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 № 8 з дисципліни
«Алгоритми та структури даних-1.
Основи алгоритмізації»

«Дослідження алгоритмів пошуку та сортування»

Варіант 27

Виконав студент ІП-13 Пархомчук Ілля Вікторович
(шифр, прізвище, ім'я, по батькові)

Перевірив Вєчерковська Анастасія Сергіївна
(прізвище, ім'я, по батькові)

Київ 2021

Лабораторна робота 8
Дослідження алгоритмів пошуку та сортування

Мета – дослідити алгоритми пошуку та сортування, набути практичних навичок використання цих алгоритмів під час складання програмних специфікацій..

Варіант 27

№ варіанта	Розмірність	Тип даних	Обчислення значень елементів одновимірного масиву
27	8 x 4	Дійсний	Із добутку від'ємних значень елементів рядків двовимірного масиву. Відсортувати обміном за зростанням.

Постановка задачі

Потрібно ініціалізувати двовимірний масив. Потім ініціалізувати одновимірний масив і відсортувати його методом обміну.

Побудова математичної моделі

Таблиця змінних

Змінна	**Тип**	**Ім'я**	**Призначення**
Двовимірний масив	Індексований дійсний	Array_1D	Проміжні дані
Одновимірний масив	Індексований дійсний	Array_2D	Проміжні дані, Вихідні дані
Тимчасовий одновимірний масив	Індексований дійсний	temp_arr	Проміжні дані
Лічильник	Цілочисельний	i	Проміжні дані
Лічильник	Цілочисельний	j	Проміжні дані
Допоміжна змінна для сортування	Цілочисельний	temp	Проміжні дані

Таблиця аргументів підпрограм

Масив до друку/опрацювання	Індексований символьний	arr	Проміжні дані, вихідні дані
Кількість рядків	Цілочисельний	m	Проміжні дані
Кількість стовпчиків	Цілочисельний	n	Проміжні дані
Розмір одновимірного масиву	Цілочисельний	size	Проміжні дані

Таблиця підпрограм

Підпрограма	**Оголошення**	**Опис**
init_2d_array	init_2d_array(arr[,], m, n)	Іініціалізує двовимірний масив випадковими значеннями
print_2d_array	print_2d_array(arr[,], m, n)	Друкує двовимірний масив
get_new_array	get_new_array(arr[,], m, n)	Повертає згенерований одновимірний масив на основі переданого двовимірного
print_1d_array	print_1d_array(arr[], size)	Друкує одновимірний масив
Sort	Sort(arr[], size)	Сортує одновимірний масив методом обміну

Таблиця констант

Змінна	**Тип**	**Ім’я**	**Значення**
Кількість рядків у масиві	Цілочисельний	M	8
Кількість стовпчиків у масиві	Цілочисельний	N	4

Таблиця операторів

Оператор	Назва	Синтаксис	Опис
rand()	Рандом	rand()	Повертає випадкове натуральне число

Потрібно заповнити двовимірними масив випадковими дійсними значеннями. Для цього можна використати функцію rand(). Так як rand() повертає випадкове натуральне число, то різниця rand() - rand() точно буде цілим числом(друге випадкове число може бути більше ніж перше). І щоб перейти до дійсних чисел можна отриману різницю поділити на деяке натуральне, відмінне від нуля число(наприклад 100).

N-й елемент одновимірного масиву потрібно заповнити добутком усіх від’ємних елементів n-ого рядка двовимірного масиву. Розмір одновимірного масиву відповідно дорівнює кількості рядків у двовимірному масиві.

Потрібно відсортувати отриманий масив методом обміну за зростанням. Для цього потрібно порівняти елементи і якщо елемент з меншим індексом більший за елемент з більшим індексом – поміняти їх місцями.

Розв'язання

Програмні специфікації запишемо у псевдокоді та графічній формі у вигляді блок-схеми.

Крок 1. Визначимо основні дії.

Крок 2. Деталізуємо ініціалізацію двовимірного масиву за допомогою підпрограми з використанням арифметичного оператору повторення.

Крок 3. Деталізуємо ініціалізацію одновимірного масиву за допомогою підпрограми з використанням арифметичного оператору повторення та умовної форми вибору.

Крок 4. Деталізуємо сортування одновимірного масиву за допомогою підпрограми з використанням арифметичного оператору повторення та умовної форми вибору.

Крок 5. Деталізуємо вивід масивів за допомогою підпрограми з використанням арифметичного оператору повторення.

Псевдокод алгоритму

Основна програма

початок

```
  init_2d_array(Array_2D, M, N)
  print_2d_array(Array_2D, M, N)
  Array_1D := get_new_array(Array_2D, M, N)
  print_1d_array(Array_1D, M);
  Sort(Array_1D, M);
  print_1d_array(Array_1D, M);
```

кінець

Підпрограми

підпрограма init_2d_array(arr[,], m, n)

  ініціалізація двовимірного масиву

все підпрограма

підпрограма get_new_array(arr[,], m, n)

  ініціалізація одновимірного масиву

все підпрограма

підпрограма Sort(arr[], size)(arr[,], m, n)

  сортування одновимірного масиву

все підпрограма

підпрограма print_2d_array(arr[,], m, n)

  друк двовимірного масиву

все підпрограма

підпрограма print_1d_array(arr[], size)

  друк одновимірного масиву

все підпрограма

Основна програма

початок

```
  init_2d_array(Array_2D, M, N)
  вивід «2d array:»
  print_2d_array(Array_2D, M, N)
  Array_1D := get_new_array(Array_2D, M, N)
  вивід «1d array:»
  print_1d_array(Array_1D, M);
  Sort(Array_1D, M);
  вивід «sorted 1d array:»
  print_1d_array(Array_1D, M);
```

кінець

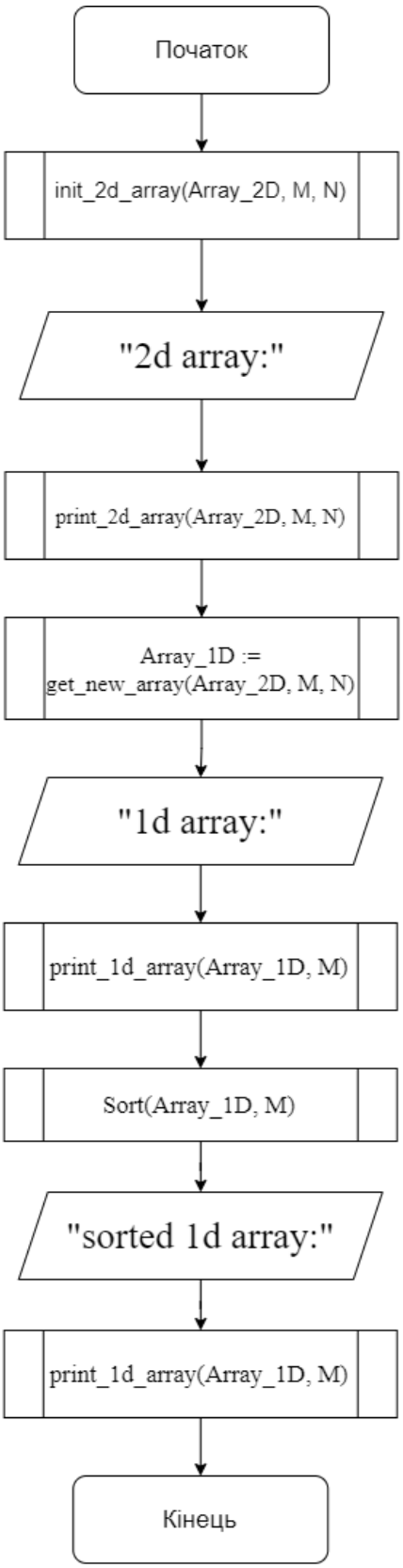
Підпрограми

```
підпрограма init_2d_array(arr[,], m, n)
  для i від 0 до m, з кроком 1 повторити
       для j від 0 до n, з кроком 1 повторити
           arr[i,j] := (rand() – rand())/100
        все повторити
  все повторити
все підпрограма

підпрограма get_new_array(arr[,], m, n)
  для i від 0 до m, з кроком 1 повторити
       для j від 0 до n, з кроком 1 повторити
          temp_arr [i] := 1
          якщо arr[i,j] < 0
            то
              temp_arr [i] := temp_arr [i] * arr[i,j]
            все якщо
        все повторити
```

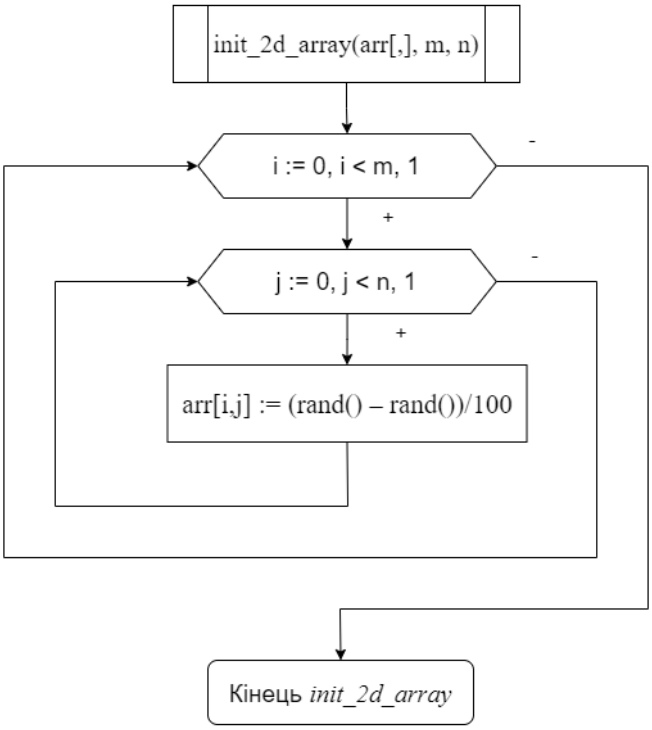
```
  все повторити
  повернути temp_arr
все підпрограма

підпрограма Sort(arr[], size)
  для i від 0 до size -1 , з кроком 1 повторити
       для j від 0 до size -1, з кроком 1 повторити
          якщо arr[j] > arr[j + 1]
             то
                temp := arr[j]
                arr[j] := arr[j + 1]
                arr[j+1] := temp
             все якщо
        все повторити
  все повторити
все підпрограма

підпрограма print_2d_array(arr[,], m, n)
  для i від 0 до m, з кроком 1 повторити
       для j від 0 до n, з кроком 1 повторити
           вивід arr[i,j]
        все повторити
  все повторити
все підпрограма

підпрограма print_1d_array(arr[], size)
  для i від 0 до size , з кроком 1 повторити
       вивід arr[i]
  все повторити
все підпрограма
```

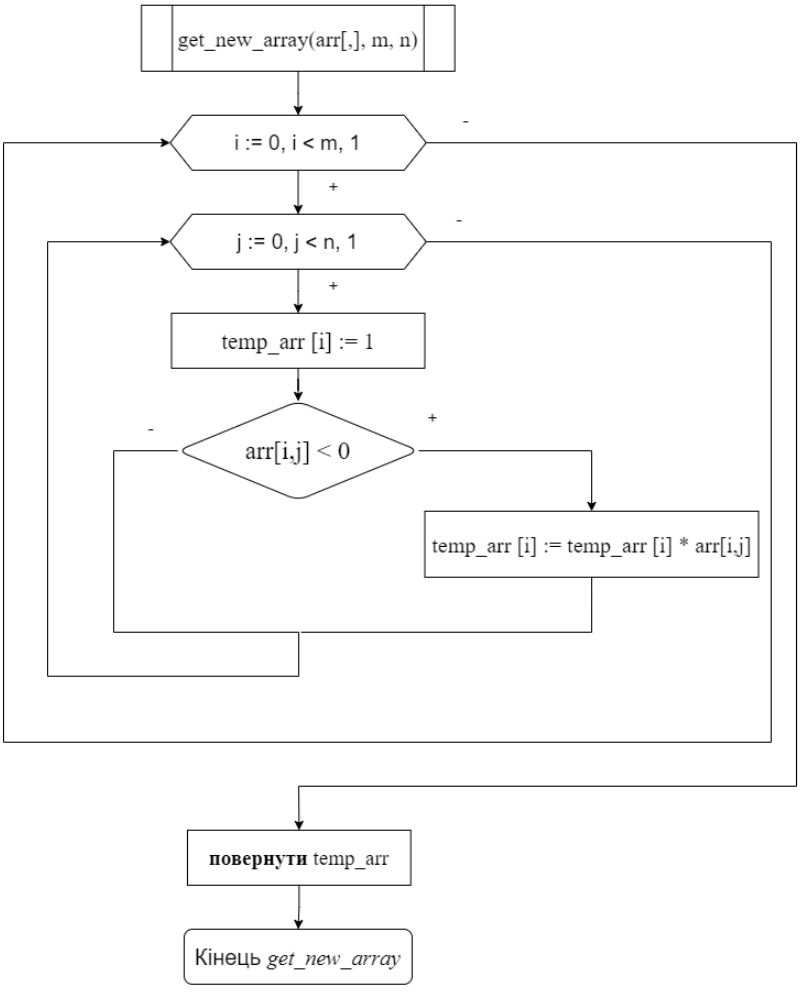
Блок-схема

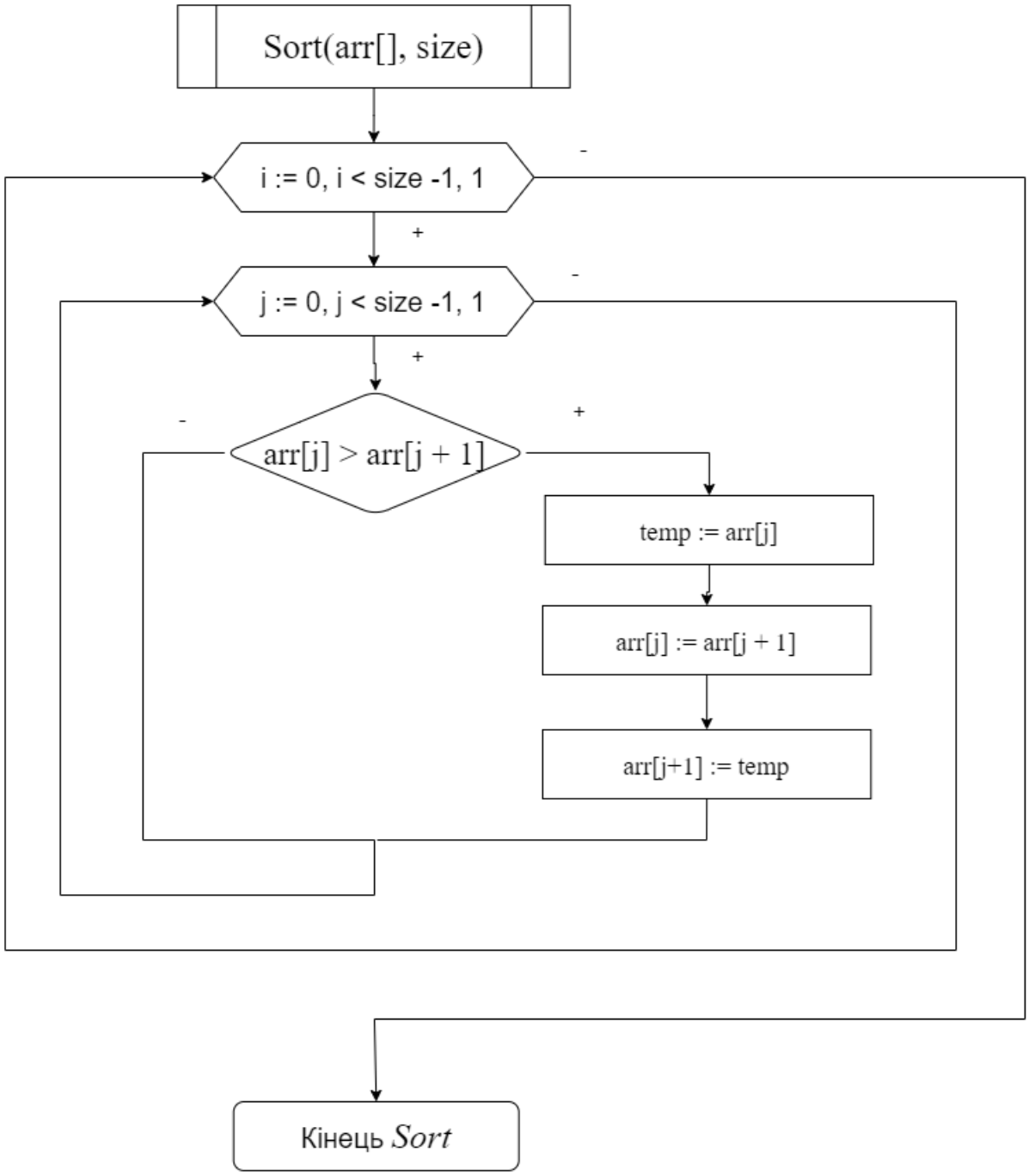
Основна програма:

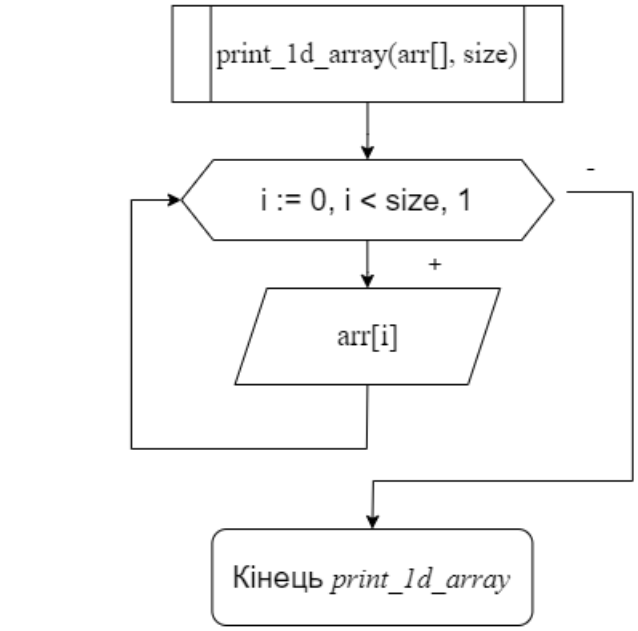


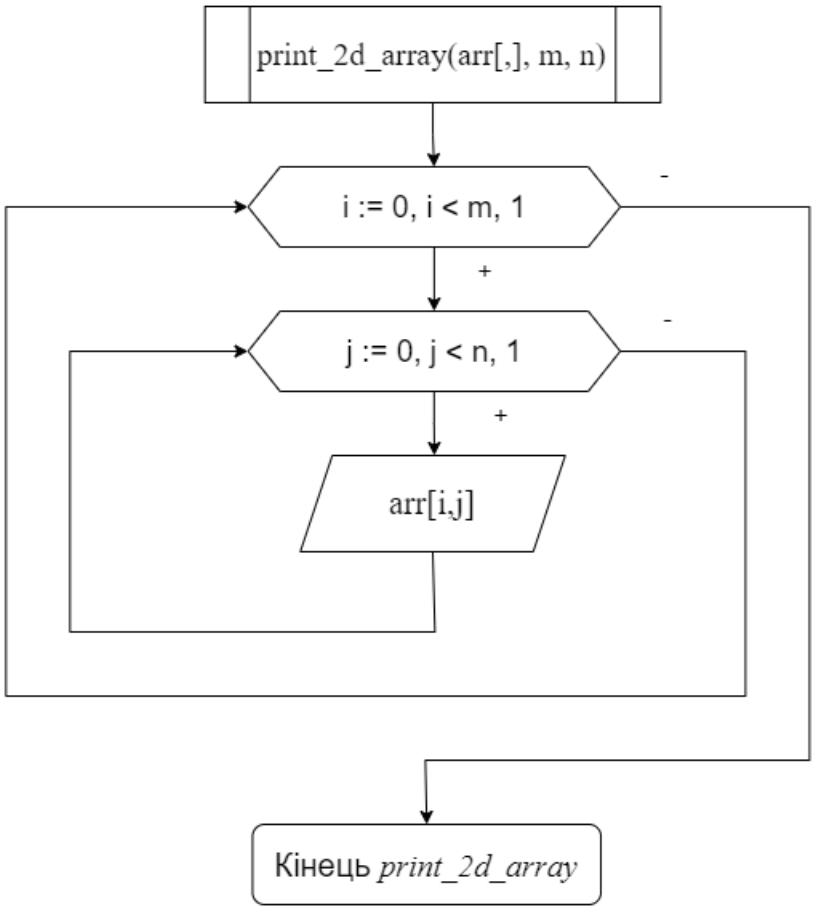
Підпрограми:






Sort(arr[], size)
i := 0, i < size -1, 1
-
+
j := 0, j < size -1, 1
-
+
-
arr[j] > arr[j + 1]
+
temp := arr[j]
arr[j] := arr[j + 1]
arr[j+1] := temp
Кінець *Sort*


print_1d_array(arr[], size)
i := 0, i < size, 1
-
+
arr[i]
Кінець print_1d_array


print_2d_array(arr[,], m, n)
i := 0, i < m, 1
-
+
j := 0, j < n, 1
-
+
arr[i,j]
Кінець print_2d_array

Код програми

```
#include <iostream>
#include <time.h>
#include <random>
#include <iomanip>

using std::cout;
using std::cin;
using std::setw;

void init_2d_array(double** arr, const int m, const int n);
double* get_new_array(double** arr, const int m, const int n);
void Sort(double * arr, const int size);
void print_1d_array(double* arr, const int size);
void print_2d_array(double** arr, const int m, const int n);


int main()
{
	srand(time(NULL));

	const int M = 8;
	const int N = 4;
	double* Array_1D;
	double** Array_2D = new double*[M];
	for (int i = 0; i < M; i++)
	{
		Array_2D[i] = new double[N];
	}
	init_2d_array(Array_2D, M, N);
	cout << "2d array:\n";
	print_2d_array(Array_2D, M, N);
	Array_1D = get_new_array(Array_2D, M, N);
	cout << "1d array:\n";
	print_1d_array(Array_1D, M);
	cout << "Sorted 1d array:\n";
	Sort(Array_1D, M);
	print_1d_array(Array_1D, M);

	delete[] Array_1D;
	for (int i = 0; i < M; i++)
	{
		delete[] Array_2D[i];
	}
	delete[] Array_2D;
	system("pause");
}



void init_2d_array(double** arr, const int m, const int n)
{
	for (int i = 0; i < m; ++i)
	{
		for (int j = 0; j < n; ++j)
		{
			arr[i][j] = ( rand() - rand() ) / 100.0;
		}
	}
}

double* get_new_array(double** arr, const int m, const int n)
{
	double* temp_arr = new double[m];
	for (int i = 0; i < m; ++i)
	{
		temp_arr[i] = 1;
```

```
                for (int j = 0; j < n; ++j)
                {
                        if (arr[i][j] < 0)
                        {
                                temp_arr[i] = temp_arr[i] * arr[i][j];
                        }
                }
        }
        return temp_arr;
}

void Sort(double* arr, const int size)
{
        for (int i = 0; i < size - 1; ++i)
        {
                for (int j = 0; j < size - 1; ++j)
                {
                        if (arr[j] > arr[j + 1])
                        {
                                double temp = arr[j];
                                arr[j] = arr[j + 1];
                                arr[j+1] = temp;
                        }
                }
        }
}

void print_1d_array(double* arr, const int size)
{
        for (int i = 0; i < size; ++i)
        {
                cout << arr[i] << ' ';
        }
        cout << '\n';
}

void print_2d_array(double** arr, const int m, const int n)
{
        for (int i = 0; i < m; ++i)
        {
                for (int j = 0; j < n; ++j)
                {
                        cout << setw(8) << arr[i][j];
                }
                cout << '\n';
        }
}
```

Випробування алгоритму

```
2d array:
   -3.55  101.22  -81.33 -216.16
  229.94   97.97 -108.33   -8.01
  202.99 -104.89    0.18  128.41
   33.91   -5.78   16.97   42.58
   16.52 -144.94 -158.28   70.51
   -91.5   10.21 -145.77  -45.52
    21.9 -244.05   120.9  220.26
  183.84  -77.85   92.59  113.81
1d array:
-62410 867.723 -104.89 -5.78 22941.1 -607144 -244.05 -77.85
Sorted 1d array:
-607144 -62410 -244.05 -104.89 -77.85 -5.78 867.723 22941.1
```

```
2d array:
  140.24   -4.88 -150.05   88.51
  -93.49  -57.52 -165.88   -8.38
  119.85   17.74  -16.02 -272.68
  -44.42  -23.33  119.05   98.18
  -29.09 -111.77  -72.19    72.1
  -60.81  -24.28  160.67  -17.76
  124.82  -33.36   54.58  -24.32
  100.54  -83.05  164.44  -63.36
1d array:
732.244 7.47519e+06 4368.33 1036.32 -234718 -26222.1 811.315 5262.05
Sorted 1d array:
-234718 -26222.1 732.244 811.315 1036.32 4368.33 5262.05 7.47519e+06
```

Висновки

На лабораторній роботі я дослідив алгоритми пошуку та сортування, набув практичних навичок використання цих алгоритмів під час складання програмних специфікацій. Спочаку я ініціалізував двовимірний масив, потім, за вказаним алгоритмом створим одновимірний масив з добутку від'ємних елементів рядків двовимірного. Потім відсортував одновимірний масив методом обміну. Також створив підпрограми для виводу масивів. Мною був написаний псевдокод, створено блок-схеми та проведено тестування алгоритму.