

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 22

Виконав студент ІП-13, Музичук Віталій Андрійович
(шифр, прізвище, ім'я, по батькові)

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

Лабораторна робота 6

Дослідження алгоритмів пошуку та сортування

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

Варіант 22

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

1. Опису змінної індексованого типу (двовимірного масиву) згідно з варіантом.
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).

22	5 x 7	Дійсний	Із середнього арифметичного додатних значень елементів стовпців двовимірного масиву. Відсортувати обміном за спаданням.
----	-------	---------	---

Постановка задачі

Спочатку ми створюємо двовимірний масив із 5 рядками та 7 стовпцями. Ініціалізація цієї змінної відбувається випадково числами з проміжку [-9, 9] за допомогою підпрограми `initMatrix()`. Далі за допомогою підпрограми `outputMatrix()` виводимо матрицю в консоль для наглядності. Після цього створюємо окрему змінну `array[]` та заповнюємо її середнім арифметичним кожного стовпця за допомогою підпрограми `averageColumn()` та арифметичного циклу. Сортуюмо цей масив методом «бульбашки» по спаданню й виводимо в консоль.

Побудова математичної моделі

Складемо таблицю змінних

Змінна	Тип	Ім'я	Призначення
Двовимірний масив	Дійсний	<i>matrix</i>	Проміжне значення

Одновимірний масив	Дійсний	<i>array</i>	Проміжне значення
Підпрограма для ініціалізації матриці	void	<i>initMatrix</i>	Ініціалізація матриці
Підпрограма для виведення матриці	void	<i>outputMatrix</i>	Виведення матриці в консоль
Підпрограма для визначення середнього арифметичного стовпця	Дійсний	<i>averageColumn</i>	Визначення середнього арифметичного стовпця
Підпрограма для сортування масиву	void	<i>sort</i>	Сортування масиву

initMatrix – підпрограма яка заповнює матрицю випадковими числами з проміжку [-9; 9].

outputMatrix – підпрограма, яка поелементно виводить в консоль дані з матриці за допомогою 2 ітераційних циклів.

averageColumn - підпрограма яка повертає середнє арифметичне значення додатних елементів матриці.

sort – підпрограма призначена для сортування одновимірного масиву методом обміну («бульбашки»).

Rand(a, b) – функція яка генерує випадкове число на проміжку [a, b]

Розв'язання

Програмні специфікації запишемо у псевдокодi та графічній формi у вигляді блок-схеми.

Крок 1. Визначаємо основні дії

Крок 2. Створення й ініціалізація матриці

Крок 3. Виведення матриці

Крок 4. Ініціалізація й заповнення одновимірного масиву

Крок 5. Сортування масиву

Крок 6. Виведення масиву

Псевдокод

крок 1

початок

створення й ініціалізація матриці

виведення матриці

ініціалізація й заповнення одновимірною масиву

сортування масиву

виведення масиву

кінець

крок 2

початок

strings = 5

columns = 7

matrix [strings, columns]

initMatrix(matrix)

виведення матриці

ініціалізація й заповнення одновимірною масиву

сортування масиву

виведення масиву

кінець

крок 3

початок

strings = 5

columns = 7

matrix [strings, columns]

initMatrix(matrix)

outputMatrix(matrix)

ініціалізація й заповнення одновимірною масиву

сортування масиву

виведення масиву

кінець

крок 4

початок

strings = 5

columns = 7

matrix [strings, columns]

initMatrix(matrix)

outputMatrix(matrix)

array[columns]

для i від 1 до columns

array[i] = averageColumn(matrix, i);

все повторити

сортування масиву

виведення масиву

кінець

крок 5

початок

strings = 5

columns = 7

matrix [strings, columns]

initMatrix(matrix)

outputMatrix(matrix)

array[columns]

для i від 1 до columns

array[i] = averageColumn(matrix, i);

все повторити

sort(array)

виведення масиву

кінець

крок 6

початок

strings = 5

columns = 7

matrix [strings, columns]

initMatrix(matrix)

outputMatrix(matrix)

array[columns]

для i від 1 до columns

 array[i] = averageColumn(matrix, i);

все повторити

sort(array)

для i від 1 до columns

виведення array[i]

все повторити

кінець

Псевдокод підпрограми:

initMatrix(matrix)

для i від 1 до strings повторити

для j від 1 до columns повторити

 matrix[i, j] = rand(-9, 9)

все повторити

все повторити

outputMatrix(matrix)

для i від 1 до strings **повторити**

для j від 1 до columns **повторити**

виведення $\text{matrix}[i, j]$

все повторити

виведення '\n'

все повторити

averageColumn(matrix, numOfColumn)

$k = 0$

$\text{sum} = 0$

для i від 1 до strings

якщо $\text{matrix}[i, \text{numOfColumn}] \geq 0$

$\text{sum} = \text{sum} + \text{matrix}[i, \text{numOfColumn}]$

$k = k + 1$

все якщо

все повторити

якщо $k == 0$

повернути 0

$\text{average} = \text{sum} / k$

повернути average

sort(array)

для i від 1 до columns - 1 **повторити**

для j від 1 до columns - 1 **повторити**

якщо $\text{array}[j] < \text{array}[j+1]$

$\text{tmp} = \text{array}[j + 1]$

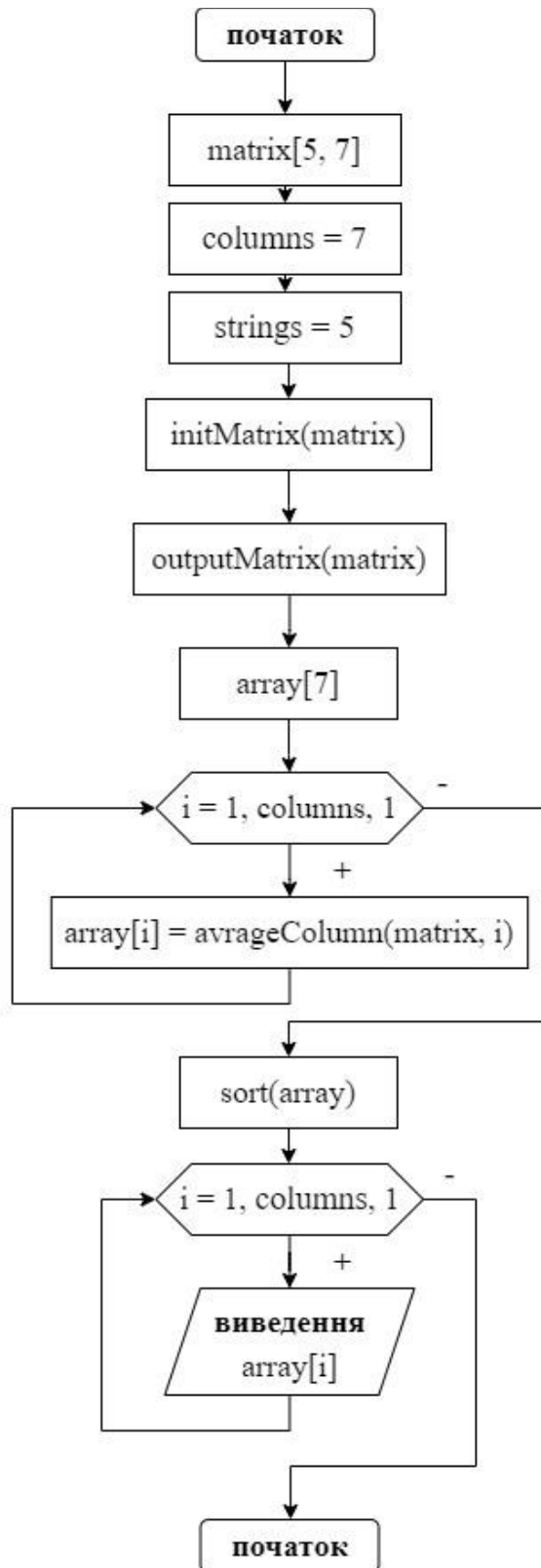
$\text{array}[j + 1] = \text{array}[j]$

$\text{array}[j] = \text{tmp}$

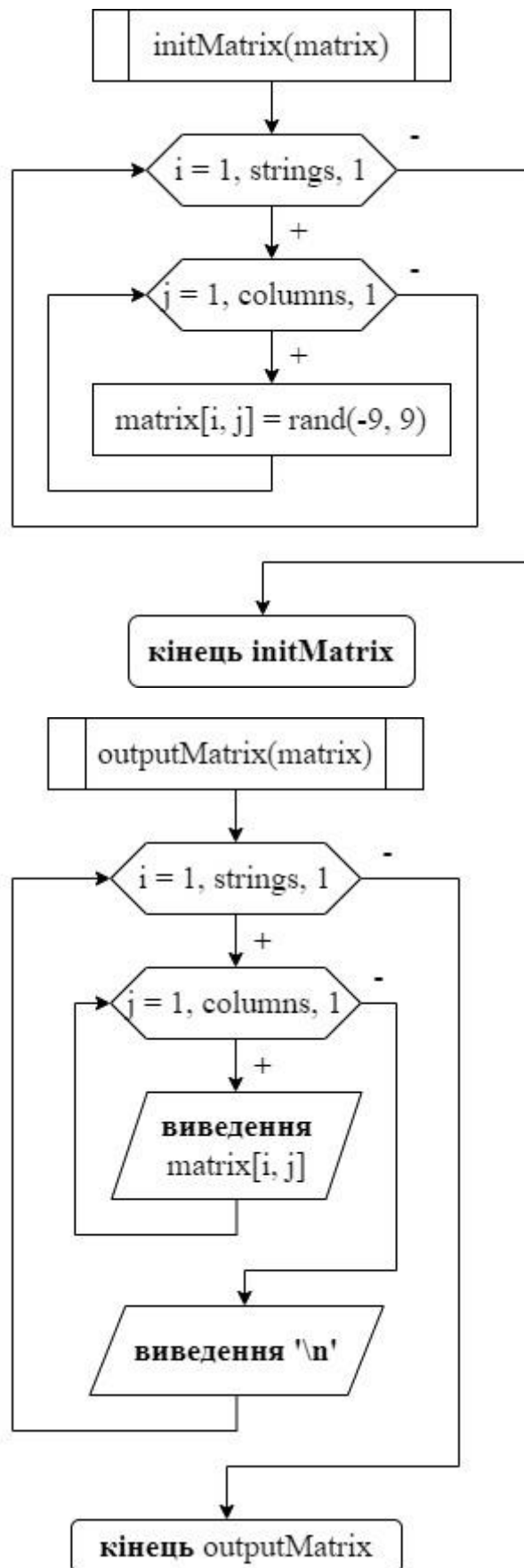
все якщо

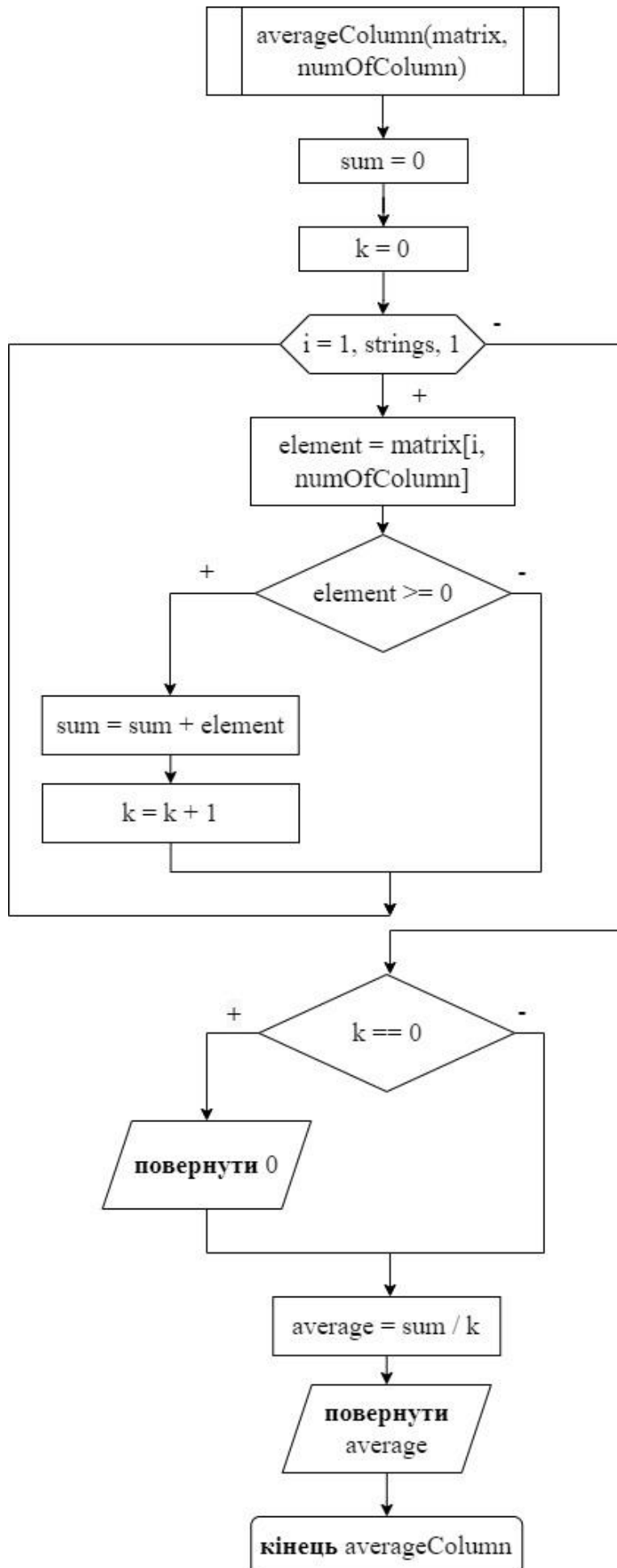
все повторити

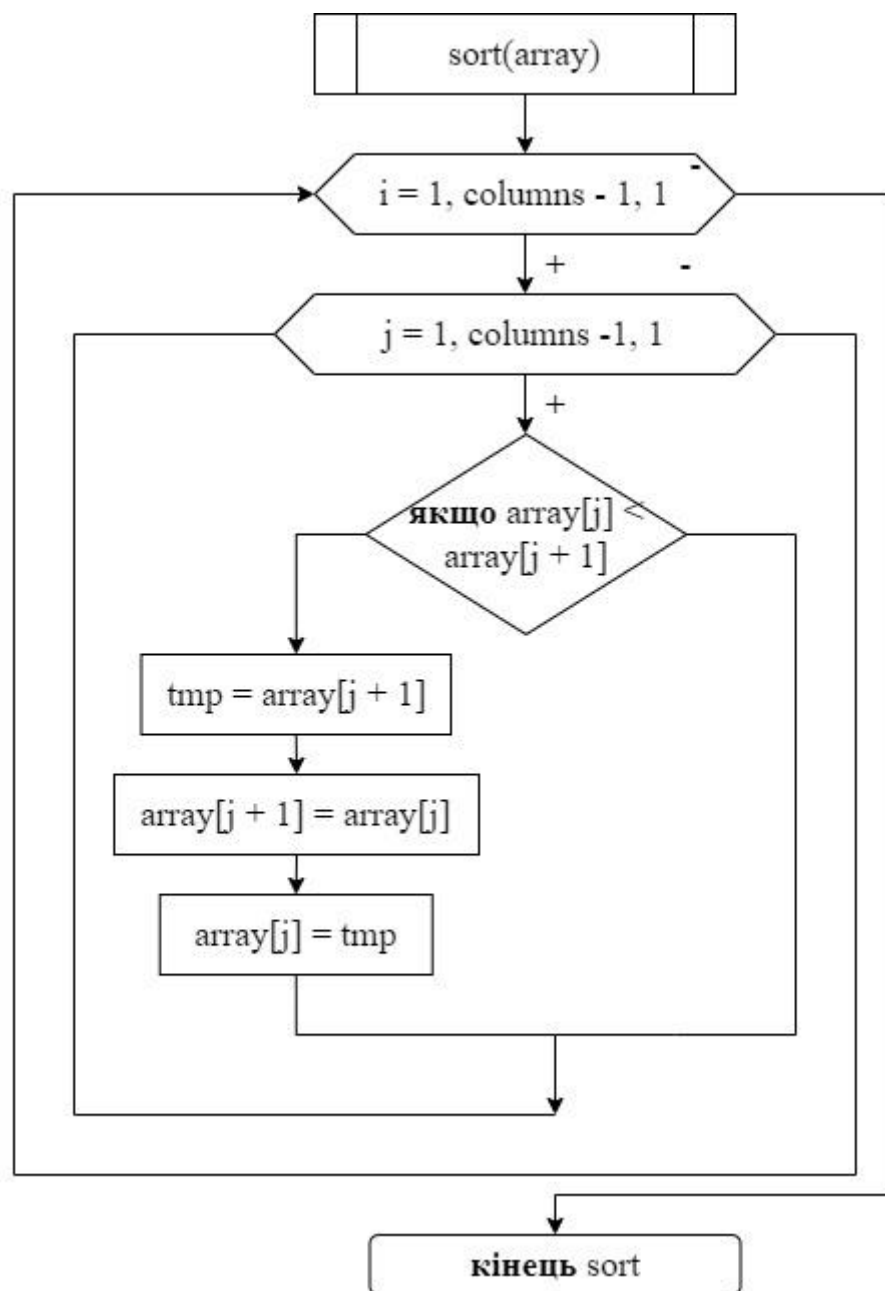
Блок-схема



Блок схеми підпрограми:







Код програми:

```
D).cpp # X
ACD) (Global Scope)

#include <iostream>
#include <iomanip>
#include <stdlib.h>
#include <time.h>
using namespace std;

const int columns = 7;
const int strings = 5;
void outputMatrix(float** matrix);
void initMatrix(float** matrix);
float averageColumn(float** matrix, int numOfColumn);
void sort(float array[]);

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

    float** matrix = new float* [strings];           // Ініціалізація випадково заданої матриці 5 x 7
    for (int i = 0; i < strings; i++)
        matrix[i] = new float [columns];

    initMatrix(matrix);

    cout << "The matrix is: " << endl;               // Виведення матриці
    outputMatrix(matrix);

    float *array = new float [columns];              // Ініціалізація й заповнення одновимірного масиву
    for (int i = 0; i < columns; i++)
        array[i] = averageColumn(matrix, i);

    for (int i = 0; i < strings; i++)                // Видалення матриці
        delete[] matrix[i];
    delete[] matrix;

    cout << "Array: " << '\n';                        // Виведення масиву
    for (int i = 0; i < columns; i++)
        cout << array[i] << " | ";
    cout << endl;


    sort(array);                                     // Сортування й виведення відсортованого масиву
    cout << "Sorted array:" << '\n';
}
```

```
40
41     sort(array); // Сор
42     cout << "Sorted array:" << '\n';
43     for (int i = 0; i < columns; i++)
44         cout << array[i] << " | ";
45
46     delete[] array; // Виде
47
48     return 0;
49 }
50
51 void sort(float array[])
52 {
53     for(int i = 0; i < columns - 1; i++)
54         for (int j = 0; j < columns - 1; j++)
55         {
56             if (array[j] < array[j + 1])
57             {
58                 float tmp = array[j + 1];
59                 array[j + 1] = array[j];
60                 array[j] = tmp;
61             }
62         }
63 }
64
65 float averageColumn(float** matrix, int numOfColumn)
66 {
67     int k = 0, sum = 0;
68     for (int i = 0; i < strings; i++)
69     {
70         if (matrix[i][numOfColumn] >= 0)
71         {
72             sum += matrix[i][numOfColumn];
73             k++;
74         }
75     }
76     if (k == 0)
77         return 0;
78     float average = (float)sum / k;
79     return average;
80 }
```

```
- void outputMatrix(float** matrix)
{
-   for (int i = 0; i < strings; i++)
    {
        for (int j = 0; j < columns; j++)
            cout << setw(3) << matrix[i][j];
        cout << '\n';
    }
}

- void initMatrix(float** matrix)
{
    for (int i = 0; i < strings; i++)
        for (int j = 0; j < columns; j++)
            matrix[i][j] = rand() % 19 - 9;
}
```

Випробування алгоритму:

 Microsoft Visual Studio Debug Console

The matrix is:

```
-6 -3 -7 3 -8 1 4
 6 -4 5 0 0 -1 -5
 8 -1 1 8 -9 -1 6
 8 -2 4 -7 -3 -6 2
-8 9 -6 -5 -2 9 -6
```

Array:

```
7.33333 | 9 | 3.33333 | 3.66667 | 0 | 5 | 4 |
```


Sorted array:

```
9 | 7.33333 | 5 | 4 | 3.66667 | 3.33333 | 0 |
```

D:\КПИ\C++ (Learning)\Stucture 1 (Lesson 64)\Debug\Lab_8 (A

To automatically close the console when debugging stops, enable when debugging stops.

Press any key to close this window . . .

 Microsoft Visual Studio Debug Console

The matrix is:

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

Array:

```
6.5 | 5.33333 | 6.66667 | 4.33333 | 6.5 | 4 | 3.66667 |
```

Sorted array:

```
6.66667 | 6.5 | 6.5 | 5.33333 | 4.33333 | 4 | 3.66667 |
```

D:\КПИ\C++ (Learning)\Stucture 1 (Lesson 64)\Debug\Lab_8 (A

To automatically close the console when debugging stops, enable when debugging stops.

Press any key to close this window . . .

Висновки:

На цій практичній ми дослідити алгоритми пошуку та сортування, набули практичних навичок використання цих алгоритмів під час складання програмних специфікацій. Також зробили постановку задачі склали матмодель написали псевдокод та намлювали блок-схему. Отримали очікуваний результат.