**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"**

Інститут **КНІТ**

Кафедра **ПЗ**

# ЗВІТ

До лабораторної роботи № 3

**На тему:** *“РОБОТА З МАСИВАМИ В С ”*

**З дисципліни:** *“Основи програмування”*

**Лектор:**

ст. викладач кафедри ПЗ

Муха Т.О.

**Виконав:**

студ. групи ПЗ-13

Топала Т.Д.

**Прийняв:**

доц. кафедри ПЗ

Дяконюк Л.М.

«\_\_\_» \_\_\_\_\_\_ 2021 р.

∑ = \_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

Львів – 2021

**Тема роботи:** РОБОТА З МАСИВАМИ В С

**Мета роботи:** навчитися організовувати такі структури даних як масиви та освоїти

основні методи програмування алгоритмів обробки масивів даних засобами мови С.

## Індивідуальне завдання

## 1. Розробити алгоритми розв’язання індивідуальних завдань і подати їх у вигляді блок-схем.

## 2. Скласти програми на мові С у відповідності із завданнями.

## 3. Виконати обчислення по програмам.

## 4. Підготувати та здати звіт про виконання лабораторної роботи.

## Завдання 1. З клавіатури ввести вектор дійсних чисел. Знайти середнє арифметичне усіх чисел, що

## розміщені між максимальним та мінімальним елементами вектора.

## Завдання 2. Ввести прямокутну матрицю дійсних чисел. Визначити номери двох рядків введеної

## матриці, скалярний добуток яких є найменшим.

## Теоретичні відомості

Масив являє собою сукупність однотипних змінних, розміщених у послідовно пронумерованих суміжних комірках пам'яті. Номер елемента масиву задається індексом. ***Індексація елементів масиву в С починається з 0***. Якщо у масиві N елементів, то перший елемент матиме індекс 0, а останній – індекс (N–1).

За способом зв'язування індексів з комірками пам'яті виділяють три категорії масивів: *статичні*, *фіксовані автоматичні* та *динамічні*.

*Статичним* називають масив, в якому зв’язування індексів та розміщення в пам'яті виконуються на етапі компіляції програми. Статичні масиви досить ефективні, оскільки для їх створення та знищення не потрібно додаткових операцій.

*Фіксованим автоматичним* називають масив, в якому індекси зв'язуються статично, а розміщення в пам'яті виконується при обробці оголошень всередині функцій.

*Динамічним* називають масив, де зв’язування індексів і розміщення в пам'яті виконується безпосередньо під час виконання програми. Цей спосіб організації даних достатньо гнучкий, оскільки розміри динамічного масиву можуть збільшуватися та зменшуватися під час виконання програми у міру необхідності.

Об’єм пам'яті, який займає масив, рівний добутку розміру типу елементів масиву та кількості елементів в масиві. Наприклад, якщо масив містить 16 чисел типу int і тип int займає в пам’яті 2 байти, то цілий масив займатиме 2×16 = 32 байти.

***У мові С не перевіряється вихід індексу масиву за межі допустимого значення***. Вихід за межі пам’яті, яку займає масив, не контролюється. При помилці програміста може відбутися запис інформації в сусідні комірки пам'яті, а ***наслідки*** цього ***непередбачувані***. Масиви можуть бути одновимірними та багатовимірними.

## Одновимірні масиви

Одновимірні масиви оголошуються таким чином:

тип ім’я\_масиву[розмір]

Наприклад, так оголошується статичний масив з іменем numbs, що складається з 10 елементів типу int:

int numbs[10];

У С доступ до елементів масиву здійснюється за допомогою спеціального оператора []. В квадратних дужках вказується індекс елемента масиву, а перед ними – його ім'я. Наприклад, присвоєння першому елементу масиву numbs значення 12 виконується так:

numbs[0] = 12;

Масиви тісно пов’язані із вказівниками. Ім’я масиву є вказівником на його перший елемент. Тому ім'я масиву можна використати в якості бази для зміщення вказівника. Наприклад, вираз

numbs[4] = 1

еквівалентний виразу

\*(numbs+4) = 1;

## Ініціалізація масивів

Для заповнення масиву початковими даними достатньо виконати ініціалізацію під час оголошення масиву:

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

При цьому не потрібно вказувати розмір масиву – він буде розпізнаний за кількістю введених елементів. Тому можна писати так:

int numbs[] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

Більше того, якщо вказати в квадратних дужках неправильний розмір, все одно компілятор автоматично обчислить правильний розмір. Наприклад, при оголошенні:

int numbs[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8}

компілятор «вважатиме», що масив містить 9 елементів. Однак, при цьому виникне проблема

при наступному присвоєнні:

numbs[10] = 10;

Компілятор розпізнає помилку, коли присвоюється більше чисел, ніж задано елементів масиву:

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

При цьому видається повідомлення «Занадто багато ініціалізаторів».

До масивів застосовний оператор **sizeof**, який повертає розмір масиву у байтах:

sizeof(numbs)

Якщо масив не підлягає змінам, його можна оголосити константним, додавши слово **const** в оголошення масиву:

const int numbs[] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

**Приклад**. З клавіатури ввести 10 цілих чисел, записавши їх у масив. Роздрукувати введені числа в рядок через два пробіли.

# include<stdio.h> # include <conio.h> int main(){

int vect[10]; // оголошення масиву з 10 цілих чисел for(int i = 0; i < 10; i++)

{/\* у циклі виводимо підказку про введення чисел \*/

printf("Enter a number");

scanf("%d", &vect[i]); /\* на і-ій ітерації циклу записуємо введене число в і-ий елемент масиву; змінна циклу має початкове значення 0, оскільки елементи масиву індексуються починаючи з 0 \*/

}

printf("You entered:"); for(int i = 0; i < 10; i++)

{/\* у циклі виводимо всі введені числа, відокремлені двома пробілами \*/ printf("%d ", vect[i]);

}

return 0;

getch();

}

**Приклад**. З клавіатури ввести 10 цілих чисел, записавши їх у масив. Роздрукувати введені числа у зворотному порядку.

int vect[10];

for(int i = 0; i < 10; i++)

{

printf("Enter a number");

scanf("%d", &vect[i]);

}

printf("You entered:");

for(int i = 9; i >= 0; i--) /\* елементи масиву «перебираємо» з кінця масиву \*/

{

printf("%d ", vect[i]);

}

Результати показані на рис. 1

Зображення, що містить текст

Автоматично згенерований опис

а) б)

Рис. 1 Введення 10 цілих чисел та роздрук їх в одному рядку в прямому (а) та зворотному (б) порядку

**Приклад**. Ввести з клавіатури 10 символів. Порахувати кількості введених голосних та вивести їх на зразок таблиці.

const char vowels[] = {'a','e','i','o','u','y'}; /\* оголошуємо масив vowels з 6 елементів типу char та одразу здійснюємо ініціалізацію латинськими голосними літерами; масив оголошено константним тому, що не передбачається його змінювати протягом виконання програми \*/

char usersymbols[10]; /\* оголошуємо масив usersymbols, в який будемо записувати

10 символів, введених користувачем з клавіатури \*/

int frequencies[6];/\* оголошуємо масив frequencies з 6 цілих чисел, який зберігатиме інформацію про те, скільки разів користувач ввів кожну з голосних літер; тому кількість елементів масиву frequencies збігається з кількістю елементів масиву vowels \*/

for(int i = 0; i < sizeof(usersymbols)/sizeof(char); i++)

{/\* в циклі заповнюємо масив usersymbols; вираз sizeof(usersymbols)/sizeof(char) рівний числу елементів в масиві usersymbols \*/

printf("Enter a symbol");

scanf("%c",&usersymbols[i]);

}

for(int i = 0; i < sizeof(vowels)/sizeof(char); i++)

{/\* у циклі «перебираємо» всі голосні літери, спочатку вважаючи, що кожна з них зустрілася серед введених користувачем символів 0 разів; і-ий елемент масиву vowels відповідає і-ому елементу масиву frequencies \*/

frequencies[i] = 0;

for(int j = 0; j < sizeof(usersymbols)/sizeof(char); j++)

{ /\* у внутрішньому циклі перевіряємо відповідність кожного введеного символу і-ій голосній літері; якщо відповідність знайдена, збільшуємо і-ий елемент масиву frequencies на 1\*/

if(usersymbols[j]==vowels[i])

frequencies[i]++;

}

}

for(int i = 0; i < sizeof(vowels)/sizeof(char); i++) {/\* виводимо результати на екран у два стовпці \*/

printf("%c %d", vowels[i], frequencies[i]);

}

Результат показаний на рис. 2

Зображення, що містить текст

Автоматично згенерований опис

Рис. 2 Підрахунок кількості введених з клавіатури голосних літер

## Двовимірні масиви

Часто за допомогою мов програмування виникає потреба обробляти таблиці або матриці. Для їх представлення у мові С передбачені багатовимірні масиви.

Найпоширеніший варіант багатовимірного масиву – двовимірний масив, який можна представити у вигляді масиву одновимірних масивів:

тип ім’я\_масиву [розмір1][розмір2]

Звертаються до елементів масиву так:

ім’я\_масиву [індекс1][індекс2]

Наприклад:

int matr[10][5]; */\* оголошується матриця з 10 рядків і 5 стовпців \*/*

matr[4][3] = 10; */\* четвертому елементу п’ятого рядка присвоюється значення 10 \*/*

Ініціалізація двовимірного масиву здійснюється по рядках:

int numbs[3][3] = {1, 2, 3, 4, 5, 6, 7, 8, 9};

В результаті перший рядок міститиме числа 1, 2, 3, другий – числа 4, 5, 6, а третій – числа 7,

8, 9.

Для наочності краще при ініціалізації використовувати групування:

int numbs[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};

Двовимірний масив, як і будь-який інший, зберігається в послідовних комірках пам'яті, а тому його можна представити як одномірний, пам’ятаючи, що ім'я масиву – це вказівник на його перший елемент. Нехай двовимірний масив mas має розмірність n\*m. Тоді елемент mas[i][j] можна представити як \*(\*mas+n\*i+j). З іншого боку, вираз mas[i] є вказівником на перший елемент і-ого рядка, а тому \*[mas[i]+j] – це j-ий елемент і-ого рядка.

**Приклад**. З клавіатури ввести квадратну матрицю. Роздрукувати її.

int matr[N][N]; for(int i = 0; i < N; i++)

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

{

printf("Enter the %d element of %d row",(j+1),(i+1));

scanf("%d",&matr[i][j]);

}

for(int i = 0; i < N; i++) for(int j = 0; j < N;j++) printf("%d\t",matr[i][j]);

Розмір масиву має бути константою. Оголошення

int N = 4; int matr[N][N];

призвело б до помилки компіляції.

Хоча у С індексація елементів масивів починається з нуля, тим не менш зручно іменувати елемент з індексом 0 першим. Тому підказку для вводу масиву виводимо зі збільшеними на 1 індексами: printf("Enter the %d element of %d row",(j+1),(i+1));

Введену матрицю виводимо по рядках, розмежовуючи елементи символами табуляції для покращення читабельності. Фрагмент результату показаний на рис. 3.

Зображення, що містить текст

Автоматично згенерований опис

Рис. 3 Ввід та роздрук квадратної матриці

Оскільки масиви тісно пов'язані зі вказівниками, то програму для вводу та роздруку квадратної матриці цілих чисел можна переписати наступним чином:

int matr[N][N];

for(int i = 0; i < N\*N; i++)

{ /\* квадратна матриця містить N\*N елементів, \*matr є вказівником на перший елемент першого рядка матриці matr, а змінна і задає зміщення елемента від початку масиву, оскільки всі елементи розміщені в пам'яті послідовно \*/

printf("Enter a number");

scanf("%d",\*matr+i);

}

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

{

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

printf("%d\t",matr[i][j]);

}

Вивід елементів масиву matr в рядок можна здійснити так:

for(int i = 0; i < N\*N; i++)

{

printf ("%d\t",\*matr+i);

}

Вивід у табличному вигляді можна здійснити лише з використанням одного циклу:

for(int i = 0; i < N\*N; i++)

{

printf("%d\t", \*(\*matr+i));

if(!((i+1)%N)) printf("\n");

}

За допомогою оператора if після кожних N виведених на екран елементів переходимо на новий рядок.

**Приклад**. З клавіатури ввести квадратну матрицю. Роздрукувати саму матрицю та номер рядка, сума елементів якого є максимальною (рядки для зручності нумеруватимемо з 1).

int matr[N][N];

for(int i = 0; i < N\*N; i++)

{

printf("Enter a number");

scanf("%d",\*matr+i);

}

int max = 0; int maxn = 1;

int sum;

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

{

sum = 0;

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

sum = sum + matr[i][j]; if(sum > max) {max = sum; maxn = i;}

}

for(int i = 0; i < N\*N; i++)

{

printf("%d\t",\*(\*matr+i));

if(!((i+1)%N)) printf("\n");

}

printf("The maximum sum is in the row #%d ",(maxn+1));

Змінна max міститиме поточну значення максимальної суми елементів в рядку (спочатку присвоїмо max = 0). У змінній maxn зберігатимемо номер рядка, сума елементів якого є максимальною. У змінній sum зберігатимемо суму елементів рядка, який опрацьовується у даний момент. Перед опрацюванням кожного рядка змінну sum «скидаємо в нуль». Якщо змінна sum більша за поточне значення max, то max присвоюємо значення sum, а номер відповідного рядка записуємо в maxn. Результати показані на рис. 4.

Зображення, що містить текст

Автоматично згенерований опис

Рис. 4 Роздрук квадратної матриці та визначення номера рядка з найбільшою сумою елементів

**Приклад**. Ввести з клавіатури квадратну матрицю. Роздрукувати її разом з сумою елементів кожного рядка.

int matr[N][N];

for(int i = 0; i < N\*N; i++)

{

printf("Enter a number");

scanf("%d",\*matr+i);

}

int max = 0; int maxn = 1;

int sum;

int vect[N]; /\* у vect[i] запишемо суму елементів і-ого рядка матриці matr\*/ for(int i = 0; i < N; i++)

{

sum = 0;

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

sum = sum + matr[i][j]; if(sum > max) {max = sum; maxn = i;} vect[i] = sum;

}

for(int i = 0; i < N; i++) { for(int j = 0; j < N;j++) printf("%d\t",matr[i][j]);

printf("Suma: %d\n",vect[i]);

}

Результат показаний на рис. 5

Зображення, що містить текст

Автоматично згенерований опис

Рис. 5 Ввід та роздрук квадратної матриці з сумою елементів кожного рядка

## Блок-схеми

Назва файлу: **Lab2Task1.png**

Зображення, що містить текст, монітор, екран, у приміщенні

Автоматично згенерований опис

Назва файлу: **Lab2Task2.png**

**Зображення, що містить текст, ноутбук, екран, у приміщенні

Автоматично згенерований опис**

## Код програми

Назва файлу: **Task1.c**

#include <stdio.h>

int main(void)

{

int iCount, iMinIndex = 0, iMaxIndex = 0;

float aNumbers[1000];

scanf\_s("%i", &iCount);

for (int i = 0; i < iCount; i++)

{

scanf\_s("%f", &aNumbers[i]);

if (aNumbers[i] > aNumbers[iMaxIndex])

iMaxIndex = i;

if (aNumbers[i] < aNumbers[iMinIndex])

iMinIndex = i;

}

int iStart, iFinish;

if (iMinIndex < iMaxIndex)

{

iStart = iMinIndex;

iFinish = iMaxIndex;

}

else

{

iStart = iMaxIndex;

iFinish = iMinIndex;

}

float fSum = 0;

for (int i = iStart; i <= iFinish; i++)

{

fSum += aNumbers[i];

}

printf("%.3f", fSum / (iFinish - iStart + 1));

}

Назва файлу: **Task2.c**

#include <stdio.h>

int main(void)

{

int iRows, iColumns;

scanf\_s("%i", &iRows);

scanf\_s("%i", &iColumns);

float aaMatrix[100][100];

for (int i = 0; i < iRows; i++)

for (int j = 0; j < iColumns; j++)

scanf\_s("%f", &aaMatrix[i][j]);

float fMinProduct = 0;

int iRow1 = 0, iRow2 = 1;

for (int k = 0; k < iColumns; k++)

{

fMinProduct += aaMatrix[0][k] \* aaMatrix[1][k];

}

for (int i = 0; i < iRows; i++)

{

for (int j = i+1; j < iRows; j++)

{

float fProduct = 0;

for (int k = 0; k < iColumns; k++)

{

fProduct += aaMatrix[i][k] \* aaMatrix[j][k];

}

if (fProduct < fMinProduct)

{

fMinProduct = fProduct;

iRow1 = i;

iRow2 = j;

}

}

}

printf("%i %i", iRow1, iRow2);

## }

## Протокол роботи

Завдання 1.

Протокол вводу:

5

13

27

9

18

45

Протокол виводу на екран:

24.000

На рис. 1 наведено результати виконання програми.

Зображення, що містить текст

Автоматично згенерований опис

Рис. 1. Результат виконання програми 1

Завдання 2.

Протокол вводу:

3

3

1

2

3

4

5

6

7

8

9

Протокол виводу на екран:

0 1

Зображення, що містить текст

Автоматично згенерований опис

Рис. 2. Результат виконання програми 2

## Висновки

На лабораторній роботі розглянуто поняття масивів в мові C та алгоритми для роботи з ними. Для обох завдань створено блок-схеми алгоритмів їх вирішення та у відповідності з ними складено програми на C.