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 7

Виконав студент ІП-15, Гуменюк Олександр Володимирович
(шифр, прізвище, ім'я, по батькові)
Перевірила Вєчерковська Анастасія Сергіївна
(прізвище, ім'я, по батькові)

Київ 2021

Лабораторна робота 8

Дослідження алгоритмів пошуку та сортування

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

Варіант 7

Завдання

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).

7	8 x 5	Дійсний	Із добутку значень елементів рядків двовимірного масиву. Відсортувати обміном за зростанням.

Постановка задачі

Використовуємо 3 підпрограми для ініціалізації матриці (двомірного масиву), ініціалізації масива і сортування цього масиву. Матриці заповнюємо випадковими дійсними числами. Масив заповнюємо добутками елементів кожного рядка матриці. Потім відсортовуємо масив обміном за зростанням.

Результатом розв’язку є обчислення матриці, обчислення і сортування масива.

Побудова математичної моделі

Таблиця імен змінних

Змінна	***Тип***	***Ім'я***	***Призначення***
Матриця	float	matrix	Проміжні дані
Масив	float	array	Проміжні дані/Результат
Кількість рядків матриці	int	row	Початкові дані
Кількість стовпців матриці	int	col	Початкові дані
Процедура для ініціалізації матриці	Процедура	initializeMatrix	Початкові дані
Лічильник i	int	i	Початкові дані
Лічильник j	int	j	Початкові дані
Процедура для ініціалізації масиву	Процедура	initializeArray	Початкові дані
Добуток елементів рядків матриці	float	product	Проміжні дані

Процедура для сортування масиву обміном за зростанням	Процедура	bubbleSortArray	Початкові дані
Змінна, яка тимчасово тримає значення елемента	float	temp	Проміжні дані

Спочатку задаємо 8 і 5 як значення змінних row і col відповідно. Після оголошення матриці matrix ініціалізуємо її, використовуючи процедуру initializeMatrix, яка приймає матрицю, кількість рядочків і стовпчиків як параметри (matrix, row, col). Процедура заповнює матрицю випадковими дійсними числами від 0 до 10, використовуючи функцію randomFloat(0,10).

Після цього ініціалізуємо масив array, використовуючи процедуру initializeArray, яка приймає масив, матрицю, кількість рядочків і стовпчиків як параметри (array, matrix, row, col). Використовуючи два арифметичних цикла (зовнішній і вкладений) з лічильниками і та j, які набувають значень від 0 до row і col відповідно, процедура рахує добуток елементів кожного рядка матриці і записує його у відповідний елемент масива.

Потім використовуємо процедуру bubbleSortArray, яка приймає масив і кількість рядків як параметри (array, row). Ця процедура реалізує сортування обміном за зростанням. Використовуючи два арифметичних цикла (зовнішній і вкладений) з лічильниками і та j, які набувають значень від 0 до row-1, вона перебирає кожне значення масива, і якщо воно більше за

наступне (array[j + 1] < array[j]), то вони переставляються (для перестановки використовується змінна temp).

Після закінчення роботи процедури bubbleSortArray виводимо масив array.

Розв'язання

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

Крок 1. Визначимо основні дії.

Крок 2. Ініціалізація змінних row і col

Крок 3. Ініціалізація матриці

Крок 4. Ініціалізація масиву

Крок 5. Сортування та виведення масиву

Псевдокод

Основна програма:

Крок 1

початок

Ініціалізація змінних row і col

Ініціалізація матриці

Ініціалізація масиву

Сортування та виведення масиву

кінець

Крок 2

початок

row := 8

col := 5

Ініціалізація матриці

Ініціалізація масиву

Сортування та виведення масиву

кінець

Крок 3

початок

row := 8

col := 5

initializeMatrix (matrix)

Ініціалізація масиву

Сортування та виведення масиву

кінець

Крок 4

початок

row := 8

col := 5

initializeMatrix (matrix)

initializeArray (array)

Сортування та виведення масиву

кінець

Крок 5

початок

row := 8

col := 5

initializeMatrix (matrix)

initializeArray (array)

bubbleSortArray(array)

виведення array

кінець

Підпрограми:

initializeMatrix (matrix, row, col)

повторити для i **від** 1 **до** row

повторити для j **від** 1 **до** col

matrix[i][j] = randomFloat(0,10)

все повторити

все повторити

кінець

initializeArray (array, matrix, row, col)

повторити для i **від** 1 **до** row

product := 1

повторити для j **від** 1 **до** col

product := product * matrix[i][j]

все повторити

array[i] = product

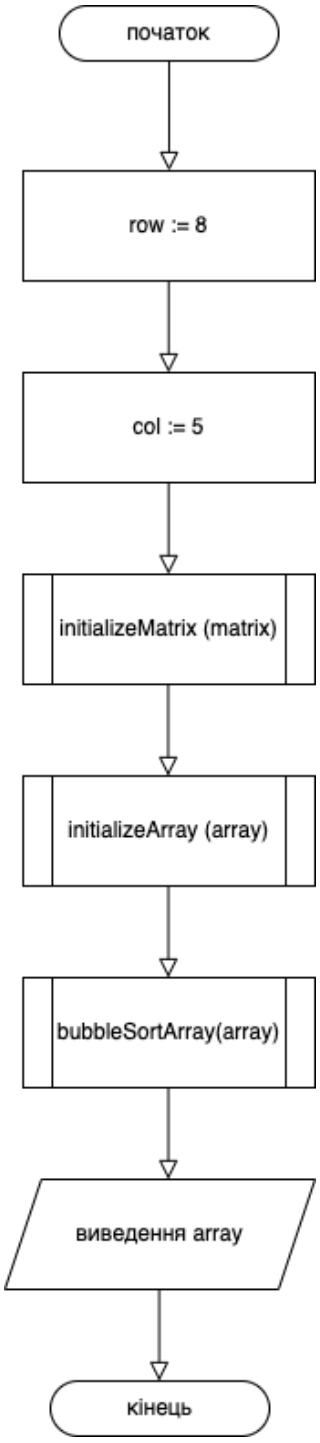
все повторити

кінець

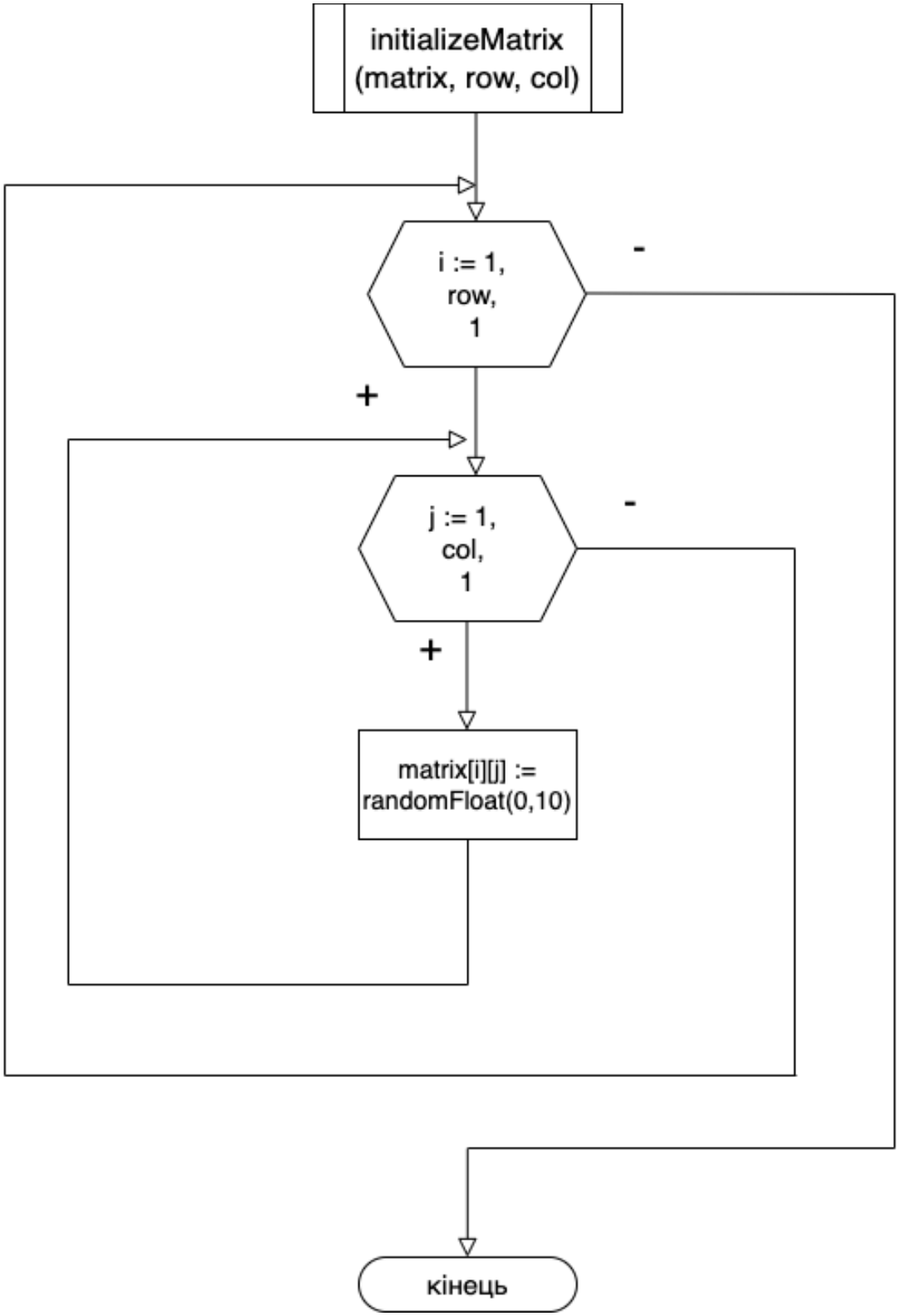
```
bubbleSortArray (array, row)
        повторити  для i від 1 до (row - 1)
                повторити для j від 1 до (row - 1)
                        якщо array[j + 1] < array[j]
                            то
                                temp := array[j + 1]
                                array[j + 1] := array[j]
                                array[j] := temp
                        все якщо
                все повторити
        все повторити
кінець
```

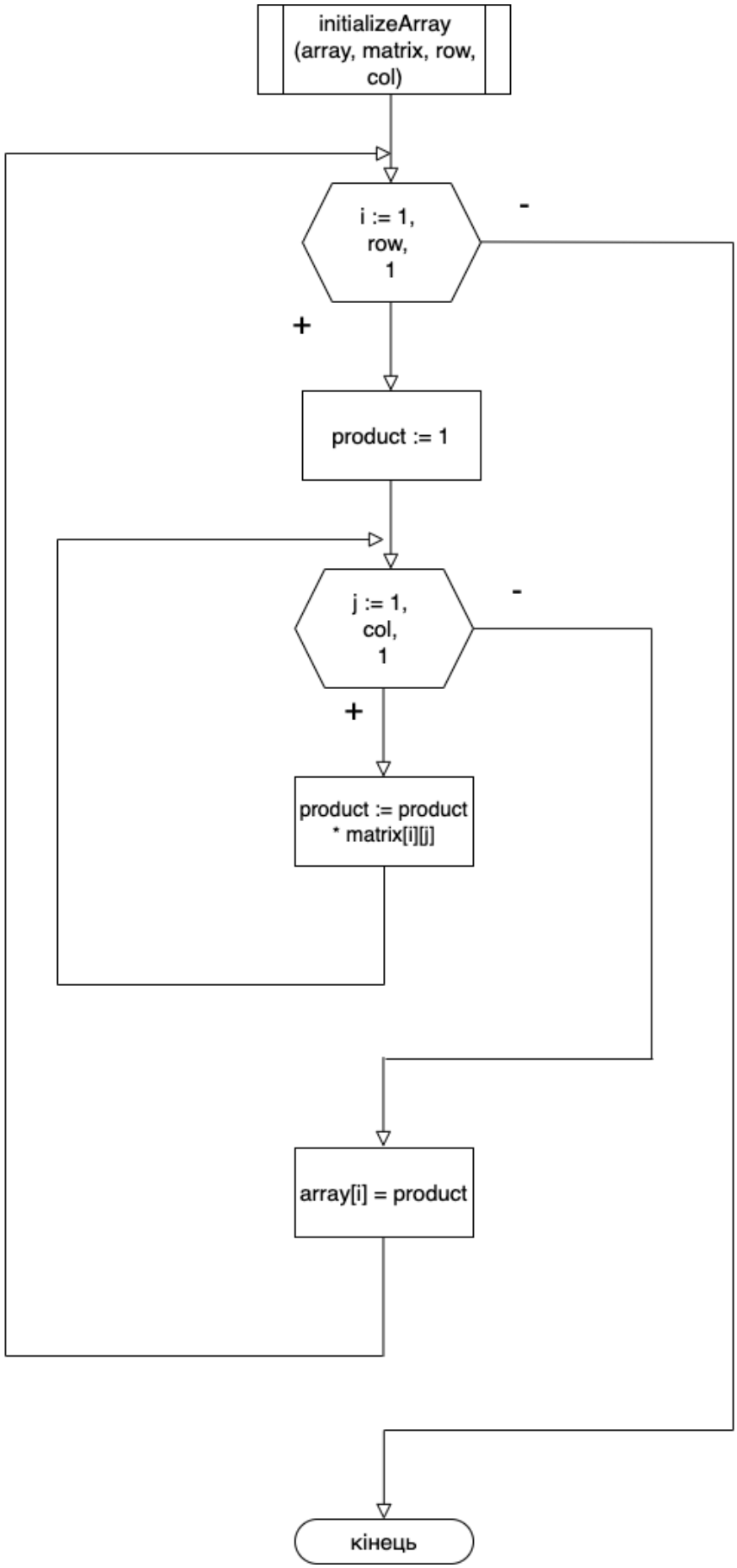
Блок-схема:

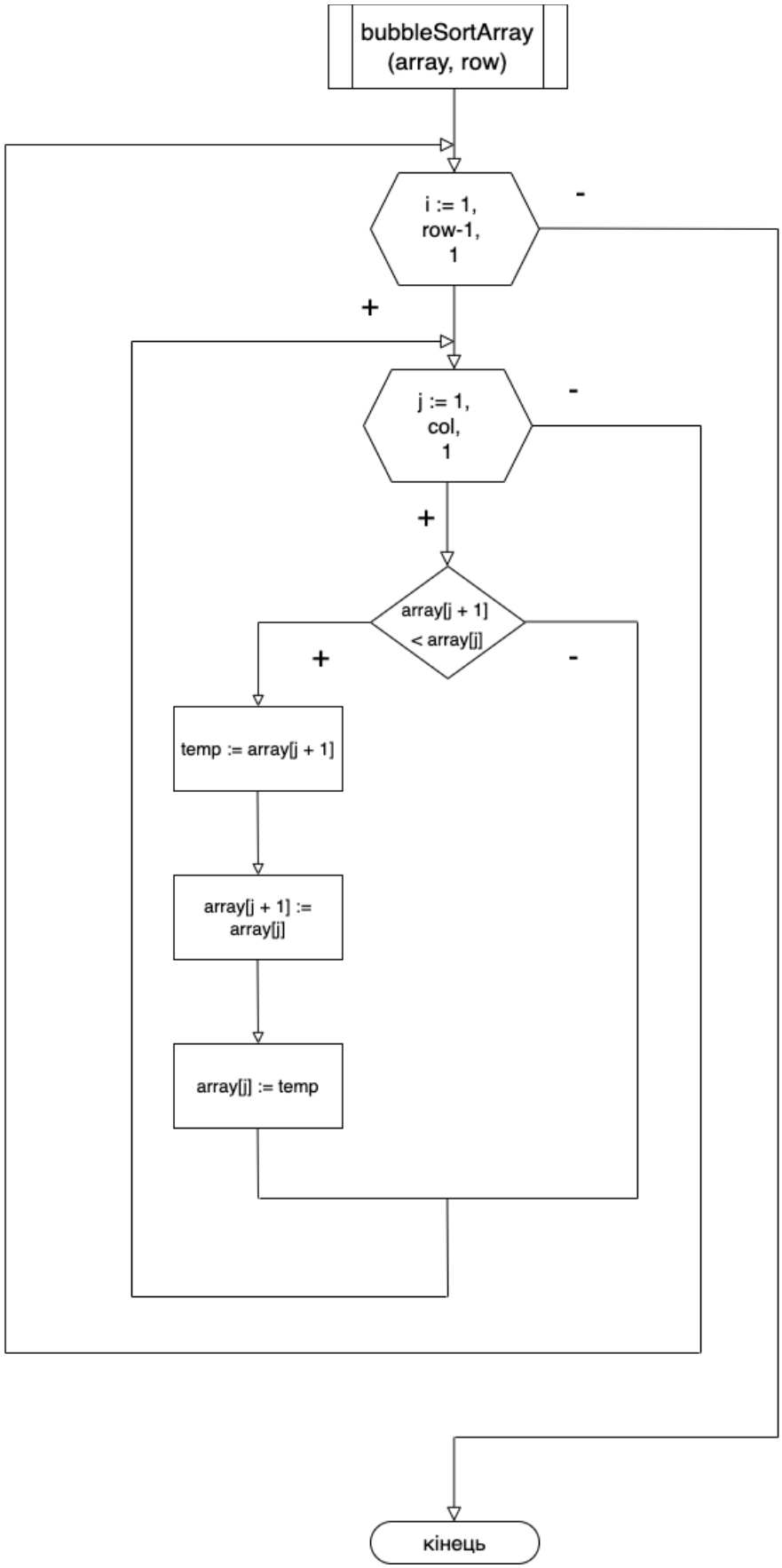
Основна програма:



Підпрограми:




initializeArray
(array, matrix, row,
col)
i := 1,
row,
1
-
+
product := 1
j := 1,
col,
1
-
+
product := product
* matrix[i][j]
array[i] = product
кінець


bubbleSortArray
(array, row)
i := 1,
row-1,
1
-
+
j := 1,
col,
1
-
+
array[j + 1]
< array[j]
+
-
temp := array[j + 1]
array[j + 1] :=
array[j]
array[j] := temp
кінець

Код:

Основна програма:

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

    int row = 8, col = 5;

    float ** matrix = new float*[row];
    for (int i = 0; i < row; i++){
        matrix[i] = new float[col];
    }

    initializeMatrix(matrix, row, col);

    float * array = new float[row];
    initializeArray(array, matrix, row, col);


    cout << "Matrix" << endl;
    for (int i = 0; i < row; i++){
        for (int j = 0; j < col; j++){
            printf("%8.3f", matrix[i][j]);
        }
        cout << endl;
    }
    cout << endl;
```

```
    cout << "Unsorted Array" << endl;
    for (int i = 0; i < row; i++){
        printf("%-11.3f", array[i]);
    }
    cout << endl;

    bubbleSortArray(array, row);

    cout << "\nSorted Array" << endl;
    for (int i = 0; i < row; i++){
        printf("%-11.3f", array[i]);
    }
    cout << "\n\n";


    delete [] array;

    for (int i = 0; i < row; i++){
        delete[] matrix[i];
    }
    delete[] matrix;

    return 0;
}
```

Підпрограми:

```
void initializeMatrix(float ** matrix, int row, int col){

    for (int i = 0; i < row; i++){
        for (int j = 0; j < col; j++){
            matrix[i][j] = (float)(rand())/(float)RAND_MAX*10;
        }
    }
}
```

```
void initializeArray(float * array, float ** matrix, int row, int col){

    float product;

    for (int i = 0; i < row; i++){
        product = 1;

        for (int j = 0; j < col; j++){
            product *= matrix[i][j];
        }
        array[i] = product;
    }
}
```

```
void bubbleSortArray(float * array, int row){

    float temp;

    for (int i = 0; i < row - 1; i++){
        for (int j = 0; j < row - 1; j++){

            if (array[j + 1] < array[j]){

                temp = array [j + 1];
                array [j + 1] = array[j];
                array [j] = temp;
            }
        }
    }
}
```

```
Matrix
   0.629   9.240   9.620   5.264   3.623
   5.112   6.163   2.537   6.128   8.687
   6.625   4.072   5.805   3.089   9.784
   3.078   9.627   8.727   9.632   7.335
   9.344   0.271   3.995   7.704   5.910
   4.923   7.678   1.006   6.011   5.268
   5.754   6.781   5.246   7.513   6.051
   4.358   0.661   0.978   9.768   8.275

Unsorted Array
1066.946   4255.637   4732.884   18272.199  460.583    1204.108   9306.803   227.675

Sorted Array
227.675    460.583    1066.946   1204.108   4255.637   4732.884   9306.803   18272.199
```

Висновки

Протягом восьмої лабораторної роботи я дослідив методи послідовного пошуку у впорядкованих і невпорядкованих послідовностях та набув практичних навичок їх використання під час складання програмних специфікацій. В результаті я отримав алгоритм, що ініціалізую матрицю випадковими дійсними числами, а також ініціалізую масив за даною умовою та сортує цей масив.