МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

Кафедра: “Систем штучного інтелекту та аналізу даних”

МЕТОДИЧНІ ВКАЗІВКИ

**для виконання лабораторних робіт
з дисципліни**

ПРОГРАМУВАННЯ

Частина 2

для студентів спеціальностей

**F3 Комп’ютерні науки,
F4 Системний аналіз та наука про дані,
F6 Інформаційні системи і технології,
F7 Комп’ютерна інженерія.**

ТЕРНОПІЛЬ 2026

Спеціальності: "Комп'ютерні науки", "Системний аналіз та наука про дані" "Інформаційні системи і технології ", "Комп'ютерна інженерія"

УДК 681,5
М 54

Методичні вказівки до виконання лабораторних робіт з курсу "Програмування". Частина 2. Для студентів спеціальностей: F3 "Комп'ютерні науки", F4 "Системний аналіз та наука про дані", F6 "Інформаційні системи і технології ", F7 "Комп'ютерна інженерія". Упоряд.: Ю. Гладьо, Г. Семенишин, С. Гладьо, Н.Гащин – Тернопіль: видавництво ТНТУ, 2026. – 40с.

Методичні вказівки розроблені у відповідності з навчальними планами F3 "Комп'ютерні науки", F4 "Системний аналіз та наука про дані", F6 "Інформаційні системи і технології ", F7 "Комп'ютерна інженерія".

Укладачі: к.т.н., доцент Гладьо Ю.Б.,
старший викладач Семенишин Г.М.,
к.т.н. старший викладач Гладьо С.В.
к.т.н., доцент Гащин Н.Б.

.

Рецензент: к.т.н., доцент Дуда О.М.

Затверджено на засіданні систем штучного інтелекту та аналізу даних

Протокол № 3 від *15 березня 2026 р.*

Схвалено та рекомендовано до друку методичною комісією факультету комп'ютерно-інформаційних систем і програмної інженерії Тернопільського національного технічного університету імені Івана Пулюя.

Протокол № 4 від *9 квітня 2026 р.*

Лабораторна робота № 6
(2 год.)

Тема: Обробка одновимірних масивів.

Мета роботи: Оволодіти практичними навичками розробки алгоритмів та програмування обчислювального процесу обробки одновимірних масивів.

За час виконання лабораторної роботи студент повинен освоїти:

- опис змінних типу масив;
- використання в програмах індексних величин;
- застосування оператора циклу з кроком при вводі та обробці масиву;
- задання значень елементів масиву під час його оголошення;
- використання вказівників при звертанні до елементів масиву;
- використання констант при задані значень індексних величин;
- порядок обробки величин індексного типу;
- вивід результатів виконаної роботи.

Завдання на лабораторну роботу: Скласти схему алгоритму та дві програми на С++ обробки одновимірного масиву, використовуючи введення значень елементів масиву з клавіатури, а згодом їх задання під час оголошення масиву. В другій програмі для звернення до елементів масиву використати вказівники. Вхідні дані вибирати довільно такі, що відповідають умові задачі. Вивід результатів передбачити на екран.

Порядок виконання лабораторної роботи:

- Використовуючи відповідний ярлик, викликати інтегроване середовище С++;
- ввести текст програми, де ввід елементів одновимірного масиву здійснюється з клавіатури (в першій стрічці обов'язково має бути коментар, в якому вказати номер роботи, групу та прізвище виконавця);
- текст програми записати у власну папку: (***Prizvyshche_L6v31_1***);
- відкомпілювати програму, виправляючи при цьому можливі помилки;
- відлагоджену програму виконати, записуючи отриманий результат;
- ввести текст програми, де значення елементів масиву задаються під час його оголошення, а обробка із застосуванням вказівників;
- текст програми записати у власну папку: (***Prizvyshche_L6v31_2***);
- відкомпілювати програму, виправляючи при цьому можливі помилки;
- відлагоджену програму виконати, записуючи отриманий результат;
- оформити звіт про виконану роботу.

Теоретичні відомості

Масив – це впорядкований скінчений набір даних одного типу, які зберігаються в ***послідовно розташованих*** комірках оперативної пам'яті і мають спільну назву. Назву масиву надає користувач.

Масив складається з ***елементів***. Кожен елемент має індекси, за якими його можна знайти у масиві. Кількість індексів визначає розмірність масиву. Розрізняють одно- та багатовимірні масиви. Наприклад, двовимірний масив даних – це таблиця, що складається з декількох рядків і стовпців. У математиці поняттю масив відповідають поняття вектору та матриці.

Загальний вигляд конструкції опису одномірного масиву такий:

<тип> <ім'я масиву>[<розмір>]

Розмір – це кількість елементів масиву. Розмір масиву необхідно знати і задавати заздалегідь, оскільки компілятор має зарезервувати для нього необхідний обсяг пам'яті. Розміром може бути лише ***стала*** величина (не змінна).

Ім'я масиву у програмі змінювати не можна – це стала величина, яка містить адресу першого елемента. Отже, назва масиву є вказівником на перший елемент.

Наприклад, команда ***int stud[5]*** оголошує масив з іменем ***stud***, який складається із п'яти цілих чисел; команда ***float rist[10]*** оголошує масив ***rist***, який містить десять чисел дійсного типу; ***char alphavit[6]*** – оголошення масиву із 6 символів.

Звернутись до елементів масиву можна двома способами: за допомогою імені масиву або використовуючи вказівники.

Нумерація елементів масиву завжди починається з нуля. Щоб звернутись до деякого елемента, необхідно зазначити ім'я масиву, а у квадратних дужках – його номер. Наприклад, змінна ***stud[2]*** є третім елементом масиву ***stud***, а ***stud[4]*** – п'ятим, оскільки масив ***stud*** має елементи

stud[0], stud[1], stud[2], stud[3] та stud[4].

За замовчуванням усім елементам масиву надається значення **0**. Масив можна ініціалізувати повністю або частково відразу під час його оголошення, записуючи значення елементів через кому у фігурних дужках. Наприклад,

```
int Stud[] = {2, 10, 5, 7, 3};
float rist[10] = {163.4, 154.6, 170, 172.8, };
char alphavit[6]  = "Абетка";
або char alphavit[6]  = {'А', 'б', 'е', 'т', 'к', 'а'};
```

Перші чотири елементи масиву ***rist*** були проініціалізовані, а решта – ні.

Назва масиву ***stud*** є вказівником на його перший елемент. Змінна ****stud*** містить значення першого елемента масиву (елемента ***stud[0]***). Оскільки всі елементи масиву розміщені у послідовних комірках оперативної пам'яті комп'ютера, то вказівник ****(stud + 1)*** вказуватиме на другий елемент масиву (зміщення відносно вказівника ****stud*** на одну одиницю пам'яті), а вказівник ****(stud + 4)*** – на п'ятий (зміщення на чотири одиниці).

Проініціалізувати масив (надати значення елементам масиву) можна одним із способів:

- використовуючи принцип замовчування;
- безпосередньо під час його оголошення;
- застосовуючи команду присвоєння;
- під час введення даних із клавіатури.

Якщо масив повністю ініціалізують під час оголошення, то його розмір зазначати не обов'язково. У цьому випадку компілятор сам визначає скільки

пам'яті необхідно зарезервувати. У наведеному прикладі масив ***stud*** складатиметься з п'яти цілих чисел.

Надати значення іншим елементам масиву ***rist*** або змінити значення вже проініціалізованих елементів можна командою присвоєння, наприклад, так**:**

```
rist[3] = 175.4; rist[9] = 184.1;
```

або так:

```
*(rist + 2) = 164.5; *(rist + 7) = 148.0;
```

тощо.

Елементи масиву також можна вводити з клавіатури під час виконання програми, як це робимо для змінних простих стандартних типів.

Масиви-сталі (значення яких змінювати у програмі не можна) оголошують так: ***const int flag[] = {1, 2}.***

Сталі масиви треба ініціалізувати під час оголошення, інакше елементам масиву автоматично будуть присвоєні нульові значення.

Для опрацювання елементів масиву найчастіше використовують команду циклу **for**, хоча можна застосувати і ***while*** або ***do while***.

Приклад1. Створити масив з перших ста цілих чисел і обчислити суму всіх його значень можна одним із способів:

```
int n[100];  // 1-й спосіб
int S = 0;
for (k = 0; k< 100;)
     {
       *(n+k) = ++k;
       S += *(n+k);
     }
```

```
int n[100];   //2-й спосіб
int S = 0;
for (k = 0; k< 100; k++)
{
  n[k] = k + 1;
  S += n[k];
}
```

Задачі відшукання в масиві конкретних даних розв'язують методом сканування (перебирання, перегляду) всіх елементів масиву за допомогою циклу й умовної команди, де зазначають умову пошуку потрібних даних.

Динамічне оголошення масивів. Під час компіляції програмного коду для статично оголошених масивів надається пам'ять. Для ефективного використання пам'яті призначене динамічне оголошення масивів, а саме:

```
<тип вказівника> *<назва> = new <тип змінної>[<кількість>];
```

Після виконання цієї команди буде виділена неперервна ділянка пам'яті обсягом

```
sizeof(тип змінної) *<кількість>
```

і назва масиву вказуватиме на початок цієї ділянки.

З динамічною змінною можна виконувати операції, визначені для даних відповідного базового типу.

Після опрацювання масиву вивільнити пам'ять можна за допомогою команди

```
delete[] <назва вказівника на масив даних>;
```

Під час вивільнення пам'яті розмір масиву зазначати не потрібно.

Спеціальності: "Комп'ютерні науки", "Системний аналіз та наука про дані" "Інформаційні системи і технології", "Комп'ютерна інженерія"

Розв'яжемо задачу з *прикладу1*, використовуючи динамічний розподіл пам'яті

```
int *n = new int[100];     // Виділяємо пам'ять для ста цілих чисел
for(int S = 0, k = 0; k < 100;)
{ // Опрацьовуємо масив
  *(n+k) = ++k;
  S += *(n+k);
}
delete[] n;    // Вивільняємо пам'ять
```

За допомогою динамічних змінних можна розв'язати задачу почергового опрацювання одною програмою деякої кількості великих масивів (якщо всі масиви неможливо одночасно ввести у пам'ять). Задачу розв'язують так. Створюють масив, наприклад, ****mas1 = new <тип>[<кількість>]*** і опрацьовують динамічні змінні ****mas1, *(mas1+1), ...***. Вивільняють пам'ять ***delete[] mas1***.

Створюють й опрацьовують елементи другого масиву

****mas2 = new <тип> [<кількість>]*** і т.д.

Упорядкування масивів. Задачі впорядкування даних у масиві мають важливе практичне значення. Прикладами таких даних можуть бути телефонні довідники (дані, упорядковані за прізвищем адресатів), бібліотечні каталоги, розклади руху потягів тощо. Упорядковані дані значно легше опрацьовувати, оскільки серед них набагато швидше можна відшукати потрібну інформацію.

Дані у масиві можна впорядковувати за зростанням (від меншого до більшого) або за спаданням (від більшого до меншого).

Задача **(про упорядкування масиву).** Розглянемо масив А, який містить цілі числа 5, 2, 3, 6, 1, 4. Упорядкувати цей масив за *зростанням* значень.

Є багато різних способів упорядкування даних в одновимірному масиві. Розглянемо один з них:

Метод обміну ("бульки"). Читаємо два перші елементи масиву (5 та 2) і порівнюємо їх між собою. Якщо перший елемент більший за другий, то міняємо їх місцями (стане *2* і 5). Далі розглядаємо другий і третій елемент (тепер це елементи 5 і 3) і порівнюємо їх між собою. Якщо треба, то міняємо їх місцями (стане 3 і 5). Потім порівнюємо третій і четвертий елементи (5 і 6) і так далі. Отримаємо масив 2, 3, 5, 1, 4, 6. Таким чином максимальний елемент (це елемент 6) опинився в самому кінці масиву, тобто там, де потрібно. Після цього знову розглядаємо масив, але вже без останнього елемента. Застосовуємо метод обміну до нового масиву (2, 3, 5, 1, 4). На останньому місці опиниться 5. Далі знов розглядатимемо масив без останнього елемента і застосовуватимемо метод обміну і т. д. Якщо масив має *n* елементів, то метод обміну треба застосувати ***n - 1*** разів до кожного разу меншої кількості

елементів. Упорядковані елементи будуть нагромаджуватись із кінця масиву. Складемо програму:

```
// Впорядкування масиву методом обміну елементів (бульки)
#include <iostream>
#include <conio.h>
using namespace std;
const int k = 6;
void sort(int m[k]);
void main()
 {
  system("color F0"); // Встановити білий фон і чорний текст
  setlocale(0, "ukr");
  clrscr();
  int i;
  int a[k];
  cout << "Введіть " << k << " чисел: \n";
  for (i = 0; i < k; i++)
       cin >> *(a + i);
  sort(a);
  cout << " Упорядкований масив: \n";
  for (i = 0; i < k; i++)
  cout << *(a + i) << "\n";
      }
//------------------------
void sort(int m[k])
{
 int temp, i, j;
 for (i = 0; i < k-1; i++)
 for (j = 0; j < k - i - 1; j++)
  if(*(m+j) > *(m + j + 1))
       {
         temp = *(m + j + 1);
         *(m + j + 1) = *(m + j);
         *(m + j) = temp;
       }
}
```

Зауваження. Оскільки назва масиву – це вказівник на перший елемент цього масиву, а функції користувача можуть повертати значення у програму не тільки за допомогою команди ***return***, а й через вказівники, то у програмі сортування масиву міняємо місцями значення елементів масиву **a** за допомогою функції ***sort()***, використовуючи вказівники.

Варіанти завдань лабораторної роботи

1. Дано масив A(15). Знайти добуток від`ємних елементів масиву.
2. Дано масив A(15). Знайти кількість елементів, більших заданого числа B.
3. Дано масив A(15). Знайти кількість елементів, менших заданого числа B.
4. Дано масив A(15). Знайти кількість елементів, рівних заданому числу B.
5. Дано масив A(15). Знайти максимальний елемент масиву.
6. Дано масив A(15). Знайти мінімальний елемент масиву.

Спеціальності: "Комп'ютерні науки", "Системний аналіз та наука про дані" "Інформаційні системи і технології", "Комп'ютерна інженерія"

7. Дано масив А(15). Знайти номери всіх додатних елементів.
8. Дано масив А(15). Знайти номери всіх від`ємних елементів.
9. Дано масив А(15). Знайти добуток елементів з парними номерами.
10. Дано масив А(15). Розділити всі елементи на найменший елемент масиву (відмінний від нуля).
11. Дано масив А(15). Розділити всі елементи на найбільший елемент масиву (відмінний від нуля).
12. Дано масив А(15). Знайти суму елементів з непарними номерами.
13. Дано масив А(15). Знайти суму елементів з парними номерами.
14. Дано масив А(15). Знайти добуток елементів з непарними номерами.
15. Дано масив А(15). Знайти кількість додатних елементів масиву.
16. Дано масив А(15). Знайти кількість від`ємних елементів масиву.
17. Дано масив А(15). Знайти суму додатних елементів масиву.
18. Дано масив А(15). Знайти суму від`ємних елементів масиву.
19. Дано масив А(15). Знайти середнє арифметичне додатних елементів масиву.
20. Дано масив А(15). Знайти середнє арифметичне від`ємних елементів масиву.
21. Дано масив А(15). Знайти мінімальний додатній елемент масиву.
22. Дано масив А(15). Знайти максимальний від`ємний елемент масиву.
23. Дано масив А(15). Знайти кількість елементів, рівних нулю.
24. Дано масив А(15). Знайти добуток додатних елементів масиву.
25. Дано масив А(15). Поміняти місцями найбільший елемент з останнім.
26. Дано масив А(15). Поміняти місцями найменший елемент з останнім.
27. Дано масив А(15). Замінити від`ємні елементи масиву найбільшим елементом.
28. Дано масив А(15). Замінити додатні елементи масиву найбільшим елементом.
29. Дано масив А(15). Замінити від`ємні елементи масиву найменшим елементом.
30. Дано масив А(15). Замінити додатні елементи масиву найменшим елементом.

Приклад виконання завдання

Завдання:

Дано масив А(15). Замінити додатні елементи масиву найменшим елементом.

Блок схема алгоритму:

(дивись лабораторну роботу №1)

Зразок програм лабораторної роботи

Програма1:

```
// Лабораторна робота №6 варіант 31 група СІ-12 ГордієнкоГордієнко М.В.
```

```
// Дано масив A(15). Замінити додатні елементи масиву найменшим
// елементом
#include <iostream>
using namespace std;

const int n = 15;
int main()
{
     system("color F0"); // Встановити білий фон і чорний текст
     setlocale(0, "ukr");
    int i;
    float min, A[n];
    for(i = 0; i < n; i++)
       { cout << "A[" << i << "]="; cin >> A[i]; }
    min = A[0];
    for(i = 1; i < n; i++)
     if (A[i] < min)  min = A[i];
    for(i = 0; i < n; i++)
     if (A[i] > 0) A[i] = min;
    for(i = 0; i < n; i++)
       cout << "A[" << i << "]=" << A[i] << endl;
    return 0;
}
```

Програма2:

```
// Лабораторна робота №6 варіант 31 група СІ-12 Гордієнко М.В.
// Дано масив A(15). Замінити додатні елементи масиву найменшим
// елементом
#include <iostream>
using namespace std;

const int n = 15;
int main()
{
     system("color F0"); // Встановити білий фон і чорний текст
     setlocale(0, "ukr");
     int i;
     float   min,   A[n]   =   {3.05,-9,-5,22.5,12,-6.045,5,9,3,-
     5,4.1,8,1,0,15};
    min = *A;
    for(i = 1; i < n; i++)
     if ( *(A + i) < min)  min = *(A + i);
    for(i = 0; i < n; i++)
     if ( *(A + i) > 0) *(A + i) = min;
    for(i = 0; i < n; i++)
       cout << "A[" << i << "]=" << *(A + i) << endl;
    return 0;
}
```

Спеціальності: "Комп'ютерні науки", "Системний аналіз та наука про дані" "Інформаційні системи і технології", "Комп'ютерна інженерія"

Лабораторна робота № 7
(2 год.)

Тема: Обробка двовимірних масивів.

Мета роботи: Оволодіти практичними навичками розробки алгоритмів та програмування обчислювального процесу обробки двовимірних масивів.

За час виконання лабораторної роботи студент повинен освоїти:

- оголошення змінних типу багатовимірний масив;
- використання в програмах індексних величин;
- застосування вкладених команд циклу з кроком при вводі та обробці багатовимірних масивів;
- задання значень елементів двовимірного масиву під час його оголошення;
- використання вказівників при звертанні до елементів масиву;
- порядок обробки величин індексного типу;
- вивід результатів виконаної роботи.

Завдання на лабораторну роботу: Скласти схему алгоритму та дві програми на С++ обробки двовимірного масиву, використовуючи введення елементів з клавіатури, а згодом задання значень елементів масиву під час його оголошення. Вхідні дані вибирати довільно такі, що відповідають умові задачі. Вивід результатів передбачити на екран.

Порядок виконання лабораторної роботи:

- Використовуючи відповідний ярлик, викликати інтегроване середовище С++;
- ввести текст програми, де ввід елементів одновимірного масиву здійснюється з клавіатури (в першій стрічці обов'язково має бути коментар, в якому вказати номер роботи, групу та прізвище виконавця);
- текст програми записати у власну папку: (***Prizvyshche_L7v31_1***);
- відкомпілювати програму, виправляючи при цьому можливі помилки;
- відлагоджену програму виконати, записуючи отриманий результат;
- ввести текст програми, де значення елементів масиву задаються під час його оголошення, а обробка із застосуванням вказівників;
- текст програми записати у власну папку: (***Prizvyshche_L7v31_2***);
- відкомпілювати програму, виправляючи при цьому можливі помилки;
- відлагоджену програму виконати, записуючи отриманий результат;
- оформити звіт про виконану роботу.

Теоретичні відомості

Якщо елемент масиву має не один, а декілька індексів, то такі масиви називаються ***багатовимірними.*** Прикладами багатовимірних масивів можуть бути різноманітні табличні дані: Семестрові оцінки групи студентів, сторінка в шкільному журналі, таблиця результатів футбольних змагань тощо. Це двовимірні таблиці, яким у математиці відповідає поняття матриці.

Загальний вигляд конструкції опису багатовимірного *(N*-вимірного) масиву такий:

```
<тип> <ім'я масиву>[<p1>][<p2>]... [<PN>] ,
```

де ***p1, p2, ..., PN*** – задають розміри для кожного виміру.

Кількість індексів визначає вимірність масиву: двовимірні масиви мають два індекси, тривимірні – три і т.д.

Усі багатовимірні масиви можна розглядати й опрацьовувати як одновимірні. Наприклад, тривимірний масив ***sal[5][20][30]*** можна інтерпретувати як п'ять масивів розміром ***20 × 30***, а інші – як ***20*** одновимірних масивів, які містять по ***30*** елементів.

Елементи двовимірного масиву визначаються іменем масиву та двома індексами: перший індекс означає номер рядка, інший – номер стовпця, на перетині яких розміщений елемент.

Наприклад, оголосимо масив ***int doba[24][60].*** Він містить елементи цілого типу і складається з ***24*** рядків і ***60*** стовпців. Елемент ***doba[23)[59]*** розміщений на перетині останнього ***24***-го рядка та останнього ***60***-го стовпця (*нумерація індексів масиву завжди починається з нуля*).

Двовимірні масиви компілятор розглядає як послідовність одновимірних. Тому до елементів двовимірного масиву, як і для одновимірних, можна також звертатись через вказівники. У такому випадку це вказівник на вказівник одновимірного масиву:

****(*(<назва вказівника> + <зміщення по рядках>) + <зміщення по стовпцях>)***

Наприклад, елемент ****(*(doba+2)+15)*** розміщений на перетині ***3***-го рядка та ***16***-го стовпця.

Під час оголошення двовимірні масиви можна частково або повністю ініціалізувати.

Оголосимо і проініціалізуємо двовимірний масив цілих чисел

int ball[2][3] = {4, 5, 3, 3, 5, 2}.

У такому випадку елементам надаються значення так:

ball[0][0] = 4, ball[0][1] = 5, ball[0][2] = 3, ball[1][0] = 3, ball[1][1]=5, ball[1][2] = 2.

Двовимірні масиви автоматично ініціалізуються "по рядках , тобто спочатку модифікується зовнішній (правіший) індекс . Надавати значення елементам масиву більш наглядно можна і так:

```
int ball[2][3] = {{4, 5, 3}, {3, 5, 2}};
```

або так:

```
int ball[2][3] = { 4, 5, 3,
                   3, 5, 2 };
```

Для опрацювання елементів багатовимірних масивів найчастіше використовують вкладені команди циклу ***for***, хоча можна застосувати і ***while*** або ***do while***.

Задача **(про таблицю множення).** Скласти програму для занесення в двовимірний масив ***p*** таблиці множення двох чисел (таблиця Піфагора) і виведення масиву на екран.

```
//Програма Піфагор
#include <iostream>
```

```
using  namespace std;

void main()
{
 system("color F0"); // Встановити білий фон і чорний текст
 setlocale(0, "ukr");
 const n = 9;
 int p[n][n];
 for (int i = 0; i < n; i++)
  {
     for (int j = 0; j < n; j++)
     {    // Множення чисел
       p[i][j] = (i + 1) * (j + 1);
       cout << p[i][j] << "\t";
     }
    cout << "\n"; // Для виводу масиву у вигляді таблиці
  }
}
```

Задача **(про добуток матриць).** Задано матрицю **A** розміром **m × n** та матрицю **B** розміром **n × p**. Обчислити їх добуток.

З курсу математики відомо, що матриця **C = A × B** матиме розмір **m × p** і її елементи визначаються за формулою $C_{ij} = \sum_{k=1}^{m} a_{ik} b_{kj}$.

```
// Програма Добуток матриць
#include <iostream>
#define N 3
#define M 4
#define P 6
using  namespace std;
void main()
{
 system("color F0"); // Встановити білий фон і чорний текст
 setlocale(0, "ukr");
  int i, j, k;
  int A[N][M] = { 4, 5, 7, 3,
                  6, 4, 8, 9,
                  9, 5, 8, 1 };
  int B[M][P] = { 4, 5, 7, 3, 8, 9,
                  6, 4, 8, 9, 2, 8,
                  9, 5, 8, 1, 7, 2,
                  3, 5, 7, 9, 4, 6 };
  int C[N][P];
  cout << "Матриця C: \n";
  for (i = 0; i < N; i++)
    {
      for (j = 0; j < P; j++)
       {
        *(*(C + i) + j) = 0;
        for (k = 0; k < M; k++)
        *(*(C + i) + j) += *(*(A + i) + k) *
                           *(*(B + k) + j);
```

```
            cout << *(*(C + i) + j) << "\t";
        }
      cout << "\n" << "\n";
    }
}
```

Варіанти завдань лабораторної роботи

1. Дано масив C(5,4). Знайти середнє арифметичне елементів кожної стрічки.
2. Дано масив H(3,5). Знайти суму від'ємних елементів кожної стрічки.
3. Дано масив T(6,4). Знайти максимальний елемент кожної стрічки.
4. Дано масив M(2,6). Знайти мінімальний елемент кожного стовпця.
5. Дано масив E(4,5). Для кожної стрічки знайти суму елементів більших числа 10.
6. Дано масив AD(3,5). Знайти добуток всіх елементів з непарних стовпців.
7. Дано масив KD(4,3). Знайти суму максимальних елементів всіх стрічок.
8. Дано масив MD(4,6). Знайти кількість від'ємних елементів кожної стрічки.
9. Дано масив OD(5,3). Знайти кількість додатних елементів кожного стовпця.
10. Дано масив HD(4,4). Для кожного стовпця знайти кількість елементів більших по модулю числа π.
11. Дано масив D(4,4). Знайти добуток елементів діагоналей матриці.
12. Дано масив KD(4,4). Знайти суму елементів що лежать над головною діагоналлю.
13. Дано масив OD(5,3). Всі додатні елементи замінити їхнім логарифмом.
14. Дано масив E(4,5). Всі від'ємні елементи замінити їхнім модулем.
15. Дано масив KD(4,3). Всі елементи кожної стрічки розділити на перший ненульовий елемент стрічки.
16. Дано масив A(5,5). Знайти суму елементів масиву.
17. Дано масив B(5,6).Знайти добуток елементів масиву.
18. Дано масив X(6,8). Знайти значення найбільшого елементу масиву.
19. Дано масив Y(4,5). Знайти значення найменшого елементу масиву.
20. Дано масив Z(6,3). Знайти кількість додатних елементів масиву.
21. Дано масив B(6,6). Знайти добуток відмінних від нуля елементів.
22. Дано масив C(3,5). Всі елементи розділити на найбільший, відмінний від нуля, елемент.
23. Дано масив D(3,4) Всі елементи помножити на найменший, відмінний від нуля, елемент.
24. Дано масив KD(4,4). Знайти суму елементів кожної з діагоналей.
25. Дано масив KD(4,4). Знайти добуток елементів що лежать нижче головної діагоналі.
26. Дано масив D(4,4). Знайти суму елементів кожної стрічки.
27. Дано масив A(3,5). Знайти суму додатних елементів кожної стрічки.
28. Дано масив P(4,3). Знайти добуток елементів кожної стрічки.
29. Дано масив K(4,3). Знайти добуток додатних елементів кожного стовпця.

30. Дано масив D(3,7). Для кожної стрічки знайти суму елементів непарних стовпців.

Приклад виконання завдання

Завдання:

Дано масив D(3,7). Для кожної стрічки знайти суму елементів непарних стовпців.

Блок схема алгоритму:

(дивись лабораторну роботу №1)

Зразок програми лабораторної роботи

Програма1:

```
// Лабораторна робота №7 варіант 31 група СІ-12 Гордієнко М.В.
// Дано масив D(3,7). Для кожної стрічки знайти суму
// елементів непарних стовпців.
#include <iostream>
#define m 3
#define n 4
using  namespace std;
int main()
{
  system("color F0"); // Встановити білий фон і чорний текст
  setlocale(0, "ukr");
    int i, j;
    float S, D[m][n];
    for(i = 0; i < m; i++)
     for(j = 0; j < n; j++)
       { cout << "D[" << i << "," << j <<"]= ";
          cin >> D[i][j];
       }
    for(i = 0; i < m; i++)
     {
      S = 0;
       for(j = 1; j < n; j += 2)
           S += D[i][j];
       cout << "стрічка " << i << "  сума " << S << endl;
     }
   return 0;
}
```

Програма2:

```
// Лабораторна робота №7 варіант 31 група СІ-12 Гордієнко М.В.
// Дано масив D(3,7). Для кожної стрічки знайти суму
// елементів непарних стовпців.
#include <iostream>
#define m 3
#define n 4
```

Спеціальності: "Комп'ютерні науки", "Системний аналіз та наука про дані" "Інформаційні системи і технології", "Комп'ютерна інженерія"

```
using  namespace std;
int main()
{ system("color F0"); // Встановити білий фон і чорний текст
  setlocale(0, "ukr");

     int i, j;
     float S, D[m][n] = { 2, 3, 4, 4,
                          7, 4, 8, 3,
                          6, 2, 9, 1 };
    for(i = 0; i < m; i++)
     {
      S = 0;
       for(j = 1; j < n; j += 2)
           S += *(*(D + i) + j);
       cout << "стрічка " << i << "  сума " << S << endl;
     }
   return 0;
}
```

Лабораторна робота № 8
(2 год.)

Тема: Використання функцій користувача.

Мета роботи: Оволодіти практичними навичками розробки та програмування обчислювального процесу з використанням функцій користувача. Засвоїти опис та їх використання у виразах.

За час виконання лабораторної роботи студент повинен освоїти:

- опис функцій користувача безпосередньо та при допомозі макросів;
- використання у виразах звернень до функцій користувача;
- взаємодію фактичних аргументів та формальних параметрів;
- використання локальних та глобальних змінних;
- вивід результатів виконаної роботи.

Завдання на лабораторну роботу: Скласти схему алгоритму та дві програми на С++ обчислення значення змінної ***F*** з використанням функції користувача ***p(k)*** (таблиця 1) безпосередньо та при допомозі макросів. Вхідні дані (таблиця 2) вводити з клавіатури. Вивід результатів передбачити на екран.

Порядок виконання лабораторної роботи:

- Використовуючи відповідний ярлик, викликати інтегроване середовище С++;
- ввести текст програми з використанням функції користувача ***p(k)*** безпосередньо (в першій стрічці обов'язково має бути коментар, в якому вказати номер роботи, групу та прізвище виконавця);
- текст програми записати у власну папку: (***Prizvyshche_L8v31_1***);
- відкомпілювати програму, виправляючи при цьому можливі помилки;
- відлагоджену програму виконати, записуючи отриманий результат;
- ввести текст програми з використанням функції користувача ***p(k)*** описаної при допомозі макросу:

- текст програми записати у власну папку: (***Prizvyshche_L8v31_2***);
- відкомпілювати програму, виправляючи при цьому можливі помилки;
- відлагоджену програму виконати, записуючи отриманий результат;
- оформити звіт про виконану роботу.

Теоретичні відомості

Мова програмування С++ дає можливість реалізувати концепцію структурного аналізу алгоритмів. ***Структурний аналіз*** полягає у попередньому опрацюванні складної задачі чи громіздкого алгоритму та поділі його на окремі простіші частини. У С++ ці частини реалізуються за допомогою ***функцій.*** Окремі функції об'єднують у спільну програму. У відкомпільованому вигляді така програма утворює модуль. Головна функція, що обов'язково входить до кожної програми, – **`main()`**. Розрізняють стандартні функції та функції користувача. Стандартні функції мови описані (визначені) у бібліотеках.

Функція користувача – це поіменована група команд, яка оголошена у файлі заголовків (або в основній програмі) та описана у модулі (в основній програмі). До функції можна звертатись (викликати) з будь-якого місця програми необхідну кількість разів.

Оголошення функцій користувача. Кожну функцію користувача перед першим викликом передусім необхідно оголосити. Функцію користувача оголошують так:

```
<тип функції> <назва функції> (<список формальних параметрів>);
```

де тип функції – це тип даного, який функція повертає в основну програму. Тип функції можна не зазначати. За замовчуванням функція повертає у програму дане цілого типу ***int***. Функцію, яка не повертає у програму жодного результату, оголошують з типом ***void***. Для функції, яка не залежить від жодних параметрів, у круглих дужках записують службове слово ***void***.

Назву функції надає користувач за правилом створення ідентифікаторів.

У списку формальних параметрів через кому записують змінні, зазначаючи їхні типи. Тип необхідно зазначати для кожної змінної окремо. Імена змінних можна опускати. Якщо функція не набуває жодних значень, то список формальних параметрів може бути відсутній. Круглі дужки опускати не можна:

```
float Suma(int kil, float cina);
void dilennya(float, float);
kod(intk1, intk2);
double loto(void);
```

У цьому випадку оголошена функція ***Suma*** типу ***float***, яка залежить від двох змінних: перша змінна цілого типу ***int***, друга – типу ***float***. Функція ***dilennya*** залежить від двох змінних дійсного типу ***float*** і не повертає у програму

жодного значення. Ціла функція ***kod*** залежить від двох змінних цілого типу, дійсна функція ***loto*** типу ***double*** не залежить від жодних параметрів.

Під час оголошення можна відразу ініціалізувати формальні параметри функції, тобто надавати їм певних значень. Такі значення називаються ***значеннями за замовчуванням***. їх записують у кінці списку. Значення таких параметрів у програмі можна змінювати. Враховуючи це, розглянемо ще один спосіб оголошення функцій:

```
float Suma(int ki1, float cina = 2.5);
void dilennya(float v = 1.2, float n = 3);
kod(int k1, int k2 = 5);
```

Опис функцій користувача. Опис функції складається із заголовка *без крапки з комою* і тіла функції, записаного у фігурних дужках, а саме:

```
<тип функції> <назва функції>(<список формальних параметрів>)
{
    <тіло функції>;
    return (<вираз>);
}
```

У тілі функції записують команди, які задають дію функції. Результат виконання функції повертається в основну програму (у точку виклику) за допомогою ***виразу*** командою ***return***. Тип ***виразу*** має збігатися з типом функції. У тілі функцій типу ***void*** команду ***return*** не зазначають. У команді ***return*** круглі дужки можна не писати.

Зауваження. Функцію можна описувати і на початку програми. У такому випадку декларувати її не потрібно.

Виклик функцій користувача. До функції користувача звертаються з розділу команд основної програми (функції ***main()***) або з іншої функції. Виклик функцій можна виконати двояко: або командою виклику, або з виразів так:

```
<назва функції>(<список фактичних параметрів>)
```

Список фактичних параметрів може містити сталі, змінні, посилання, вказівники, вирази. Списки формальних і фактичних параметрів мають бути узгодженими за типами та кількістю елементів. Якщо у списку формальних параметрів є проініціалізовані змінні, то у списку фактичних параметрів ці змінні можуть бути відсутні, їм будуть надані значення за замовчуванням.

Оголошення функцій при допомозі макросів: ***Директива*** ***#define*** має подвійне значення. По-перше, вона може задати стале значення (оголошує сталу). Наприклад, якщо у програмі задано ***#define N 25***, то ***N*** під час виконання програми матиме значення ***25.*** По-друге, вона дає змогу описати ***макроси*** - короткі команди (переозначити команди) чи записати функції, наприклад, так:

```
#define D(a, b, c) ((b) * (b) - 4 * (a) * (c)).
```

Необхідно зауважити, що текст макроса рекомендовано брати у круглі дужки для правильної інтерпретапції результуючого виразу.

Тепер скрізь для обчислення дискримінанта замість команди

```
d = b*b-4*a*c
```

можна записувати

```
d = D(a, b ,c).
```

Директива **`#undef`** скасовує дію директиви **`#define`**. Наприклад,

```
#define D(a,b,c) ((b) * (b) - 4 * (a) * (c))
#undef D
#define D(a,b,c) ((a) * (b) * (c))
```

Варіанти завдань лабораторної роботи

Таблиця 1.

№ п/п	Функція F	Підпрограма-функція
1.	$F = \frac{p(x) + p(y)}{p(z)} \cos x$	$p(k) = \frac{\ln^2 ak + \operatorname{tg} ak}{ak}$
2.	$F = \frac{\sqrt{p(x)}}{\sin x} + \frac{\operatorname{tg} p(y)}{\lg y} - p(z)$	$p(k) = \frac{\sqrt{k^2 + a^2}}{ak} + ak$
3.	$F = \frac{\sin p(x)}{\sqrt{x + \lvert p(y) \rvert}} + \frac{p^2(z)}{2x} - y \lg x$	$p(k) = \operatorname{tg}(ak) + \sin(k)$
4.	$F = \frac{ap^2(x) + bp^2(y)}{xy} - \sqrt{p(z)}$	$p(k) = \frac{\lg(ak + k)}{k^2}$
5.	$F = \frac{p(x)^{3/2}}{\ln x} + \frac{xp(y) - yp(z)}{\sqrt[3]{xy}}$	$p(k) = \operatorname{tg} k + \lg(ak + k)$
6.	$F = \frac{\lg p(x) - \lg x \lg y}{\sqrt[3]{\operatorname{tg} p(y)}} - \sqrt{p(z)}$	$p(k) = \frac{(k + a)^4}{\sqrt[4]{(ak + 1)}} a - k$
7.	$F = e^{p(x)} + \sqrt{\ln(x^2 p(y))} + p(z)$	$p(k) = \ln^2 (a + k)^2 + ak^2$
8.	$F = \lg p(x) + e^x - \sqrt{p(y)} + p(z)$	$p(k) = \frac{1 + \cos ak}{1 + \sin ak}$
9.	$F = \lg(p(x) + p(y)) - \ln p(z)$	$p(k) = \frac{\operatorname{tg} ak + \operatorname{ctg} ak}{a + k}$
10.	$F = \frac{\sqrt[3]{p(x) + xy}}{p(y) + x} + \frac{p(z) + y}{\sqrt{z}}$	$p(k) = \frac{\sin^2 k + a}{\cos^2 k + 1}$
11.	$F = \frac{p^2(x) + p(y)x + y}{p(z)xy}$	$p(k) = \frac{\lg ak + \ln ak}{a}$

12.	$F = \frac{\sqrt{p(x)} + p(y)}{1+y} - \ln p(z)$	$p(k) = \frac{\operatorname{tg} ak + 1}{\operatorname{tg} k + a}$
13.	$F = \lg p(x) + \operatorname{tg} p(y) + e^{p(z)}$	$p(k) = \frac{ak^2 + ka^2}{ak+1}$
14.	$F = \operatorname{tg} p(x) + \frac{\sin^2 p(y)}{\cos p(z)}$	$p(k) = \arcsin ak + \operatorname{arctg} k$
15.	$F = \cos^2 p(x) + \sin^2 p(y) + p(z)$	$p(k) = \arccos ak + \arcsin k$
16.	$F = \frac{1 + \lg p(x)}{\ln p(y)} + p(z)$	$p(k) = \frac{\sqrt[3]{ak^2} + \sqrt[3]{k^2}}{ak}$
17.	$F = \frac{p^2(x) + p(y)}{2 + p(z)} e^a$	$p(k) = \sqrt[3]{ak} + \sqrt[3]{k}$
18.	$F = p^2(x) + \frac{p(y)}{xy} + e^{p(z)}$	$p(k) = \frac{k}{a^2} + \frac{a}{k^2} + ak$
19.	$F = \sqrt[3]{p(x) + p(z)x - p(y)}$	$p(k) = (e^{ak} + e^2) \ln ak$
20.	$F = \operatorname{tg} p(x) + \frac{\sin p(y) + x}{\cos p(z) + y}$	$p(k) = \frac{\lg ak}{\ln a} - ak$
21.	$F = \lg(p(x) + p(y)) - \ln p(z)$	$p(k) = \operatorname{arctg} k + \frac{\arcsin a}{k}$
22.	$F = \frac{p(x)^2 - 1}{p(y)^{3,2} + xy} + p(z)$	$p(k) = \frac{e^a + k}{1 + e^k}$
23.	$F = x^2 + \sqrt{p(x)} + y\sqrt[3]{p(y) + p(z)}$	$p(k) = \frac{\lg a + \ln ak}{ak}$
24.	$F = xyp(x) + p(y)e^{p(z)}$	$p(k) = \frac{\sin ak + \cos ak}{1 + ak}$
25.	$F = \frac{\sqrt{p(y) + \sqrt[3]{p(x)}}}{\ln p(z)}$	$p(k) = \frac{a^2 + k^2 + ak}{\ln ak}$
26.	$F = e^{p(x)} + p^2(y) + \sqrt{p(z)}$	$p(k) = k^a + a^k$
27.	$F = p(x)e^x - p^2(y)\ e^y + p(z)$	$p(k) = \ln k + \lg ak + k$
28.	$F = xp(x) + p(y)e^{p(z)}$	$p(k) = \frac{e^a + e^k}{ak + k^2}$
29.	$F = \lg p(x) + e^{p(y)} + p(z)$	$p(k) = \frac{a^2 k + k^2 a}{a + k}$
30.	$F = (\sqrt{p(x)} + \sqrt[3]{p(y)})e^{p(z)}$	$p(k) = \frac{\sin ak + \cos ak}{ak}$

Таблиця 2.

№ п/п	a	x	y	z
1.	0.11	2.13	0.17	0.23
2.	1.42	1.39	0.68	0.98
3.	3.18	2.64	3.27	2.64
4.	7.29	3.28	4.38	2.59
5.	0.38	0.34	0.98	0.58
6.	8.95	9.14	7.83	5.47
7.	3.14	4.38	5.32	8.96
8.	7.43	6.53	0.74	8.38
9.	0.02	0.30	0.98	0.38
10.	0.42	0.13	0.65	0.02
11.	4.76	6.54	8.20	5.89
12.	0.46	0.63	1.00	0.87
13.	0,38	0,78	0,47	0,28
14.	1.03	0.57	0.98	0.26
15.	0.01	1.20	3.05	2.00
16.	9.64	7.46	4.54	8.54
17.	5.46	8.65	7.59	8.03
18.	5.36	2.07	3.94	7.78
19.	3.28	3.14	7.28	3.941
20.	1.00	0.83	2.16	0.93
21.	2.05	0.33	0.74	0.96
22.	7.83	4.15	5.37	4.01
23.	2.64	5.46	8.34	6.55
24.	1.30	0.98	0.99	0.97
25.	5.98	3.97	4.32	5.43
26.	9.74	8.00	4.03	7.00
27.	4.39	9.84	4.00	4.29
28.	4.59	4.87	9.00	3.21
29.	2.59	10.1	23.1	12.4
30.	2.11	1.00	0.93	1.00

Зразок програми лабораторної роботи

$F = (\sqrt{p(x)} + \sqrt[3]{p(y)})e^{p(z)}$		$p(k) = \frac{\sin ak + \cos ak}{ak}$	
a	***x***	***y***	***z***
2.11	***1.00***	***0.93***	***1.00***

```
// Лабораторна робота №8 варіант 31 група СІ-12 Гордієнко М.В.
#include <iostream>
#include <cmath>
using  namespace std;
```

Спеціальності: "Комп'ютерні науки", "Системний аналіз та наука про дані" "Інформаційні системи і технології", "Комп'ютерна інженерія"

```
const float a=2.11;

#define p(k) (sin(a*k)+cos(a*k)/(a*k))

int main()
{  system("color F0"); // Встановити білий фон і чорний текст
   setlocale(0, "ukr");
   double x,y,z,F;
    cout<<"x="; cin>>x;
    cout<<"y="; cin>>y;
    cout<<"z="; cin>>z;
    F=(sqrt(p(x))+pow(p(y),1.0/3))*exp(p(z));
    cout<<"F= "<<F<<endl;
    return 0;
}

// Лабораторна робота №8 варіант 31 група СІ-12 Гордієнко М.В.
#include <iostream>
#include <cmath>
using  namespace std;

const float a=2.11;

double p(double k)
{   double p1;
       p1= sin(a*k)+cos(a*k)/(a*k);
       return(p1);
}
int main()
{  system("color F0"); // Встановити білий фон і чорний текст
   setlocale(0, "ukr");
    double x,y,z,F;
    cout<<"x="; cin>>x;
    cout<<"y="; cin>>y;
    cout<<"z="; cin>>z;
    F=(sqrt(p(x))+pow(p(y),1.0/3))*exp(p(z));
    cout<<"F= "<<F<<endl;
    return 0;
}
```

Лабораторна робота № 9
(2 год.)

Тема: Використання функцій користувача при роботі з масивами.

Мета роботи: Оволодіти практичними навичками розробки та програмування обчислювального процесу обробки масивів з використанням функцій користувача. Засвоїти їх опис та звернення до них у програмах.

За час виконання лабораторної роботи студент повинен освоїти:

- опис функцій користувача з використанням вказівників;
- використання у виразах звернень до функцій користувача;
- взаємодію фактичних аргументів та формальних параметрів через вказівники при зверненні до функцій користувача;

- використання локальних та глобальних змінних;
- сторонні ефекти при використанні функцій користувача;
- вивід результатів виконаної роботи.

Завдання на лабораторну роботу: Скласти схему алгоритму та програму на С++ обчислення значення змінної ***F*** з використанням функцій користувача обчислення елементів масиву x_i (таблиця 1). Вхідні дані з таблиці 3 вибирати згідно даних варіанту, що заданий в таблиці 2. Дані ввести з клавіатури чи через опис типізованих констант. Вивід результатів передбачити на екран

Порядок виконання лабораторної роботи:

- Використовуючи відповідний ярлик, викликати інтегроване середовище С++;
- ввести текст програми (в першій стрічці обов'язково має бути коментар, в якому вказати номер роботи, групу та прізвище виконавця);
- текст програми записати у власну папку: (***Prizvyshche_L9v31***);
- відкомпілювати програму, виправляючи при цьому можливі помилки;
- відлагоджену програму виконати, записуючи отриманий результат;
- оформити звіт про виконану роботу.

Теоретичні відомості

Глобальні та локальні змінні. Операція видимості.

Під-час оголошення змінних в оперативній пам'яті комп'ютера резервується місце для зберігання їхніх значень. Обсяг наданої пам'яті залежить від типів змінної та компілятора. Кожна змінна характеризується областю дії та областю видимості. ***Область дії*** – це частина програми, де змінна придатна для повноцінного опрацювання. ***Область видимості*** – це частина програми, де змінна оголошена або, де до неї можна отримати доступ за допомогою операції надання видимості, що позначається "***::***". Змінні можна оголошувати у блоці команд, у тілі деякої функції або поза всіма функціями (**глобальні змінні**). Області дії та можливої видимості, час дії змінної у певній програмі залежать від того, де і як оголошені змінні.

Змінні, які оголошені у тілі деякої функції або у блоці, називаються ***локальними.*** Область дії локальних змінних поширюється лише на відповідну функцію чи блок. Під час виходу із функції (блока) частина оперативної пам'яті, відведеної під локальні змінні, вивільняється, тобто закінчується область дії змінної. Тому у різних функціях можна використовувати змінні з однаковими іменами.

Змінні, які описані поза всіма функціями, тобто на початку програми або у файлі заголовка, називають ***глобальними***. Вони видимі та доступні під час виконання всієї програми. До таких змінних можна звернутись з будь-якої функції чи блока. Функція може повертати значення у основну програму за допомогою команди ***return, вказівників, посилань*** або через ***глобальні змінні***. Значення глобальних змінних можна змінювати у тілі функцій. З огляду на це у великій програмі важко проаналізувати значення

глобальної змінної на певному етапі її виконання. Тому варто якомога менше використовувати глобальні змінні, бо це ускладнює розуміння програми.

Звернутись до елементів масиву можна двома способами: за допомогою імені масиву або використовуючи ***вказівники***.

Нумерація елементів масиву завжди починається з нуля. Щоб звернутись до деякого елемента, необхідно зазначити ім'я масиву, а у квадратних дужках – його номер. Наприклад, змінна **`stud[2]`** є третім елементом масиву **`stud`**, а **`stud[4]`** – п'ятим, оскільки масив **`stud`** має елементи

`stud[0], stud[1], stud[2], stud[3] та stud[4].`

За замовчуванням усім елементам масиву надається значення **0**. Масив можна ініціалізувати повністю або частково відразу під час його оголошення, записуючи значення елементів через кому у фігурних дужках. Наприклад,

```
int Stud[] = {2, 10, 5, 7, 3};
```

Назва масиву **`stud`** є вказівником на його перший елемент. Змінна **`*stud`** містить значення першого елемента масиву (елемента **`stud[0]`**). Оскільки всі елементи масиву розміщені у послідовних комірках оперативної пам'яті комп'ютера, то вказівник **`*(stud + 1)`** вказуватиме на другий елемент масиву (зміщення відносно вказівника `*stud` на одну одиницю пам'яті), а вказівник **`*(stud + 4)`** – на п'ятий (зміщення на чотири одиниці).

Варіанти завдань лабораторної роботи:

Сталі a = 0.98, c = 1.1.

Таблиця 1**.**

№ п/п	Вираз для змінної ***F***	Функція користувача для масиву x_i
1.	$F = \prod_{i=1}^{N} x_i + a\sum_{i=1}^{n} x_i$	$x_i = a\sum_{j=1}^{m} \sin b_{ij} + c$
2.	$F = \frac{\sum_{i=1}^{n} x_i}{a} + \sum_{i=1}^{n} a^2 x_i$	$x_i = \sum_{j=1}^{m} \frac{ab_{ij}}{cb_{ij}^2}$
3.	$F = a + \prod_{i=1}^{n} x_i$	$x_i = \sum_{j=1}^{m} \left(ab_{ij} + cb_{ij}\right)$
4.	$F = a\sum_{i=1}^{n} x_i$	$x_i = a\prod_{j=1}^{m} b_{ij} + c\sum_{j=1}^{m} b_{ij}$

5.	$F = a\dfrac{\sum_{i=1}^{n} x_i}{\prod_{i=1}^{n} x_i}$	$x_i = a\sum_{j=1}^{m}\left(b_{ij}^2 + cb_{ij}\right)$
6.	$F = a\sqrt{\sum_{i=1}^{n} x_i}$	$x_i = c\sum_{j=1}^{m}\dfrac{b_{ij} + a}{\sqrt{b_{ij}}}$
7.	$F = \sum_{i=1}^{n}\left(x_i^2 + ax_i\right)$	$x_i = \sum_{j=1}^{m}\left(\sqrt{b_{ij}} + cb_{ij}\right) + a$
8.	$F = \prod_{i=1}^{n}\left(x_i^2 + a^2\right)$	$x_i = \prod_{j=1}^{m}\left(c\sqrt{b_{ij}} + ab_{ij}\right)$
9.	$F = \sum_{i=1}^{n}\left(a + x_i + x_i^{0,5}\right)$	$x_i = \prod_{j=1}^{m}\left(ab_{ij}^2 - c\right)$
10.	$F = \sqrt{\prod_{i=1}^{n}\left(x_i + a\right)}$	$x_i = \dfrac{\sum_{j=1}^{m}\left(b_{ij} + a\right)}{\sum_{j=1}^{m}\left(b_{ij} + c\right)}$
11.	$F = a\sum_{i=1}^{n} x_i - \dfrac{1}{\sqrt{\prod_{i=1}^{n} x_i}}$	$x_i = \dfrac{\prod_{j=1}^{m}\left(b_{ij} + c\right)}{\prod_{j=1}^{m}\left(b_{ij}^2 + a\right)}$
12.	$F = a + \prod_{i=1}^{n}\left(x_i^2 + 1\right)$	$x_i = \sum_{j=1}^{m}\dfrac{ab_{ij}}{c + b_{ij}}$
13.	$F = \dfrac{\prod_{i=1}^{n}\left(x_i + x_i^2 + a\right)}{\sum_{i=1}^{n} x_i + 1}$	$x_i = \prod_{j=1}^{m}\left(\dfrac{a + b_{ij}^2}{c}\right)$
14.	$F = \sum_{i=1}^{n}\left(x_i^2 + a\sqrt{x_i}\right)\sum_{i=1}^{n} x_i$	$x_i = \sum_{j=1}^{m}\left(\dfrac{\sqrt{b_{ij} + c}}{b_{ij}^2}\right)$
15.	$F = \sqrt[3]{a\sum_{i=1}^{n} x_i + \sum_{i=1}^{n} x_i^2}$	$x_i = \prod_{j=1}^{m}\left(c + \sqrt[3]{b_{ij}}\right)$

16.	$F = \sum_{i=1}^{n} a x_i$	$x_i = \frac{\prod_{j=1}^{m} \left(a + b_{ij}\right)}{\prod_{j=1}^{m} \left(b_{ij} + c\right) + 1}$
17.	$F = \sum_{i=1}^{n} \left(x_i + \frac{\sqrt{x_i}}{a} \right) + \prod_{i=1}^{n} x_i$	$x_i = \sum_{j=1}^{m} \left(\frac{c b_{ij} + \cos b_{ij}}{a} \right)$
18.	$F = a \sum_{i=1}^{n} \sqrt{x_i + x_i^2}$	$x_i = \sum_{j=1}^{m} \left(a \sin b_{ij} + c \cos b_{ij} \right)$
19.	$F = \prod_{i=1}^{n} \left(x_i^2 + a \sqrt{x_i} \right)$	$x_i = \prod_{j=1}^{m} \left(a\, tg\left(c b_{ij} \right) \right)$
20.	$F = \frac{\prod_{i=1}^{n} \left(x_i^2 + a^2 \right)}{\sum_{i=1}^{n} \left(a - \sqrt{x_i} \right)}$	$x_i = \prod_{j=1}^{m} \left(c e^{b_{ij}} + a \right)$
21.	$F = e^{\sum_{i=1}^{n} x_i} + c$	$x_i = a^{\sum_{j=1}^{m} b_{ij}} + c$
22.	$F = \frac{1 + e^{\prod_{i=1}^{n} x_j}}{2}$	$x_i = \sum_{j=1}^{m} \frac{a - e^{b_{ij}}}{c}$
23.	$F = a^{\sum_{i=1}^{n} x_i} + 1$	$x_i = \prod_{j=1}^{m} \left(c^{b_{ij}} + a \right)$
24.	$F = a^{\prod_{i=1}^{n} x_i} + 1$	$x_i = \sum_{j=1}^{m} \left(a^{b_{ij}} + c b_{ij} \right)$
25.	$F = \mathrm{tg}\left(\sum_{i=1}^{n} x_i \right)$	$x_i = \prod_{j=1}^{m} \left(a + \cos\left(c b_{ij} \right) \right)$
26.	$F = \frac{na + \sum_{i=1}^{n} x_i}{\sum_{i=1}^{n} x_i}$	$x_i = e^{\sum_{j=1}^{m} (b_{ij} + a)} + c$
27.	$F = a \prod_{i=1}^{n} \left(x_i + 1 \right) + 2$	$x_i = c e^{\left(a + \sum_{j=1}^{m} b_{ij} \right)}$
28.	$F = \prod_{i=1}^{n} \left(a + x_i \right) + \prod_{i=1}^{n} \left(a + x_i^2 \right)$	$x_i = c \sum_{j=1}^{m} b_{ij}^2 + a^2$

29.	$F = \sum_{i=1}^{n} x_i + \frac{\sum_{i=1}^{n} x_i}{a}$	$x_i = a\sum_{j=1}^{m} b_{ij} + e^{c\sum_{j=1}^{m} b_{ij}}$
30.	$F = a + \frac{\sum_{i=1}^{n} x_i}{\prod_{i=1}^{n} x_i}$	$x_i = \prod_{j=1}^{m}\left(\frac{ab_{ij}^2 + c}{b_{ij}}\right)$

Таблиця 2.

№ п/п	Кількість рядків **n**	Кількість стовпців **m**	Початковий номер рядка	Початковий номер стовпця
1.	3	10	1	1
2.	4	9	2	2
3.	5	8	1	3
4.	3	9	2	1
5.	4	8	1	2
6.	5	7	2	3
7.	3	8	1	2
8.	4	7	2	3
9.	5	6	1	4
10.	3	5	2	2
11.	4	4	1	3
12.	5	3	2	4
13.	3	7	1	2
14.	4	6	2	3
15.	5	5	1	4
16.	3	6	2	3
17.	4	5	1	4
18.	5	4	2	5
19.	3	5	1	3
20.	4	4	2	2
21.	5	6	1	3
22.	3	7	2	2
23.	4	6	1	1
24.	5	7	2	3
25.	3	6	1	1
26.	4	5	2	2
27.	5	10	1	1
28.	3	9	2	1
29.	4	8	1	1
30.	5	9	2	1

Таблиця 3

дк	с т о в п ц і

Спеціальності: "Комп'ютерні науки", "Системний аналіз та наука про дані" "Інформаційні системи і технології", "Комп'ютерна інженерія"

	1	2	3	4	5	6	7	8	9	10
1	1.3	1.9	2.6	3.7	2.1	2.7	3.6	2.9	1.2	2.0
2	2.4	3.8	3.6	2.6	1.7	1.5	1.4	1.8	1.3	1.5
3	3.5	2.9	1.4	4.3	3.8	1.4	1.5	1.5	0.5	1.4
4	1.2	3.2	1.3	2.8	2.4	2.7	2.5	1.3	0.8	0.5
5	2.3	2.3	2.3	2.5	1.4	3.2	3.8	2.6	1.1	0.8
6	1.9	4.2	1.5	3.2	2.9	1.4	4.6	1.0	0.7	0.7

Затіненням виділено дані масиву для програми зразка.

Зразок програми лабораторної роботи

$F = a + \frac{\sum_{i=1}^{n} x_i}{\prod_{i=1}^{n} x_i}$	$x_i = \prod_{j=1}^{m} \left(\frac{ab_{ij}^2 + c}{b_{ij}} \right)$

Кількість рядків **n**	Кількість стовпців **m**	Початковий номер рядка	Початковий номер стовпця
5	9	2	1

```
// Лабораторна робота №9 варіант 31 група СІ-12 Гордієнко М.В.
#include <iostream>
#include <math.h>
using  namespace std;
const float a=0.98, c=1.1;
const int n=5, m=9;
void fx (double b[n][m], double *x)
{
      int i,j;
      for (i=0;i<n;i++)
      {
         x[i]=1;
         for (j=0;j<m;j++)
         x[i]*=(a*pow(b[i][j],2.0)+c)/b[i][j];
      }
}
double ff(double b[n][m])
{
  double x[n], ch=0, zn=1, f;
  int i;
        fx(b, x);
        for (i=0;i<n;i++)
```

```
        {
            ch+=x[i];
            zn*=x[i];
        }
        f=a + ch/zn;
        return f;
}
int main()
{  system("color F0"); // Встановити білий фон і чорний текст
   setlocale(0, "ukr");
    double b[n][m] = { 2.4,3.8,3.6,2.6,1.7,1.5,1.4,1.8,1.3,
                       3.5,2.9,1.4,4.3,3.8,1.4,1.5,1.5,0.5,
                       1.2,3.2,1.3,2.8,2.4,2.7,2.5,1.3,0.8,
                       2.3,2.3,2.3,2.5,1.4,3.2,3.8,2.6,1.1,
                       1.9,4.2,1.5,3.2,2.9,1.4,4.6,1.0,0.7 };
    cout<<"F= "<<ff(b)<<endl;
    return 0;
}
```

Лабораторна робота № 10
(2 год.)

Тема: **Робота із рядками символів**.

Мета роботи: Оволодіти практичними навичками розробки та програмування обчислювального процесу з використанням даних рядкового типу. Засвоїти застосування функцій роботи із рядковими величинами.

За час виконання лабораторної роботи студент повинен освоїти:

- опис рядкових констант та змінних;
- дії над рядковими величинами;
- застосування функцій роботи із рядковими величинами;
- вивід результатів виконаної роботи.

Завдання для виконання роботи: Скласти програму завдання, вказаного у таблиці 1. Текст вибрати самостійно, ввести і отримати результат Під текстом в даному завданні слід розуміти послідовність, що містить від **1** до **255** символів. Групи символів, що розділені пропусками і не містять пропусків всередині, назвемо словами. У звіт записати введений текст і одержаний результат.

Порядок виконання лабораторної роботи:

- використовуючи відповідний ярлик, викликати інтегроване середовище ***C++;***
- ввести текст програми (в першій стрічці обов'язково має бути коментар, в якому вказати номер роботи, групу та прізвище виконавця);
- текст програми записати у власну папку (***Prizvyshche_L10v31***);
- відкомпілювати програму, виправляючи при цьому можливі помилки;
- відлагоджену програм виконати, записуючи отриманий результат;
- оформити звіт про виконану роботу.

Теоретичні відомості

Рядки символів і дії з ними. На відміну від інших мов програмування у **C++** не визначено спеціального типу для опрацювання рядків. Рядок символів

розглядається як масив елементів типу ***char,*** який закінчується символом "**\0**" (нуль-символ) що є ознакою кінця рядка. Такі рядки називають ***ASCII-рядками.*** Сталі типу рядок записують у лапках, наприклад, ***"Тернопільська політехніка", "студенти", " "*** - рядок, що містить один символ-пропуск. У сталих рядках нуль-символ дописується автоматично.

Масиви символів оголошують так:

```
char <назва рядка>[довжина рядка];
```

Якщо масив символів оголошують й ініціалізують одночасно, то довжину можна не зазначати, компілятор визначить її. Оскільки рядки є масивами символів, то назва рядка є *вказівником* на його перший елемент (на перший символ).

Приклад 1. Розглянемо оголошення та ініціалізацію рядків

```
const char text1[] = "Ми вивчаємо програмування";
char slovo[] = "University";
char fraza1[11], fraza2[40];
```

Тут оголошено сталу ***text1***, яка має значення "***Ми вивчаємо програмування"***, символьні масиви: ***slovo*** (без зазначення розміру), ***fraza1*** (може містити до 10 символів) та ***fraza2*** (до 39 символів).

Символьний масив ***slovo*** ще можна оголосити так:

```
char slovo[11] = "University";
```

або так

```
char slovo[]={'U','n','i','v','e','r','s','i','t','y','\0'};
```

Тут потрібно ***вручну*** записати нуль-символ, інакше компілятор трактуватиме змінну ***slovo*** не як рядок, а як масив.

Рядки можна опрацьовувати посимвольно за допомогою вказівників або назви масиву, наприклад, так:

```
for (int n = 0; n < 11; n++)   *(fraza1+n) = *(slovo+n);
cout << fraza1;
```

Змінній ***fraza1*** надається значення "***University***" і ця фраза виводиться на екран. Інакше це можна зробити так:

```
for (int n = 0; n < 11; n++)
fraza1[n] = slovo[n];
cout << fraza1;
```

Увести весь масив символів можна:

```
cin >> <назва масиву>;
```

Якщо рядок даних містить символ пропуску, то команда ***cin>>*** зчитає дані лише до першого пропуску.

Вивести значення рядка на екран можна за допомогою команди

```
cout << <назва рядка>;
```

Посимвольно вводити чи виводити елементи рядка можна за допомогою команд циклу ***for*** або ***while.*** Наприклад,

```
for (int n = 0; n < 11; n++) cin >> *(fraza1 + n);
```

В кінці рядка необхідно поставити нуль-символ, тобто ****(fraza1 +n + 1) = '\0';***

У бібліотеці ***conio.h*** визначені стандартні функції введення-виведення рядків. Наприклад, ***getc(), getchar()*** зчитують по одному символу рядка, введеного з клавіатури, ***putc()*** та ***putchar()*** виводять окремі символи рядка тощо. У бібліотеці ***stdio.h*** описані функції для введення ***gets()*** та виведення ***puts()*** усього рядка.

Функції для опрацювання рядків. Для опрацювання масивів символів у мові ***C++*** є стандартні функції, які описані у модулі ***string.h***. Розглянемо деякі з них.

```
char Ternopil[] = "Тернопільська політехніка",
Un[30] = "НУ ",
r1[30] = "";
char  *p; int n;
```

strlen(<рядок>) - визначає фактичну кількість символів у рядку, застосовується у виразах;

`n = strlen(Ternopil)`	`n = 25`

strcat(r1, r2) - команда з'єднання рядків ***r1, r2*** в один рядок, результат присвоює змінній r1;

`strcat(Un, Ternopil)`	`Un = "НУ Тернопільська політехніка"`

strncat(r1, r2, n) – до змінної ***r1*** додає перших ***n*** символів рядка ***r2***,

`strncat(Un, Ternopil, 13)`	`r1 = "НУ Тернопільська"`

strcpy(r1, r2) - копіює символи з рядка ***r2*** в рядок ***r1***, команда;

`strcpy(r1, Ternopil)`	`r1 = "Тернопільська політехніка"`

strncpy(r1, r2, n) - копіює перших ***n*** символів рядка ***r2*** в рядок ***r1***, команда;

`strncpy(r1, Ternopil, 13)`	`r1 = "Тернопільська"`

strchr(r1, <символ>) - визначає перше входження деякого символу у рядок ***r1*** так: повертає рядок, який починається від першого входження заданого символу до кінця рядка ***r1***, застосовується у виразах;

`p = strchr(Ternopil, 'п')`	`p = "пільська політехніка"`

strrchr(r1, <символ>) -: повертає рядок, який починається від останнього входження заданого символу до кінця рядка ***r1***, застосовується у виразах;

p = strrchr(Ternopil, 'i')	***p = "іка"***

strspn(r1, r2) - визначає номер першого символу, який входить у рядок ***r1,*** але не входить у рядок ***r2***, застосовується у виразах;

n = strspn(Ternopil, "Тернопіль")	***n = 10***

strstr(r1, r2) - визначає в рядку ***r1*** підрядок, що починається з першого входження рядка ***r2*** у рядок ***r1,*** застосовується у виразах;

p = strstr(Ternopil, "тех")	***p = "техніка"***

strtok(r1, r2) — визначає частину рядка **r1**, яка закінчується перед першим однаковим символом рядків ***r1*** та ***r2***;

p = strtok(Ternopil, "ckc")	***p = "Ternopil"***

strnset(r1, <символ>, n) - вставляє ***n*** разів заданий символ в рядок ***r1***, застосовується у виразах;

p = strnset(Ternopil, 'x', 14)	***p = "xxxxxxxxxxxxxxполітехніка"***

strupr(r1) - перетворює усі малі літери рядка у великі;

p = strupr("I Love You")	***p = "I LOVE YOU"***

strlwr(r1) - перетворює усі великі літери рядка у малі;

p = strlwr("I Love You")	***p = "i love you"***

strrev(r1) - записує рядок у зворотному порядку.

p = strrev("техніка")	***p = "акінхет"***

Зауваження. Функції перетворення літер ***strlwr*** і ***strupr*** діють лише для латинського алфавіту. Крім того, у деяких версіях мови ***C++*** ці функції можуть записуватись інакше: ***_strlwr***, ***_strupr***.

У модулі ***ctype.h*** оголошено групу функцій, призначених для перевірки та класифікації окремих символів (тут ***c*** – окремий символ):

isalpha(c) – перевіряє, чи ***c*** є малою або великою латинською літерою;

isdigit(c) – перевіряє, чи ***c*** є десятковою цифрою;

isalnum(c) – перевіряє, чи ***c*** є латинською літерою або десятковою цифрою;

islower(c) – перевіряє, чи ***c*** є малою латинською літерою;

isupper(c) – перевіряє, чи ***c*** є великою латинською літерою.

Рядки символів можна порівнювати між собою.

Для цього застосовують функцію

strcmp(r1, r2)

у якій результат виконання може бути рівним нулю, якщо рядки рівні між собою, додатним, якщо ***r1*** більший за ***r2*** або від'ємним, якщо ***r1*** менший за ***r2***. Два рядки порівнюються зліва направо посимвольно, причому 'A' < 'B', 'B' < 'C' тощо. Більшим уважається символ, який розміщений в алфавіті дальше (він має більший номер у таблиці кодів ***ASCII***).

Варіанти завдань лабораторної роботи:

Таблиця 1

№ п/п	Завдання
1.	Дано текст. Визначити, скільки яких і яких цифр він містить.
2.	Дано текст. Визначити, скільки і яких букв він містить.
3.	Дано текст. Визначити, чи є в ньому символи +,-,=,* і скільки їх.
4.	Дано число. Кожну групу символів *120* у цьому числі замінити на символи *478*.
5.	Дано текст. Визначити, чи містить він символи, відмінні від латинських букв і цифр і які саме.
6.	Дано текст. Кожну малу латинську літеру замінити на цифру 1, а велику на цифру 5.
7.	Дано текст. Якщо в ньому є цифра 2, то кожен символ перед цією цифрою замінити на букву **a** (якщо цифри 2 йдуть підряд 22 222 тощо. то заміну робити тільки перед першою цифрою).
8.	Дано текст. Якщо в ньому є символ +, то кожен наступний після + символ замінити символом -, а після - на + відповідно.
9.	Дано текст. Якщо в ньому є малі латинські букви то кожен символ, що знаходиться за латинською буквою, замінити крапкою.
10.	Дано текст. Визначити скільки і яких букв є в назві місяців року.
11.	Дано текст. Порахувати кількість слів у тексті та визначити довжину найдовшого слова.
12.	Дано текст. Знайти всі слова, що містять букву **a** і скільки раз.
13.	Дано текст. Знайти всі слова, що не містять букву **o.**
14.	Дано текст. Знайти всі слова, що починаються з букви **k** і закінчуються буквою **a**.
15.	Дано текст. У словах, що закінчуються на ***ing*** замінити це закінчення на ***ed***.
16.	Дано текст. Визначити, чи є в тексті слово ***one*** і їх кількість.
17.	Дано текст. Порахувати, скільки латинських букв а є в перших трьох словах.

Спеціальності: "Комп'ютерні науки", "Системний аналіз та наука про дані" "Інформаційні системи і технології", "Комп'ютерна інженерія"

18.	Дано натуральне число n. Отримати це ж число у вигляді комбінації цифр і пропусків: пропуски розділяють між собою групи по три цифри, починаючи справа. Наприклад: якщо n=12345791, то одержимо n=12 345 791.
19.	Дано текст. Визначити кількість слів, в які буква а входить не менше двох разів.
20.	Дано текст. Знайти найдовше слово.
21.	Дано текст. Якщо першим символом тексту є цифра, то всі цифри, що входять в текст, замінити символом *. Всі слова записати згодом в зворотньому порядку.
22.	Дано текст. Всі маленькі приголосні літери замінити на великі.
23.	Дано текст. Витерти всі голосні літери, що зустрічаються в тексті.
24.	Дано текст. Після кожної цифри **1** вставити символи ***one***.
25.	Дано текст. Визначити довжину найкоротшого слова.
26.	Дано текст. Визначити кількість слів, що містять чотири символи.
27.	Дано текст. Для кожного із слів вказати, скільки разів воно зустрічається у тексті.
28.	Дано текст. Визначити найбільшу кількість цифр, що йдуть в ньому підряд.
29.	Дано текст. Визначити кількість слів, які починаються і закінчуються однією і тією ж буквою.
30.	Дано натуральне число **n (n<100).** Записати це число українською мовою.

Приклад:
Підрахувати кількість появи заданої букви в тексті.

```
// Лабораторна робота №10 варіант 31 група СІ-12 Гордієнко М.В.
#include <iostream>
#include <conio.h>
#include <stdio.h>
#include <string.h>

using namespace std;

int main()
{  system("color F0"); // Встановити білий фон і чорний текст
   setlocale(0, "ukr");
    char s[80],c;
    int i,k;
    cout<<"Enter string, please "; gets(s);
    cout<<"Enter symbol, please "; cin>>c;
    k=0;
    for(i=0;i<strlen(s);i++)
              if (s[i]==c) k++;
    cout<<"k="<<k<<endl;
    return 0;
}
```

Лабораторна робота № 11
(2 год.)

Тема: Робота з текстовими файлами.

Мета роботи: Оволодіти практичними навичками розробки та програмування обчислювального процесу з організацією збереження

результатів у текстових файлах. Засвоїти застосування стандартних функцій роботи з файлами.

За час виконання лабораторної роботи студент повинен освоїти:

- опис змінних файлового типу;
- організацію зв'язку між логічним та фізичним іменем файлу;
- відкриття файлу для обробки;
- застосування процедур та функцій при роботі з файлами;
- перевірка та вивід результатів виконаної роботи.

Завдання для виконання роботи:

Модифікувати раніше створені програми лабораторних робіт **№5** (передбачити вивід результатів на екран та в текстовий файл, який згодом переглянути, наприклад, програмою Блокнот) та **№9** (передбачити ввід елементів двовимірного масиву із попередньо створеного текстового файлу і вивід результату на екран)

Порядок виконання лабораторної роботи:

- використовуючи відповідний ярлик, викликати інтегроване середовище C++;
- ввести текст програми (в першій стрічці обов'язково має бути коментар, в якому вказати номер роботи, групу та прізвище виконавця);
- тексти програм записувати у власну папку (***Prizvyshche_L11v31_1***) та (***Prizvyshche_L11v31_2***) відповідно;
- відкомпілювати програму, виправляючи при цьому можливі помилки;
- відлагоджену програму виконати, записуючи отриманий результат;
- оформити звіт про виконану роботу.

Теоретичні відомості

Часто виникає потреба опрацьовувати інформацію, розміщену на зовнішніх носіях (на дисках), або виводити результати програми не на екран монітора, а у файл.

Файл — це сукупність даних, які розміщені на зовнішньому носії, зокрема, на жорсткому диску. Дані у файлі називаються елементами. Кількість даних під час опрацювання файлів не зазначають. Файли можуть містити як текстову, так і числову інформацію.

Функції, що дозволяють зчитати інформацію з файлу або спрямовувати потік виведення у файл, описані в бібліотеці ***fstream.***

- У модулі ***fstream*** також містяться описи стандартних потоків ***cout*** і ***cin***. Тому при роботі з файлами можна також здійснювати ввід з клавіатури та вивід на монітор без підключення модуля ***iostream.***

Для опрацювання файлу його необхідно відкрити, виконати потрібні дії та закрити.

Щоб зчитати вхідні дані з файлу, необхідно оголосити файлову змінну та відкрити файл для читання так (розглянемо два способи):

```
ifstream <назва файлової змінної>(<зовнішня назва>,
ознака1 |,ознака2 | ... |, ознакаN);
```

або так:

- ***ifstream <назва файлової змінної>; <назва файлової змінної>.open(<зовнішня назва>);***

Ознаки. Ознаки слугують для зміни правил доступу до файлу. Розглянемо деякі з них:

Ознака	Призначення
ios::in	Відкриває файл для читання з нього інформації. Вміст файлу
ios::out	Відкриває файл для записування
ios::app	Відкриває файл для дописування
ios::trunk	Якщо файл, який відкривають для записування вже існує, то його вміст буде вилучено
ios::nocreate	Забороняє створювати файл, який відкривають
ios::noreplace	Забороняє перезаписати існуючий

Дія команд. Підключаємо потік введення даних із файлу та налагоджуємо зв'язок між файловою змінною у програмі та файлом на зовнішньому носії. Назву зовнішнього файлу треба записувати у лапках. Ознаки слугують для визначення прав доступу до файлу (див. пункт "**Ознаки**"). Ознаки зазначати необов'язково. У випадку, коли їх задавати не потрібно, наведені дві форми запису тотожні, і можна обирати одну з них.

Відкрити зовнішній файл ***MyText.txt*** для читання з нього даних можна так:

```
ifstream MyFile("MyText.txt");
```

або так:

```
ifstream MyFile; MyFile.open("MyText.txt");
```

Тут ***MyFile*** - це файлова змінна, що асоціюється у програмі з відповідним зовнішнім файлом на диску. Введення даних виконують командою введення >>. Наприклад, команда

```
MyFile >> text;
```

зчитає одне дане з початку файлу ***MyText.txt*** до першого пропуску чи до символу "кінець рядка" у змінну text. Якщо треба зчитати наступне дане (яке розміщене між першим і другим пропусками), необхідно повторити попередню команду, тобто записати команди ***MyFile >> text; MyFile >> text1***; або використати команду ***MyFile >> text>> text1;.***

Отже, зчитати дані з файлу можна за допомогою команди:

<назва файлової змінної> >> <змінна1> >> <змінна2> >>…>> <змінnaN>

Після опрацювання файлу його потрібно закрити. Це роблять так:

<назва файлової змінної>.close()

Наприклад, ***MyFile.close().*** Файл ***MyText.txt*** буде закрито. Після цього файлову змінну ***MyFile*** у разі потреби можна зв'язати з іншим файлом на диску.

Визначити кінець файлу можна за допомогою функції ***eof() (eof - end of file –*** кінець файлу) так:

<назва файлової змінної>.eof()

Ця функція повертає ненульове значення, якщо досягнуто кінця файлу.

Виведення даних у файл. Відкрити файл для записування у нього даних можна так

ofstream <назва файлової змінної>(<зовнішня назва>, ознака1 |ознака2 | ... | ознакаN);

або так:

ofstream <назва файлової змінної>;
<назва файлової змінної>.open(<зовнішня назва>);

Дія команд. Підключаємо потік виведення у файл і налагоджуємо зв'язок між файловою змінною у програмі та файлом на зовнішньому носії. Ознаки зазначати не обов'язково.

Приклад. Відкрити файл ***MyText1.txt*** для запису у нього даних можна так (тут ***FileForZap*** – файлова змінна):

ofstream FileForZap("MyText1.txt");

або так:

ofstream FileForZap; FileForZap.open("MyText1.txt");

Щоб занести дані у файл, використовують команду виведення даних ***<<***. Наприклад, після виконання команд

int n = 10; FileForZap << n;

у файл ***MyText1.txt*** буде занесено число ***10***. Занести декілька даних у файл можна так само, як і вивести їх на екран: використовуючи стандартні правила команди ***<<*** та керуючі послідовності.

Отже, записати дані у файл можна так:

<назва файлової змінної> << <змінна1> << <змінна2> << ... << <зміннаN>

Приклад Після оголошення

ofstream Flags("text1.dat", ios::app, ios::noreplace);

можна дописати до кінця вже існуючого файлу ***text1.dat*** потрібну інформацію. Якщо ж оголосити потік ***Flags*** так:

ofstream Flags("text1.dat", ios::noreplace);

і спробувати щось записати у файл, то жодних дій не відбудеться, оскільки ознака ***ios::noreplace*** забороняє змінювати існуючий файл. Проте якщо цю ознаку застосувати до нового файлу, якого ще немає на диску, наприклад, записати

ofstream Flags("text3.dat", ios::noreplace);

Спеціальності: "Комп'ютерні науки", "Системний аналіз та наука про дані" "Інформаційні системи і технології", "Комп'ютерна інженерія"
то буде створено файл ***text3.dat***, в який можна буде заносити дані.

Якщо потік оголосити так:

```
ofstream Flags("text4.dat", ios::nocreate);
```

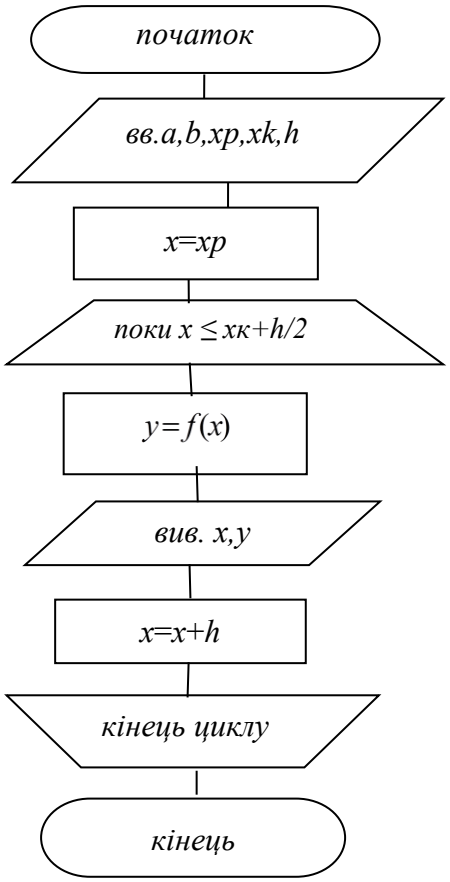
то він може бути відкритий для записування даних, якщо файл ***text4.dat*** вже існує.

Зразки програм:

Завдання1:

функція	a	b	xp	xk	h
$y = e^{0.4}\arcsin x - \lg\left\|\mathrm{tg}\left(x+\sqrt{b}\right)\right\| + \sqrt[7]{x^2+a}$	0.88	0.77	0.24	0.64	0.04

Блок схема для програми з командою while:



Програма з командою while:

```
// Лабораторна робота №11 варіант 31 група СІ-12 Гордієнко М.В.
#include <iostream>
#include <cmath>
#include<fstream>
#include <iomanip.h>

int main()
{
```

```
        system("color F0"); // Встановити білий фон і чорний
текст
        setlocale(0, "ukr");
        ofstream F_for_vyv("rez.txt");
        float a,b,xp,xk,h,x,y;
        cout << "a=  ";  cin >> a;
        cout << "b=  ";  cin >> b;
        cout << "xp= ";  cin >> xp;
        cout << "xk= ";  cin >> xk;
        cout << "h=  ";  cin >> h;
        x = xp;
        cout << "аргумент" << "\t" << "функція " << endl;
        cout << setiosflags(ios::scientific);
        while(x <= xk + h/2)
        {
         y = exp(0.4) * asin(x) - log10(fabs(tan(x + sqrt(b))))
+
             pow(x*x + a,1.0/7);
         cout << x << "\t" << "\t" << y << endl;
         F_for_vyv<<"x= "<<x<<"   y= "<<y<<endl;
         x += h;
        }
        return 0;
     }
```

Завдання2:

$F = a + \frac{\sum_{i=1}^{n} x_i}{\prod_{i=1}^{n} x_i}$	$x_i = \prod_{j=1}^{m} \left(\frac{ab_{ij}^2 + c}{b_{ij}} \right)$

Кількість рядків **n**	Кількість стовпців **m**	Початковий номер рядка	Початковий номер стовпця
5	9	2	1

```
// Лабораторна робота №11 варіант 31 група СІ-12 Гордієнко М.В.
#include <iostream>
#include <math.h>
#include<fstream>
using  namespace std;
const float a=0.98, c=1.1;
const int n=5, m=9;
void fx (double b[n][m], double *x)
{
     int i,j;
     for (i=0;i<n;i++)
     {
        x[i]=1;
        for (j=0;j<m;j++)
```

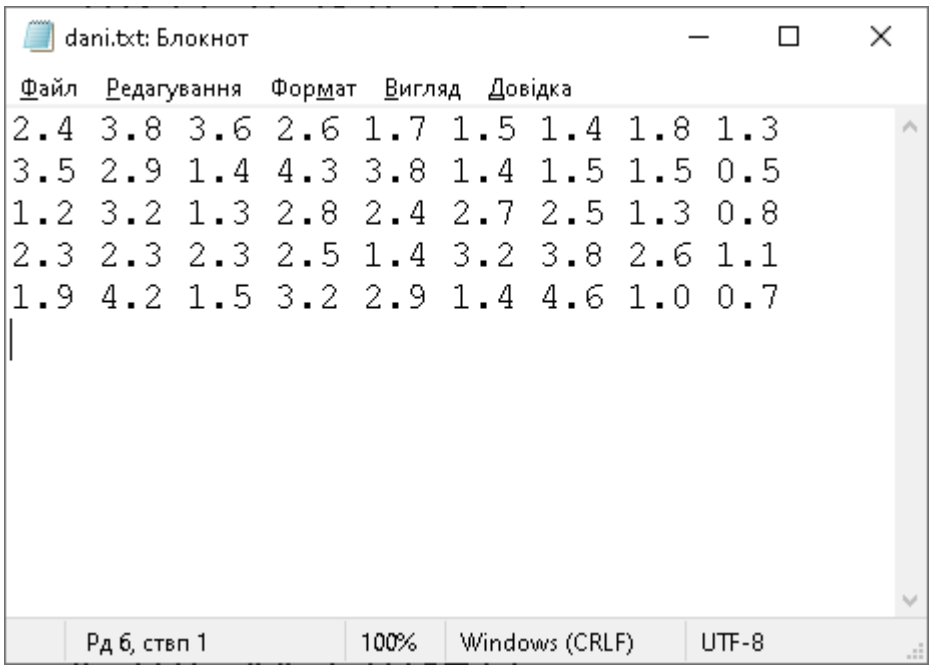
```
            x[i]*=(a*pow(b[i][j],2.0)+c)/b[i][j];
        }
}

double ff(double b[n][m])
{
  double x[n], ch=0, zn=1, f;
  int i;
        fx(b, x);
        for (i=0;i<n;i++)
        {
            ch+=x[i];
            zn*=x[i];
        }
        f=a + ch/zn;
        return f;
}

int main()
{  ifstream F_for_vv("dani.txt");
   system("color F0"); // Встановити білий фон і чорний текст
   setlocale(0, "ukr");
    double b[n][m];
    int i,j;
    for(i=0;i<n;i++)
    for(j=0;j<m;j++)
     F_for_vv>>b[i][j];

    for(i=0;i<n;i++)
    {
      for(j=0;j<m;j++)
        cout<<b[i][j]<<"  ";
      cout<<endl;
    }
    cout<<"F= "<<ff(b)<<endl;
    F_for_vv.close();
    return 0;
}
```

Файл вхідних даних підготовлений у програмі Блокнот:


dani.txt: Блокнот
Файл Редагування Формат Вигляд Довідка
2.4 3.8 3.6 2.6 1.7 1.5 1.4 1.8 1.3
3.5 2.9 1.4 4.3 3.8 1.4 1.5 1.5 0.5
1.2 3.2 1.3 2.8 2.4 2.7 2.5 1.3 0.8
2.3 2.3 2.3 2.5 1.4 3.2 3.8 2.6 1.1
1.9 4.2 1.5 3.2 2.9 1.4 4.6 1.0 0.7
Рд 6, ствп 1
100%
Windows (CRLF)
UTF-8

Спеціальності: "Комп'ютерні науки", "Системний аналіз та наука про дані" "Інформаційні системи і технології", "Комп'ютерна інженерія"

НАВЧАЛЬНЕ ВИДАННЯ

МЕТОДИЧНІ ВКАЗІВКИ
для виконання лабораторних робіт
з дисципліни "ПРОГРАМУВАННЯ"
Частина 2

для студентів спеціальностей

F3 Комп'ютерні науки,
F4 Системний аналіз та наука про дані,
F6 Інформаційні системи і технології,
F7 Комп'ютерна інженерія**.**

Укладачі: к.т.н., доцент Гладьо Ю.Б.,
старший викладач Семенишин Г.М.,
к.т.н.старший викладач Гладьо С.В..
к.т.н., доцент Гащин Н.Б.

Підписано до друку _____ Формат 60x84 1/16. Ум. др. арк. 1.
Друк лазерний. Замовлення № _____. Наклад 100 пр.
Віддруковано у видавництві ТНТУ.