МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«УЖГОРОДСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ»
ІНЖЕНЕРНО-ТЕХНІЧНИЙ ФАКУЛЬТЕТ
КАФЕДРА КОМП'ЮТЕРНИХ СИСТЕМ ТА МЕРЕЖ

КОМП'ЮТЕРНІ ТА КОМУНІКАЦІЙНІ ТЕХНОЛОГІЇ

Методичні вказівки та завдання до лабораторних робіт
для студентів 1-го курсу інженерно-технічного факультету,
спеціальності G7 Автоматизація, комп'ютерно-інтегровані технології
та робототехніка

Ужгород – 2025 р.

Комп’ютерні та комунікаційні технології. Методичні вказівки і завдання до лабораторних робіт для студентів 1-го курсу інженерно-технічного факультету, спеціальності G7 Автоматизація, комп’ютерно-інтегровані технології та робототехніка.

Укладачі: Пойда В.Ю., доцент, канд.фіз.-мат.наук, доцент кафедри комп’ютерних систем та мереж інженерно-технічного факультету ДВНЗ «УжНУ»;
Балога С.І., доцент, канд.фіз.-мат.наук, доцент кафедри комп’ютерних систем та мереж інженерно-технічного факультету ДВНЗ «УжНУ»;
Тютюнникова Г.С., старший викладач кафедри комп’ютерних систем та мереж інженерно-технічного факультету ДВНЗ «УжНУ».

Рецензент: Глебена М.І., канд. фіз.-мат. наук, доцент, зав. кафедри системного аналізу та теорії оптимізації факультету математики та цифрових технологій ДВНЗ «УжНУ».

Відповідальний за випуск – Голик Й.М., доцент, канд. техн. наук, декан інженерно-технічного факультету ДВНЗ «УжНУ».

Дані методичні вказівки розглянуто та схвалено на засіданні кафедри комп’ютерних систем та мереж, протокол № 1 від «29» серпня 2025 року та науково-методичної комісії інженерно-технічного факультету протокол № 1 від «10» вересня 2025 року.

ВСТУП

Мета вивчення навчальної дисципліни “Комп’ютерні та комунікаційні технології” - отримання студентами базових знань з програмування та комп’ютерної техніки. Отримані знання з даної дисципліни дозволять застосовувати класичні та сучасні методи програмування в практичній роботі. Майбутній спеціаліст може застосовувати знання, які отримав при вивченні цієї дисципліни, як при подальшому навчанні, так і після отримання освітньо-кваліфікаційного рівня – бакалавр у своїй професійній діяльності.

Завдання дисципліни – формувати теоретичні знання та практичні навички у майбутніх фахівців відповідно до поставленої мети.

Вивчення даної дисципліни надасть студентам необхідну теоретичну і практичну підготовку для того, щоб вміти розробляти і аналізувати блок-схеми, переводити блок-схеми у програмний код, готувати тести, аналізувати отримані результати та формувати висновки.

Дані методичні вказівки допоможуть студентам засвоїти базові поняття програмування. Вивчити і осмислити основні алгоритми виконання простіших задач.

Виконання студентами лабораторних робіт з курсу “Комп’ютерні та комунікаційні технології” дозволить закріпити теоретичні знання і надасть можливість набути практичних навичок роботи із застосуванням Visual Studio.

Лабораторна робота № 1

Тема: Програмування лінійних та розгалужених алгоритмів

Мета роботи: Ознайомитись із роботою програмного додатку Online C++ Compiler. Оволодіти навичками даного додатку для створення програм на мові С++. Навчитись створювати програми на мові С++, в яких реалізуються лінійні та розгалужені алгоритми. Набути навичок налагодження та тестування програм.

Наголосимо, що дана лабораторна робота складається із двох окремих частин:

1. *Лінійні алгоритми.*
2. *Розгалужені алгоритми.*

1 Лінійні алгоритми

Зміст роботи:

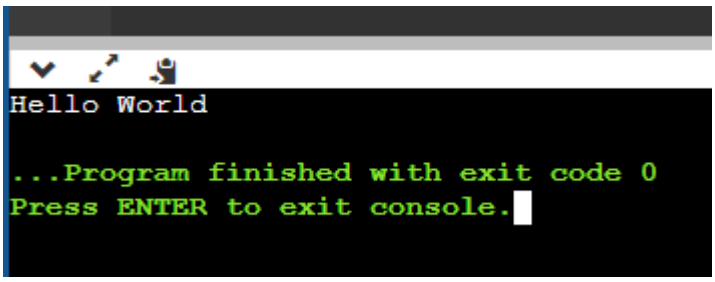
1. Ознайомитись із роботою і основними можливостями програмного додатку Online C++ Compiler.
2. За допомогою даного додатку створити перші програми на мові С/С++.
3. Запустити їх на виконання, перевірити роботу програм і якщо потрібно здійснити її налагодження.
4. Виконати всі завдання лабораторної роботи, та індивідуальне завдання згідно варіанту.

Хід роботи

Завдання 0. Ознайомлення із роботою Online C++ Compiler

1. У Інтернеті набираємо **Online C++ Compiler.**
2. Вибираємо і запускаємо **Online C++ Compiler**. Відкриється вікно з наступним кодом:

3. Для запуску запропонованої програми на виконання, натискаємо зелену клавішу **Run** у верхній частині екрану. Результат `Hello World` появиться у нижній частині екрану. Далі слідує повідомлення (зеленого кольору), що програма завершила виконання успішно з кодом 0.



Коротко пояснимо код, або текст, даної програми.

Перш 7 рядків – це коментарі до програми. Їх можна до уваги і не брати.

Код програми – це рядки 9-18. Порожні рядки 10, 12, 16 – це пропуски, які зроблені для зручності, і в коді програми ніякої ролі не виконують. Отже програма закодована у семи рядках 9, 11, 13, 14, 15, 17, 18.

Перші два з них:

```
#include <iostream>
using namespace std;
```

- це заголовок програми на мові С++. Він є стандартним. І поки що деталізацію опустимо.

Наступний рядок

```
int main( )
```

- тут вказане ім'я головної функції **main**, воно типу **int**. Цей рядок теж є стандартним.

Рядки 14 і 18 це відкриваюча і закриваюча фігурні дужки. Між ними поміщається тіло функції, або розділ операторів. В рядку 17 конструкція, яка керує виведенням на екран тексту **Hello World**. Останній рядок **return 0;** – повернення із функції. Його можна і опустити. Наголосимо, що кожний оператор закінчується крапкою з комою **;**.

Завдання 1. Переведемо дану програму із стилю С++ у стиль С.

```
#include <stdio.h>
int main()
{
     printf("Hello C \n ");
}
```

Тут код програми скоротився до п'яти рядків, а заголовок - до одного рядка. У заголовку підключається бібліотека **stdio.h**, з функціями для управління введенням/виведенням інформації. Для виведення тексту **Hello C** на екран ми скористались бібліотечною функцією форматованого виводу **printf**. **\n** – пара управляючих символів (перехід на новий рядок).

Завдання 2. Перші обчислення. Тут залишимо стиль С.

```
#include <stdio.h>
int main()
{
      int a=2, b=3;        //   ініціалізація даних
      int c=a+b;           //   обчислення
      printf("c= %d",c);   //   виведення результату
}                          //   на екран: c=5
```

Ініціалізація даних – це опис змінних, та присвоєння їм початкових значень.

Завдання 3. Перейдемо у стиль c++. Програму з попереднього завдання переведемо у стиль C++.

```
#include <iostream>
using namespace std;
int main()
{
      int a=2, b=3;            //   ініціалізація даних
      int c=a+b;               //   обчислення
      cout << "c=" << c << endl;// виведення результ.
}                              //   на екран
```

Як бачимо помінявся заголовок програми та **printf** замінився на конструкцію **cout**.

Завдання 3а. Замінимо змінні із цілих на дійсні. Для цього замінимо **int** на **float**, а значенням змінних добавимо дробові частини.

```
#include <iostream>
using namespace std;
int main()
{
      float a=2.1, b=3.3;      //   ініціалізація даних
      float c=a+b;             //   обчислення
      cout << "c=" << c << endl; // на екран: c=5.4
}
```

Звернемо увагу, що у мові C++ розділювачем цілої і дробової частин дійсного числа служить крапка, а не кома.

Завдання 4. Перевірити формулу скороченого множення, формулу квадрата суми.

$$(a+b)^2 = a^2 + 2ab + b^2$$

Змінній **c** присвоїмо значення лівої частини рівності, а змінній **d** – правої.

```
#include <iostream>
using namespace std;
int main()
{
      float a=2.1, b=3.3;      //   ініціалізація даних
      float c=(a+b)*(a+b);     //   обчислення
      float d=a*a+2*a*b+b*b;   //        на екрані:
      cout << "c=" << c << endl;    //   c=29.16
      cout << "d=" << d << endl;    //   d=29.16
}
```

Як і очікувалось, значення **c** і **d** співпали.

Завдання 5. Тригонометрія. Синус суми двох кутів.

$$\sin(\alpha+\beta) = sin\alpha\ cos\beta + cos\alpha\ sin\beta$$

Дане завдання виконаємо аналогічно як попереднє. Для обчислення тригонометричних функцій підключаємо бібліотеку **cmath**. Це другий рядок заголовку програми.

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
```

```
        float a=0.3, b=0.4;       //   ініціалізація даних
        float c=sin(a+b);         //   обчислення
        float d=sin(a)*cos(b)+cos(a)*sin(b);
        cout << "c=" << c << endl;          //   c=0.644218
        cout << "d=" << d << endl;          //   d=0.644218
}
```

Завдання 6. Розв'язування квадратного рівняння.

$$ax^2 + bx + c = 0$$
$$D = b^2 - 4ac$$
$$x = \frac{-b \pm \sqrt{D}}{2a}$$

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
        float a=1, b=7, c=12;          //   ініціалізація даних
        float d=b*b-4*a*c;
        float x1=(-b-sqrt(d))/(2*a);
        float x2=(-b+sqrt(d))/(2*a);
        cout << "x1=" << x1 << endl;  //   x1=-4
        cout << "x2=" << x2 << endl;  //   x2=-3
}
```

Варіант 2. Замінимо в цьому прикладі ініціалізацію даних на введення з клавіатури. Для цього перший рядок у тілі функції замінимо на наступні рядки:

```
float a, b, c;
cout << "a="; cin >> a;
cout << "b="; cin >> b;
cout << "c="; cin >> c;
```

Після запуску програми початкові дані вводимо з клавіатури в діалоговому режимі.

Завдання 7. Створення програми з реалізацією лінійного алгоритму

1. Розглянемо створення програми для реалізації лінійного алгоритму на наступному прикладі.

Написати програму для розрахунку по двох формулах. Результати обчислень по двох формулах повинні співпадати. Для обчислень потрібно використовувати функції з математичної бібліотеки мови С++, які знаходяться у файлі **<cmath>**. Ввід вхідних даних здійснюється із клавіатури, а результати виводяться на екран.

Формули:

$$z_1 = \frac{(m-1)\sqrt{m} - (n-1)\sqrt{n}}{\sqrt{m^3 n} + nm + m^2 - m}, \qquad z_2 = \frac{\sqrt{m} - \sqrt{n}}{m}.$$

2. На початку програми підключимо необхідні файли:

```
#include <iostream>
#include <cmath>
using namespace std;
```

3. Із умови задачі випливає, що вводитись будуть два числа **m, n**. Реалізуємо програмно їх опис і ввід з клавіатури в тілі головної функції програми **main**:

```
int main()
{
  double n,m;
```

```
    cout << "n = "; cin  >> n;
    cout << "m = "; cin  >> m;
...
}
```

Оператор **cout << "n = ";** здійснює виведення на екран підказки **n =** (пропонується ввести дійсне число). За ним слідує оператор введення **cin >> n;**, який вводить набране число в комп'ютер.

4. Далі обчислимо z_1 і z_2 по формулах, використовуючи математичні функції **sqrt** та **pow** з файлу **<cmath>**. Нові оператори добавляємо на місце три крапок.

```
double z1=((m-1)*sqrt(m)-(n-1)*sqrt(n))/
      (sqrt(pow(m,3)*n)+n*m+m*m-m);
double z2=(sqrt(m)-sqrt(n))/m;
```

5. Далі будуть слідувати оператори виводу результату:

```
cout << "z1 = " << z1 << endl;
cout << "z2 = " << z2 << endl;
```

Код програми завершується закритою фігурною дужкою, яка фактично закриває розділ операторів функції **main**.

6. Для компіляції і запуску програми натиснемо **Run**. Водимо додатні значення **n** і **m**. Останнє диктується областю визначення функцій z_1 і z_2 від аргументів *n* і *m*. Переконуємось в тому, що обчислені комп'ютером значення z_1 і z_2 співпадають.

7. Наведемо дані для тестування:

```
n = 2
m = 3
z1 = 0,105946
z2 = 0,105946
```

Завдання 8. Створення лінійного алгоритму та написання відповідної програми

Сторони трикутника рівні *a, b, c*. Скласти програму для обчислення площі трикутника S, радіуса вписаного кола *r*, та описаного кола *R*, його висот, медіан та внутрішніх кутів. Протестувати програму для вказаних значень *a, b, c*.

$$S=\sqrt{p(p-a)(p-b)(p-c)},\ p=\frac{a+b+c}{2},$$

$$r=\frac{S}{p}, R=\frac{abc}{4S};$$

$$h_a=\frac{2S}{a}, h_b=\frac{2S}{b}, h_c=\frac{2S}{c};$$

$$m_a=\frac{1}{2}\sqrt{2b^2+2c^2-a^2}, m_b=\frac{1}{2}\sqrt{2a^2+2c^2-b^2}, m_c=\frac{1}{2}\sqrt{2a^2+2b^2-c^2},$$

$$\angle A=\arccos\left(\frac{b^2+c^2-a^2}{2bc}\right),\ \angle B=\arccos\left(\frac{a^2+c^2-b^2}{2ac}\right),\ \angle C=\arccos\left(\frac{a^2+b^2-c^2}{2ab}\right);$$

Дані для тестування:

$a=5;\ b=6;\ c=7.$

$S=14{,}6969;\ \ r=1{,}6330;\ \ R=3{,}5722;$

$h_a=5{,}8788;\quad m_a=6{,}0208;\quad \angle A=44{,}4153°;$

$h_b=4{,}8990;\quad m_b=5{,}2915;\quad \angle B=57{,}1217°;$

$h_c=4{,}1991;\quad m_c=4{,}2720;\quad \angle C=78{,}463°.$

Короткі рекомендації при написанні програм

1. Вибирайте тип змінних з урахуванням діапазону і необхідної точності представлення даних.
2. Давайте змінним імена, які відображають їх призначення.
3. Введення з клавіатури супроводжуйте запитом. Для контролю зразу ж після вводу даних виведіть вхідні дані на дисплей. Після завершення налагодження програми цей пункт можна вилучити з програми.
4. До запуску програми підготуйте тестові приклади, які містять необхідні дані і очікувані результати.
5. При записі виразів звертайте увагу на пріоритет операцій.
6. У функціях printf і scanf для кожної змінної вказуйте специфікацію формату, що відповідає її типу. Пам'ятайте, що в scanf передається адреса змінної, а не її значення.
7. При використанні стандартних функцій або класів потрібно за допомогою директиви #include підключити до програми відповідні заготовочні файли.
8. Не змішуйте в одній програмі ввід/вивід за допомогою класів (в стилі C++) і за допомогою функцій бібліотеки (в стилі C).
9. Надавайте перевагу локальним змінним перед глобальними. Змінна повинна мати мінімальну із можливих областей дії.
10. Дані при вводі розділяйте пропусками, символами переводу рядка або табуляції.

Протокол лабораторної роботи

Завершальним етапом виконання лабораторної роботи є оформлення протоколу лабораторної роботи. В заголовку протоколу потрібно вказати номер, назву та мету роботи. Далі потрібно описати хід роботи. Протоколюючи виконання індивідуального завдання обов'язково записати умову задачі, код програми з поясненнями окремих фрагментів, результати тестування, та результати моніторингу в середовищі **MathCad**. Протокол лабораторної роботи завершується висновками. Висновки повинні бути змістовними, з обговоренням проблемних ситуацій, які виникали при виконанні роботи.

Індивідуальне завдання

Студент в обовязковому порядку виконує одне завдання згідно варіанту. Прототипом індивідуального завдання є розглянуте вище завдання 7.

Наголосимо, що для довільних значень аргументів обчислені комп'ютером значення z_1 і z_2 повинні співпадати.

Обчислити z_1 і z_2 також у **MathCad**, і переконатись, що вони співпадають як між собою, так з відповідними результатами отриманими у Visual Studio.

У варіантах де у виразах фігурує константа π, потрібно ініціалізувати константу, наприклад так: `double pi=3.1415926;`.

Фрагмент виразу, який починається із раціонального дробу, потрібно почати із дійсної константи. Наприклад фрагмент $z_1 = \frac{1}{4} - \sin\left(\frac{5}{2}\pi - \alpha\right)$ доцільно запрограмувати так:

```
z1 = 1./4-sin(5./2*pi-a);
```

Саме символ крапка приводить до сприйняття компіляторим чисельника як дійсного числа. І тоді результатом виконання фрагменту `1./4` буде число `0.25`. Якщо ж крапку опустити то результатом буде `0`, і результат обчислення всього виразу буде неправильними.

Варіанти індивідуальних завдань

1. $z_1 = 2\sin^2(3\pi - 2\alpha)\cdot\cos^2(5\pi + 2\alpha)$;
$z_2 = \frac{1}{4} - \frac{1}{4}\sin\left(\frac{5}{2}\pi - 8\alpha\right)$.

2. $z_1 = \cos\alpha + \sin\alpha + \cos 3\alpha + \sin 3\alpha$;
$z_2 = 2\sqrt{2}\cos\alpha\cdot\sin\left(\frac{\pi}{4} + 2\alpha\right)$.

3. $z_1 = \frac{\sin 2\alpha + \sin 5\alpha - \sin 3\alpha}{\cos\alpha + 1 - 2\sin^2 2\alpha}$;
$z_2 = 2\sin\alpha$.

4. $z_1 = \frac{\sin 2\alpha + \sin 5\alpha - \sin 3\alpha}{\cos\alpha - \cos 3\alpha + \cos 5\alpha}$;
$z_2 = tg3\alpha$.

5. $z_1 = 1 - \frac{1}{4}\sin^2 2\alpha + \cos 2\alpha$;
$z_2 = \cos^2\alpha + \cos^4\alpha$.

6. $z_1 = \cos\alpha + \cos 2\alpha + \cos 6\alpha + \cos 7\alpha$;
$z_2 = 4\cos\frac{\alpha}{2}\cdot\cos\frac{5}{2}\alpha\cdot\cos 4\alpha$.

7. $z_1 = \cos^2\left(\frac{3}{8}\pi - \frac{\alpha}{4}\right) - \cos^2\left(\frac{11}{8}\pi + \frac{\alpha}{4}\right)$;
$z_2 = \frac{\sqrt{2}}{2}\sin\frac{\alpha}{2}$.

8. $z_1 = \cos^4 x + \sin^2 y + \frac{1}{4}\cdot\sin^2 2x - 1$;
$z_2 = \sin(y + x)\cdot\sin(y - x)$.

9. $z_1 = (\cos\alpha - \cos\beta)^2 - (\sin\alpha - \sin\beta)^2$;
$z_2 = -4\sin^2\frac{\alpha - \beta}{2}\cdot\cos(\alpha + \beta)$.

10. $z_1 = \frac{\sin\left(\frac{\pi}{2} + 3\alpha\right)}{1 - \sin(3\alpha - \pi)}$;
$z_2 = ctg\left(\frac{5}{4}\pi + \frac{3}{2}\alpha\right)$.

11. $z_1 = \frac{1 - 2\sin^2\alpha}{1 + \sin 2\alpha}$;
$z_2 = \frac{1 - tg\alpha}{1 + tg\alpha}$.

12. $z_1 = \frac{\sin 4\alpha}{1 + \cos 4\alpha}\cdot\frac{\cos 2\alpha}{1 + \cos 2\alpha}$;
$z_2 = ctg\left(\frac{3}{2}\pi - \alpha\right)$.

13. $z_1 = \frac{\sin\alpha + \cos(2\beta - \alpha)}{\cos\alpha - \sin(2\beta - \alpha)}$;
$z_2 = \frac{1 + \sin 2\beta}{\cos 2\beta}$.

14. $z_1 = \frac{\cos\alpha + \sin\alpha}{\cos\alpha - \sin\alpha}$;
$z_2 = tg2\alpha + \sec 2\alpha$.

15. $z_1 = \frac{\sqrt{2b + 2\sqrt{b^2 - 4}}}{\sqrt{b^2 - 4} + b + 2}$;
$z_2 = \frac{1}{\sqrt{b + 2}}$.

16. $z_1 = \frac{x^2 + 2x - 3 + (x + 1)\sqrt{x^2 - 9}}{x^2 - 2x - 3 + (x - 1)\sqrt{x^2 - 9}}$;
$z_2 = \sqrt{\frac{x + 3}{x - 3}}$.

17. $z_1 = \frac{\sqrt{(3m + 2)^2 - 24m}}{3\sqrt{m} - \frac{2}{\sqrt{m}}}$;
$z_2 = -\sqrt{m}$.

18. $z_1 = \left(\frac{a + 2}{\sqrt{2a}} - \frac{a}{\sqrt{2a} + 2} + \frac{2}{a - \sqrt{2a}}\right)\frac{\sqrt{a} - \sqrt{2}}{a + 2}$;
$z_2 = \frac{1}{\sqrt{a} + \sqrt{2}}$.

19. $z_1 = \left(\frac{1 + a + a^2}{2a + a^2} + 2 - \frac{1 - a + a^2}{2a - a^2}\right)^{-1}(5 - 2a^2)$;
$z_2 = \frac{4 - a^2}{2}$.

20. $z_1 = \frac{(m - 1)\sqrt{m} - (n - 1)\sqrt{n}}{\sqrt{m^3 n} + nm + m^2 - m}$;
$z_2 = \frac{\sqrt{m} - \sqrt{n}}{m}$.

Контрольні завдання

1. У коді програми завдання 0 організуйте виведення кількох рядків тексту одним оператором.
2. У коді програми завдання 1 організуйте виведення кількох рядків тексту одним оператором.
3. У коді програми завдання 6 замініти типи змінних **`float`** на **`double`**.
4. У коді програми завдання 7 замініть оператори введення виведення на відповідні функції у стилі С.
5. У коді програми завдання 7 оператори виведання z_1 і z_2 замініть одним оператором.
6. Складіть програму для перевірки головної тригонометричної тотожності.

Контрольні питання

1. Які типи даних вам відомо?
2. Які існують операції, та їхні приорітети?
3. Для чого потрібні функції **printf** і **scanf**?
4. Для чого призначена деректива **include**?
5. Чи можлива робота програми без **include**?
6. Як виділяється розділ операторів у програмі?
7. Який зв'язок між типом функції і оператором повернення з функції?
8. Які це глобальні і локальні змінні?
9. Що таке область дії змінної?
10. Які це прості і складені оператори просвоєння?

2. Програмування розгалужених алгоритмів

Теоретичні відомості

Умовний оператор **`if`**. Даний оператор реалізує алгоритмічну структуру – розгалуження. Реалізується у двох варіантах:

- повне розгалуження: **`if (P) S1; else S2;`**
- неповне розгалуження: **`if (P) S1;`**

`P` – логічний вираз, приймає значення *true* або *false* (істинне або хибне). Дужки, у які береться логічний вираз, є обов'язковими. **`S1`** – оператр, виконується, якщо **`P`** – `істинне.` Оператор **`S2`** виконується у повному розгалуженні, якщо **`P`** – хибне.

Якщо у якісь із віток потрібно виконати кілька операторів, то їх потрібно заключити у блок {…}. У блоці можуть бути будь-які оператири в тому числі описи і інші умовні оператири, але блок не може складатись тільки з одних описів.

Приклад 1. Скласти програму для обчислення виразу:

$$y = \begin{cases} \sin x, & x \ge 0; \\ \cos x, & x < 0. \end{cases}$$

Наведемо текст програми з використанням оператора **if** у форматі повного розгалуження:

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
     float x,y;
     cout << "x="; cin >> x;
     if (x>=0) y=sin(x); else
```

```
        y=cos(x);
    cout << "y=" << y << endl;
    return 0;
}
```

Дані для тестування програми: для $x = 1,\ y = 0{,}841471$, для $x = -1,\ y = 0{,}540302$.

Зауважимо, що тригонометричні функції обчислюються від аргументів заданих у радіанах.

В даній програмі оператор **if** можна замінити на послідовність двох операторів **if** у форматі неповного розгалуження:

```
if (x>=0) y=sin(x);
if (x< 0) y=cos(x);
```

Приклад 2. Скласти програму для обчислення виразу:

$$y = \begin{cases} 0, & x \le 0 \text{ or } x \ge 1000; \\ 1, & 1 \le x \le 9; \\ 2, & 10 \le x \le 99; \\ 1, & 100 \le x \le 999. \end{cases}$$

Текст програми з використанням оператора **if** у форматі повного розгалуження:

```
include <iostream>
using namespace std;
int main()
{
    int x,y;
    cout << "x="; cin >> x;
    if (x<=0 || x>=1000) y=0; else
        if (x>=1 && x<=9) y=1; else
            if (x>=10 && x<=99) y=2; else y=3;
    cout << "y=" << y << endl;
    return 0;
}
```

В даній програмі при формуванні логічних виразів використані конструкції з пар символів: **||** – або, **&&** - і.

Приклад 3. Знайти максимальне із двох заданих чисел x, y: $u = \max(x, y)$.

Оскільки u може приймати одне із двох значень: x або y, то умову даної задачі можна переписати у наступному виді:

$$u = \begin{cases} x, & x \ge y; \\ y, & y > x. \end{cases}$$

Тоді для розв'язання можна запропонувати наступну програму:

```
#include <iostream>
using namespace std;
int main()
{
    float x,y,u;
    cout << "x="; cin >> x;
    cout << "y="; cin >> y;
    if (x>=y) u=x; else u=y;
    cout << "u=" << u << endl;
}
```

Приклад 4. Знайти максимальне із трьох чисел x, y і z: $u = \max(x, y, z)$:

Перепишемо умову задачі у наступному виді:

$$u = \begin{cases} x, & x \ge y,\ x \ge z, \\ y, & y \ge x,\ y \ge z, \\ z, & z \ge x,\ z \ge y, \end{cases}$$

Тоді по аналогії з попередньою задачею для розв'язання запропонуємо наступну програму:

```
#include <iostream>
using namespace std;
int main()
{
      float x,y,z,u;
      cout << "x="; cin >> x;
      cout << "y="; cin >> y;
      cout << "z="; cin >> z;
      if (x>=y && x>=z) u=x; else
            if (y>=x && y>=z) u=y; else u=z;
      cout << "u=" << u << endl;
}
```

Приклад 5. Знайти полярні координати точки на площині по її прямокутним координатам x і y. При цьому скористатись формулами:

$r = \sqrt{x^2 + y^2}$;

$$\varphi = \begin{cases} arctg\dfrac{y}{x}, & x > 0, y \ge 0, \\ \dfrac{\pi}{2}, & x = 0, y > 0, \\ \pi + arctg\dfrac{y}{x}, & x < 0, \\ \dfrac{3\pi}{2}, & x = 0, y < 0, \\ 2\pi + arctg\dfrac{y}{x}, & x > 0, y < 0. \end{cases}$$

Текст програми до даної задачі:

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
      float x,y,r,fi,pi=3.141592654;
      cout << "x="; cin >> x;
      cout << "y="; cin >> y;
      r=sqrt(x*x+y*y);
      if (x>0 && y>=0) fi=atan(y/x); else
            if (x==0 && y>0) fi=pi/2; else
                  if (x<0) fi=pi+atan(y/x); else
                        if (x==0 && y<0) fi=3.*pi/2; else
                              if (x>0 && y<0)
                                    fi=2*pi+atan(y/x);
```

```
        cout << "r=" << r << endl;
        cout << "fi=" << fi << endl;
}
```

Константу π ми задали в програмі явно шляхом ініціалізації.

Приклад 6. Скласти програму для визначення коренів квадратного рівняння

$$ax^2 + bx + c = 0.$$

Дискримінант квадратного рівняння обчислюється за формулою

$$D = b^2 - 4ac\,.$$

Далі розглядаються три випадки:

1. Якщо D>0, то рівняння має два розв'язки, які знаходяться за формулами

$$x_1 = \frac{-b - \sqrt{D}}{2a},$$

$$x_2 = \frac{-b + \sqrt{D}}{2a}.$$

2. Якщо D=0, то рівняння має один розв'язок, який знаходиться за формулою

$$x = \frac{-b}{2a}.$$

3. Якщо D<0, то рівняння немає розв'язків.

Таким чином початковими даними задачі є коефіцієнти *a, b, c*. Опишемо їх як дійсні змінні і введемо з клавіатури. Обчислимо *D*. Далі обчислювальний процес може піти по одній із трьох описаних вище віток. Скористаємось оператором **if** у форматі повного розгалуження:

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
      double a,b,c;
      cout << "a="; cin >> a;
      cout << "b="; cin >> b;
      cout << "c="; cin >> c;
      double d=b*b-4*a*c;
      if (d>0)
      {
            double x1=(-b-sqrt(d))/(2*a);
            double x2=(-b+sqrt(d))/(2*a);
            cout << "x1=" << x1 << endl;
            cout << "x2=" << x2 << endl;
      } else
            if (d==0)
            {
                  double x=-b/(2*a);
                  cout << "x=" << x << endl;
            } else
            if (d<0) cout << "nema rozvjazku" << endl;
}
```

У даній програмі ми скористались вкладеними блоками {…} у першій та у другій вітках розгалуження.

Оператор **switch**. Оператор реалізує алгоритмічну структуру – вибір.

```
switch (v)
{
     case c1 : s1;
     case c2 : s2;
     ........
     case cn : sn;
     default s;
}
```

v - цілочисельний вираз,

ci- константний вираз.

Вихід з перемикача, як правило, здійснюється за допомогою **break** або **return**.

Всі константні вирази повинні мати різні значення, але повинні бути одного і того ж цілочисельного типу.

Приклад 7. Скласти програму – калькулятор.

```
#include <iostream>
using namespace std;
int main()
{
     float a,b,res;
     char op;
     cout << " 1-operand:"; cin >> a;
     cout << " operacija:"; cin >> op;
     cout << " 2-operand:"; cin >> b;
     bool f=true;
     switch (op)
     {
     case '+': res=a+b; break;
     case '-': res=a-b; break;
     case '*': res=a*b; break;
     case '/': res=a/b; break;
     default :  cout << "error" << endl; f=false;
     }
     if (f) cout << "result:" << res  << endl;
}
```

Самостійні завдання-2 першого рівня

вар	Завдання	вар	Завдання
1	$y=\begin{cases}\cos(A+C+N), \text{якщо } A=C=N,\\ \cos(A\cdot C\cdot N), \text{якщо } A<C=N,\\ \cos((A+C)\cdot N), \text{якщо } A<C<N,\\ 0, \quad \text{в інших випадках}.\end{cases}$	2	$y=\begin{cases}\ln\lvert x\rvert - n, \text{якщо } x<n,\\ \ln\lvert x\rvert - n, \text{якщо } x=n,\\ \cos(nx), \text{якщо } x>n.\end{cases}$
3	$y=\begin{cases}\cos(x^2+\ln(x)), \text{якщо } x^2+\ln(x)>0\\ \dfrac{1}{(x^2+\ln(x))}, \text{якщо } x^2+\ln(x),\\ \cos(x), \text{якщо } x^2+\ln(x).\end{cases}$	4	$y=\begin{cases}1, \text{якщо } a<b<c,\\ 2, \text{якщо } a=b=c,\\ 3, \text{якщо } b<a<c,\\ 4, \text{якщо } b<a<c,\\ 0, \text{в інших випадках}.\end{cases}$
5	$z=\begin{cases}ye^x, \text{якщо } y<x,\\ y\cdot x, \text{якщо } y=x,\\ x\cdot e^y, \text{якщо } y>x.\end{cases}$	6	$y=\begin{cases}1, \text{якщо } 0\le x<5,\\ 2, \text{якщо } 5\le x<8,\\ 3, \quad \text{якщо } x<0,\\ 4, \quad \text{якщо } x\ge 8.\end{cases}$
7	$y=\begin{cases}1, \text{якщо } 2x^2-x-3=0,\\ 2, \text{якщо } 2x^2-x-3>0,\\ 0, \text{якщо } 2x^2-x-3<0.\end{cases}$	8	$y=\begin{cases}0, \text{якщо } 0\le n<5,\\ 1, \text{якщо } 5\le n<10,\\ 2, \text{якщо } 10\le n<15,\\ 3, \text{в інших випадках}.\end{cases}$
9	$y=\begin{cases}e^t\cdot\sin(t), \text{якщо } t<0\\ e^t\cdot\cos(t), \text{якщо } t\ge 0\end{cases}$ $t=\begin{cases}\cos(x), \text{якщо } x<2,\\ \ln(x), \text{якщо } x\ge 2.\end{cases}$	10	$z=\begin{cases}\ln(x)-tg(x), 1\le x\le 3,\\ tg(x), 3<x\le 4,\\ 0,\end{cases}$
11	$y=\begin{cases}(\cos(x))^n, \text{якщо } n=1,\\ x^n, \quad \text{якщо } n=2,\\ n^x, \quad \text{якщо } n=3,\\ 0, \text{в інших випадках}.\end{cases}$	12	$y=\begin{cases}0, \text{якщо } n=1,2,3,4,\\ 1, \text{якщо } n=10,11,12,\\ 2, \text{якщо } n=15,19,\\ 3, \text{в інших випадках}.\end{cases}$
13	$y=\begin{cases}1, \text{якщо } 2x^2-3=0,\\ 2x, \text{якщо } 2x^2-3>0,\\ x, \text{якщо } 2x^2-3<0.\end{cases}$	14	$z=\begin{cases}(\sin(x))^n, \text{якщо } n=10,\\ x^n, \quad \text{якщо } n=22,\\ n^x, \quad \text{якщо } n=3,\\ 1, \text{в інших випадках}.\end{cases}$
15	$y=\begin{cases}x^3-3x+8, \text{якщо } x^2-3x<-1,\\ \dfrac{1}{x^3-3x+8}, \text{якщо } x^2-3x>1,\\ 0, \quad \text{якщо } -1<x^2-3x<1.\end{cases}$	16	$y=\begin{cases}-4, \quad \text{якщо } x<0,\\ x^2+3x+4, \text{якщо } 0\le x<1,\\ (x^2+3x+4)^2-1, \quad \text{якщо } x\ge 1.\end{cases}$

Самостійні завдання-2 другого рівня

Задача 1. По заданим значенням *x, y, z* обчислити $u = \min(x, \max(y, z))$.

Задача 2. По заданим значенням *x, y, z* обчислити $u = \dfrac{\min\left(\dfrac{x+y+z}{3}, xyz\right)}{1+\min^2\left(\dfrac{x+y+z}{3}, xyz\right)}$.

Задача 3. По заданим значенням *x, y* обчислити $u = \dfrac{\min(x, y)+0.5}{1+\max^2(x, y)}$.

Задача 4. Скласти програму для визначення коренів біквадратного рівняння

$$ax^4 + bx^2 + c = 0.$$

Задача 5. Нехай дано дійсні числа *a, b, c, d.* Виясніть, чи можна прямокутник з сторонами *a, b* вмістити в всередину прямокутника з сторонами *c, d* так, щоб кожна із сторін одного прямокутника була паралельною або перпендикулярною кожній стороні другого прямокутника.

Задача 6. Складіть програму, яка перевіряє, чи пройде цеглина з ребрами *a, b, c* в прямокутний отвір зі сторонами *x, y*. Просовувати цеглину в отвір дозволяється тільки так, щоб кожне з її ребер було паралельне або перпендикулярне кожній із сторін отвору.

Задача 7. Скласти програму, яка по введеному числу з діапазону 1-12, визначає назву місяця року.

Задача 8. Скласти програму, яка по введеному числу з діапазону 1-7, визначає назву дня тижня.

Задача 9. Нехай дане ціле число k, $1 \le k \le 180$. Визначити, яка цифра знаходиться в k-й позиції послідовності 101112131415…9899.

Задача 10*. Скільки спільних точок у прямої $y = kx + b$ і кола $x^2 + y^2 = R^2$?

Задача 11*. Нехай дано дійсні числа $x_1, x_2, x_3, y_1, .y_2, y_3$. Вияснити, чи належить початок координат трикутнику з вершинами $(x_1, y_1), (x_2, y_2), (x_3, y_3)$.

Задача 12*. Нехай дано дійсні числа $x_1, x_2, x_3, y_1, .y_2, y_3$. Вияснити, чи належить точка з координатами (x, y) трикутнику з вершинами $(x_1, y_1), (x_2, y_2), (x_3, y_3)$.

Задача 13*. Вияснити, чи є випуклим чотирикутник з координатами вершин $(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)$.

Задача 14. З клавіатури вводяться три числа. Якщо вони можуть бути довжинами сторін прямокутного трикутника, то вивести їх у порядку зростання.

Задача 15. З клавіатури вводяться три числа. Якщо вони можуть бути довжинами сторін гострокутного трикутника, то вивести їх у порядку зростання.

Задача 16. З клавіатури вводяться три числа. Якщо вони можуть бути довжинами сторін тупокутного трикутника, то вивести їх у порядку зростання.

Задача 17*. Трикутник заданий на координатній площині своїми вершинами $(x_1, y_1), (x_2, y_2), (x_3, y_3)$. Вияснити тип трикутника: прямокутний, гострокутний, тупокутний.

Задача 18*. Скласти програму, яка по введеній даті (місяць і день) визначає знак Зодіаку.

Контрольні завдання

1. У коді програми прикладу 2 замініть оператор **if** на послідовність чотирьох операторів **`if`** у форматі неповного розгалуження.
2. У коді програми прикладу 3 замініть оператор **if** на послідовність двох операторів **`if`** у форматі неповного розгалуження.
3. У коді програми прикладу 7 замініть оператор **`switch`** на послідовність операторів **`if`**.
4. У коді програми прикладу 7 замініть тип змінної **`f`** із **`bool`** на **`int`**.
5. У коді програми завдання 1 замініть оператори введення виведення на відповідні функції у стилі С.

Контрольні питання

1. Яка структура оператора **if** у форматі повного розгалуження?
2. Яка структура оператора **if** у форматі неповного розгалуження?
3. Коли використовуються блоки у операторах **if**?
4. Як конструюються складі логічні умови у операторах **if**?
5. Як можна записати розгалуження на три вітки за допомогою оператора **if**?
6. Поясніть структуру оператора **`switch`.**
7. Як організовується вихід з оператора **`switch`**?
8. Які вимоги ставляться до константних виразів у операторі **`switch`**?
9. Які службові слова використовуються у операторі **`switch`**?
10. Як використовуються у операторі **if** пари символів **`||`** і **`&&`**?

Приклад 1. Обчислити алгебраїчні вирази.

$y = 1 + x + x^2 + x^3 + x^4;$

$y_1 = 1 + 2x + 3x^2 + 4x^3;$

$y_2 = 2 + 6x + 12x^2;$

$y_3 = 6 + 24x;$

Дані для тестування:

$x = 0$	$x = 1$	$x = 2$
$y = 1$	$y = 5$	$y = 31$
$y_1 = 1$	$y_1 = 10$	$y_1 = 49$
$y_2 = 2$	$y_2 = 20$	$y_2 = 62$
$y_3 = 6$	$y_3 = 30$	$y_3 = 54$

Приклад 2. Обчислити алгебраїчні вирази.

$y = (a + b)(a + b) + (a + b + c)(a - b + c);$

$y_1 = \frac{a+b+c}{a+b-c};$

$y_2 = \frac{1}{a} + \frac{1}{b} + \frac{1}{c};$

Дані для тестування підготувати самостійно.

Лабораторна робота № 2

Тема: Програмування циклічних алгоритмів

Мета роботи: Набуття навичок використання мови С++ для програмування арифметичних та ітераційних циклів. Навчитись виводити рекурентні співвідношення.

1 Арифметичні цикли

Теоретичні відомості

Цикл з параметром **for**. Загальний вид даного оператора разом з початковими установками:

```
U;
for ( I; P; M ) O;
```

U - установки;

I - ініціалізація;

P - вираз;

M - модифікація;

O - оператор.

Початкові установки **U** використовуються для задання початкових значень змінним перед входженням в цикл.

Ініціалізація **I** використовується для оголошення і присвоєння початкових значень величинам, що використовуються в циклі. В частині **I** можна записувати кілька операторів розділених комою. Наприклад:

```
for ( int i=0,j=1; ...
int k,m;
for ( k=1,m=1; ...
```

Область дії змінних, оголошених в частині ініціалізації циклу – даний цикл.

Вираз **P** – логічний вираз визначає умову виконання циклу. *true* – цикл виконується, в противному випадку (*false*) цикл не виконується. Цикл з параметром реалізований як цикл з передумовою. Наведемо приклад:

```
for ( int i=1; i<n; ...
```

Тут логічний вираз - **i<n**. Цикл буде виконуватись до тих пір, поки **i** залишається меншим **n**.

Модифікація **M** – виконується після кожної ітерації циклу, і служить для зміни параметра циклу. Доповнимо попередній приклад частиною модифікації:

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

Тут модифікація - вираз **i=i+1**. Він модифікує, тобто змінює **i**. У даному випадку в кожному наступному циклі **i** збільшується на **1**. **i=i+1** можна замінити на операцію інкрементуа **i++**.

В частині модифікації можна записувати кілька операторів через кому. Наприклад:

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

Оператор **O** – це тіло циклу. Якщо тіло циклу це кілька операторів, то вони заключаються у блок **{ ...}**.

Наведемо фрагмент програми обчислення суми чисел від 1 до *n* включно:

```
int s=0;
for ( int i=1; i<=n; i++) s=s+i;
```

У даному фрагменті програми простий оператор присвоєння **s=s+i;** можна замінити на складний оператор присвоєння **s+=i;**.

Будь-яка частина оператора **for** може бути опущена, але **;** потрібно залишити. Так у наведеному фрагменті програми модифікацію, тобто інкремент, можна перенести у оператор. І тоді наведений фрагмент програми буде виглядіти так:

```
int s=0;
for ( int i=1; i<=n; ) s+=i++;
```

Таким чином, до оператора складного присвоєння ”причеплений” інкремент, а частина модифікації відсутня. Побудовану конструкцію **s+=i++** можна перенести на місце модифікації:

```
int s=0;
for ( int i=1; i<=n; s+=i++) ;
```

Тепер у нас відсутній оператор.

Ініціалізацію також можна винести з оператора **for** наперед, і тоді фрагмент програми буде мати вид:

```
int s=0,i=1;
for ( ; i<=n; ) s+=i++;
```

Така зміна не є оптимальною, оскільки опис і ініціалізацію змінної, згідно стилю С++, потрібно по максимуму наближати до місця її використання. Змінна **i** використовується тільки всередині оператора **for**, і тому доцільно її всередині оператора **for** описати та ініціалізувати.

Практикум з програмування

Приклад 1. Обчислити суму чисел від 1 до *n*:

Остаточний текст програми, фрагменти до якої наводились вище:

```
#include <iostream>
using namespace std;
int main()
{
      int n;
      cout << "n="; cin >> n;
      int s=0;
      for (int i=1; i<=n; i++) s+=i;
      cout << " s=" << s << endl;
}
```

Не дивлячись на широкі можливості мови С++, наведений текст програми є найбільш оптимальним. В ньому враховані як можливості мови, так і прийнятий стиль програмування в С++.

Приклад 2. Обчислити n!

По постановці дана задача подібна до попередньої. Тоді для її розв’язання можна запропонувати наступну програму:

```
#include <iostream>
using namespace std;
int main()
{
      int n;
      cout << "n="; cin >> n;
      int fact=1;
      for (int i=1; i<=n;i++) fact*=i;
      cout << "fact=" << fact << endl;
}
```

На відміну від попереднього прикладу тут початкове значення встановлюється рівним 1, (**fact=1;**), а оператор **s+=i;** замінений на **fact*=i;**. Зауважимо, що вся конструкція програми залишилась без змін.

Приклад 3. Обчислити суму $1-2+3-4+5-6+...+n$.

Тут потрібно обчислити суму членів знакозмінного ряду. Для зміни знаку ініціюємо додаткову змінну **z=1**, і на кожному кроці будемо міняти її знак на протилежний **z=-z.** Внісши у програму прикладу 1 вказані зміни, отримаємо наступну програму:

```
#include <iostream>
using namespace std;
int main() {
     int n;
     cout << "n=";cin >> n;
     int s=0;
     for (int i=1,z=1; i<=n; i++,z=-z) s+=z*i;
     cout << "s=" << s << endl;
}
```

Приклад 4. Скласти програму для побудови таблиці відповідності між футами, дюймами та сантиметрами, якщо відповідність між величинами: 1 фут = 12 дюймів, 1 дюйм = 2,54 см,. Початкове значення першої величини, крок зміни цього значення та кількість рядків у таблиці вводяться з клавіатури.

Опишемо і введемо з клавіатури **xp** - початкове значення першої величини, **h** - крок, **n** - кількість рядків. Далі опишемо другу і третю величини, та ініціюємо першу величину початковим її значенням. На завершення організовуємо арифметичний цикл, де параметр циклу міняється від **1** до **n**. В тілі циклу обчислюються друга та третя величина, та всі три величини у виді рядка виводяться на екран.

```
#include <iostream>
using namespace std;
int main()
{
     float xp,h; int n;
     cout << "xp="; cin >> xp;
     cout << "h="; cin >> h;
     cout << "n="; cin >> n;
     float y,z,x=xp;
     for (int i=1;i<=n;i++,x+=h)
     {
          y=2.54*x;
          z=12*y;
          cout << x << ' ' << y << ' ' << z << endl;
     }
}
```

Якщо з клавіатури задати **xp=1,h=1, n=3,** отримаємо на екрані:

```
1 12 30.48
2 24 60.96
3 36 91.44
```

Приклад 5. Протабулювати функцію $y = x\sin x$. на інтервалі [a;b] з кроком h. Знайти серед обчислених значень найбільше, та вказати значення аргументу при якому воно досягається.

Спочатку організуємо просте табулювання функції, без знаходження максимального значення:

```
#include <iostream>
#include <cmath>
using namespace std;
```

```
int main()
{
     float a,b,h;
     cout << "a="; cin >> a;
     cout << "b="; cin >> b;
     cout << "h="; cin >> h;
     for (float x=a,y;x<=b;x+=h)
     {
          y=x*sin(x);
          cout << x << " " << y  << endl;
     }
}
```

Добавимо до даного коду програми оператори для знаходження максимального значення **y**. Перед циклом ініціалізуємо **ym** та **xm**, а в циклі добавимо оператор **if.** Після циклу виведемо **xm** та **ym.** Наведемо новий код програми:

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
     float a,b,h;
     cout << "a="; cin >> a;
     cout << "b="; cin >> b;
     cout << "h="; cin >> h;
     float ym=a*sin(a),xm=a;
     for (float x=a,y;x<=b;x+=h)
     {
          y=x*sin(x);
          cout << x << " " << y  << endl;
          if (y>ym)  ym=y,xm=x;
     }
     cout << "xm=" << xm << " ym=" << ym  << endl;
}
```

Два оператори, які потрібно виконати у операторі розгалуження **if**, ми розділили комою. Це альтернатива блоку **{ym=y;xm=x;}** .

Приклад 6. Скласти програму для знаходження кількості різних трикутників сторони яких дорівнюють цілим числам меншим за число *n*.

Для сторін трикутника введемо цілі змінні **a,b,c** (**a>=b>=c**). Для підрахунку кількості трикутників, що задовольняють умову задачі, ініціюємо змінну **int k=0**. Організуємо потрійний цикл по всім сторонам трикутника, і провірятимемо умову, що більша сторона менша за суму двох інших сторін трикутника. При виконанні умови виконуємо інкремент **k++.** Наведемо текст програми:

```
#include <iostream>
using namespace std;
int main()
{
     int n;
     cout << "n="; cin >> n;
     int k=0;
     for (int a=1;a<=n;a++)
          for (int b=1;b<=a;b++)
```

```
                for (int c=1;c<=b;c++) if (a<b+c) k++;
        cout << "k=" << k << endl;
}
```

Приклад 7. Скласти програму для знаходження кількості різних прямокутних трикутників сторони яких дорівнюють цілим числам меншим за число *n*.

Дана задача подібна до попередньої. Тут провірятимемо умову $a^2 = b^2 + c^2$. Замінимо `if (a<b+c) k++;` на оператор `if (a*a==b*b+c*c) k++;`

Індивідуальні завдання

Завдання 1. Обчислення скінчених сум та добутків.

1. Обчислити суму $1^2 + 2^2 + 3^2 + ... + n^2$.
2. Обчислити суму $1^2 - 2^2 + 3^2 - 4^2 + ... + n^2$.
3. Обчислити суму парних чисел від 1 до *n*.
4. Обчислити суму чисел від 1 до *n*, кратних числу 3.
5. Обчислити суму $\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{n}$.
6. Обчислити суму $\frac{1}{1} - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + ... + \frac{1}{n}$.
7. Обчислити суму $\frac{1}{2} + \frac{1}{4} + \frac{1}{6} + ... + \frac{1}{2n}$.
8. Обчислити суму $\frac{1}{2} - \frac{1}{4} + \frac{1}{6} - \frac{1}{8} + ... + \frac{1}{2n}$.
9. Обчислити добуток парних чисел від 1 до *n*.
10. Обчислити кількість чисел від 1 до *n*, кратних числу 5.
11. Обчислити суму $\frac{1}{2^2} + \frac{1}{4^2} + \frac{1}{6^2} + ... + \frac{1}{(2n)^2}$.
12. Обчислити суму $1 + \frac{1}{2^3} + \frac{1}{3^3} + ... + \frac{1}{n^3}$.
13. Обчислити суму $\frac{1}{2^2} - \frac{1}{4^2} + \frac{1}{6^2} - \frac{1}{8^2} + ... + \frac{1}{(2n)^2}$.
14. Обчислити суму $\frac{1}{3} + \frac{1}{6} + \frac{1}{9} + ... + \frac{1}{3n}$.
15. Обчислити суму $\frac{1}{3} - \frac{1}{6} + \frac{1}{9} - \frac{1}{12} + ... + \frac{1}{3n}$.

Завдання 2. Табулювання функції.

Протабулювати функцію $y = f(x)$ на інтервалі [a;b] з кроком h. Знайти серед обчислених значень найбільше і найменше, а також вказати значення аргументів при яких вони досягаються. Дані для варіантів:

№ варіанта	Функція	інтервал	крок h
1	$y = x^2 \sin x$	[0;1]	0,1
2	$y = \sqrt{x} \cos^2 x$	[0;10]	1
3	$y = \sqrt{x} \cdot tgx$	[0;1]	0,1
4	$y = ctgx / \sqrt{x}$	[1;2]	0,1

5	$y = x^2 e^{-x}$	[0;1]	0,1
6	$y = x^2 \cdot 2^x$	[0;5]	0,5
7	$y = \sqrt{x}\ln\left(1 + x^2\right)$	[0;2]	0,2
8	$y = x\sqrt{x} \cdot e^x$	[0,5;3]	0,25
9	$y = x \cdot arctgx$	[0;6]	0,6
10	$y = \sin x \cdot e^x$	[0;1]	0,1
11	$y = \sqrt{x}\lg x$	[0;10]	1
12	$y = x \cdot \ln^2 x$	[0;2]	0,2
13	$y = x^3 \sin^2 x$	[0;10]	1
14	$y = \sqrt{x}\cos^2 x$	[2;3]	0,1

Контрольні завдання

1. У коді програми прикладу 1 винести ініціалізацію із оператора **for.**
2. У коді програми прикладу 1 винести модифікацію із оператора **for.**
3. Переробити код програми прикладу 1 для знаходження суми парних чисел.
4. У прикладі 2 замінити введення числа **n** з клавіатури ініціалізацією.
5. У прикладі 3 винесіть опис та модифікацію **z** за круглі дужки оператора **for**.
6. У першому коді програми прикладу 5 добавити підказку при виведенні **x** і **y.**
7. У коді програми прикладу 6 добавити у середині внутрішнього циклу оператори виведення на екран значень сторін трикутника.

Контрольні питання

1. Яка структура оператора **for**?
2. Яку алгоритмічну конструкцію реалізує оператор **for**?
3. Скільки разів фігурує параметр циклу в операторі **for**?
4. Куди можна перенести ініціалізацію із оператора **for**?
5. Куди можна перенести модифікацію у операторі **for**?
6. Куди можна перенести оператор у блоці **for**?
7. Яким знаком розділяється кілька операторів у частині ініціалізації?
8. Яким знаком розділяється кілька операторів у частині модифікації?
9. Яким знаком розділяється кілька операторів у тілі циклу оператора **for**?
10. Скільки операторів може бути у тілі циклу оператора **for**?

2 Програмування ітераційних циклів

Теоретичні відомості

По постановці задач цикли поділяються на ***арифметичні*** та ***ітераційні***.

Арифметичні цикли – це цикли кількість повторень у яких наперед відома.

Ітераційні цикли – це цикли кількість повторень у яких наперед невідома, а умова виходу з циклу визначається в процесі його виконання.

Для програмування ітераційних циклів використовуються оператори циклу з перед умовою, та після умовою.

Оператор **`while`** реалізує цикл з перед умовою. Загальний вид даного оператора:

```
while (P) S;
```
(1)

`P` - логічний вираз (логічна умова);

`S` - тіло циклу;

Логічний вираз **P** визначає умову виконання циклу. *true* – цикл виконується, в противному випадку (*false*) цикл не виконується. Наведемо приклад:

```
while (i<=n) ...                                    (2)
```

Тут логічний вираз **i<=n**. Цикл буде виконуватись до тих пір, поки **i** залишається меншим або рівним **n**.

Тіло циклу – це оператор, який виконується при кожному повторенні. Якщо в тілі циклу потрібно виконати кілька операторів, то вони беруться у фігурні дужки. При кожному повторенні повинен змінюватись логічний вираз, інакше відбудеться зациклення. Наведемо приклад обчислення суми цілих чисел від 1 до *n*:

```
int s=0,i=1;                                        (3)
while (i<=n) {s=s+i; i=i+1;}
```

В даному прикладі в тілі циклу **s** збільшується на **i**, а **i** збільшується на **1**.

Реалізовуються дані дії за допомогою двох операторів, які розділені крапкою з комою і взяті у фігурні дужки. У рядку перед циклом виконуються початкові установки, описуються і ініціюються змінні **s** та **i**.

В даному прикладі запис тіла циклу можна скоротити з 14 до 12 символів, розділивши оператори комою і опустивши фігурні дужки.

```
while (i<=n) s=s+i,i=i+1;                           (4)
```

Запис тіла циклу в одному і другому варіанті є наглядним і добре читається. Та у С++ прийнята скорочена форма запису: із складним присвоєнням та інкрементом:

```
while (i<=n) s+=i,i++;                              (5)
```

Кількість символів у тілі циклу ще більше скоротилась: з 12 до 9. Причепимо інкремент до оператора присвоєння і отримаємо ще коротший запис тіла циклу, з 7 символів:

```
while (i<=n) s+=i++;                                (6)
```

Оператор **do while** реалізує цикл з після умовою. Загальний вид даного оператора:

```
do S while (P);                                     (7)
```

Тут спочатку виконується тіло циклу **S**, а далі провіряється логічна умова **P**. Тобто один раз тіло циклу обов'язково виконується.

Наведемо приклад обчислення суми цілих чисел від 1 до *n*. Переробимо фрагмент програми (3), реалізувавши цикл з після умовою:

```
int s=0,i=1;                                        (8)
do {s=s+i; i=i+1;} while (i<=n);
```

Скоротивши тіло циклу, отримаємо:

```
int s=0,i=1;                                        (9)
do s+=i++; while (i<=n);
```

Оператори **while** та **do while** можна використовувати як для програмування ітераційних так і арифметичних циклів.

Практикум з програмування

Приклад 1. Обчислити n!, використавши оператор **while.**

По постановці задачі, це арифметичний цикл. Кількість повторень у ньому наперед відома, вона рівна n. Ми вже обчислювали n! в попередній лабораторній роботі. Там для обчислення ми використали оператор **for**. Програма з використанням оператора **while:**

```
#include <iostream>
using namespace std;
int main()
{
     int n;
     cout << "n="; cin >> n;
```

```
        int f=1,i=1;
        while (i<=n)
        {
              f*=i;
              i++;
        }
        cout << "n!=" << f << endl;
}
```

Приклад 2. Обчислити $y=\sqrt{x}$ за рекурентною формулою $y_n=\left(y_{n-1}+x/y_{n-1}\right)/2$, з точністю e, $\left|y_n-y_{n-1}\right|<\varepsilon$.

Процес обчислення по даній формулі збігається швидко, тобто достатньо кілька ітерацій, щоб отримати потрібну точність. Наведемо результати обчислення по даній формулі при обчисленні $\sqrt{2}$ при початковому значенні y_0=1.

y_0=1
y_1=1,5
y_2=1,416666667
y_3=1,414215686
y_4=1,414213562
y_5=1,414213562

Як бачимо на п’ятому кроці вже стабілізувалось 10 значущих цифр. Тут початкове значення такого ж порядку що і шукане. Якщо початкове значення відрізняється на порядки від точного, то для отримання результату із заданою точністю циклів потрібно виконати трохи більше.

Для розв’язання задачі використаємо цикл **do while.**

```
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;
int main()
{
      double x,eps;
      cout << "x="; cin >> x;
      cout << "eps="; cin >> eps;
      double yp,y=1;
      do
      {
            yp=y;
            y=(yp+x/yp)/2;
      }
      while (abs(y-yp)>=eps);
      cout << "y=" << setprecision(12)<< y << endl;
}
```

Початкове наближення **y** ми задали рівним 1. При реалізації алгоритму індекс у змінної *y* відсутній. Оскільки на кожному кроці циклу присутні тільки два значення y_n - нове та y_{n-1} – старе, то ми відповідно вводимо дві змінні **y** та **yp**. Старе значення поновлюється, а нове обчислюється по формулі.

В даній задачі реалізовується ітераційний цикл, оскільки кількість повторень наперед невідома. Не відомо на якому кроці модуль різниці y_n та y_{n-1} стане меншим за мале наперед задане число ε.

У програмі ми підключили бібліотеку **iomanip** щоб скористатись параметризованим маніпулятором **setprecision(12)**. Для збільшення кількості цифр у результаті до 12.

Приклад 3. Обчислити скінчену суму $S_n = \sum_{i=0}^{n} \frac{x^i}{i!}$.

Виведемо рекурентну формулу. Перепишемо задану формулу у наступному виді

$$S_n = \sum_{i=0}^{n} a_i(x), \qquad (1)$$

де

$a_i(x) = \frac{x^i}{i!}$ –*i*-й член суми. Тоді

$a_{i-1}(x) = \frac{x^{i-1}}{(i-1)!}$ –*i-1* - й член суми.

Поділимо $a_i(x)$ на $a_{i-1}(x)$:

$$\frac{a_i(x)}{a_{i-1}(x)} = \frac{a_i}{a_{i-1}} = \frac{\frac{x^i}{i!}}{\frac{x^{i-1}}{(i-1)!}} = \frac{x^i \cdot (i-1)!}{x^{i-1} \cdot i!} = \frac{x^{i-1} \cdot x \cdot (i-1)!}{x^{i-1} \cdot (i-1)! \cdot i} = \frac{x}{i}.$$

Звідси рекурентна формула:

$$a_i = a_{i-1} \frac{x}{i}. \qquad (2)$$

Початкове значення:

$$a_0 = \frac{x^0}{0!} = 1. \qquad (3)$$

Формула (3) задає найменший член суми, а формула (2) виражає кожний член суми через попередній її член.

З самого початку, в блоці операторів, опишемо і введемо з клавіатури **x** і **n**. Далі задамо початкові значення для **a**, **s**, **i**. Індекс у **a** відсутній. Роль індексу виконує змінна цілого типу **i**, яка супроводжує змінну **a**.

Для організації циклу використаємо оператор **do while**. У тілі циклу **i** кожний раз збільшується на 1, нове значення **a** отримуємо по рекурентній формулі (2), і до **s** додаємо наступний член суми. Цикл виконується до тих пір, поки **i** залишається меншим-рівним за **n**. Останній цикл виконається при **i=n**. Наведемо код програми:

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      double x; int n;
      cout << "x=";cin >> x;
      cout << "n=";cin >> n;
      double a=1,s=a;int i=0;
      do
      {
            i++;
            a*=x/i;
            s+=a;
      }
```

```
        while (i<n);
        cout << "s=" << setprecision(12) << s << endl;
}
```

Дослідимо циклічний процес. Для цього виконаємо обчислення за допомогою даної програми для кількох значень *n* (x=1):

n = 2, *S* =2.5;
n = 4, *S* =2.78333333333;
n = 8, *S* =2.71827876984;
n =14, *S* =2.71828182846;
n =16, *S* =2.71828182846;

Як бачимо, починаючи з *n* =14 результат не міняється. Це означає, що за 14 повторень ми отримали результат з точністю 12 значущих цифр. Наступні повторення не додають до суми нічого, тобто втрачається їх зміст. Тоді як припинити обчислення? Відповідь на питання дає наступний приклад.

Приклад 4. Обчислити нескінчену суму $S = \sum_{i=0}^{\infty} \frac{x^i}{i!}$ з точністю ε.

Дана задача по постановці схожа на попередню. Відрізняється тільки критерієм завершення обчислень: $|a_i(x)| < \varepsilon$, і навпаки, обчислення продовжуються до тих пір, поки $|a_i(x)| > \varepsilon$. Тоді замість опису і введення **n** потрібно описати і ввести **eps**. Логічний вираз у операторі **do while** потрібно замінити на **abs(a)>eps**. Програма, в результаті, стане наступною:

```
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;
int main()
{
      double x,eps;
      cout << "x=";cin >> x;
      cout << "eps=";cin >> eps;
      double a=1,s=a;int i=0;
      do
      {
            i++;
            a*=x/i;
            s+=a;
      }
      while (abs(a)>eps);
      cout << "s=" << setprecision(12) << s << endl;
}
```

Дослідимо як результат залежить від ε:

ε = 0.001 *S*=2.71825396825;
ε = 0.00001 *S*=2.71828152557;
ε = 0.0000001 *S*=2.7182818262;

Приклад 5. Використовуючи метод ділення відрізку пополам, з точністю ε знайдіть найменший додатний корінь рівняння $\sin x = x^2$.

Шукатимемо розв'язок на інтервалі від 0 до π. Позначимо через **xl** - лівий кінець інтервалу в який потрапляє розв'язок рівняння, **xp** – правий, та поточне значення змінної **x**, яке є серединою проміжку [**xl,xp**], ε позначимо через **eps**. Тоді до початку циклу присвоюємо: **xl=0,xp=3.14**. В тілі циклу інтервал [**xl,xp**] поступово звужуватиметься: якщо $\sin x > x^2$, то лівій межі **xl** присвоюється значення **x**, якщо $\sin x < x^2$, то правій межі **xp** присвоюється значення **x.** Повторення продовжуватимуться до тих пір, поки довжина проміжку [**xl,xp**] залишатиметься більшою за **eps**.

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
      double eps;
      cout << "eps=";cin >> eps;
      double xl=0,xp=3.14,x;
      do
      {
            x=(xl+xp)/2;
            if (sin(x)>x*x) xl=x;else xp=x;
      }
      while (xp-xl>eps);
      cout << "x=" << x << endl;
}
```

Індивідуальні завдання

Завдання 1. Обчислення скінченої суми.

Вивести рекурентну формулу. Скласти блок-схему і програму для обчислення суми. Обчислити S_n для n=2; 4; 8; 16; 32; … Дослідити суму на збіжність. Результати дослідження внести в протокол лабораторної роботи.

Варіанти завдань

1. $S_n = \sum_{i=1}^{n} (-1)^{i+1} \frac{x^i}{i}$, $S_n\big|_{n\to\infty} \to S(x) = \ln(1+x)$.
2. $S_n = \sum_{i=0}^{n} \frac{x^i}{i!}$, $S_n\big|_{n\to\infty} \to S(x) = e^x$.
3. $S_n = \sum_{i=0}^{n} (-1)^i \frac{x^{2i+1}}{(2i+1)!}$, $S_n\big|_{n\to\infty} \to S(x) = \sin x$.
4. $S_n = \sum_{i=0}^{n} (-1)^i \frac{x^{2i}}{(2i)!}$, $S_n\big|_{n\to\infty} \to S(x) = \cos x$.
5. $S_n = \sum_{i=0}^{n} \frac{x^{2i+1}}{(2i+1)!}$, $S_n\big|_{n\to\infty} \to S(x) = shx$.
6. $S_n = \sum_{i=0}^{n} \frac{x^{2i}}{(2i)!}$, $S_n\big|_{n\to\infty} \to S(x) = chx$.
7. $S_n = \sum_{i=0}^{n} (-1)^i \frac{x^{2i+1}}{2i+1}, |x| < 1$, $S_n\big|_{n\to\infty} \to S(x) = arctgx$.
8. $S_n = \ln|x| + \sum_{i=1}^{n} \frac{(ax)^i}{i \cdot i!}$, $S_n\big|_{n\to\infty} \to S(a,x) = \int \frac{e^{ax} dx}{x}$

9. $S_n = \sum_{i=0}^{n} (-1)^i \frac{\left(\frac{1}{2}x\right)^{2i}}{(i!)^2}$, $S_n\big|_{n\to\infty} \to J_0(x); J_0(1) = 0{,}7651976865$.

10. $S_n = \sum_{i=0}^{n} (-1)^i \frac{\left(\frac{1}{2}x\right)^{2i+1}}{(i!)^2 \cdot (i+1)}$, $S_n\big|_{n\to\infty} \to J_1(x); J_1(1) = 0{,}4400505857$.

11. $S_n = \sum_{i=0}^{n} (-1)^i \frac{\left(\frac{1}{2}x\right)^{2i+2}}{2^{2i+2} i!\cdot(i+2)!}$, $S_n\big|_{n\to\infty} \to J_2(x); J_2(1) = 0{,}1149034849$.

12. $S_n = \sum_{i=1}^{n} \frac{1}{i^p}$, $S_n\big|_{n\to\infty} \to \varsigma(p); \varsigma(2) = 1{,}6449340668$.

13. $S_n = 2 \cdot \sum_{i=0}^{n} \frac{x^{2i+1}}{2i+1}$, $x^2 < 1$, $S_n\big|_{n\to\infty} \to S(x) = \ln\left(\frac{1+x}{1-x}\right)$.

14. $S_n = 2 \cdot \sum_{i=0}^{n} \frac{1}{x^{2i+1}(2i+1)}$, $x^2 > 1$, $S_n\big|_{n\to\infty} \to S(x) = \ln\left(\frac{x+1}{x-1}\right)$.

15. $S_n = \sum_{i=1}^{n} \frac{1}{i}\left(\frac{x-1}{x}\right)^i$, $x > \frac{1}{2}$,, $S_n\big|_{n\to\infty} \to S(x) = \ln x$.

Завдання 2. Обчислення нескінченої суми.

Постановка індивідуальних завдань:

1. Скористатись рекурентною формулою з попереднього завдання.
2. Скласти блок-схему і програму для обчислення нескінченого ряду з точністю ε.
3. Обчислити S для ε=0.1; 0.01; 0.001; 0.0001; 0.00001; 0.000001. Дослідити ряд на збіжність і переконатись у правильності отриманого результату.
4. Результати дослідження внести в протокол лабораторної роботи.

Варіанти завдань

1. $S(x) = \sum_{i=1}^{\infty} (-1)^{i+1} \frac{x^i}{i} = \ln(1+x)$.
2. $S(x) = \sum_{i=0}^{\infty} \frac{x^i}{i!} = e^x$.
3. $S(x) = \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i+1}}{(2i+1)!} = \sin x$.
4. $S(x) = \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i}}{(2i)!} = \cos x$.
5. $S(x) = \sum_{i=0}^{\infty} \frac{x^{2i+1}}{(2i+1)!} = shx$.
6. $S(x) = \sum_{i=0}^{\infty} \frac{x^{2i}}{(2i)!} = chx$.
7. $S(x) = \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i+1}}{2i+1} = arctgx$, $|x| < 1$.

8. $S(a,x)=\ln|x|+\sum_{i=1}^{\infty}\frac{(ax)^i}{i\cdot i!}$, $\qquad S_n\big|_{n\to\infty}\to S(a,x)=\int\frac{e^{ax}dx}{x}$

9. $S(x)=\sum_{i=0}^{\infty}(-1)^i\frac{\left(\frac{1}{2}x\right)^{2i}}{(i!)^2}=J_0(x),$, $\qquad J_0(1)=0{,}7651976865$.

10. $S(x)=\sum_{i=0}^{\infty}(-1)^i\frac{\left(\frac{1}{2}x\right)^{2i+1}}{(i!)^2\cdot(i+1)}=J_1(x),$ $\qquad J_1(1)=0{,}4400505857$.

11. $S(x)=\sum_{i=0}^{\infty}(-1)^i\frac{\left(\frac{1}{2}x\right)^{2i+2}}{2^{2i+2}i!\cdot(i+2)!}=J_2(x),$ $\qquad J_2(1)=0{,}1149034849$.

12. $S(x)=\sum_{i=1}^{\infty}\frac{1}{i^p}=\varsigma(p),$ $\qquad \varsigma(2)=1{,}6449340668$.

13. $S(x)=2\cdot\sum_{i=0}^{\infty}\frac{x^{2i+1}}{2i+1}=\ln\left(\frac{1+x}{1-x}\right),\ x^2<1$.

14. $S(x)=2\cdot\sum_{i=0}^{\infty}\frac{1}{x^{2i+1}(2i+1)}=\ln\left(\frac{x+1}{x-1}\right),\ x^2>1.$

15. $S(x)=\sum_{i=1}^{\infty}\frac{1}{i}\left(\frac{x-1}{x}\right)^i=\ln x,\ x>\frac{1}{2},$

Контрольні завдання

1. У коді програми прикладу 1 замініть оператор **while** на **do while.**
2. У коді програми прикладу 1 скоротіть тіло циклу до одного оператора.
3. У коді програми прикладу 2 замініть оператор **do while** на **while.**
4. У коді програми прикладу 3 замініть оператор **do while** на **while.**
5. У коді програми прикладу 4 зменшити кількість операторів у тілі циклу до 2.
6. У коді програми прикладу 4 зменшити кількість цифр при виводі *S* до 10.
7. Добавити у коді програми прикладу 5 маніпулятор **`setprecision`**.

Контрольні питання

1. Яка структура оператора **while**?
2. Яка структура оператора **do while**?
3. Яким знаком розділяється кілька операторів у тілі циклу?
4. Яким оператором реалізується цикл з перед умовою?
5. Яким оператором реалізується цикл з після умовою?
6. Чому цикл **while** більш універсальний ніж цикл **do while**?
7. Для чого використовується бібліотека **`iomanip`**?
8. Для чого використовується бібліотека **`iostream`**?
9. Для чого використовується параметризований маніпулятор **`setprecision`**?
10. В чому доцільність використання рекурентних формул?

Лабораторна робота № 3

Тема: Масиви.

Мета роботи: Набуття навичок використання мови С++ для обробки масивів. Навчитись використовувати статичні та динамічні масиви.

Статичні масиви

Масив – скінчена послідовність однакових за типом елементів.

Опис 1-вимірного масиву: `t im[k];`

`t` - тип елементів масиву,

`im` – ім'я масиву,

`k` - кількість елементів масиву,

`[]` - елементи синтаксису С++.

При описі масиву можна використовувати ті ж самі модифікатори, що і для простих змінних:

- клас пам'яті;
- const;
- ініціалізатор.

Ініціалізуючі змінні для елементів масиву записуються у {}.

Приклад. `int a[5]={3,0,-1};`

`a[0]=3;  a[1]=0;  a[2]=-1;  a[3]=0;  a[4]=0;`

Якщо елементів масиву більше ніж ініціалізаторів, то елементи, значення яких не визначені, зануляються.

Розмір масиву разом з типом елементів визначає об'єм пам'яті, необхідну для розміщення масиву. У наведеному вище прикладі під масив **`a`** буде виділено 20 байт. Розмір пам'яті, що виділяється під масив можна визначити за допомогою операції **`sizeof`**:

```
int a[5]={3,0,-1};
cout << "sizeof(a)=" << sizeof(a)<< endl;
```

Якщо при описі масиву не вказана розмірність, то повинен бути ініціалізатор. Тоді компілятор виділить пам'ять по кількості ініціалізованих елементів.

Приклад. `int b[]={1,-1,1,-1,0,1};`

Під масив **`b`** буде виділено 24 байти.

Для доступу до елементів масиву після його імені вказується номер елемента в [].

Приклад. `a[1]=13;`

Елементу масиву з індексом 1 присвоюється значення 13.

Розмірність масиву доцільно задавати за допомогою іменованих констант.

Приклад. `int const n=4;` або `const int n=4;`

Практикум з програмування

Приклад 1. Знайти суму елементів масиву. Масив ініціалізується.

```
#include <iostream>
using namespace std;
int main()
{
     int a[]={2,4,6,8,10};
     int s=0;
     for (int i=0;i<5;i++) s+=a[i];//s+=*(a+i);
     cout << "s=" << s << endl;
```

```
}
```

Тут ініціалізується 5 елементів масиву, індекси яких міняються від 0 до 4. Умова виходу з циклу теж задається явно **i<5**.

Ідентифікатор масиву є константним покажчиком на його нульовий елемент. Тому фрагмент програми **s+=a[i];** можна замінити на **s+=*(a+i);**.

Приклад 2. Знайти максимальний елемент масиву. Масив ініціалізується.

```
#include <iostream>
using namespace std;
int main()
{
     int a[]={2,4,16,8,10,0,1};
     int const n=7;
     int amax=a[0];
     for (int i=1;i<n;i++)
          if (a[i]>amax) amax=a[i];
     cout << "amax=" << amax << endl;
}
```

Тут ініціалізується 7 елементів масиву, індекси яких міняються від 0 до 6, і описується іменована константа **const n=7.** Умова виходу з циклу задається через іменовану константу **i<n**.

Допустимо спочатку, що максимальний елемент масиву **amax** рівний його нульовому елементу **a[0]** . Далі порівнюватимемо всі елементи масиву, починаючи з 1-го, з **amax** і якщо **a[i]** виявлятиметься більшим за **amax** то присвоюватимемо **amax** нове значення **amax=a[i]**.

Приклад 3. Знайти індекс максимального елементу масиву. Масив ініціалізується.

```
#include <iostream>
using namespace std;
int main()
{
     int a[]={2,4,16,8,10};
     int const n=5;
     int imax=0;
     for (int i=0;i<n;i++)
          if (a[i]>a[imax]) imax=i;
     cout << "imax=" << imax << endl;
}
```

Приклад 4. Поміняти місцями елементи масиву з індексами *k* та *l*. Масив ініціалізується. Вивести на екран перетворений масив.

```
#include <iostream>
using namespace std;
int main()
{
     int a[]={2,4,16,8,1};
     int const n=5,k=1,l=3;
     int buf=a[k];
     a[k]=a[l];
     a[l]=buf;
     for (int i=0;i<n;i++)
          cout << a[i] << ' ';
     cout << endl;
```

```
}
```

Приклад 5. Поміняти місцями мінімальний та максимальний елементи масиву. Масив ініціалізується. Вивести на екран перетворений масив.

Спочатку знайдемо індекси максимального **imax** та індекс мінімального **imin** елементів (див. Приклад 3). Далі обміняємо елементи із знайденими індексами (див. Приклад 4).

```
#include <iostream>
using namespace std;
int main()
{
     int a[]={2,4,16,8,1};
     int const n=5;
     int imax=0,imin=0;
     for (int i=0;i<n;i++)
     {
          if (a[i]>a[imax]) imax=i;
          if (a[i]<a[imin]) imin=i;
     }
     int buf=a[imin];
     a[imin]=a[imax];
     a[imax]=buf;
     for (int i=0;i<n;i++)
          cout << a[i] << ' ';
     cout << endl;
}
```

Приклад 6. Циклічно зсунути елементи масиву на один елемент вліво. Масив ініціалізується. Вивести на екран перетворений масив.

Запам’ятаємо нульовий елемент масиву (запис у буфер **buf=a[0];**). Далі на його місце запишемо перший, на місце першого другий і так далі, поки останній не пропишеться на місце передостаннього. На місце останнього запишемо нульовий елемент, значення якого знаходиться у буфері (витягування із буфера **a[n-1]=buf;**). Останній елемент має номер **n-1**, передостанній **n-2**.

```
#include <iostream>
using namespace std;
int main()
{
     int a[]={2,4,16,8,1};
     int const n=5;
     int buf=a[0];
     for (int i=1;i<n;i++) a[i-1]=a[i];
     a[n-1]=buf;
     for (int i=0;i<n;i++)
          cout << a[i] << ' ';
     cout << endl;
}
```

Приклад 7. Циклічно зсунути елементи масиву на один елемент вправо. Масив ініціалізується. Вивести на екран перетворений масив.

Запам’ятаємо останній елемент масиву (запис у буфер **buf=a[n-1];**). Далі на його місце запишемо передостанній, на місце передостаннього перед передостанній і так

далі, поки нульовий не пропишеться на місце першого. Останній елемент має номер **n-1**, передостанній **n-2**, а перед передостанній номер **n-3**.

```
#include <iostream>
using namespace std;
int main()
{
	int a[]={2,4,16,8,1};
	int const n=5;
	int buf=a[n-1];
	for (int i=n;i>0;i--) a[i-1]=a[i-2];
	a[0]=buf;
	for (int i=0;i<n;i++)
		cout << a[i] << ' ';
	cout << endl;
}
```

Приклад 8. Циклічно зсунути елементи масиву на *k* елементів вправо. Масив ініціалізується. Вивести на екран перетворений масив.

Повторимо у циклі **for** операцію зсуву *k* разів. За основу візьмемо алгоритм із попереднього прикладу.

```
#include <iostream>
using namespace std;
int main()
{
	int a[]={2,4,16,8,1};
	int const n=5,k=2;
	for (int j=1;j<=k;j++)
	{
		int buf=a[n-1];
		for (int i=n;i>=1;i--) a[i-1]=a[i-2];
		a[0]=buf;
	}
	for (int i=0;i<n;i++)
		cout << a[i] << ' ';
	cout << endl;
}
```

Приклад 9. Зсунути ненульові елементи масиву вліво, не міняючи їхній порядок слідування. Масив ініціалізується. Вивести на екран перетворений масив.

```
#include <iostream>
using namespace std;
int main()  {
	int a[]={0,2,1,0,16,0,0,0,3,13,-1},n=10;
	int k;
	do
	{
		k=0;
		for (int i=0;i<n;i++)
			if (a[i]==0 && a[i+1]!=0)
			{
				a[i]=a[i+1];
				a[i+1]=0;
				k++;
```

```
                }
        }
        while (k>0);
        for (int i=0;i<=n;i++)
             cout << a[i] << ' ';
        cout << endl;
}
```

Приклад 10. Сортування масиву методом бульбашки**.** Розмістити елементи масиву в порядку зростання. Масив ініціалізується. Вивести на екран перетворений масив.

```
#include <iostream>
using namespace std;
int main()
{
     int a[]={0,2,1,0,16,0,5,-2,3,13,-1},n=10;
     int k;
     do
     {
          k=0;
          for (int i=0;i<n;i++)
               if (a[i]>a[i+1])
               {
                    int buf=a[i];
                    a[i]=a[i+1];
                    a[i+1]=buf;
                    k++;
               }
     }
     while (k>0);
     for (int i=0;i<=n;i++)
          cout << a[i] << ' ';
     cout << endl;
}
```

Замінимо в даній програмі ініціалізацію масиву, введенням масиву з клавіатури. При описі масиву задамо його розмірність із запасом **a[100]**. Далі опишемо і введемо з клавіатури **n** – реальну кількість елементів масиву. Далі організовуємо цикл **for** для введення масиву з клавіатури. Виведення підсказки і введення елементу масиву складаються з двох операторів заключених у блок.

```
#include <iostream>
using namespace std;
int main()
{
     int a[100],n;
     cout << "n="; cin >> n;
     for (int i=0;i<=n;i++)
     {
          cout << "a[" << i << "]=";
          cin >> a[i];
     }
     int k;
     do
```

```
        {
            k=0;
            for (int i=0;i<n;i++)
                if (a[i]>a[i+1])
                {
                    int buf=a[i];
                    a[i]=a[i+1];
                    a[i+1]=buf;
                    k++;
                }
        }
        while (k>0);
        for (int i=0;i<=n;i++)
            cout << a[i] << ' ';
        cout << endl;
    }
```

Динамічні масиви

Динамічні масиви створюють за допомогою операції **new** .

Приклад.
```
int n=100;
float *p = new float[n];
```

Тут створюється змінна-покажчик на **float**. В динамічній пам’яті виділяється неперервна область, достатня для розміщення 100 елементів дійсного типу, а адресе її початку записується в покажчик **p**.

Динамічні масиви не можна при створенні ініціалізувати, і вони не зануляються.

Перевага динамічного масиву: об’єм пам’яті під масив визначається на етапі виконання програми.

Доступ до елементів динамічного масиву такий самий, як і до елементів статичного масиву.

Приклад. **p[5]** або ***(p+5)**

Пам'ять зарезервована за допомогою оператора **new** звільняється за допомогою оператора **delete [].**

Розмір масиву в **delete** не вказується, але дужки **[]** обов'язкові.

Коли потрібно звільнити пам'ять від динамічного масиву?

Якщо масив в якийсь момент роботи програми перестав бути потрібним і ми збираємось використати цю пам'ять повторно.

Приклад 11. Знайти суму додатних елементів масиву. Пам’ять під масив виділити динамічно.

```
#include <iostream>
using namespace std;
int main()
{
    int n;
    cout << "n="; cin >> n;
    int *a = new int[n];
    for (int i=0;i<=n;i++)
    {
        cout << "a[" << i << "]=";
        cin >> a[i];
    }
    int s=0;
```

```
        for (int i=0;i<=n;i++) if (a[i]>0) s+=a[i];
        cout << "s=" << s << endl;
        delete [] a;
}
```

Індивідуальні завдання

Завдання 1. Статичний масив. Написати програму згідно отриманого варіанту використовуючи статичний масив. Всі пункти завдання реалізувати в одній програмі.

Завдання 2. Динамічний масив. Переробити програму завдання 1, використавши динамічний масив.

Варіанти завдань

1. Дано одновимірний масив $a=(a_1,a_2,a_3,...,a_n)$. Поміняти місцями максимальний і мінімальний елементи масиву. Вивести на екран перетворений масив.
2. Дано одновимірний масив $a=(a_1,a_2,a_3,...,a_n)$. Поміняти місцями початковий та останній елементи масиву. Вивести на екран перетворений масив.
3. Дано вектор $a=(a_1,a_2,a_3,...,a_n)$. Обчислити модуль вектора $|a|=\sqrt{a_1^2+a_2^2+a_3^2+...+a_n^2}$. Кожний елемент вектора *a* розділити на $|a|$. Вивести на екран перетворений масив *a*.
4. Дано одновимірний масив $a=(a_1,a_2,a_3,...,a_n)$. Від'ємні елементи масиву замінити на –1, додатні на 1. Вивести на екран перетворений масив.
5. Дано одновимірний масив $a=(a_1,a_2,a_3,...,a_n)$. Знайти суму додатних S_+, та суму від'ємних S_- елементи масиву. Додатні елементи масиву розділити на S_+, а від'ємні на S_-. Вивести на екран перетворений масив.
6. Дано одновимірний масив $a=(a_1,a_2,a_3,...,a_n)$. Знайти кількість додатних k_+, та кількість від'ємних k_- елементи масиву. Додатні елементи масиву помножити на k_+, а від'ємні на k_-. Вивести на екран перетворений масив.
7. Дано одновимірний масив $a=(a_1,a_2,a_3,...,a_n)$. Елементи масиву індекси яких кратні числу 3 замінити нулями. Вивести на екран перетворений масив.
8. Дано одновимірний масив $a=(a_1,a_2,a_3,...,a_n)$. Елементи масиву, індекси яких при діленні на 3 дають в остачі 2, замінити нулями. Вивести на екран перетворений масив.
9. Дано одновимірні масиви $a=(a_1,a_2,a_3,...,a_n)$ та $b=(b_1,b_2,b_3,...,b_n)$. Від'ємні елементи масиву *a* замінити відповідними елементами масиву *b*. Вивести на екран перетворений масив *a*.
10. Дано вектори $a=(a_1,a_2,a_3,...,a_n)$ та $b=(b_1,b_2,b_3,...,b_n)$. Знайти їх скалярний добуток. Розділити кожний елемент масиву *a* на отриманий добуток. Вивести на екран перетворений масив *a*.
11. Дано вектори $a=(a_1,a_2,a_3,...,a_n)$ та $b=(b_1,b_2,b_3,...,b_n)$. Знайти суму векторів. Вивести на екран отриманий вектор.
12. Дано вектори $a=(a_1,a_2,a_3,...,a_n)$ та $b=(b_1,b_2,b_3,...,b_n)$. Знайти різницю векторів. Вивести на екран отриманий вектор.
13. Дано одновимірний масив $a=(a_1,a_2,a_3,...,a_n)$. Циклічно зсунути елементи першої половини масиву на одну позицію вліво. Вивести на екран перетворений масив.

14. Дано одновимірний масив $a=(a_1,a_2,a_3,...,a_n)$. Циклічно зсунути елементи першої половини масиву на одну позицію вправо. Вивести на екран перетворений масив.
15. Дано одновимірний масив $a=(a_1,a_2,a_3,...,a_n)$. Циклічно зсунути елементи другої половини масиву на одну позицію вліво. Вивести на екран перетворений масив.
16. Дано одновимірний масив $a=(a_1,a_2,a_3,...,a_n)$. Циклічно зсунути елементи другої половини масиву на одну позицію вправо. Вивести на екран перетворений масив.

Контрольні завдання

1. У коді програми прикладу 1 замініть ініціалізацію масиву на введення масиву з клавіатури.
2. У коді програми прикладу 2 замініть ініціалізацію масиву на введення масиву з клавіатури.
3. У коді програми прикладу 4 замініть масив із цілих чисел на масив із дійсних чисел.
4. У коді програми прикладу 6 кількість елементів масиву збільшити до 10, і циклічно зсунути першу половину масиву.

Контрольні питання

1. Як описується одновимірний статичний масив?
2. Як ініціалізується одновимірний масив?
3. Як здійснюється доступ до елементів масиву?
4. Як заборонити зміну ініціалізованих елементів масиву?
5. Чи може елементів масиву бути більше ніж ініціалізаторів?
6. Як створюється одновимірний динамічний масив?
7. Які переваги має динамічний масив?
8. Як звільняється пам'ять виділена під динамічний масив?
9. Як визначити розмір пам'яті виділену під масив?
10. Коли і як використовується буфер у алгоритмах з масивами?

Лабораторна робота № 4

Тема: Функції.

Мета роботи: Познайомитись з організацією програмування з описом та використанням функцій в С++.

Оголошення, виклик та опис функції

Функція – це іменована послідовність описів і операторів, яка виконує певні завершені дії.

Функція може приймати параметри і повертати значення.

Будь-яка програма на С++ складається з функцій, одна з яких повинна мати назву main.

Функція починає виконуватись з моменту її виклику.

Будь-яка функція повинна бути *оголошена* і *визначена*. Як і для інших об'єктів, оголошень може бути кілька, а визначення тільки одне.

Оголошення функції в тексті програми повинно знаходитись раніше від її викликів.

Оголошення функції задає:

- Ім'я функції;
- Тип повернутого значення;
- Список переданих параметрів.

Визначення функції. Крім оголошення у визначенні добавляється тіло функції, яке складається з описів і операторів.

```
Тип ім'я ( [список параметрів] )
      { тіло функції };
```

Тип – це тип повернутого функцією значення. Може бути будь-яким крім масиву і функції, але може бути покажчиком на масив або функцію. Якщо функція не повинна повертати значення, то вказується тип **`void`**.

Список параметрів визначає величини які потрібно передати у функцію при її виклику.

Елементи списку розділяються комами.

Для кожного параметру, що передається у функцію, вказується його тип і ім'я. У оголошенні імена можна опускати.

При оголошенні, опису і виклику типи і порядок слідування параметрів повинні співпадати.

Тут наведений простіший варіант визначення функції**.**

Глобальні параметри. Глобальні параметри видимі у всіх функціях, у яких не описані локальні змінні з такими ж самими іменами.

Не рекомендується використовувати глобальні параметри для передачі даних між функціями. Потрібно старатись, щоб функції були максимально незалежними.

Повернуте значення. Механізм повернення – за допомогою оператора:

```
return [вираз];
```

У функції може бути кілька операторів **return**. Якщо функція описана як **void**, то **return** опускається.

Вираз вказаний після **return**, неявно перетворюється до типу повернутого функцією значення, і передається в місце виклику функції.

Приклад 1. Функція обчислює суму двох чисел.

```
#include <iostream>
using namespace std;
int suma(int , int );          //Оголошення функції
int main()
{
     int a=3, b=4;
     int c=suma(a,b);          //Виклик функції
     cout << a << '+' << b
          << '=' << c << endl;
}
int suma(int a, int b)   //Визначення функції
{
     int y=a+b;
     return y;
}
```

Параметри функції

Механізм параметрів є основним способом обміну інформацією між функціями, яка викликає і яку викликають.

Формальні параметри - параметри перераховані в заголовку опису функції.

Фактичні параметри – записані в операторі виклику функції.

При виклику функції в першу чергу обчислюються вирази, які знаходяться на місці аргументів (частіше всього на місці аргументів знаходяться змінні, значення яких уже відомі); потім в стеку виділяється пам'ять під формальні параметри функції у відповідності з їх типом, і кожному з них присвоюється значення відповідного аргументу. При цьому провіряється відповідність типів і при необхідності виконується їх перетворення. При невідповідності типів видається діагностичне повідомлення.

В С++ є два способи передачі параметрів у функцію:

- по значенню;
- за адресою.

Передача по значенню: у стек записуються копії значень аргументів і оператори функції працюють із цими копіями. Доступ до оригіналів у функції нема, а отже вона і не може їх змінити.

Передача за адресою: у стек заносяться копії адрес аргументів, а функція здійснює доступ до комірок пам'яті за цими адресами і тим самим може міняти початкові значення аргументів.

Приклад 2. Три різні способи передачі параметрів у функцію.

У функції **main** ініціалізуються три змінні **int i=1, j=2, k=3;**, і їх значення виводяться на екран. Далі, різними способами вони передаються у функцію **f**. У функції над переданими значеннями виконується інкремент, і знову, після виклику функції **f**, їх значення виводяться на екран.

```
#include <iostream>
using namespace std;
void f( int, int *, int &);
int main()
{
     int i=1, j=2, k=3;
     cout << i << ' ' << j << ' ' << k << endl;
     f(i,&j,k);
     cout << i << ' ' << j << ' ' << k << endl;
}
void f( int i, int *j, int &k)
{
     i++;
     (*j)++;
     k++;
}
```

Отримаємо на екрані:

```
1 2 3
1 3 4
```

Як бачимо, першу змінну функція не змінила, а другу і третю змінила.

i - передається по значенню. Його зміна у функції не впливає на початкове значення.

j - передається за адресою за допомогою покажчика. Для цього для передачі у функцію адреси фактичного параметра використовується операція взяття адреси, а для отримання його значення у функції потрібно виконати операцію розіменування.

k - передається за адресою за допомогою посилання.

Передача за посиланням

У функцію передається адреса вказаного при виклику параметра, а в середині функції всі звернення до параметра неявно розіменовуються.

Використання посилань замість покажчиків покращує читаність програми, звільняючи її від операцій отримання адрес і розіменувань.

Використання посилань замість передачі по значенню більш ефективне, оскільки не потребує копіювання параметрів, що має значення при передачі структур даних великого об'єму.

Якщо потрібно заборонити зміну параметрів всередині функції, то використовується модифікатор **const**.

Наприклад: **int f(const int *m);**

Рекомендується вказувати **const** перед усіма параметрами, зміна яких у функції не передбачається.

Передача масивів у ролі параметрів

При використанні в ролі параметрів масиву у функцію передається покажчик на його перший елемент, тобто масив завжди передається за адресою. При цьому інформація про кількість елементів масиву втрачається і слід передавати його розмір через окремий параметр.

При передачі багатомірних масивів всі розміри, якщо вони не відомі на етапі компіляції, повинні передаватись у ролі параметрів. Всередині функції масив інтерпретується як одномірний, а його індекс перераховується.

Приклад 4. Циклічно зсунути елементи одномірного масиву на k позицій вліво. Використати динамічні масиви та покажчики.

```
#include <iostream> //   циклічний зсув
using namespace std;     //   одномірного масиву
void zsuvl(int n, int  *m);
void zsuvl_k(int n, int k, int  *m);
void cinm(int n, int *m);
void coutm(int n, int *m);
int main()
{
     int n; cout << "n="; cin >> n; //розмір мас.
     int *m = new int [n];
     cinm(n,m);
     int k; cout << "k=";
     cin >> k;                          //к-ть позицій
     coutm(n,m);                        //   зсуву
```

```
     zsuvl_k(n,k,m);
     coutm(n,m);
}
void zsuvl(int n, int  *m)    //   ф-я цикл. зсуву
{                             //на одну позицію вліво
     int buf=*m;
     for (int *p=m+1;p<m+n;p++) *(p-1)=*p;
     *(m+n-1)=buf;
}
void zsuvl_k(int n, int k, int  *m)     //ф-я цикл.
{                                  //зсуву вліво
     for (int i=0;i<k;i++)         // на k позицій
          zsuvl(n,m);
}
void cinm(int n, int *m) //   ф-я введення мас.
{
     for (int *p=m;p<m+n;p++)
     {
          cout << "a[" << p-m << "]=";
          cin >> *p;
     }
}
void coutm(int n, int *m)     //   ф-я виведення мас.
{
     for (int *p=m;p<m+n;p++)
          cout <<  *p << ' ';
     cout << endl;
}
```

Короткі підсумки

1. Функція – це іменована послідовність операторів, яка виконує завершені дії.
2. Функції потрібні для спрощення структури програми.
3. Інтерфейс грамотно написаної функції визначається її заголовком.
4. Для виклику функції потрібно вказати її ім’я і набір аргументів.
5. При визначенні, оголошенні і при виклику функції типи і порядок слідування аргументів і параметрів повинні співпадати.
6. Передача параметрів у функцію може виконуватись по значенню або по адресі.

Самостійні завдання

1. Функції і. Повернення значення через return

Виконати завдання 1 лабораторної роботи № 1 „Програмування лінійних та розгалужених алгоритмів ”, оформивши обчислення **z1** і **z2** у виді функцій. Всі необхідні дані для функцій повинні передаватись їм в ролі параметрів. Використання глобальних параметрів у функціях не допускається.

2. Функції і масиви

Виконати завдання лабораторної роботи № 3 „Масиви”.

Оголосити, описати та викликати наступні функції:

1. Функція введення масиву з клавіатури.
2. Функція виведення масиву на екран.
3. Функція перетворення масиву.

Всі необхідні дані для функцій повинні передаватись їм в ролі параметрів. Використання глобальних параметрів у функціях не допускається.

Контрольні питання

1. Для чого у програмах потрібні функції?
2. Яка різниця між оголошенням і визначенням функції?
3. Чим відрізняється функція від блоку операторів?
4. Що визначає тип функції?
5. Де у коді програми повинні співпадати типи і порядок слідування параметрів?
6. Скільки значень можна повернути із функції за допомогою **`return`**?
7. Скільки може бути операторів **`return`** у функції?
8. Скільки разів можна викликати функцію.
9. Які є механізми передачі із функції кількох значень?

Лабораторна робота № 5

Тема: Двовимірні масиви.

Мета роботи: Набуття навичок використання мови С++ для обробки двовимірних масивів. Навчитись використовувати статичні та динамічні масиви. Познайомитись із технікою використання покажчиків у двовимірних масивах.

Статичні масиви

Опис 2-вимірного масиву: `t im[nr][ns];`

t - тип елементів масиву,

im – ім'я масиву,

nr - кількість рядків у масиві,

ns - кількість стовпців у масиві.

Приклад. `int a[3][4];`

У пам'яті елементи 2-х мірного масиву розміщаються в послідовних комірках по рядках.

Елементи багатомірного масиву розміщаються так, що при переході до наступного елемента швидше міняється останній індекс.

Для доступу до елементів багатомірного масиву вказуються всі його індекси:

```
a[i][k]
```

Ініціалізація двовимірного масиву:

Приклад.
```
int a[2][2]={{1,2},{3,4}};
int a[2][2]={1,2,3,4};
int a[ ][2]={{1,2},{3,4}};
```

При ініціалізації багатомірного масиву він представляється:

1. Як масив із масивів. При цьому кожний масив заключається в свої **{}**, і тоді лівий розмір (кількість рядків) можна не вказувати.
2. задається загальний список в такому порядку, в якому елементи розташовуються в пам'яті.

Практикум з програмування

Приклад 1. У ініціалізованому масиві з трьох рядків і трьох стовпців замінити другий рядок на нулі, а третій стовпець на одиниці. Вивести на екран перетворений масив у виді таблиці.

```
#include <iostream>
using namespace std;
```

```
int main()
{
     int a[3][3]={
          {1,2,3},
          {4,5,6},
          {7,8,9}};
     for (int i=0;i<3;i++) a[1][i]=0;
     for (int i=0;i<3;i++) a[i][2]=1;
     for (int i=0;i<3;i++)
     {
          cout << endl;
          for (int j=0;j<3;j++) cout << a[i][j] << ' ';
     }
     cout << endl;
}
```

Після запуску програми на екрані отримаємо:

```
1 2 1
0 0 1
7 8 1
```

Тут ініціалізується 3*3 елементів масиву, індекси яких міняються від 0 до 2.

У першому циклі другий рядок, тобто послідовність елементів з першим індексом рівним 1, міняються на 0.

У другому циклі третій стовпець, тобто послідовність елементів з другим індексом рівним 2, міняються на 1.

У наступному складеному циклі організовується виведення масиву. Зовнішній цикл по **i** – це цикл по рядкам. Внутрішній цикл по **j** – цикл по стовпцям. Перший оператор у блоці **cout << endl;** забезпечує перехід на новий рядок після виведення всіх елементів чергового рядка.

Приклад 2. У ініціалізованому масиві з трьох рядків і трьох стовпців переставити перший і третій стовпці. Вивести на екран початковий і перетворений масив у виді таблиці.

```
#include <iostream>
using namespace std;
int main()
{
     int a[3][3]={{1,2,3},{4,5,6},{7,8,9}};
     for (int i=0;i<3;i++)
     {
          cout << endl;
          for (int j=0;j<3;j++) cout << a[i][j] << ' ';
     }
     cout << endl;
```

```
    for (int i=0;i<3;i++)
    {
        int buf=a[i][0];
        a[i][0]=a[i][1];
        a[i][1]=buf;
    }
    for (int i=0;i<3;i++)
    {
        cout << endl;
        for (int j=0;j<3;j++) cout << a[i][j] << ' ';
    }
    cout << endl;
}
```

Після запуску програми на екрані отримаємо:

```
1 2 3
4 5 6
7 8 9

3 2 1
6 5 4
9 8 7
```

Динамічні масиви

В динамічній області пам'яті можна створювати двовимірні масиви за допомогою оператора **new**.

При виділенні пам'яті відразу під весь масив кількість рядків (лівий розмір) можна задавати за допомогою змінної або виразу, а кількість стовпців повинна бути константним виразом, тобто явно визначена до початку виконання програми.

Після слова **new** записується тип створюваного масиву, а потім його розмірності в квадратних дужках (аналогічно опису нединамічних масивів).

Приклад.
```
const int m=5; cin >> n;
int (*a)[m]  = new int[n][m];
```

Тут адреса початку виділеної за допомогою **new** ділянки памяті присвоюється змінній **a**, яка визначена як покажчик на масив з **m** елементів мипу **int**.

Дужки потрібні, оскільки без них конструкція буде інтерпретуватись як масив покажчиків.

Більш безпечний і універсальний спосіб виділення памяті під двохмірний масив, коли обидва розміри задаються на етапі виконання програми.

Приклад.
```
int nr,ns;
cin >> nr >> ns;
int **a = new int *[nr];          // 1
```

```
          for (int i=0; i<nr; i++)          // 2
               a[i] = new int[ns];          // 3
```

В 1 оголошується змінна типу “покажчик на покажчик на **int**” і виділяється пам'ять під масив покажчиків на рядки масиву (**nr** - кількість рядків).

У 2–організований цикл для виділення пам’яті під кожен рядок масиву.

У 3 – кожному елементу масиву покажчиків на рядки присвоюється адреса початків ділянок пам’яті виділених під рядки 2-х мірного масиву. Кожен рядок складається з **ns** - елементів типу **int**.

Приклад 3. Двовимірний динамічний масив (квадратна матриця). Розмірність задається константою на початку програми. Виконати операції над масивом (див. коментарі у коді програми).

```
#include <iostream>
using namespace std;
int main()
{                    // опис масиву                          (1)
     const int n=3;
     int (*a)[n] = new int [n][n];
     //        введення масиву з клавіатури                  (2)
     for (int i=0; i<n; i++)
     {
          cout << endl;
          for (int j=0; j<n; j++)
          {
               cout << " a[" << i << "][" << j << "]=";
               cin >> a[i][j];
          }
     }
     //             виведення масиву на екран                (3)
     for (int i=0; i<n; i++)
     {
          cout << endl;
          for (int j=0; j<n; j++)
               cout.width(4),cout << a[i][j];
     }
     cout << endl;
     //        сума додатніх ел вище гол діагоналі          (4)
     int s=0;
     for (int i=0; i<n; i++)
          for (int j=0; j<n; j++)
               if (j>i && a[i][j]>0) s+=a[i][j];
     cout << "s=" << s << endl;
     //        к-ть нульових ел на гол діагоналі            (5)
     int k=0;
     for (int i=0; i<n; i++) if (a[i][i]==0) k++;
     cout << "k=" << k << endl;
     //   добуток від’ємн. ел. нижче гол діагоналі          (6)
```

```
     int d=1;
     for (int i=0; i<n; i++)
          for (int j=0; j<n; j++)
               if (j<i && a[i][j]<0) d*=a[i][j];
     cout << "d=" << d << endl;
     //        звільнення памяті з під масиву          (7)
     delete [] a;
}
```

Етапи виконання програми прокоментовані в її тексті. В пункті (3) для вирівнювання задіяний простий маніпулятор **cout.width(4)**. В результаті його дії під кожне число виділяється по 4 позиції.

Тест. З клавіатури вводимо двовимірний масив:

$$a_{00}=1;\ a_{01}=2;\ a_{02}=3;\ a_{10}=-4;\ a_{11}=0;\ a_{12}=-6;\ a_{20}=7;\ a_{21}=-8;\ a_{22}=9$$

На екрані отримаємо:

```
   1  2  3
  -4  0 -6
   7 -8  9
s=5
k=1
d=32
```

Приклад 4. Двовимірний динамічний масив (прямокутна матриця). Розмірності задаються на етапі виконання програми. Розмістити елементи рядків в порядку зростання.

```
#include <iostream>
using namespace std;
int main()
{
     //   опис масиву                                    (1)
     int ns,nr;
     cout << "ns="; cin >> ns;
     cout << "nr="; cin >> nr;
     int **a = new int *[nr];
     for (int i=0;i<nr;i++)
          a[i] = new int[ns];
     //   введення масиву з клавіатури                   (2)
     for (int i=0;i<nr;i++)
     {
          cout << endl;
          for (int j=0;j<ns;j++)
          {
               cout << "a[" << i << "][" << j << "]=";
               cin >> a[i][j];
          }
     }
     //   виведення поч масиву на екран                  (3)
```

```
     for (int i=0;i<nr;i++)
     {
          cout << endl;
          for (int j=0; j<ns; j++)
               cout.width(4),cout << a[i][j];
     }
     cout << endl;
     //   сортування елементів рядків м. бульбашки     (4)
     int k;
     for (int i=0;i<nr;i++)
          do
          {
               k=0;
                    for (int j=1; j<ns; j++) if (a[i][j]<a[i][j-
     1])
               {
                    int buf=a[i][j];
                    a[i][j]=a[i][j-1];
                    a[i][j-1]=buf;
                    k++;
               }
          }
          while (k>0);
     //   виведення кінцевого масиву на екран     (5)
     for (int i=0;i<nr;i++)
     {
          cout << endl;
          for (int j=0; j<ns; j++)
               cout.width(4),cout << a[i][j];
     }
     cout << endl;
     //        звільнення пам'яті з під масиву    (6)
delete [] a;
}
```

Самостійні завдання

Завдання 1. Статичний масив. Написати програму згідно отриманого варіанту використовуючи статичний масив.

Завдання 2. Динамічний масив. Переробити програму завдання 1, використавши динамічний масив.

Варіанти завдань

1. В матриці A *i*-й рядок замінити нулями, а *j*-й стовпчик одиницями. Вивести на екран початкову та перетворену матрицю A.
2. Елементи матриці A, сума індексів яких кратна числу 3, замінити нулями. Вивести на екран початкову та перетворену матрицю A.
3. Просумувати абсолютні значення діагональних елементів квадратної матриці A. Кожний елемент матриці A розділити на результат сумування. Вивести на екран початкову та перетворену матрицю A.
4. Побудувати матрицю A, діагональні елементи якої є сумою діагональних елементів матриць B і C, а всі інші співпадають з елементами матриці C. Вивести на екран початкову та перетворену матрицю A.
5. Від'ємні елементи матриці A вище головної діагоналі замінити нулями. Вивести на екран початкову та перетворену матрицю A.
6. Від'ємні елементи перших *k* рядків матриці A замінити нулями. Вивести на екран початкову та перетворену матрицю A.
7. Додатні елементи матриці A помножити на *k*, а від'ємні збільшити на *b*. Вивести на екран початкову та перетворену матрицю A.
8. Елементи матриці A, які більші *b*, зменшити на *c*. Від'ємні елементи взяти за модулем. Вивести на екран початкову та перетворену матрицю A.
9. Елементи матриці A, сума індексів яких дорівнює *k*, замінити на 0. Вивести на екран початкову та перетворену матрицю A.
10. Елементи матриці A, сума індексів яких менша за *k*, замінити відповідними елементами матриці B, а сума індексів яких більша за *k* збільшити на 1. Вивести на екран початкову та перетворену матрицю A.
11. Дано матрицю A. Якщо діагональний елемент матриці від'ємний, то відповідний рядок матриці замінити нулями. Вивести на екран початкову та перетворену матрицю A.
12. Дано матрицю B. Знайти матрицю $A=B^2$. Вивести на екран матриці B та A
13. Замінити в матриці A перший рядок відповідними її діагональними елементами. Вивести на екран початкову та перетворену матрицю A.
14. В квадратній матриці A *i*-й рядок і *j*-й стовпчик поміняти місцями. Вивести на екран початкову та перетворену матрицю A.
15. В матриці A поміняти місцями максимальний і мінімальний елементи. Вивести на екран початкову та перетворену матрицю A.

Контрольні питання

1. Як описується двовимірний статичний масив?
2. Як ініціалізується двовимірний масив?
3. Як здійснюється доступ до елементів двовимірного масиву?
4. Як створюється двовимірний динамічний масив?
5. Як звільняється пам'ять виділена під двовимірний динамічний масив?
6. Як визначити розмір пам'яті виділену під двовимірний масив?
7. Як розміщаються елементи двовимірного масиву у пам’яті комп’ютера?
8. Для чого використовується простий маніпулятор **`cout.width`**?

Лабораторна робота № 6

Тема: Рядки і файли.

Мета роботи: Познайомитись з організацією обробки рядків, та роботи з текстовими файлами в С++.

Хід виконання роботи

Етап-1. Вивчити теорію, та проробити і осмислити приклади 1-8 з лекції.

Етап-2. Проробити контрольні завдання до лабораторної роботи (див. в кінці даного файлу).

Етап-3. Виконати самостійне завдання згідно свого варіанту.

Етап-4. У кінці лабораторної роботи сформулювати висновки.

Самостійні завдання

Словом називається послідовність символів без пропусків і довжиною не більше ніж 255 символів.

Створіть текстовий файл із 5-10 рядків. У кожному рядку по кілька різних слів, так, щоб можна було продемонструвати правильність роботи програми.

Програма повинна проаналізувати створений файл, зробити в тексті необхідні перетворення, і результат записати у новий файл.

Варіанти завдань

Варіант 1. Із кожного слова вилучити останню його літеру, якщо вона голосна.

Варіант 2. Із кожного слова вилучити останню його літеру, якщо вона приголосна.

Варіант 3. Із кожного слова вилучити початкову його літеру, якщо вона голосна.

Варіант 4. Із кожного слова вилучити початкову його літеру, якщо вона приголосна

Варіант 5. У кінці кожного слова в середині рядка поставити кому, а в кінці рядка крапку.

Варіант 6. Поміняти місцями першу і останню букви кожного рядка тексту.

Варіант 7. Якщо слово містить у собі непарну кількість літер, то замінити середню літеру цього слова „апострофом ”.

Варіант 8. З’ясувати скільки слів у тексті мають більше п’яти букв. Вивести список таких слів.

Варіант 9. В кожному слові тексту довжиною більше шести літер поміняти місцями першу і останню літери.

Варіант 10. У кожному слові тексту, якщо воно починається з голосної букви, замінити її на велику букву.

Варіант 11. У кожному слові тексту, в якому більше двох букв, поміняти місцями останню і передостанню букви.

Варіант 12. У кожному слові тексту, в якому більше двох букв, поміняти місцями першу і другу букви.

Контрольні завдання

6. У першому коді програми прикладу 1 збільшити довжини рядків до 80.
7. У коді програми прикладу 1 перепишіть символи із змінної **`a`** у змінну **`b`** у зворотному порядку.
8. У коді програми прикладу 2 виведіть слова по одному в рядок.
9. У коді програми прикладу 3 виведіть слова по одному в рядок.
10. У коді програми прикладу 6 замініть ініціалізацію тексту на введення тексту з клавіатури.
11. У коді програми прикладу 7 замінити першу букву кожного слова на символ **`'!'`**.
12. У коді програми прикладу 7 замінити останню букву кожного слова на символ **`'!'`**.

Контрольні питання

1. Як описується рядкова змінна?
2. Як ініціалізується рядкова змінна?
3. Яку роль відіграє нуль-символ у рядкових змінних?
4. Як можна визначити довжину рядка у програмі?
5. Як прочитати рядок з пропусками в одну змінну?
6. Для чого використовується метод **`getline`**?
7. За допомогою яких функцій можна перетворити рядкові змінні у числові?
8. Для чого використовується бібліотека **`fstream`**?
9. Як створюється програмно текстовий файл?
10. Як організовується читання даних із текстового файлу?

Лабораторна робота № 7

Тема: Структури.

Мета роботи: Познайомитись з організацією обробки структур в С++.

Хід виконання роботи

Етап-1. Вивчити теорію, та проробити і осмислити приклади 1-8 з лекції.

Етап-2. Проробити контрольні завдання до лабораторної роботи (див. в кінці даного файлу).

Етап-3. Виконати самостійне завдання згідно свого варіанту.

Етап-4. У кінці лабораторної роботи сформулювати висновки.

Порядок виконання індивідуального завдання

1. Створити програму для введення даних в масив, який складається з десяти структур типу заданого індивідуальним завданням і запису даних у файл.
2. Створити другу програму для читання створеного файлу, виконання пункту 1 індивідуального завдання і запису відсортованих даних у новий файл.
3. Створити третю програму для читання створеного файлу і виконання пунктів 2-3 індивідуального завдання.

Варіанти індивідуальних завдань

Варіант 1. Описати структуру з іменем **STUDENT**, яка містить наступні поля:

1. Прізвище і ініціали.
2. Номер групи.
3. Успішність (масив із чотирьох елементів).

Написати програму, яка виконує наступні дії:

1. Впорядковує записи по зростанню номера групи.
2. Виведення на екран прізвищ і номерів груп для всіх студентів, включених в масив, якщо середній бал студента більший 4.0.
3. Якщо таких студентів нема, то вивести відповідне повідомлення.

Варіант 2. Описати структуру з іменем **STUDENT**, яка містить наступні поля:

1. Прізвище і ініціали.
2. Номер групи.
3. Успішність (масив із чотирьох елементів).

Написати програму, яка виконує наступні дії:

1. Впорядковує записи STUDENT по зростанню середнього балу.
2. Виведення на екран прізвищ і номерів груп для всіх студентів, включених в масив, які мають оцінки 4 і 5.
3. Якщо таких студентів нема, то вивести відповідне повідомлення.

Варіант 3. Описати структуру з іменем **STUDENT**, яка містить наступні поля:

1. Прізвище і ініціали.
2. Номер групи.
3. Успішність (масив із чотирьох елементів).

Написати програму, яка виконує наступні дії:

1. Впорядковує записи STUDENT по алфавіту першого поля.
2. Виведення на екран прізвищ і номерів груп для всіх студентів, включених в масив, які мають хоча б одну оцінку 2.
3. Якщо таких студентів нема, то вивести відповідне повідомлення.

Варіант 4. Описати структуру з іменем **AVIA**, яка містить наступні поля:

1. Назва пункту призначення рейсу.
2. Номер рейсу.
3. Тип літака.

Написати програму, яка виконує наступні дії:

1. Впорядковує записи AVIA по зростанню номера рейсу.
2. Виведення на екран номерів рейсів і типів літаків, які вилітають в пункт призначення, назва якого співпадає з назвою, введеною з клавіатури.
3. Якщо таких рейсів нема, то вивести відповідне повідомлення.

Варіант 5. Описати структуру з іменем **AVIA**, яка містить наступні поля:

1. Назва пункту призначення рейсу.
2. Номер рейсу.
3. Тип літака.

Написати програму, яка виконує наступні дії:

1. Впорядковує записи AVIA в алфавітному порядку по назві пункту призначення.
2. Виведення на екран пунктів призначення і номера рейсів, які обслуговуються літаком, тип якого введений з клавіатури.
3. Якщо таких рейсів нема, то вивести відповідне повідомлення.

Варіант 6. Описати структуру з іменем **FIRMA**, яка містить наступні поля:

1. Прізвище і ініціали працівника.
2. Назву займаної посади.
3. Рік поступлення на роботу.

Написати програму, яка виконує наступні дії:

1. Впорядковує записи FIRMA по алфавіту прізвищ працівників.
2. Виведення на екран прізвищ працівників, чий стаж роботи у фірмі перевищує значення введене з клавіатури.
3. Якщо таких працівників нема, то вивести відповідне повідомлення.

Варіант 7. Описати структуру з іменем **TRANS**, яка містить наступні поля:

1. Назва пункту призначення рейсу.
2. Номер поїзда.
3. Час відправлення.

Написати програму, яка виконує наступні дії:

1. Впорядковує записи TRANS в алфавітному порядку по назві пунктів призначення.
2. Виведення на екран інформації про поїзди, які відправляються після введеного з клавіатури часу.
3. Якщо таких поїздів нема, то вивести відповідне повідомлення.

Варіант 8. Описати структуру з іменем **TRANS**, яка містить наступні поля:

1. Назва пункту призначення рейсу.
2. Номер поїзда.
3. Час відправлення.

Написати програму, яка виконує наступні дії:

1. Впорядковує записи TRANS по часу відправлення поїздів.
2. Виведення на екран інформації про поїзди, які направляються в пункт назва якого введена з клавіатури.
3. Якщо таких поїздів нема, то вивести відповідне повідомлення.

Варіант 9. Описати структуру з іменем **TRANS**, яка містить наступні поля:

1. Назва пункту призначення рейсу.
2. Номер поїзда.
3. Час відправлення.

Написати програму, яка виконує наступні дії:

1. Впорядковує записи TRANS по номерах поїздів.
2. Виведення на екран інформацію про поїзд, номер якого введений з клавіатури.
3. Якщо таких поїздів нема, то вивести відповідне повідомлення.

Варіант 10. Описати структуру з іменем **MARSH**, яка містить наступні поля:

1. Назва початкового пункту маршруту.
2. Назва кінцевого пункту маршруту.
3. Номер маршруту.

Написати програму, яка виконує наступні дії:

1. Впорядковує записи MARSH по номерах маршрутів.
2. Виведення на екран інформацію про маршрути, номер яких введений з клавіатури.
3. Якщо таких маршрутів нема, то вивести відповідне повідомлення.

Варіант 11. Описати структуру з іменем **MARSH**, яка містить наступні поля:

1. Назва початкового пункту маршруту.
2. Назва кінцевого пункту маршруту.
3. Номер маршруту.

Написати програму, яка виконує наступні дії:

1. Впорядковує записи MARSH по назвах кінцевих пунктів маршрутів.
2. Виведення на екран інформацію про маршрути, які починаються і закінчуються в пункті, назва якого введена з клавіатури.
3. Якщо таких маршрутів нема, то вивести відповідне повідомлення.

Варіант 12. Описати структуру з іменем **FRIEND**, яка містить наступні поля:

1. Прізвище, ім’я.
2. Номер телефону.
3. Дата народження (масив із трьох чисел).

Написати програму, яка виконує наступні дії:

1. Впорядковує записи FRIEND по датам народження.
2. Виведення на екран інформації про друга, номер телефону якого введений з клавіатури.
3. Якщо такого номера нема, то вивести відповідне повідомлення.

Варіант 13. Описати структуру з іменем **FRIEND**, яка містить наступні поля:

1. Прізвище, ім’я.
2. Номер телефону.
3. Дата народження (масив із трьох чисел).

Написати програму, яка виконує наступні дії:

1. Впорядковує записи FRIEND по алфавіту першого поля.
2. Виведення на екран інформації про друзів, чиї дні народження припадають на місяць, значення якого введено з клавіатури.
3. Якщо таких нема, то вивести відповідне повідомлення.

Варіант 14. Описати структуру з іменем **FRIEND**, яка містить наступні поля:

1. Прізвище, ім’я.
2. Номер телефону.
3. Дата народження (масив із трьох чисел).

Написати програму, яка виконує наступні дії:

1. Впорядковує записи FRIEND по номерам телефонів.
2. Виведення на екран інформації про друга, прізвище якого введено з клавіатури.
3. Якщо такого нема, то вивести відповідне повідомлення.

Варіант 15. Описати структуру з іменем **ZODIAK**, яка містить наступні поля:

1. Прізвище, ім’я.
2. Знак Зодіаку.
3. Дата народження (масив із трьох чисел).

Написати програму, яка виконує наступні дії:

1. Впорядковує записи ZODIAK по датам народження.
2. Виведення на екран інформації про людину, прізвище якої введено з клавіатури.
3. Якщо такої нема, то вивести відповідне повідомлення.

Варіант 16. Описати структуру з іменем **ZODIAK**, яка містить наступні поля:

1. Прізвище, ім’я.
2. Знак Зодіаку.
3. Дата народження (масив із трьох чисел).

Написати програму, яка виконує наступні дії:

1. Впорядковує записи ZODIAK по алфавіту першого поля.
2. Виведення на екран інформації про людей, які народилися під знаком, назва якого введена з клавіатури.
3. Якщо таких нема, то вивести відповідне повідомлення.

Варіант 17. Описати структуру з іменем **ZODIAK**, яка містить наступні поля:

1. Прізвище, ім’я.
2. Знак Зодіаку.
3. Дата народження (масив із трьох чисел).

Написати програму, яка виконує наступні дії:

1. Впорядковує записи ZODIAK по знаку Зодіаку.
2. Виведення на екран інформації про людей, які народилися в місяць, значення якого введено з клавіатури.
3. Якщо таких нема, то вивести відповідне повідомлення.

Варіант 18. Описати структуру з іменем **PRICE**, яка містить наступні поля:

1. Назва товару.
2. Назва магазину в якому продається товар.
3. Вартість товару в грн.

Написати програму, яка виконує наступні дії:

1. Впорядковує записи PRICE в алфавітному порядку по назві товару.
2. Виведення на екран інформації про товар, назва якого введена з клавіатури.
3. Якщо таких товарів нема, то вивести відповідне повідомлення.

Варіант 19. Описати структуру з іменем **PRICE**, яка містить наступні поля:

1. Назва товару.
2. Назва магазину в якому продається товар.
3. Вартість товару в грн.

Написати програму, яка виконує наступні дії:

1. Впорядковує записи PRICE в алфавітному порядку по назвах магазинів.
2. Виведення на екран інформації про товари, які продаються в магазині, назва якого введена з клавіатури.
3. Якщо такого магазину нема, то вивести відповідне повідомлення.

Варіант 20. Описати структуру з іменем **ORDER**, яка містить наступні поля:

1. Розрахунковий рахунок платника.
2. Розрахунковий рахунок одержувача.
3. Переведена сума в грн.

Написати програму, яка виконує наступні дії:

1. Впорядковує записи ORDER в алфавітному порядку по розрахунковим рахункам платників.
2. Виведення на екран інформації про суму, зняту з рахунку платника, введеного з клавіатури.
3. Якщо такого рахунку нема, то вивести відповідне повідомлення.

Контрольні завдання

1. У коді програми прикладу 1 додати ще один елемент масиву структур.
2. У коді програми прикладу 1 додати ще один елемент до структури.
3. У коді програми прикладу 2 визначте об’єм пам’яті виділену під структуру **`s`**.
4. У коді програми прикладу 4 знайти елемент масиву з максимальним значенням поля **`y`**.
5. У коді програми прикладу 5 відсортувати масив по другому полю методом бульбашки.
6. коді програми прикладу 6 замінити ініціалізацію на введення елементів структури з клавіатури.
7. У програмі до прикладу 8 відсортувати структури по першому полю.
8. У програмі до прикладу 8 перейти до динамічного масиву структур.

Контрольні питання

1. Як описується структура?
2. Що таке список описувачів?
3. Якого типу можуть бути елементи структури?
4. Коли можна опустити ім’я типу при описі структури?
5. Як визначити програмно об’єм пам’яті виділеної під структуру?
6. Який механізм звернення до елементів структури?
7. Як відбувається присвоєння для змінних структурованого типу?
8. Для чого використовується функція **`strcmp`**?
9. З якою метою до програми підключається файл **`fstream`**?
10. Яка різниця між **`cout`** і **`fout`**?

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Васильєв О.М. Програмування на С++. К.: Вид-во «Ліра-К», 2019. – 382с.
2. Ковалюк Т. В. Алгоритмізація та програмування: підручник. – Львів: Вид-во «Магнолія 2006», 2013. – 400 с.
3. Михайленич П. Основи програмування мовою С++. Львів: Вид-во «Львівська політехніка», 2016 р. – 204с.
4. Матвієнко М.П., Розен В.П., Закладний О.М. Архітектура комп'ютерів. К.: Вид-во «Ліра-К», 2019. – 264с.
5. Ivor Horton, Peter Van Weert. Beginning C++23: From Beginner to Pro. Apress, 2023. – 948p.
6. С++. Основи програмування. Теорія та практика; підручник/ за ред. О.Г. Трофименка. – Одеса: Вид-во «Фенікс», 2010. – 544 с.
7. Шпак З. Я. Програмування мовою С. Львів: Вид-во «Львівська політехніка», 2011. – 436с.

ЗМІСТ

Гарнітура Times New Roman. Папір офсетний.
Формат видання 60х84/16.
Умов. друк. арк.3.54 Зам. № 22. Наклад 50 прим.
Видруковано ПП «АУТДОР - ШАРК»
88000, м. Ужгород, пл. Жупанатська, 15/1. Тел.: 3-51-25.
E-mail: office@shark.com.ua
Свідоцтво про внесення суб'єкта видавничої справи
до державного реєстру видавців,
виготівників і розповсюджувачів
видавничої продукції
Серія 3т № 40 від 29 жовтня 2012 року