# Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"

# Факультет інформатики та обчислювальної техніки Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 7 з дисципліни

«Алгоритми та структури даних-1.

Основи алгоритмізації»

«Дослідження лінійних алгоритмів» Варіант 2

Виконав студент Боровков Іван Ігорович

(шифр, прізвище, ім'я, по батькові)

Перевірив

( прізвище, ім'я, по батькові)

Київ 2021

**Лабораторна робота 7**

**Дослідження лінійного пошуку в послідовностях**

**Мета** – дослідити методи послідовного пошуку у впорядкованих і невпорядкованих послідовностях та набути практичних навичок їх використання під час складання програмних специфікацій.

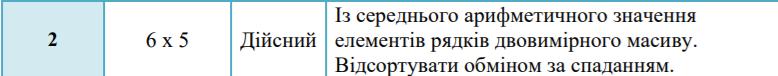
**Умова задачі** –

Розробити алгоритм та написати програму, яка складається з наступних дій:

1. Опису змінної індексованого типу (двовимірний масив) згідно з варіантом (табл. 1).

2. Ініціювання змінної, що описана в п.1 даного завдання.

3. Створення нової змінної індексованого типу (одновимірний масив) та її ініціювання значеннями, що обчислюються згідно з варіантом (табл. 1).



**Постановка задачі**

За допомогою алгоритму обходу матриці сгенерувати матрицю, обчислити середнє значення її рядків та відсортувати за спаданням отриманий масив середніх значень за допомогою алгоритму сортування обміном.

**Математична модель.**

**Складемо таблицю імен змінних:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Змінна** | **Тип** | **Ім’я** | **Призначення** |
| Матриця | Матриця 6 х 5 дійсних чисел | matrix | Вхідні дані |
| Масив середніх значень рядків | Масив дійсних чисел | array\_of\_average | Вихідні дані |
| Відсортований масив середніх значень рядків | Масив дійсних чисел | sorted\_array\_of\_average | Вихідні дані |
| Ширина матриці | Цілий | height | Допоміжна змінна |
| Довжина матриці | Цілий | weight | Допоміжна змінна |

Для генерації матриці скористаємося двома арифметичними циклами для її обходу та функцією генерації випадкових дійсних значень.

Для формування масиву середніх значень скористаємося алгоритмом обходу матриці по рядкам, обраховуючи суму значень кожного рядка та розділяючи цю суму на кількість елементів масиву.

Для сортування масиву скористаємося алгоритмом сортування обміном.

***Розв’язання:***

**Програмні специфікації** пишемо у псевдокоді та графічній формі у вигляді блок схеми та підпрограмм.

*Крок 1:* Визначимо основні дії ;

*Крок 2:* Ініціалізуємо двовимірний массив;

*Крок 3:* Створюємо та ініціалізуємо одновимірний массив;

*Крок 4:* Відстортуємо цей массив за спаданням;

*Псевдокод*

*Крок 1*

**Початок**

Ініціалізуємо двовимірний массив;

Створюємо та ініціалізуємо одновимірний массив;

Відстортуємо цей массив за спаданням;

**Кінець.**

*Крок 2*

**Початок**

width = 6, height = 5;

**generate\_matrix**(width, height);

Створюємо та ініціалізуємо одновимірний массив;

Відстортуємо цей массив за спаданням;

**Кінець.**

*Крок 3*

**Початок**

width = 6, height = 5;

matrix = **generate\_matrix**(width, height);

array\_of\_average =  **form\_array\_of\_average**(matrix, width, height);

Відстортуємо цей массив за спаданням;

**Кінець.**

*Крок 4*

**Початок**

width = 6, height = 5;

matrix = **generate\_matrix**(width, height);

array\_of\_average =  **form\_array\_of\_average**(matrix, width, height);

sorted\_array\_of\_average = **sort\_array(**array\_of\_average, width);

**Кінець.**

*Підпрограмми:*

1.

**generate\_matrix**(width, height)

**Початок**

**Повторити** **для** i **від** 1 **до** width

**Повторити** **для** j **від** 1 **до** height

matrix[i][j] = rand();

**Все повторити;**

**Все повторити;**

**Повернути** matrix;

**Кінець.**

2.

**form\_array\_of\_average**(matrix, width, height)

**Початок**

**Повторити** **для** i **від** 1 **до** width

array\_of\_average[i] = **find\_average**(matrix[i], height);

**Все повторити**

**Повернути**  array\_of\_average;

**Кінець.**

3.

**find\_average**(some\_array, array\_size)

**Початок**

sum = 0

**Повторити** **для** i **від** 1 **до** array\_size

sum+=some\_array[i];

**Все повторити;**

average = sum/array\_size;

**Повернути** average;

**Кінець.**

4.

**sort\_array(**some\_array, array\_size)

**Початок**

**Повторити** **для** i **від** 1 **до** array\_size – 1

**Повторити** **для** j **від** 1 **до** array\_size - 1

**Якщо (**some\_array[j]<some\_array[j + 1])

**То**

save = some\_array[j];

some\_array[j] = some\_array[j + 1];

some\_array[j + 1] = save;

**Все якщо;**

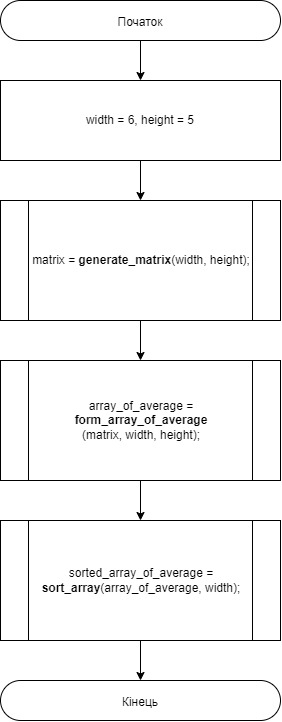
**Все повторити;**

**Все повторити;**

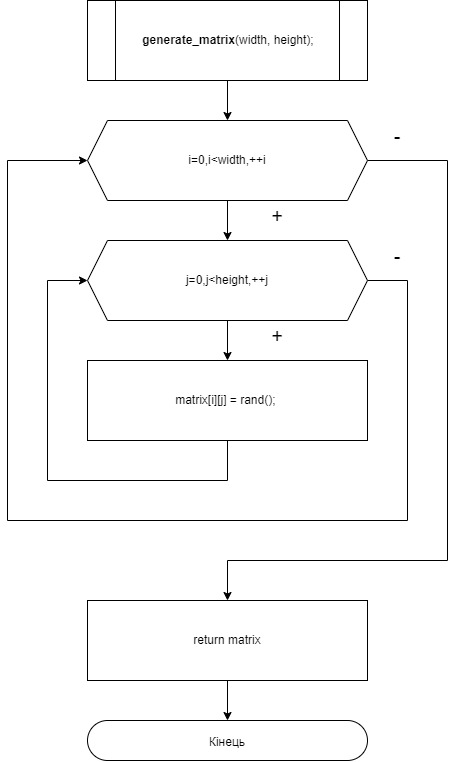
**Повернути** some\_array;

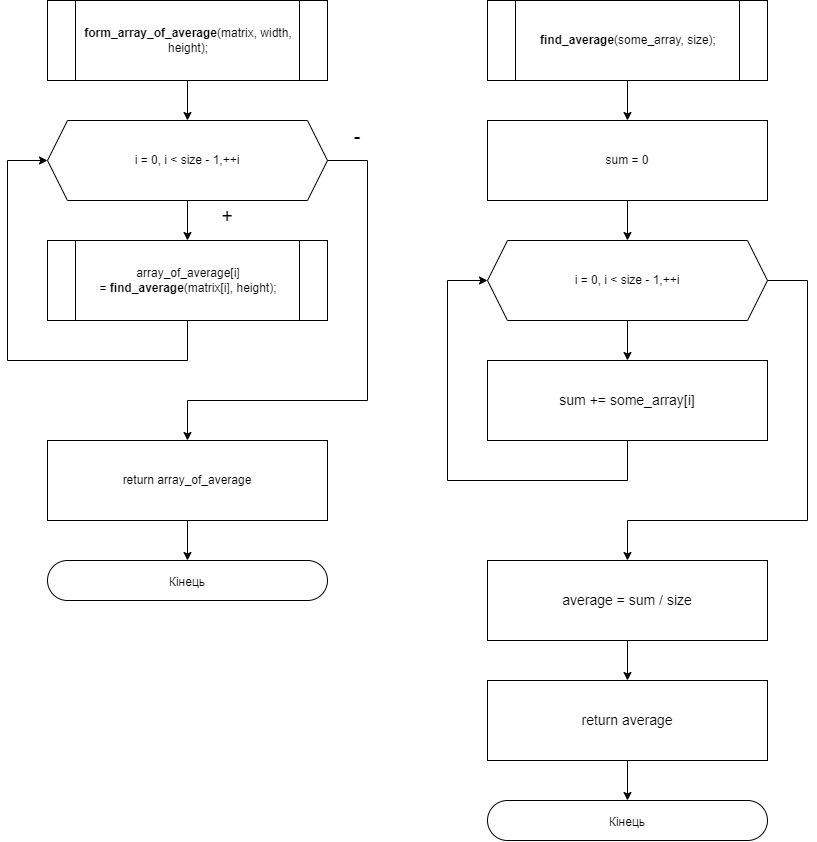
**Кінець.**

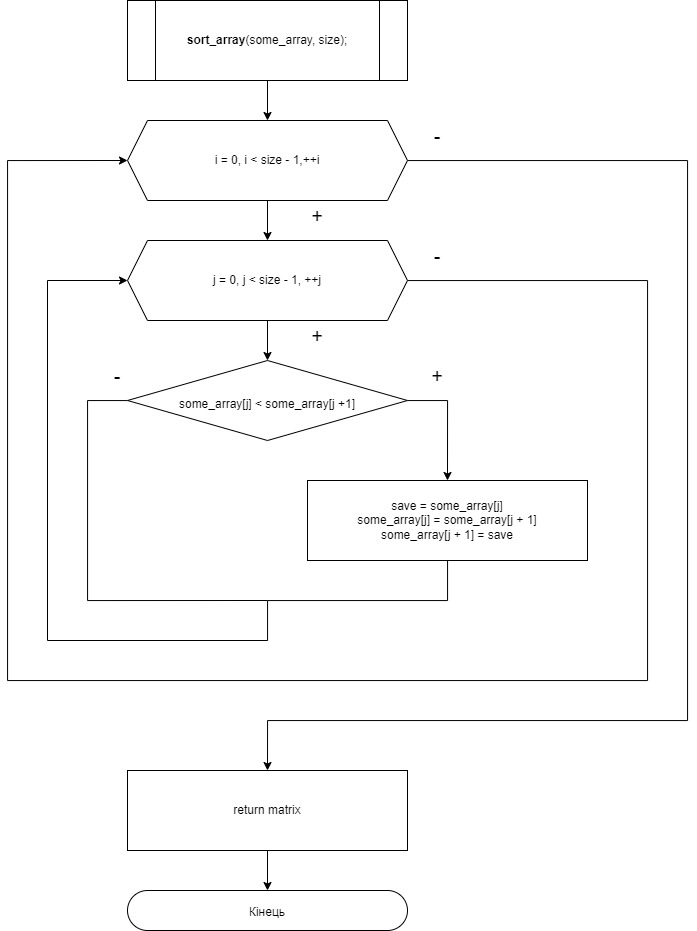
*Блок-схема:*

****

*Підпрограми:*



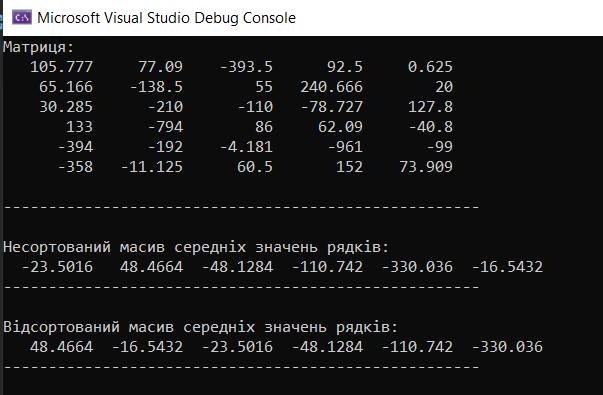
****

****

**Випробування алгоритму:**

|  |  |
| --- | --- |
| **Блок** | **Дія** |
|  | Початок |
| **1.** | Формується матриця 6 х 5 випадкових дійсних чисел:  1.5 2.5 3 7 8  1 1 2.4 3 6.5  3.5 2 4.5 4 5  2 5 2.5 4.5 6  3 6.2 3.4 5 7.8  1 4.4 2.3 6.5 2 |
| **2.** | Формується массив середніх значень рядків:  4.4 2.78 3.8 4 5.08 3.24 |
| **3.** | Масив сортується в порядку спадання:  5.08 4.4 4 3.8 3.24 2.78 |
|  | Кінець |

*Копії екранних форм:*



**Код программи:**

#include <iostream>

#include <iomanip>

#include <time.h>

using namespace std;

double\*\* generate\_matrix(int, int);

double generate\_random\_double();

double\* form\_array\_of\_average(double\*\*, int, int);

double find\_avarage(double\*, int);

void swap(double&, double&);

void sort\_array(double\*, int);

void delete\_matrix(double\*\*, int);

void delete\_array(double\*);

void output\_matrix(double\*\*, int, int);

void output\_array(double\*, int);

void output\_line();

int main()

{

srand(time(NULL));

setlocale(LC\_ALL, "");

const int width = 6, height = 5;

//Генеруємо матрицю

cout << "Матриця: \n";

double\*\* matrix = generate\_matrix(width, height);

output\_matrix(matrix, width, height);

output\_line();

//формуємо массив середніх значень

cout << "\nНесортований масив середнiх значень рядкiв: \n";

double\* array\_of\_avarage = form\_array\_of\_average(matrix, width, height);

output\_array(array\_of\_avarage, width);

output\_line();

//Сортуємо массив

cout << "\nВiдсортований масив середнiх значень рядкiв: \n";

sort\_array(array\_of\_avarage, width);

output\_array(array\_of\_avarage, width);

output\_line();

//Очищаємо пам'ять

delete\_array(array\_of\_avarage);

delete\_matrix(matrix, height);

return 1;

}

//Рішення задачі

double\* form\_array\_of\_average(double\*\* matrix, int width, int height)

{

double\* array\_of\_avarage = new double [width];

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

{

array\_of\_avarage[i] = find\_avarage(matrix[i], height);

}

return array\_of\_avarage;

}

double find\_avarage(double\* some\_array, int size)

{

double sum = 0, avarage;

for (int i = 0; i < size; ++i) sum += some\_array[i];

avarage = sum / size;

return avarage;

}

void sort\_array(double\* some\_array, int size)

{

for (int i = 0; i < size - 1; ++i)

for (int j = 0; j < size - 1; ++j)

if (some\_array[j] < some\_array[j + 1]) swap(some\_array[j], some\_array[j+1]);

}

void swap(double& a, double& b)

{

double save = a;

a = b;

b = save;

}

// Робота з массивами та матрицями

void output\_line()

{

cout << "\n-----------------------------------------------------\n";

}

void output\_array(double\* some\_array, int size)

{

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

cout << setw(10) << some\_array[i];

}

void delete\_array(double\* some\_array)

{

delete some\_array;

}

double generate\_random\_double()

{

return double(int((double(rand() % 2001 - 1000) / double(rand() % 11 + 1)) \* 1000)) / 1000; // Формуємо випадкове дійсне число від -10000 до 10000 з точністю до трьох знаків після коми

}

double\*\* generate\_matrix(int width, int height)

{

double\*\* matrix = new double\* [width];

for (int i = 0; i < width; ++i) matrix[i] = new double[height];

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

{

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

{

matrix[i][j] = generate\_random\_double();

}

}

return matrix;

}

void output\_matrix(double\*\* matrix, int width, int height)

{

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

{

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

cout << setw(10) << matrix[i][j];

cout << '\n';

}

}

void delete\_matrix(double\*\* matrix, int height)

{

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

{

delete[] matrix[i];

}

delete[] matrix;

}

**Висновок:** В ході лабораторної роботи я дослідив алгоритм сортування обміном, набув навичок використання цього алгоритму під час складання програмних специфікацій.