3+b}-2x^2,\ x<0\ i\ c\geq 0 \\ \dfrac{x(x+3)}{4+x}-(c-2x)^2,\ в\ інших\ випадках \end{cases}$

15. $f(x)=\begin{cases} \dfrac{2x-c}{cx+a},\ x<0\ i\ b+c\geq 5 \\ \dfrac{ax-b\sqrt{x}}{x^2-ac},\ x\geq 10\ i\ b+c<5 \\ \dfrac{\sqrt{x+2}}{1+cx}-(ax)^2,\ в\ інших\ випадках \end{cases}$

16. $f(x)=\begin{cases}\dfrac{5x}{c}+\dfrac{-b}{4x(x^3+1)}, & x<0 \text{ і } b+c>5\\ \dfrac{x-a}{\sqrt{x}+c} & x>0 \text{ і } b-c<0\\ -\dfrac{2x+c}{4cx+3a}, & \text{в інших випадках}\end{cases}$

17. $f(x)=\begin{cases}\dfrac{bx+x^2}{2a-x}, & x<2 \text{ і } ac<0\\ \sqrt{b+x^3}+2x^2, & x>2 \text{ і } c\geq 0\\ \dfrac{x^2(x+1)}{x+4}+(c-2)^2, & \text{в інших випадках}\end{cases}$

18. $f(x)=\begin{cases}2ax+bx^3, & x<0 \text{ і } b<0\\ \sqrt[2]{x^3}+\dfrac{2x^2}{cx+a}, & x>0 \text{ і } c\geq 0\\ ax^2(x+c)+(x-ab)^2, & \text{â ³í ø èõ âèï àäêàõ}\end{cases}$

19. $f(x)=\begin{cases}\dfrac{b+1}{ax}-\dfrac{1}{2}x, & x+10>0 \text{ і } c>3\\ \dfrac{x-2a}{b^2}+x, & x+10<\text{ і } b-c<0\\ \dfrac{\sqrt{x^2+1}}{15abc}, & \text{в інших випадках}\end{cases}$

20. $f(x)=\begin{cases}\dfrac{-ax-c}{x^2+c}-\dfrac{bx}{2}, & x<0 \text{ і } c>0\\ a+\dfrac{x-2a}{b^2+x}, & x>0 \text{ і } bc<0\\ \dfrac{3x^3}{ab}-\dfrac{2x}{cx+5a}, & \text{в інших випадках}\end{cases}$

21. $f(x)=\begin{cases}\dfrac{b+1}{ax}-\dfrac{1+c}{3}x,\ x+2>0\ i\ c>0\\ \dfrac{x-2a}{b^2}+x,\ x+2<0\ i\ b-c<0\\ \dfrac{\sqrt{x^2+1}}{3abc},\ \textit{в інших випадках}\end{cases}$

22. $f(x)=\begin{cases}\dfrac{ax^2+bx}{c},\ x-a<0\ i\ b\geq 0\\ \dfrac{\sqrt{x}}{x-a}+\dfrac{a+c}{3x},\ x-a>0\\ \dfrac{x\sqrt{x+abc}}{a+b+c},\ \textit{в інших випадках}\end{cases}$

23. $f(x)=\begin{cases}ax^3+bx+c,\ x<0\ i\ a\geq 0\\ \sqrt{x}+\dfrac{ax-c}{x+b},\ x>0\ i\ a<0\\ x\cdot(ax^2-c)+b^2,\ \textit{в інших випадках}\end{cases}$

24. $f(x)=\begin{cases}\dfrac{2x-bc}{x+a},\ x<0\ i\ b+c\geq 5\\ \dfrac{ax-b\sqrt{x+c}}{x^2-c},\ x\geq 10\ i\ b+c<5\\ \dfrac{\sqrt{ax+2b}}{1+x}-(ax-c)^2,\ \textit{в інших випадках}\end{cases}$

25. $f(x)=\begin{cases}\dfrac{\sqrt{x+b}}{2x^2}+\dfrac{b}{2x},\ x-a>0\ i\ c<0\\ b(x^2a+c),\ x-a<0\ i\ c\geq 0\\ -\dfrac{\sqrt{x+bc}}{x(\sqrt{x^2+1})+b},\ \textit{в інших випадках}\end{cases}$

Теоретичні відомості

Управляючі структури. Структури вибору

У С++ -програмах може бути передбачений перехід з однієї частини програми

до іншої в залежності від виконання або невиконання деякої умови. Оператори, що реалізують подібні переходи, називають *умовними операторами*. Умовні оператори поділяються на дві основні категорії: повторення (цикли) та розгалуження (вибору) [1 - 4].

В С++ існує декілька типів розгалужень, найбільш важливим з яких є розгалуження if ... else, що здійснює вибір між двома альтернативами. Для вибору однієї з багатьох альтернатив використовують розгалуження switch … case, дія якого визначається набором значень керуючої змінної.

Умовний оператор if

Оператор if є найбільш простим з операторів розгалуження. Його синтаксис такий [5 - 9]:

```
if ( умова )
  {
   інструкція;
   ……                    // Тіло розгалуження з кількох операторів
   інструкція;
  }
```

За ключовим словом if слідує *умова* розгалуження, поміщена в круглі дужки. Умова – це вираз, від істинності чи хибності якого залежить виконання або невиконання певної частини програми. Якщо умова істинна, *інструкції* виконуються, в іншому випадку – ні.

Тіло розгалуження може складатися як з одного оператора, так і з кількох операторів (інструкцій), укладених у фігурні дужки.

Оператор if ... else

Оператор if дозволяє здійснювати певну дію лише у тому випадку, коли виконується деяка умова [1 - 4]. Якщо ж умова не виконується, то й ніякої дії не виконується. Коли необхідно зробити одну дію у разі виконання умови, а іншу дію у разі невиконання цієї умови, то використовують розгалуження if ... else. Воно

складається з оператора if, за яким слідує блок інструкцій, і ключового слова else, за яким слідує ще один блок інструкцій.

Синтаксис розгалуження if ... else такий:

```
if ( умова )
  {
   інструкція;
   ……                // Тіло if з кількох операторів
   інструкція;
  }
else
  {
   інструкція;
   ……                // Тіло else з кількох операторів
   інструкція;
  }
```

Для множинного вибору можна використовувати *вкладені* структури if ... else, розміщуючи одну структуру if ... else всередину іншої.

Умовний оператор ?:

Замість оператора if…else у С++-програмі може використовуватися у*мовний оператор* [5 - 9]. Умовний оператор **?:** дає можливість створювати прості однорядкові умовні вирази, в яких виконується одна з двох дій залежно від значення умови.

Синтаксис умовного оператора такий:

умова ? оператор1 : оператор2;

Якщо *умова* приймає ненульове значення, то виконується *оператор1*, інакше – *оператор2*.

Оператор switch

Для організації структур із множинним вибором у мові С++ передбачено

використання оператору **switch** [1 - 4]. Ця структура складається з ряду міток **case** і необов'язкового блока **default**.

Синтаксис структури switch наступний:

```
switch (вираз умови)
   {
    case константа 1:
        інструкція;
        . . . . .
        інструкція;
        break;              //вихід із switch
    case константа 2:
        інструкція;
        . . . . .
        інструкція;
        break;              //вихід із switch
    case константа 3:
        інструкція;
        . . . . .
        інструкція;
        break;              //вихід із switch
    default:
        інструкція;
        . . . . .           // оператори за замовчуванням
        інструкція;
   }
```

За ключовим словом switch в круглих дужках слідує *вираз умови*. Цим виразом може бути будь-який, допустимий у мові С++ вираз, значення якого повинно бути цілим.

Тіло оператора switch складається з декількох фрагментів, що складаються з ключового слова case і *константного виразу*, з яким порівнюється вираз умови. У якості константного виразу використовуються цілі або символьні константи. Всі константні вирази в операторі switch повинні бути унікальні.

Крім операторів, помічених ключовим словом case, може бути, але обов'язково один, фрагмент, позначений ключовим словом default. Інструкції, відповідні цій мітці, виконуються, якщо вираз умови не відповідає жодній з міток case. Мітка default є аналогом частини else інструкції if-else.

В абсолютній більшості випадків за кожною міткою case повинен слідувати відповідний оператор break. Використання оператора break дозволяє в необхідний момент перервати виконання операторів у тілі switch і передати управління першому після структури switch оператору, тобто виконати вихід зі структури. Якщо немає оператора break, то відбувається виконання всіх наступних виразів до фігурної дужки, що завершує структуру switch. Таким чином, оператор break реалізує схему альтернативного вибору.

Зверніть увагу! В операторі switch не потрібно укладати послідовність операторів у фігурні дужки.

Операції відношень

Вираз умови розгалуження записують за допомогою особливого типу операцій, що їх називають *операціями відношень* [5 - 9]. Операція відношення порівнює між собою два значення. Результатом порівняння є значення *істина* або *брехня*.

В C++ визначено повний набір операторів відношень, які можуть бути використані в умовному виразі:

== Дорівнює

!= Не дорівнює

> Більше

< Менше

>= Більше або дорівнює

<= Менше або дорівнює

Логічні операції

С++ надає також три *логічні операції*, які можна використовувати для формування складних умов шляхом комбінування простих.

Логічними операціями є [1 - 4]:

&& - Логічне І

|| - Логічне АБО

! - Логічне НЕ (логічне заперечення)

Для оцінки дії логічних операцій використовують таблиці істинності.

Таблиця істинності операції &&

Вираз 1	Вираз 2	Вираз 1 && Вираз 2
false	false	false
false	true	false
true	false	false
true	true	true

Таблиця істинності операції ||

Вираз 1	Вираз 2	Вираз 1 || Вираз 2
false	false	false
false	true	true
true	false	true
true	true	true

Дія операції логічного заперечення **(**!) полягає в тім, що вона змінює значення свого операнда на протилежне: якщо операнд мав істинне значення, то після застосування операції ! він стає хибним, і навпаки.

Операції відношень мають вищий пріоритет, ніж логічні операції. А операція логічного І (&&) має більш високий пріоритет, ніж операція логічного АБО (||).

Структури повторення (цикли)

У С++-програмах може бути передбачене послідовне повторюване виконання певної частини програми декілька разів. Оператори, що реалізують подібні повторення, називають *операторами повторення* чи *операторами циклу* [5 - 9].

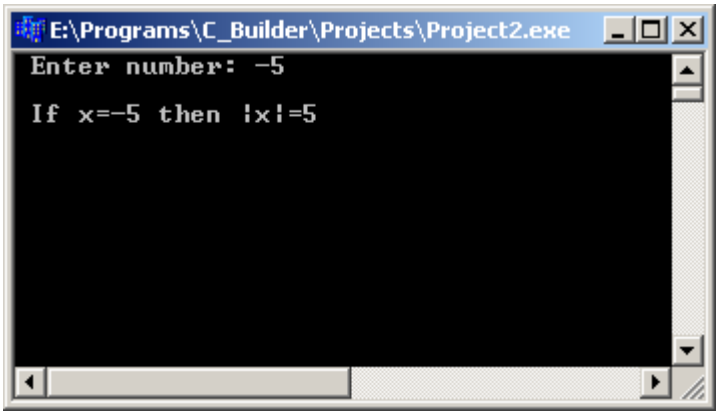
Повторення продовжується до тих пір, поки виконується відповідна умова. Коли значення виразу, що задає умову, стає помилковим, виконання циклу припиняється, і управління передається оператору, що слідує за циклом.

Приклади програми

Приклад 1: Обчислити і вивести на екран значення функції $y = f(x)$:

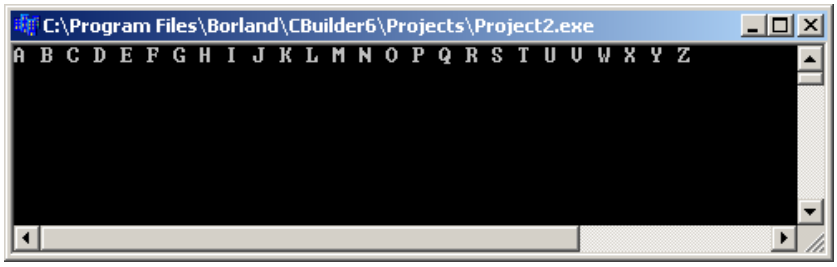
$$y=\begin{cases} x, & x >= 0; \\ -x, & x < 0. \end{cases}$$

```
#include <iostream>
using namespace std;
int main (int argc, char* argv[])
{
float x, y;
cout << " Enter number: ";
cin >> x;
if ( x >= 0 )
  y=x;
else
  y=-x;
cout << "If x=" << x << " then |x|=" << y << endl;
return 0;
}
```



Приклад 2: Вивести на екран літери англійського алфавіту.

```
#include <iostream>
using namespace std;
int main (int argc, char* argv[])
{
char letter;          // лічильник циклу - змінна типу char
for (letter ='A'; letter <='Z'; letter ++) // цикл від A до Z з кроком 1
    cout << letter << " ";          // виведення літери
return 0;
}
```



Контрольні запитання

1. Які оператори називають умовними операторами?
2. Назвіть умовні оператори C++.
3. Назвіть типи розгалужень C + +.
4. Що таке повна та неповна форма розгалуження?
5. Наведіть загальний синтаксис структур вибору C++.
6. Наведіть синтаксис умовного оператора if.

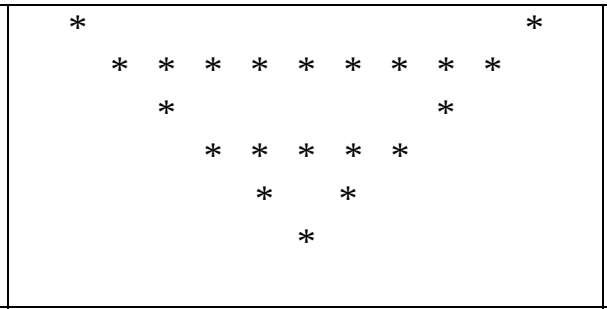
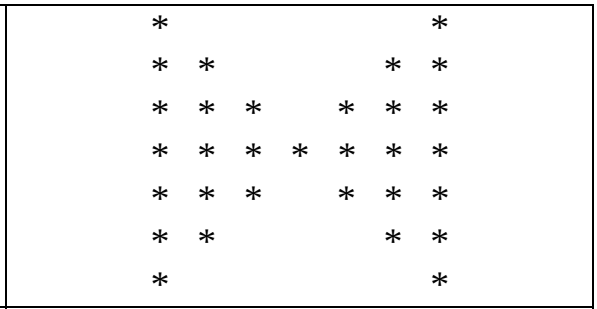
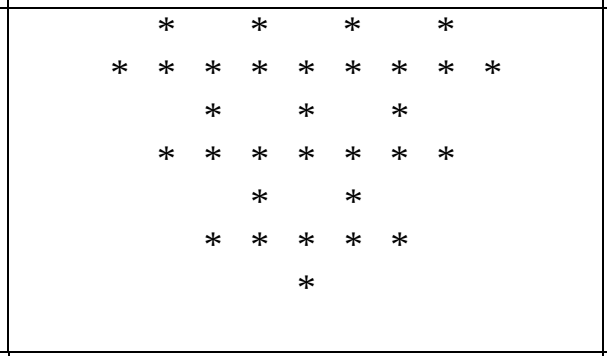
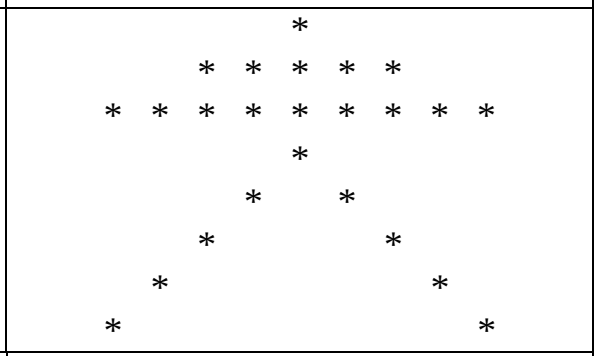
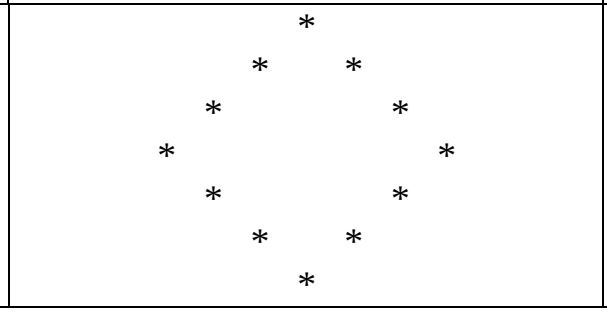
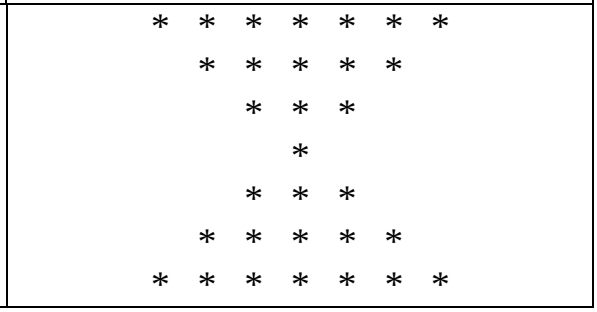
7. Наведіть синтаксис умовного оператора if…else.
8. Наведіть синтаксис умовного оператора ?:.
9. Наведіть синтаксис умовного оператора switch.
10. Які операції називають операціями відношень?
11. Назвіть оператори відношень, визначені у C++.
12. Яке призначення логічних операцій?
13. Назвіть логічні операції C++.
14. Які оператори називають операторами повторення?

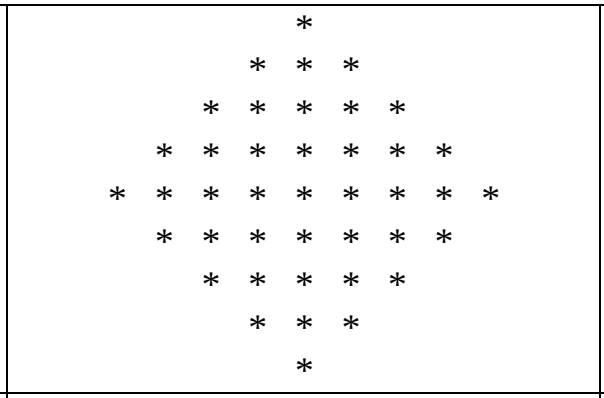
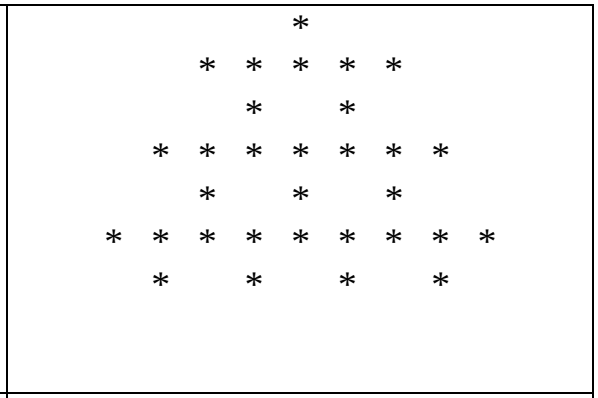
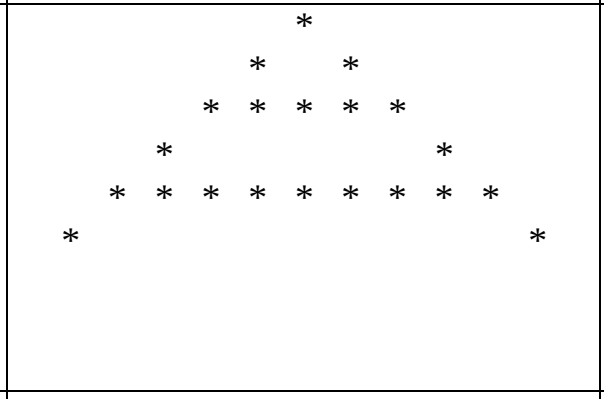
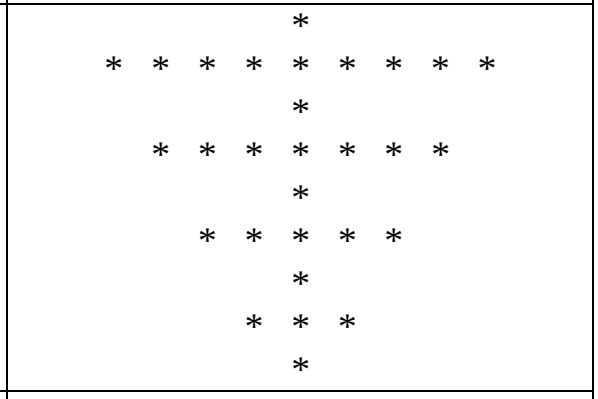
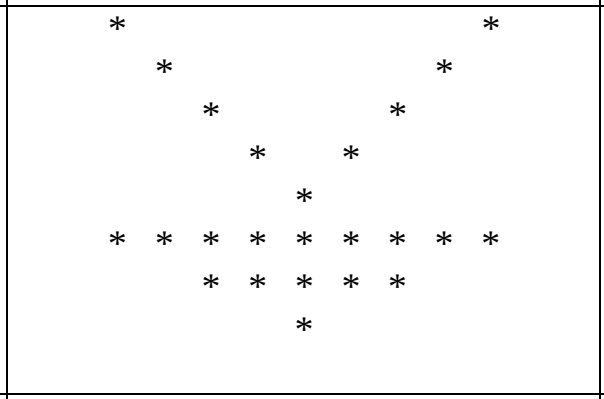
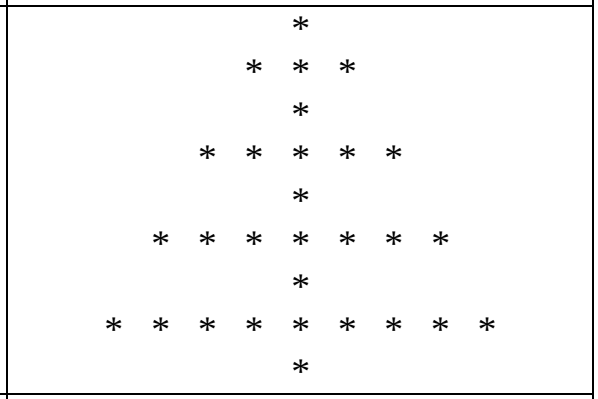
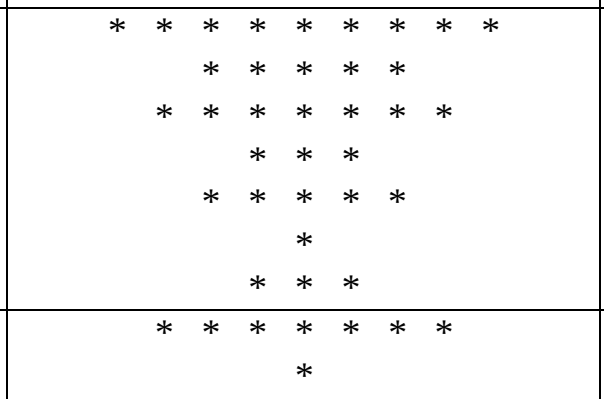
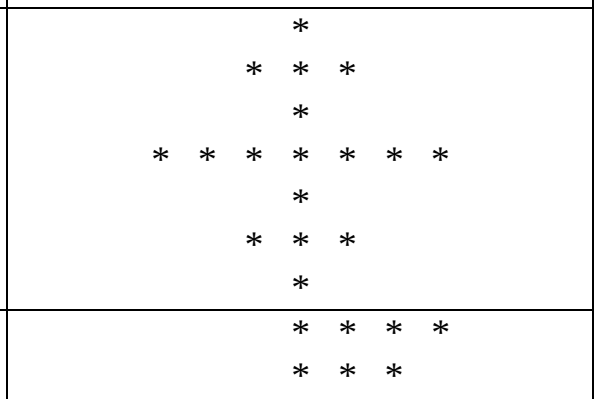
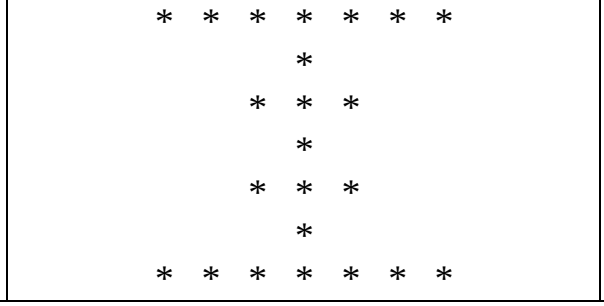
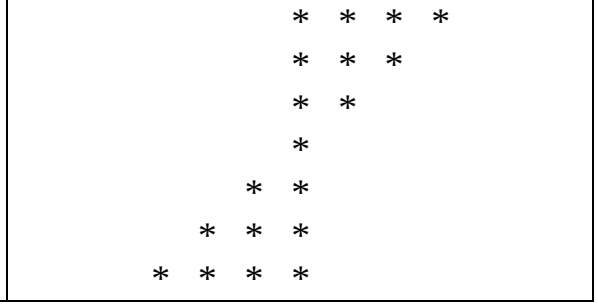
Лабораторна робота №2. Вкладені цикли

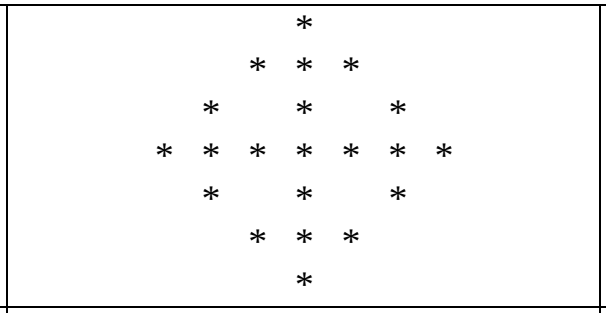
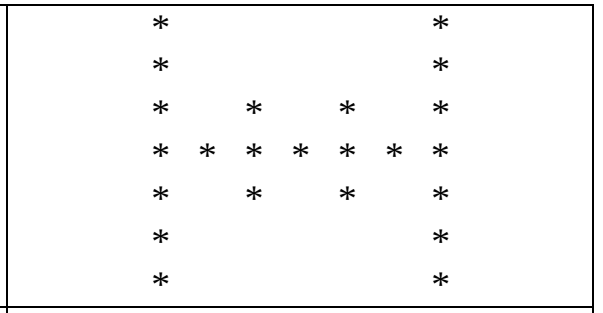
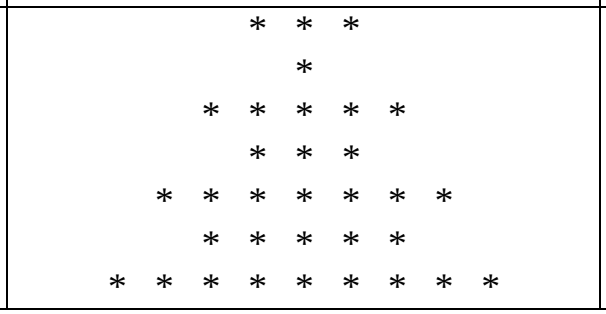
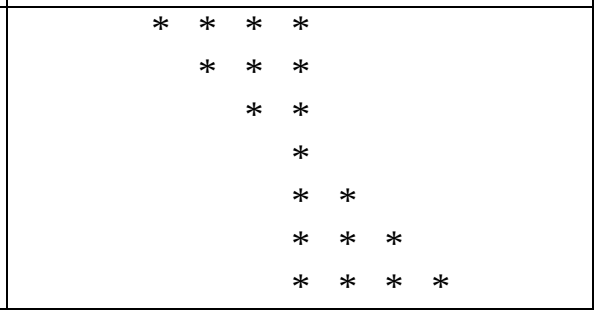
Мета: набуття навичок створення С++-програм із використанням циклічних структур із заданим числом повторень та вкладених циклів.

Завдання: Написати програму, яка виводить на екран зображений шаблон.

В програмі повинен задаватися максимальний розмір шаблона за горизонталлю (розмір має бути непарним цілим числом). Програма може використовувати лише три оператори виведення: [cout << “*”;], [cout << “ ”;] та [cout << endl;]. Для генерації шаблонів необхідно максимально використовувати структури повторення та умови (з вкладеними циклами for та умовами if).

1		2	
3		4	
5		6	

7		8	
9		10	
11		12	
13		14	
15		16	

17		18	
19		20	

Теоретичні відомості

Як було зазначено вище, у С++-програмах послідовне повторюване виконання певного фрагмента програми реалізують за допомогою *операторів повторення* (циклу) [1 - 4].

У мові С++ існує чотири типи циклів: for, while, do while і foreach.

Цикл for

Цикл for використовують тоді, коли заздалегідь відомо скільки разів повинно повторитися виконання фрагмента програми [5 - 9]. Тобто, цикл for організовує виконання фрагмента програми фіксовану кількість разів.

Синтаксис структури for наступний:

for (вираз1 ; вираз2 ; вираз3)

{

інструкція;

……// Тіло циклу з кількох операторів – блок

інструкція;

}

Зверніть увагу! Після оператора for та після закриваючої фігурної дужки крапка з комою не ставиться.

Цикл for управляється керуючою змінною, яку називають *лічильником циклу*. Вона має бути визначена до того, як почне виконуватися тіло циклу.

Вираз1 – слугує для ініціалізації керуючої змінної. Він являє собою інструкцію присвоювання, яка ініціалізує лічильник циклу, тобто встановлює керуючу змінну в деяке початкове значення. Вираз обчислюється тільки один раз.

Вираз2 – умова виконання циклу. Являє собою умовний вираз (містить в собі операцію відношень), в якому тестується значення керуючої змінної. Умова перевіряється кожен раз перед виконанням тіла циклу і визначає, чи потрібно виконувати цикл ще раз чи ні. Якщо умова істинна, то цикл виконується ще раз. У протилежному випадку виконання циклу припиняється, а управління передається оператору, що слідує за структурою циклу.

Вираз3 – інкремент керуючої змінної – вираз, що визначає як змінюється значення керуючої змінної після кожного проходження тіла циклу.

Цикл while

Цикл **while** називають *циклом з передумовою*. Синтаксис структури while наступний [1 - 5]:

```
while (умова)
{
… // тіло циклу;
}
```

В якості виразу *умови* допускається використання будь-якого виразу мови С++. Якщо умова істинна, то виконується тіло циклу while. Якщо умова хибна з самого початку або стає хибною у процесі виконання тіла циклу, то виконання циклу припиняється і виконується наступний за ним оператор.

В якості *тіла циклу* допускається використання будь-якого оператору, в тому числі порожнього або складеного. Якщо тіло циклу містить більше одного

оператора, то група операторів повинна бути укладена у фігурні дужки.

Зверніть увагу! Потрібно ініціалізувати змінну циклу while до початку виконання тіла циклу. Крім того, тіло циклу повинно містити оператор, який змінює значення змінної циклу, інакше цикл буде нескінченним.

Оператор циклу for виду:

```
for (вираз1; вираз2; вираз3)
{
…  // тіло циклу;
}
```

може бути замінений оператором while наступним чином:

```
вираз1;
while (вираз2)
{
…      // тіло циклу;
вираз3;
}
```

Цикл do while

Цикл **do while** називають *циклом з постумовою*: умова завершення циклу перевіряється не на його початку, як це мало місце у циклах while і for, а у кінці, вже після проходження тілом циклу [6 - 9]. Як наслідок, тіло циклу обов'язково виконується принаймні один раз.

Цей тип циклу зустрічається нечасто, але іноді буває корисний. Синтаксис структури do while наступний:

```
do     {
…      // тіло циклу;
}
while (умова);
```

Переривання циклу: оператори break, continue, return

У деяких випадках буває необхідно перервати повторення циклу.

Один із способів розв'язання цієї задачі – використання оператору break [6 - 9]. Він призначений для передчасного виходу з тіла циклу. Оператор break перериває виконання тіла будь-якого циклу (for, do чи while) та передає управління наступному за циклом оператору. При цьому умова виходу з циклу перевіряється в будь-якому його місті.

Ще один спосіб переривання циклу – використання оператору goto, що передає управління певному оператору за межами тіла циклу.

Для переривання циклів, розміщених у функціях, можна використати оператор return. На відміну від оператору break, оператор return перериває не лише виконання циклу, але й виконання тієї функції, у якій розміщений цикл. Розглянуті способи переривали виконання циклу. Але є ще процедура continue, яка перериває лише виконання поточної ітерації, поточного виконання тіла циклу та передає управління на наступну ітерацію. Тобто виконання оператора continue призводить до пропускання операторів, що слідують за ним у тілі циклу.

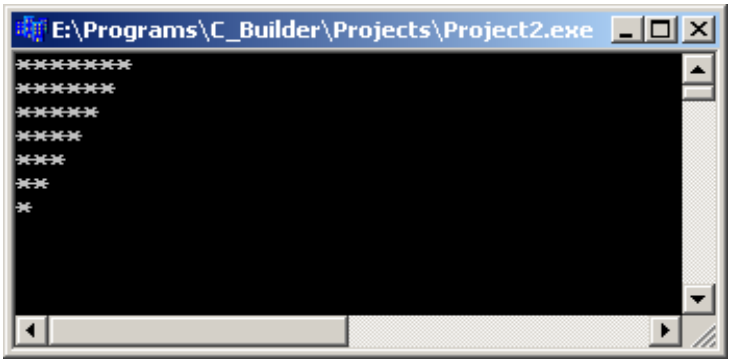
Приклад програми

Завдання: Написати програму, яка виводить на екран зображений шаблон:

```
*******
******
*****
****
***
**
*
```

```
#include <iostream> using namespace std;
int main (int argc, char* argv[])
```

```
{
int n, i, j;
cout<<"Введіть розмір шаблона: "; cin>>n;        // n=7
for (i=0; i<n; i++)
{
for (j = n - i; j > 0; j--) cout <<"*";
for (j = i; j < n; j++)
cout << " "; cout<<endl;
}
return 0;}
```



Контрольні запитання

1. Назвіть оператори повторення (циклу) C++.
2. Наведіть загальний синтаксис циклу for.
3. Яку змінну називають лічильником циклу?
4. Який цикл називають циклом з передумовою? Наведіть приклад.
5. Наведіть загальний синтаксис циклу while.
6. Який цикл називають циклом з постумовою? Наведіть приклад.
7. Наведіть загальний синтаксис циклу do while.
8. Яка різниця між циклом, що керується лічильником і циклом, що керується контрольним значенням?
9. Чи може цикл for мати кілька лічильників?

10. Що називають вкладеним циклом?
11. Як змінюються параметри зовнішнього та внутрішнього циклів?
12. Яке призначення оператору break?
13. Яке призначення оператору goto?
14. Яке призначення оператору continue?

Лабораторна робота №3. Рекурсивні функції

Мета: набуття навичок створення C++-програм із використанням рекурсивних функцій.

Завдання: Написати рекурсивну функцію, яка розв’язує задачу без використання циклів:

1. Напишіть рекурсивну функцію для обчислення суми квадратів чисел від 1 до n.
2. Створіть рекурсивну функцію, яка знаходить добуток усіх цифр цілого числа.
3. Напишіть функцію, що рахує, скільки разів певна цифра k зустрічається у числі n.
4. Реалізуйте рекурсивний пошук мінімальної цифри у заданому числі.
5. Напишіть рекурсивну функцію для знаходження суми елементів масиву, щоб потім обчислити їхнє середнє значення.
6. Створіть рекурсивну функцію, яка перевіряє, чи всі цифри числа є парними.
7. Реалізуйте рекурсивну функцію, яка виводить елементи масиву у зворотному порядку.
8. Напишіть функцію, яка перевіряє, чи є число точним степенем двійки (наприклад, 16 — так, 18 — ні).
9. Обчисліть суму всіх непарних чисел у діапазоні від a до b за допомогою рекурсії.
10. Напишіть функцію, яка підраховує кількість нулів у десятковому записі числа.
11. Реалізуйте алгоритм Евкліда для двох чисел через рекурсію.
12. Напишіть рекурсивну функцію, яка перевіряє, чи міститься цифра 7 у записі числа.
13. Обчисліть суму ряду $1 - 2 + 3 - 4 + ... \pm n$ за допомогою рекурсії.
14. Створіть рекурсивну функцію, яка повертає *true*, якщо задане число x є в масиві.

15. Реалізуйте рекурсивну функцію для знаходження найбільшої цифри в числі.

16. Напишіть функцію для обчислення *n!!* (добуток чисел одного з *n* паритету, наприклад 5!! = 5 · 3 · 1).

17. Обчисліть цілу частину від $log_2(n)$ за допомогою рекурсії (шляхом ділення на 2).

18. Напишіть функцію, яка знаходить суму всіх дільників числа *n* (крім самого числа).

19. Реалізуйте рекурсивну функцію для підрахунку кількості цифр у числі.

20. Обчисліть *n*-й член арифметичної прогресії з першим членом *a* та різницею *d*.

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 [1 - 3].

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 [4 - 6]. 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_типів параметрів);

Наприклад,

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

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

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

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

Зверніть увагу! 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 [1 - 5]. 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
}
```

Зверніть увагу! 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 [6 - 9].

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()

Зверніть увагу! 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.

Рекурсивні функції

Рекурсивними називаються функції, які в процесі виконання звертаються самі до себе [1 - 5]. Рекурсія може бути безпосередньою (пряма рекурсія) – коли з тіла функції викликається та ж сама функція, або посередньою (непряма рекурсія) – коли функція викликає інші функції, які в свою чергу безпосередньо або через треті функції звертаються до даної функції.

Здебільшого рекурсивні функції є лаконічними в записі й дають змогу наочно та стисло відобразити алгоритм розв’язування задачі. Існує цілий ряд

задач, яким рекурсивні алгоритми й конструкції найбільш притаманні, тому запрограмувати такі задачі без використання рекурсії достатньо складно, а деколи практично неможливо. З іншого боку, рекурсивні звертання часто не легкі для сприйняття, аналізу та контролю. Вони можуть вимагати значних обсягів оперативної пам’яті та призводити до часових затримок у процесі реалізації програми.

Як і у випадку з циклами, які містять умову продовження циклу, щоб не утворювати нескінченні цикли, рекурсивні функції також повинні містити умову виходу з рекурсії.

Умова виходу з рекурсії – це умова, при якій рекурсивна функція перестає викликати саму себе. В цій умові зазвичай використовується оператор *if*.

При кожному виклику функції, певні дані поміщаються в стек викликів, і за відсутності умови виходу з рекурсії, функції будуть викликатись до того моменту, поки не закінчиться пам’ять стеку і відбудеться переповнення стеку.

Приклад програми

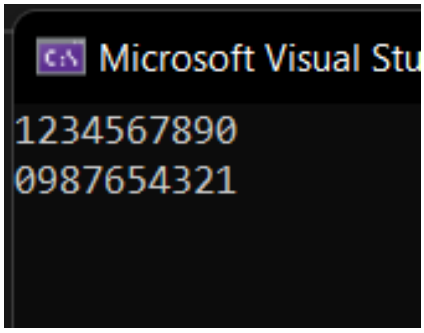
Завдання 1: Написати програму, яка використовує рекурсивну функцію для виведення всіх цифр числа задом наперед (Наприклад: число 1234 -> 4321).

```
#include <iostream>

using namespace std;

void print(int num) {
   if (num != 0) { //Умова виходу з рекурсії
      cout << num % 10;
      print(num / 10);    //Рекурсія
   }
   else {
      return;
   }
}
```

```
int main()
{
   int a;
   cin >> a;
   print(a);
   return 0;
}
}
```



Завдання 2: Написати програму, яка використовує рекурсивну функцію для виведення числа у двійковій формі

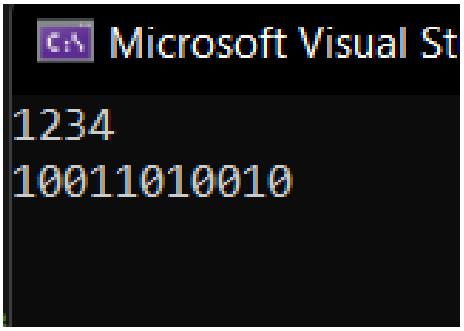
```
#include <iostream>

using namespace std;

void print_b(int num) {
   if (num == 0)   //Умова виходу з рекурсії
      return;
   print_b(num >> 1);  //Рекурсія
   cout << (num & 1);
}

int main()
{
   int a;
```

```
    cin >> a;
    print_b(a);
    return 0;
}
```



Контрольні запитання

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?
16. Які функції називають рекурсивними?
17. У чому переваги і недоліки рекурсивних функцій?
18. Що таке пряма та непряма рекурсія?

Лабораторна робота №4. Динамічне виділення пам’яті та масиви

Мета: набуття навичок створення С++-програм для обробки багатомірних динамічних масивів даних: зберігання, сортування, пошук.

Завдання: Задано двомірний масив дійсних чисел m x n. Реалізувати обробку даних масиву.

Елементи масиву вводяться з клавіатури. Пам’ять для роботи з масивом необхідно виділити динамічно, а також вивільнити її після завершення роботи з масивом.

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 елементів, що дорівнюють заданому. Вивести на екран вихідний масив і новий, елементи якого впорядковано за зростанням.

10. Визначити номер стовпчика 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 число.

13. Знайти кількість нульових елементів в кожному стовпчику масиву. Вивести на екран вихідний масив і новий, в якому всі нульові елементи замінено знайденим 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 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 знайденим 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 такі елементи замінено нулями.

18. Визначити номер стовпчика з найбільшою кількістю ненульових

елементів. Вивести на екран вихідний масив і новий, в якому до елементів знайденого стовпчика додано задане ціле число.

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.

21. Знайти кількість елементів, що перевищують значення першого елемента масиву. Вивести на екран вихідний масив і новий, в якому всі елементи, менші за значення першого елемента масиву, замінено середнім арифметичним значенням елементів масиву.

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

23. Знайти кількість нульових елементів в кожному рядку масиву. Вивести на екран вихідний масив і новий, в якому до всіх елементів рядка з мінімальною кількістю нульових елементів додане задане ціле число.

24. Знайти найбільший елемент масиву. Вивести на екран вихідний масив і новий, в якому елементи головної діагоналі помножено на цей елемент.

25. Знайти найменший елемент масиву. Вивести на екран вихідний масив і новий, в якому всі елементи головної діагоналі замінено знайденим значенням.

Теоретичні відомості

У C++ можна використовувати багатомірні масиви. Найпростіший багатомірний масив – двомірний. Двомірний масив, по суті, являє собою масив одномірних масивів [1 - 7].

Визначення двомірного масиву

Загальний синтаксис оголошення двомірного масиву наступний:

тип_даних ім’я_масиву [розмір1][розмір2];

Наприклад,

int b[10][10]; // оголошення двомірного масиву цілих чисел розміром 10x10

double s[3][4]; // оголошення двомірного масиву дійсних чисел розміром 3x4

Зверніть увагу! При оголошенні двомірного масиву у С++ кожна розмірність укладається у власну пару квадратних дужок.

Доступ до елементів двомірного масиву

Для роботи з масивом його елементи нумерують. Номер позиції елемента всередині масиву називають його *індексом*. У С++ індексація масивів починається з нуля [4 - 6].

У двомірному масиві позиція будь-якого елемента визначається двома індексами. Якщо уявити двомірний масив у вигляді таблиці, то перший індекс (за угодою) означає рядок, а другий – стовпчик. З цього випливає, що, якщо доступ до елементів масиву надати у порядку, в якому вони реально зберігаються у пам’яті, то правий індекс буде змінюватися швидше, ніж лівий.

Для звернення до певного елемента двомірного масиву вказують ім'я масиву та індекси елемента, укладені у квадратні дужки:

ім’я_масиву [індекс1][індекс2];

Наприклад,

int b[3][5]; // доступ до елемента двомірного масиву з координатами 3, 5

float list[2][3]; // доступ до елемента двомірного масиву з координатами 2, 3

Ініціалізація двомірного масиву

Багатомірні масиви можуть отримувати початкові значення у своїх оголошеннях так само, як одномірні масиви, за допомогою списку ініціалізації [6 - 9].

Формат ініціалізації двомірного масиву базується на тому факті, що двомірний масив, по суті, являє собою список одномірних масивів. Ініціалізуючі значення для кожного підмасиву укладають у фігурні дужки, розділяючи комами, а потім всі ці підмасиви також укладають у дужки та розділяють комами.

Наприклад,

int b[2][3]={

{1, 2, 3},

{5, 6, 7},

}; // ініціалізація двомірного масиву цілих чисел розміром 2x3

Якщо початкових значень у підмасиві не вистачає, то решті елементів автоматично присвоюється нульове початкове значення.

Динамічний розподіл пам’яті

C++ дозволяє керувати виділенням пам’яті та її поверненням у програмі для будь-якого типу даних. Це називають *динамічним розподілом пам’яті* [1 - 4]. Динамічне виділення пам’яті широко застосовують для економії обчислювальних ресурсів, адже пам'ять для даних виділяється у міру необхідності з області вільної пам’яті. Цю область називають «*кучею*» (heap).

Динамічне виділення пам’яті – це отримання програмою пам’яті під час її виконання. Іншими словами, таким чином програма може створювати змінні під час виконання, причому у потрібній (в залежності від ситуації) кількості.

У C++ динамічне керування пам’яттю здійснюється за допомогою двох операторів new та delete, які виконують функції з виділення та вивільнення пам’яті відповідно.

Оператор **new** динамічно виділяє пам'ять та повертає покажчик відповідного типу на цю область пам’яті (розмішений в пам'яті об'єкт).

Загальний формат використання оператора new наступний:

змінна-покажчик = new тип_змінної;

За допомогою оператора **new** можна виділити пам’ять для значень будь-якого допустимого типу. Оператор автоматично виділяє достатній об’єм пам’яті для зберігання значення заданого типу.

Наприклад:

```
double *A;
A= new double;
```

В даному випадку в пам'яті динамічно створюється об'єкт – дійсне число. Тут елемент **A** – змінна-покажчик, що буде приймати адресу виділеної пам’яті, а елемент **double** – це тип даних, які будуть зберігатися у цій пам’яті.

Після того як динамічно розміщений об’єкт стає непотрібним, необхідно вивільнити виділену йому оператором new ділянку пам'яті. Вивільняє динамічну пам'ять оператор **delete**.

Загальний формат використання оператора delete наступний:

delete змінна-покажчик;

Наприклад:

delete A; // вивільняємо пам’ять, виділену під об’єкт

Зверніть увагу! Оператор delete необхідно використовувати лише з тим покажчиком на пам'ять, який був повернутий в результаті new-запита на виділення пам’яті. Використання оператора delete з іншим типом адреси може викликати серйозні проблеми.

Ініціалізація динамічно виділеної пам’яті

Створення динамічно розміщеного об'єкта можна поєднати з його ініціалізацією – ініціалізувати динамічно виділену пам'ять, задавши ініціалізатор [5 - 9].

Загальний формат запису оператора new з використанням ініціалізатора наступний:

змінна-покажчик = new тип_змінної (ініціалізатор);

Тут ініціалізатор – це значення, яке буде присвоєне виділеній пам’яті. Очевидно, що тип ініціалізатора має бути сумісним із типом об’єкта, для якого виділяється пам'ять.

Наприклад:

```
double *A;
A = new double(5.1);          // ініціалізуємо пам'ять дійсним числом 5.1
int *p;
p= new int(87);               // ініціалізуємо память цілим числом 87
```

Виділення пам’яті для масивів

За допомогою new можна виділяти пам’ять й для масивів. Загальний формат операції виділення пам’яті для одномірного масиву наступний [1 - 4]:

змінна-покажчик = new тип_масива [розмір];

Тут елемент **розмір** задає кількість елементів у масиві.

Наприклад:

```
double *A;
A = new double[100];
```

Цей оператор new динамічно розміщує в пам’яті масив дійсних чисел на 100 елементів.

У подальшому до елементів масиву можна звертатись з використанням звичайного синтаксису, наприклад: **A[i].**

У випадку динамічного виділення пам’яті для масиву важливо пам’ятати, що його не можна одночасно й ініціалізувати.

Щоб вивільнити пам'ять, виділену для динамічно створеного масиву, використовують наступний формат оператора delete:

```
delete [] змінна-покажчик;
```

Наприклад:

```
delete [] A;          // вивільняємо пам'ять, зайняту масивом
```

Оскільки двомірний масив є таким масивом, в якому всі елементи також є масивами, то для динамічного виділення пам’яті під двомірний масив із заданою користувачем розмірністю доцільно використовувати наступний підхід:

```
float ** ptrarr;
ptrarr = new float* [2];
for (int count = 0; count < 2; count++)
    ptrarr[count] = new float [5];
```

Для вивільнення пам'яті, що виділяється під двомірний динамічний масив, діють наступним чином:

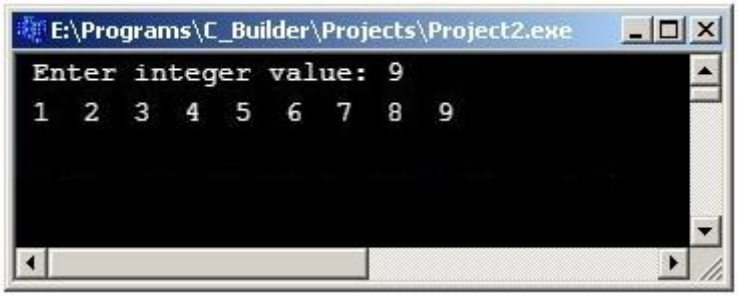
```
for (int count = 0; count < 2; count++)
    delete [] ptrarr[count];
```

Приклад програми

Завдання: Створити динамічний масив розмірності n та ініціалізувати його числами від 1 до n. Розмірність масиву користувач вводить з клавіатури.

```
#include <iostream>
```

```
using namespace std;
int main (int argc, char* argv[])
{
int num;
cout << " Enter integer value: ";
cin >> num;
int *p;
p = new int[num];
for (int i = 0; i < num; i++)
   {
   p[i] = i+1;
   cout << " " << p[i] << " ";
   }
delete [] p;
return 0;
}
```



Контрольні запитання

1. Що називають масивом?
2. Які масиви називають двомірними?
3. Наведіть синтаксис оголошення двомірного масиву.
4. Наведіть синтаксис звернення до певного елемента двомірного масиву.
5. В який спосіб можна задавати початкові значення елементам масиву?

6. Як ініціалізувати двомірний масив за допомогою списку ініціалізації?

7. Як ініціалізувати двомірний масив за допомогою циклу for?

8. Які початкові значення отримають елементи двомірного масиву у наступному випадку ініціалізації: array[2][3] = {1, 2, 3, 4, 5, 6};

9. Які початкові значення отримають елементи двомірного масиву у наступному випадку ініціалізації: array[2][3]= {{1, 2}, {4}};

10. Чим визначається обсяг пам’яті, що виділяється для багатомірного масиву?

11. Які операції над багатомірними масивами підтримуються C++?

12. Для чого використовують динамічне виділення пам’яті?

13. Який оператор дозволяє виділити пам'ять для змінних?

14. Який оператор вивільняє виділену пам'ять?

15. Чи можна ініціалізувати пам'ять у випадку її динамічного виділення?

16. Як ініціалізувати динамічно розміщений об'єкт?

17. Наведіть загальний формат операції виділення пам’яті для одномірного масиву.

Лабораторна робота №5. Класи

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

Завдання: 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 дробу. 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.yyyy, 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.

Теоретичні відомості

Клас – це базова C++-одиниця інкапсуляції [1 - 4]. 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 [5 - 9].

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* [1 - 5]. 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 С++ мають спеціальну функцію, яку називають *конструктором* [5 - 9].

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 [3 - 7].

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.

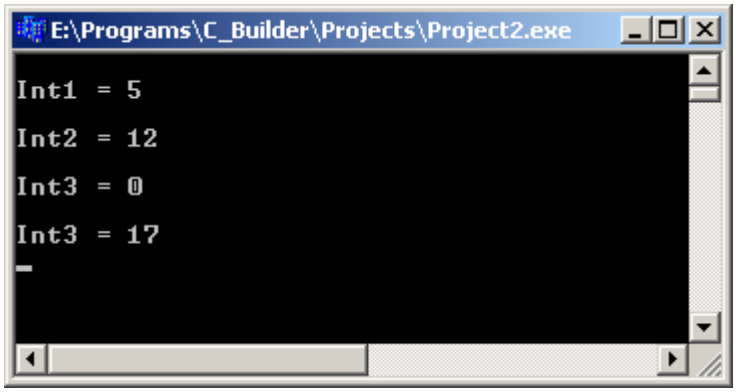
Якщо деструктор явним чином в класі не оголошений, компілятор сам генерує необхідні коди звільнення пам'яті.

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);  // створення 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();
}
```



Контрольні запитання

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.

Лабораторна робота №6. Наслідування

Мета: розробка програмного забезпечення з реалізації алгоритмів із використанням механізму наслідування.

Завдання: Написати програму для тестування похідного класу.

1. На основі базового класу «Геометрична фігура», в якому зберігаються основні параметри (наприклад, ширина та висота) двовимірного об’єкта, створити похідний клас «Прямокутник». У похідному класі передбачити функцію, яка визначає, чи є прямокутник квадратом та функцію, що обчислює площу прямокутника.

2. На основі базового класу «Відрізок», в якому зберігається довжина відрізка, створити похідний клас «Коло». У похідному класі передбачити функції, що обчислюють довжину кола та площу круга.

3. На основі базового класу «Точка», в якому зберігаються координати точки на площині (x та y), створити похідний клас «Вектор». У похідному класі передбачити функції для обчислення довжини вектора та суми двох векторів.

4. На основі базового класу «Особа», в якому зберігаються П.І.Б. особи, її стать та вік, створити похідний клас «Викладач», що містить інформацію про факультет, кафедру, де працює викладач, та займану посаду. У похідному класі передбачити також функції для виведення інформації щодо викладача на екран та визначення окладу в залежності від посади.

5. На основі базового класу «Геометрична фігура», в якому зберігаються основні параметри (наприклад, ширина та висота) двовимірного об’єкта, створити похідний клас «Паралелепіпед». У похідному класі передбачити функції, що обчислюють площу поверхні та об’єм паралелепіпеда.

6. На основі базового класу «Число» створити похідний клас «Дії». У похідному класі передбачити функції для додавання, віднімання, множення двох цілих чисел.

7. На основі базового класу «Відрізок», в якому зберігається довжина відрізка, створити похідний клас «Куля». У похідному класі передбачити функції, що обчислюють площу поверхні та об’єм кулі.

8. На основі базового класу «Геометрична фігура», в якому зберігаються основні параметри (наприклад, ширина та висота) двовимірного об’єкта, створити похідний клас «Паралелепіпед». У похідному класі передбачити функцію, що визначає, чи є паралелепіпед кубом та функцію, яка обчислює його площу поверхні.

9. На основі базового класу «Особа», в якому зберігаються П.І.Б. особи, її стать та вік, створити похідний клас «Автовласник», що містить інформацію про марку автомобіля, рік випуску, колір, державний номер. У похідному класі передбачити також функції для виведення інформації щодо автовласника на екран та визначення приналежності автомобіля із заданим державним реєстраційним номером конкретній особі.

10. На основі базового класу «Точка», в якому зберігаються координати точки на площині (x та y), створити похідний клас «Вектор». У похідному класі передбачити функцію, що визначає, чи є два вектори рівними, та функцію для обчислення скалярного добутку двох векторів.

11. На основі базового класу «Геометрична фігура», в якому зберігаються основні параметри (наприклад, ширина та висота) двовимірного об’єкта, створити похідний клас «Трапеція». У похідному класі передбачити функцію, що обчислює площу трапеції.

12. На основі базового класу «Відрізок», в якому зберігається довжина відрізка, створити похідний клас «Куб». У похідному класі передбачити функції, що обчислюють площу поверхні та об’єм куба.

13. На основі базового класу «Число» створити похідний клас «Дії». У похідному класі передбачити функції для множення та ділення двох цілих чисел, а також функцію піднесення до степеня.

14. На основі базового класу «Особа», в якому зберігаються П.І.Б. особи, її стать та вік, створити похідний клас «Працівник», що містить інформацію про посаду та розмір посадового окладу. У похідному класі передбачити також функції для виведення інформації щодо працівника на екран та визначення надбавки до окладу в залежності від вислуги років.

15. На основі базового класу «Геометрична фігура», в якому зберігаються основні параметри (наприклад, ширина та висота) двовимірного об’єкта, створити похідний клас «Циліндр». У похідному класі передбачити функції для обчислення площі бічної поверхні циліндра та його об’єм.

16. На основі базового класу «Відрізок», в якому зберігається довжина відрізка, створити похідний клас «Кільце». У похідному класі передбачити функцію, що обчислює площу кільця.

17. На основі базового класу «Точка», в якому зберігаються координати точки на площині (x та y), створити похідний клас «Вектор». У похідному класі передбачити функцію, що визначає, чи є два вектори колінеарними, та функцію для обчислення добутку вектора на число.

18. На основі базового класу «Геометрична фігура», в якому зберігаються основні параметри (наприклад, ширина та висота) двовимірного об’єкта, створити похідний клас «Конус». У похідному класі передбачити функції для обчислення площі бічної поверхні конуса та його об’єм.

19. На основі базового класу «Особа», в якому зберігаються П.І.Б. особи, її стать та вік, створити похідний клас «Студент», який зберігає результати сесії. У похідному класі передбачити також функції для виведення інформації щодо студента на екран, для визначення середнього балу студента та розміру його стипендії в залежності від середнього балу.

20. На основі базового класу «Хлопець», в якому зберігаються П.І.Б. та вік хлопця, створити похідний клас «Студент», що містить інформацію про місце навчання. У похідному класі передбачити функцію для визначення, чи є студент повнолітнім, та функцію для виведення інформації щодо певного студента на екран.

Теоретичні відомості

Однією з найбільш потужних властивостей класів у C++ є можливість їхнього розширення шляхом *наслідування* [1 - 5]. Наслідування дозволяє одному класу наслідувати характеристики іншого.

У стандартній термінології мови C++ клас, що наслідується, називають

базовим. Клас, який наслідує базовий клас, називають *похідним*.

Похідний клас наслідує всі члени, визначені в базовому класі, й додає до них власні унікальні елементи. Т. б., похідний клас являє собою спеціалізовану версію базового класу.

Базовий клас при наслідуванні залишається незмінним.

Оголошення похідного класу

Загальний формат оголошення класу, що наслідує базовий клас, має такий вигляд [2 - 5]:

```
class ім’я_похідного_класу : специфікатор_доступу ім’я_базового_класу  {
  // тіло похідного класу
};
```

При цьому використовується *операція двокрапка* (**:**), що встановлює відносини між класами.

Статус доступу членів базового класу в похідному класі визначається специфікатором доступу (private, public, protected), що використовується для наслідування базового класу.

Доступ до базового класу

Для об’єктів похідного класу можуть бути використані відкриті методи базового класу.

Проте, наслідування не працює у зворотному напрямку. Базовому класу та його об’єктам недоступні похідні класи.

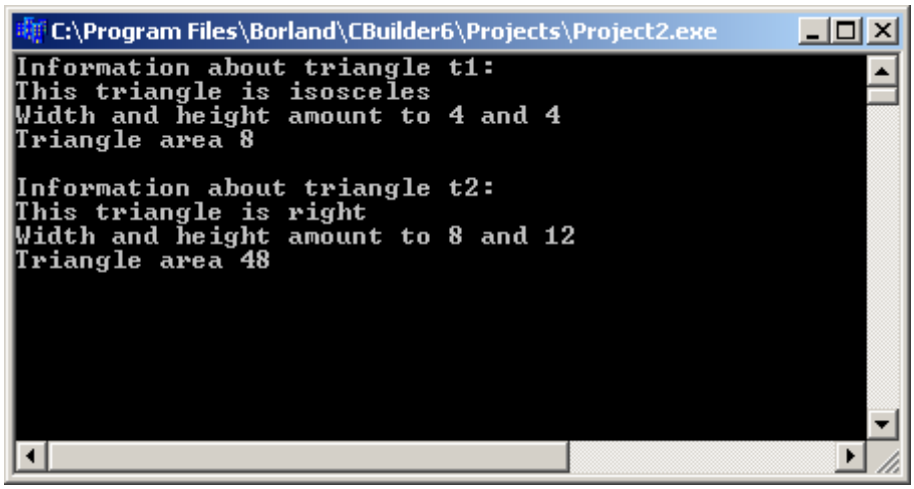
Приклад програми

Завдання: На основі базового класу TwoShape, в якому зберігаються основні параметри (ширина та висота) двовимірного об’єкта, створити похідний клас Triangle, який інкапсулює трикутники.

```
#include <iostream>
#include <cstring>
#include <conio.h>
using namespace std;
// оголошення базового класу
```

```
class TwoShape {
 public:
        double width;
        double height;
        void showDim()
        { cout << "Width and height amount to "
              << width << " and " << height << "\n"; }
};
// оголошення похідного класу
class Triangle : public TwoShape {
 public:
       char style[20];
       double area()
       { return width* height/2; }
       void showStyle()
       {cout << "This triangle is" << style << "\n"; }
};
void main()
{
Triangle t1, t2;
t1.width=4.0;
t1.height =4.0;
strcpy(t1.style," isosceles ");
t2.width=8.0;
t2.height =12.0;
strcpy(t2.style," right ");
cout << "Information about triangle t1:\n";
t1.showStyle();
t1.showDim();
cout<<"Triangle area "<<t1.area()<<"\n"<<"\n";
```

```
cout << "Information about triangle t2:\n";
t2.showStyle();
t2.showDim();
cout<<"Triangle area "<<t2.area()<<"\n";
getch();
}
```



Контрольні запитання

1. У чому полягає механізм наслідування?
2. Як називають клас, що породжує інші класи?
3. Як називають клас, що є нащадком іншого класу?
4. Які рівні наслідування базового класу Вам відомі?
5. Наведіть загальний формат оголошення похідного класу.
6. Як змінюється базовий клас при наслідуванні?
7. Як називають випадок створення класу на основі двох чи більше класів?
8. Який клас називають абстрактним базовим класом?

Список використаних джерел

1. Lafore R. Object-Oriented Programming in C++. 4th ed. Indianapolis : Sams Publishing, 2002. 1012 p.

2. Stroustrup, Bjarne. The C++ Programming Language. 4th ed., Addison-Wesley, 2013.

3. Langr, Jeff. Modern C++ Programming with Test-Driven Development: Code Better, Sleep Better. Dallas, TX: Pragmatic Bookshelf, 2013.

4. Barr, Michael. Programming Embedded Systems in C and C++. O'Reilly Media, 1999.

5. Modern C++ Programming Cookbook: Recipes to explore data structure, multithreading, and networking in C++17 , ISBN 978-1-78646-518-4, 2017. – 559 p.

6. Amini K. Extreme C: Taking You To The Limit In Concurrency, OOP, And The Most Advanced Capabilities Of C, New York: Packt Publishing, 2019. – 823 p.

7. Я. М. Глинський, В. Є. Анохін, В. А. Ряжська, C++ і C++ Builder, навчальний посіб. 5-те вид. – Львів: [СПД Глинський], 2011. – 192 с.

8. О. Васильєв Програмування C++ у прикладах і задачах. – Ліра-К, 2017. – 382 с.

9. О. Трофименко C++ Основи програмування. Теорія і практика. – Одеса: Фенікс, 2010. – 544 с.