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

«Дослідження алгоритмів обходу масивів»

Варіант 26

Виконав студент ІП-12, Саркісян Валерія Георгіївна

Перевірив _____
(прізвище, ім'я, по батькові)

Київ 2021

Лабораторна робота 9
Дослідження алгоритмів обходу масивів

Мета – дослідити алгоритми обходу масивів, набути практичних навичок використання цих алгоритмів під час складання програмних специфікацій.

Завдання. Розробити алгоритм та написати програму, яка складається з наступних дій:
1. Опису змінної індексованого типу (двовимірний масив) згідно з варіантом.
2. Ініціювання змінної, що описана в п.1 даного завдання.
3. Обчислення змінної, що описана в п.1, згідно з варіантом.

26	Задано матрицю дійсних чисел A[m,n]. При обході матриці по стовпчиках знайти в ній останній додатний елемент X і його місцезнаходження. Підрахувати кількість елементів над побічною діагоналлю, більших за X.

1. ***Постановка задачі.***
   Результатом розв'язку є кількість елементів матриці, що знаходяться над побічною діагоналлю і більші за останній додатній елемент матриці. За умовою спочатку потрібно згенерувати матрицю розмірністю m×n та заповнити її випадковими дійсними числами з діапазону від -100 до 100. Далі обходом по стовпчиках знайти останній додатній елемент X матриці і його місцезнаходження. Потім підрахувати кількість елементів матриці, що знаходяться над побічною діагоналлю і більші за X. У разі якщо матриця не є квадратною, тобто m != n, за діагональ візьмемо ту, де m = n.
2. ***Побудова математичної моделі***.

Змінна	Тип	Ім'я	Призначення
Кількість рядків масиву	цілий, сталий	m	Початкове дане, константа
Кількість стовпців масиву	цілий, сталий	n	Початкове дане, константа
Матриця	дійсний	arr[m][n]	Вхідне дане
Генерація двовимірного масива (матриці)	Функція, порожній	generateArr()	Проміжне дане
останній додатній елемент матриці	дійсний	X	Проміжне дане
Знаходження X	Функція, порожній	find_X()	Проміжне дане
Обчислення кількості елементів що знаходяться над побічною діагоналлю і більші за X	Функція, порожній	count_elements()	Проміжне дане
матриця	дійсний	A	Формальний параметр, проміжне дане
Кількість рядків	цілий	rows	Формальний параметр, проміжне дане

Кількість стовпців	цілий	columns	Формальний параметр, проміжне дане
Допоміжна змінна	цілий	temp	Проміжне дане
рядок в якому знаходиться X	цілий	Xm	Проміжне дане
стовпець в якому знаходиться X	цілий	Xn	Проміжне дане
Лічильник	цілий	i	Проміжне дане
Лічильник	цілий	j	Проміжне дане
Лічильник	цілий	k	Проміжне дане
Елемент X	дійсний	x	Формальний параметр, проміжне дане
Кількість елементів що знаходяться над побічною діагоналлю і більші за X	цілий	kilk	Проміжне дане
Генерація випадкових чисел	Функція	rand()	Проміжне дане

3. Розв'язання.

Основна програма:

Крок 1. Визначимо основні дії.

Крок 2. Деталізуємо дію ініціалізації змінних.

Крок 3. Деталізуємо дію заповнення матриці випадковими дійсними числами і її виведення.

Крок 4. Деталізуємо дію знаходження останнього додатного елемента матриці.

Крок 5. Деталізуємо дію обчислення кількості елементів що знаходяться над побічною діагоналлю і більші за X.

Підпрограми:

Крок 3.1. Деталізуємо дію заповнення матриці випадковими дійсними числами і її виведення.

Крок 4.1. Деталізуємо дію знаходження останнього додатного елемента матриці.

Крок 5.1. Деталізуємо дію обчислення кількості елементів що знаходяться над побічною діагоналлю і більші за X.

Крок 1

Початок

Ініціалізація змінних

заповнення і виведення матриці

знаходження останнього додатного елемента матриці

обчислення кількості елементів що знаходяться над побічною діагоналлю і більші за X

Кінець

Крок 2

Початок

m = 6, n = 5, arr[m][n]

заповнення і виведення матриці

знаходження останнього додатного елемента матриці

обчислення кількості елементів що знаходяться над побічною діагоналлю і більші за X

Кінець

Крок 3

Початок

m = 6, n = 5, arr[m][n]

generateArr (arr, m, n)

знаходження останнього додатного елемента матриці

обчислення кількості елементів що знаходяться над побічною діагоналлю і більші за X

Кінець

Крок 4

Початок

m = 6, n = 5, arr[m][n]

generateArr (arr, m, n)

X=arr[0][0]

find_X(arr, m, n, X)

обчислення кількості елементів що знаходяться над побічною діагоналлю і більші за X

Кінець

Крок 5

Початок

m = 6, n = 5, arr[m][n]

generateArr (arr, m, n)

X=arr[0][0]

find_X(arr, m, n, X)

count_elements(arr, m, n, X)

Кінець

Псевдокод підпрограм.

Крок 3.1. Функція **generateArr(A, rows, columns)**

Початок generateArr(A, rows, columns)

Для i від 0 до rows із кроком 1 **повторити**

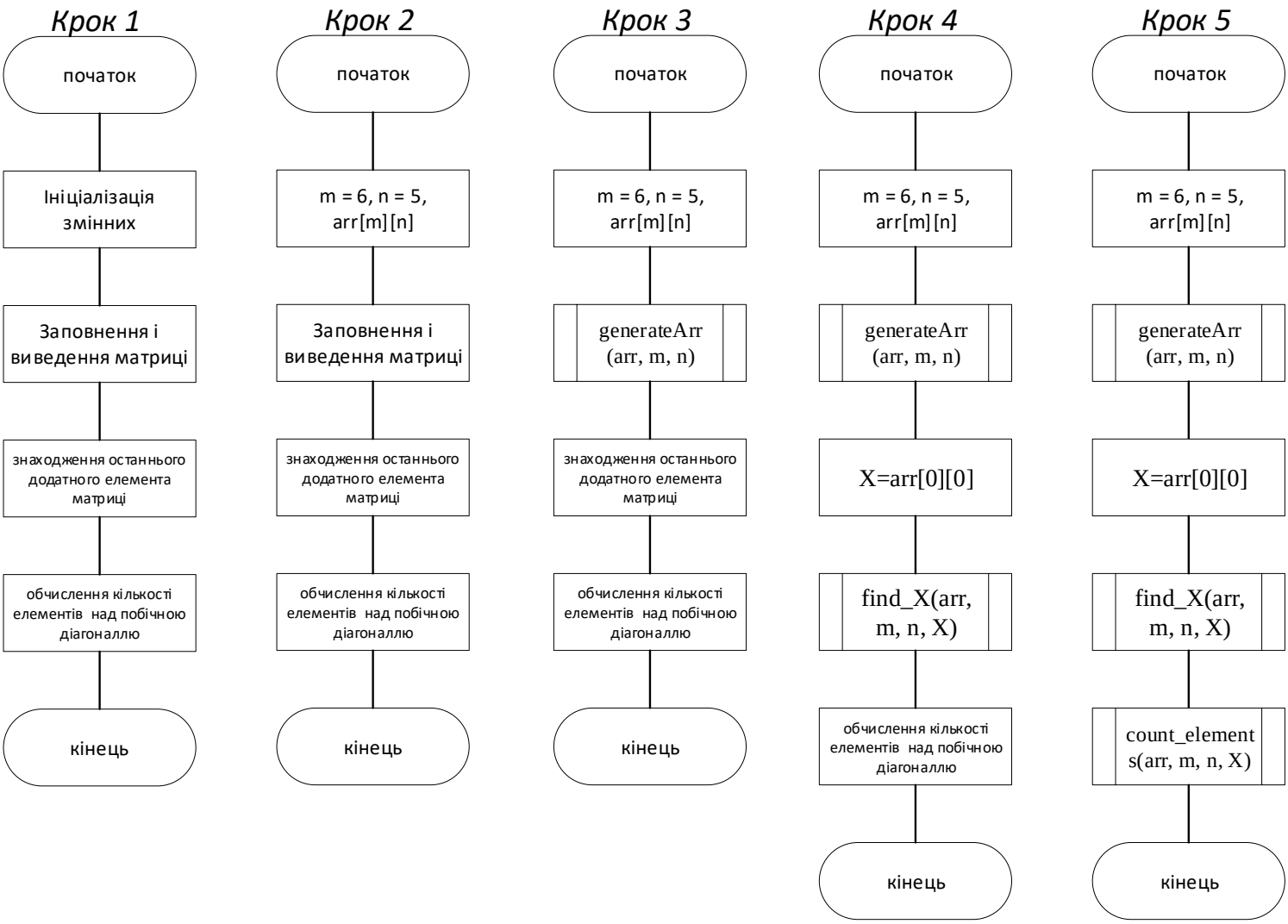
Для j від 0 до columns із кроком 1 **повторити**

A[i][j] = rand() / RAND_MAX * 2 * 100 – 100
Вивести A[i][j]
Все повторити
Все повторити
Кінець generateArr

Крок 4.1. Функція **find_X (A, rows, columns, x)**
Початок find_X (A, rows, columns, x)
i = 0, temp, Xm = 0, Xn = 0
Для j від 0 до columns із кроком 1 **повторити**
Якщо i==0 або i==-1 **то**
i=0, temp=1
інакше якщо i== rows **то**
temp=-1, i=i-1
все якщо
Для k від 0 до rows із кроком 1 **повторити**
Якщо A[i][j] > 0 **то**
x = A[i][j]
Xm = i + 1
Xn = j + 1
все якщо
i = i + temp
Все повторити
Все повторити
Вивести x, Xm, Xn
Кінець find_X

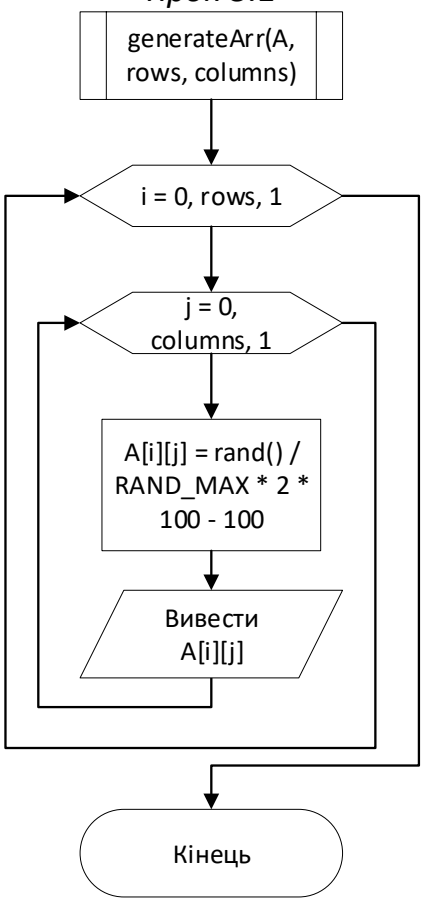
Крок 5.1. Функція **count_elements (A, rows, columns, x)**
Початок count_elements (A, rows, columns, x)
kilk=0
Для i від 0 до rows із кроком 1 **повторити**
Для j від 0 до columns із кроком 1 **повторити**
Якщо i + j < rows – 1 та A[i][j] > x **то**
kilk= kilk+1
все якщо
Все повторити
Все повторити
Вивести kilk
Кінець count_elements

4. ***Блок-схема.***

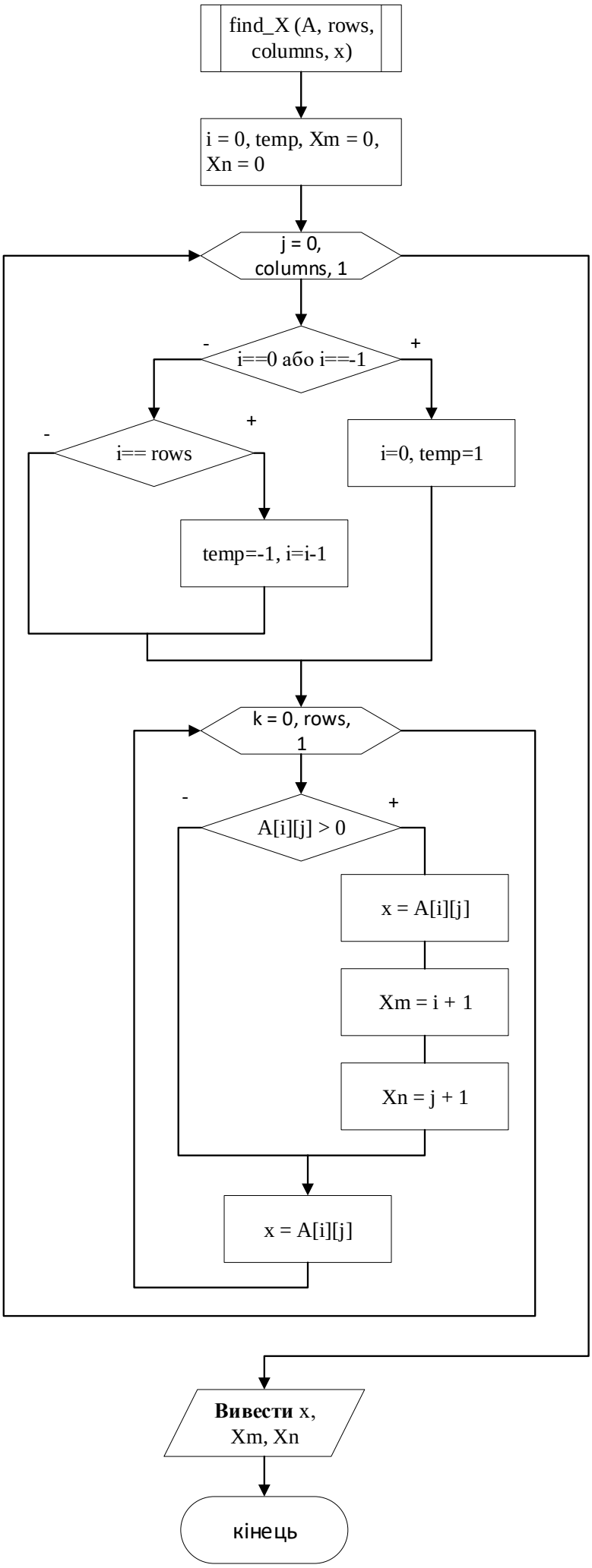


Підпрограми:

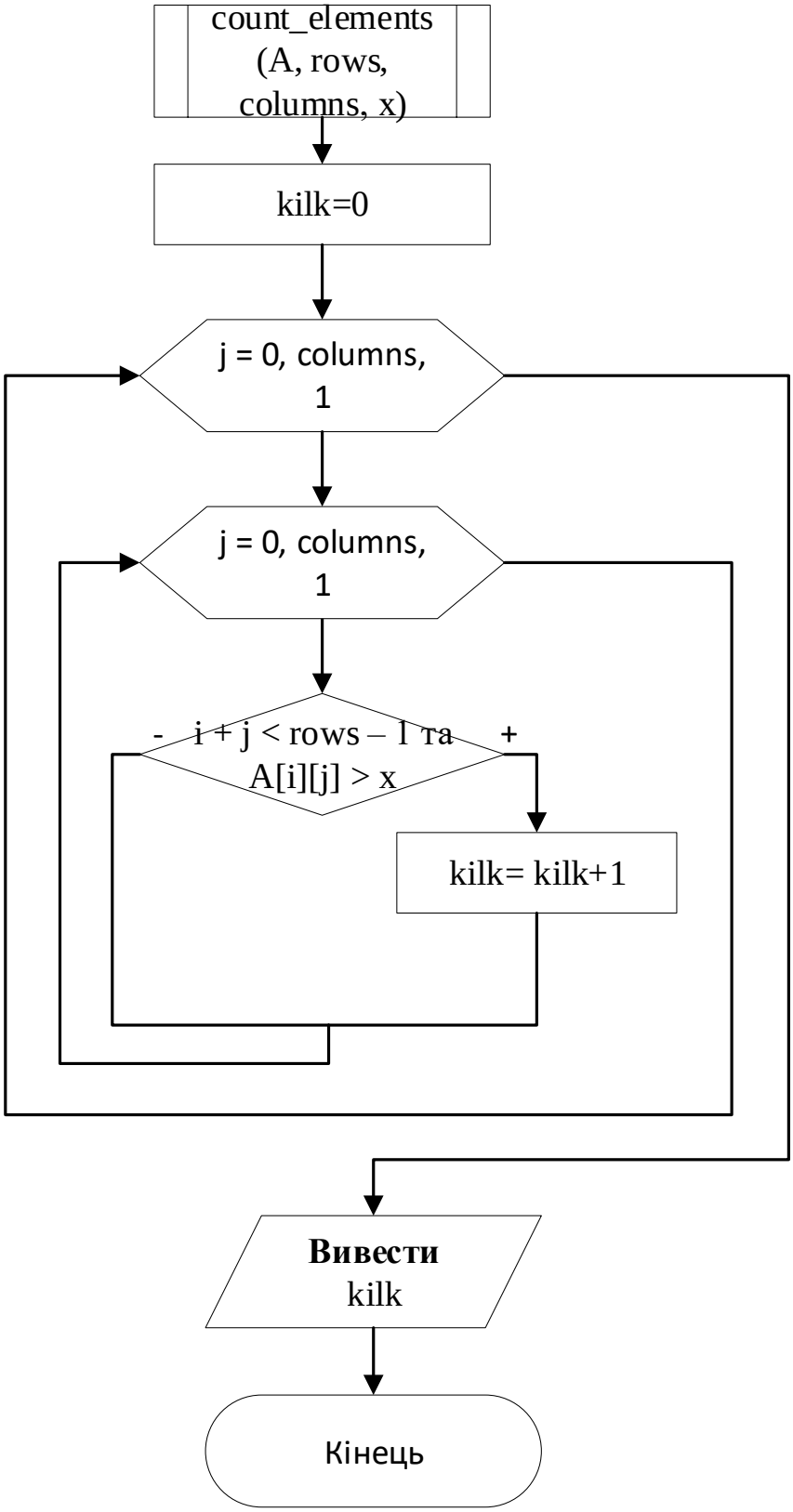
Крок 3.1



Крок 4.1



Крок 5.1



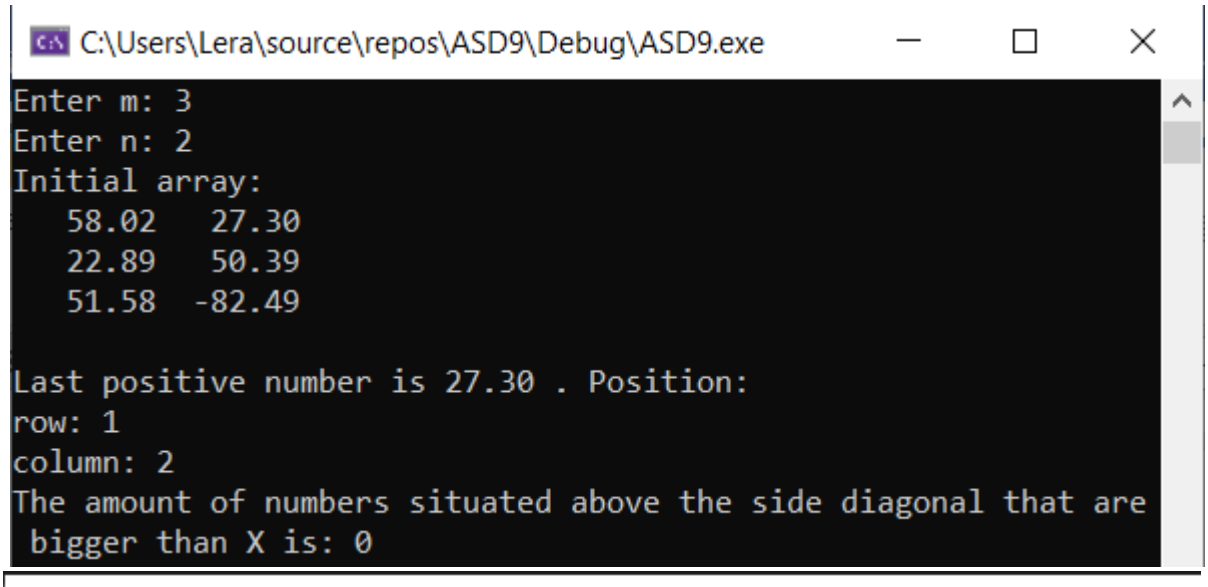
5. *Код програми (C++):*

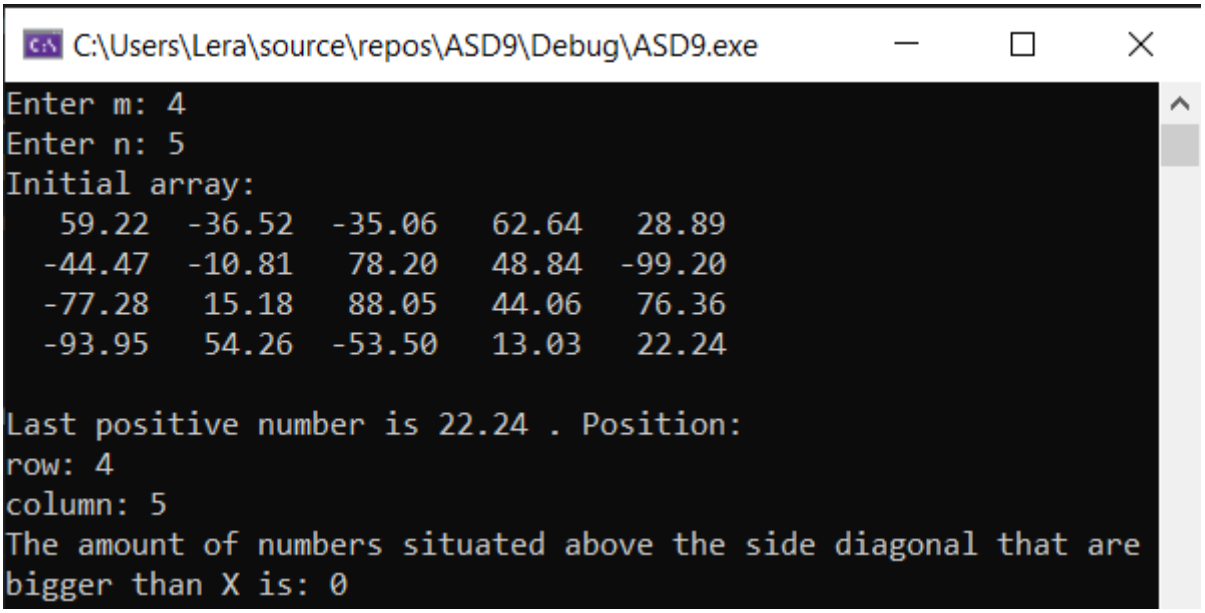
```
#include <iostream>
#include <time.h>
using namespace std;

void generateArr(float**, int, int);
```

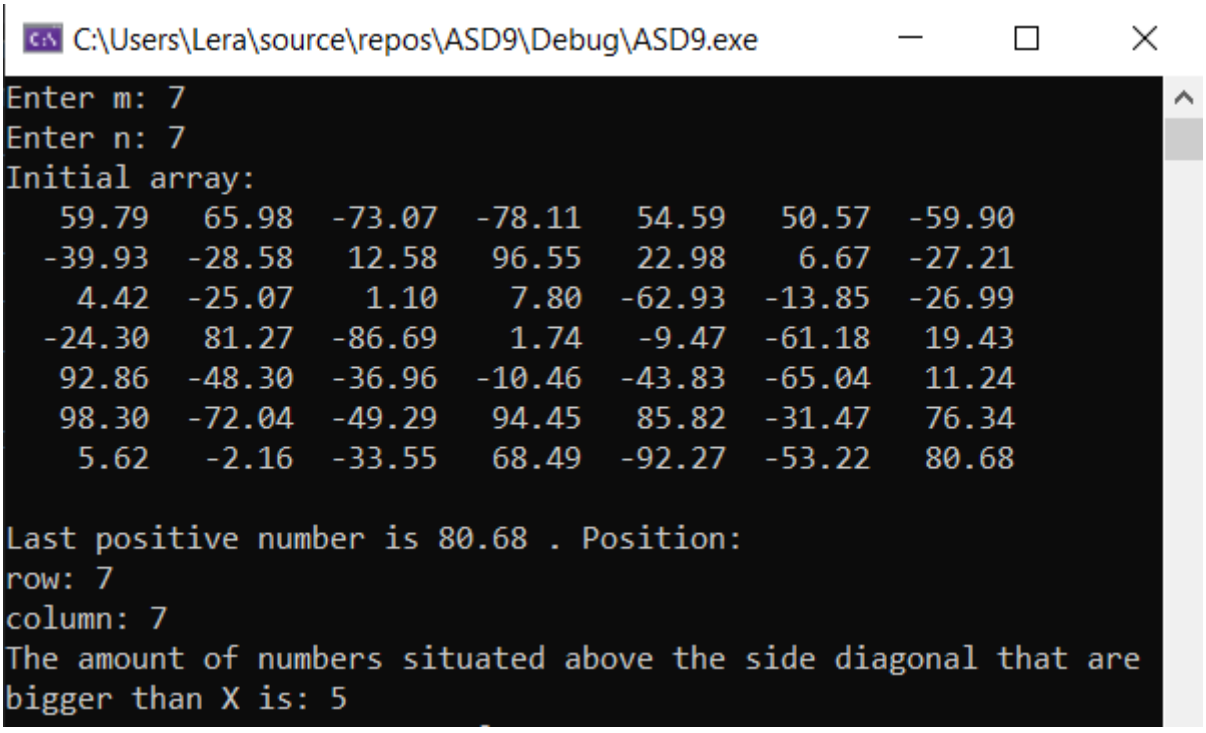
```
void find_X(float**, int, int, float);
void count_elements(float**, int, int, float);

int main()
{
   int m, n;
   cout << "Enter m: "; cin >> m;
   cout << "Enter n: "; cin >> n;
   float** arr = new float* [m];
   for (int i = 0; i < m; i++) {
         arr[i] = new float[n];
   }
   cout << "Initial array:\n";
   generateArr(arr, m, n);
   float X=arr[0][0];
   find_X(arr, m, n, X);
   count_elements(arr, m, n, X);
   for (int i = 0; i < m; i++) {
         delete[] arr[i];
   }
   delete[] arr;
   system("pause");
}

void generateArr(float** A, int rows, int columns) {
   srand(time(NULL));
   for (int i = 0; i < rows; i++) {
         for (int j = 0; j < columns; j++) {
               A[i][j] = (float)rand() / RAND_MAX * 2 * 100 - 100;
               printf("%8.2f", A[i][j]);
         }
         printf("%s\n", "");
   }
}

void find_X(float** A, int rows, int columns, float x) {
   int i = 0, temp, Xm = 0, Xn = 0;
   for (int j = 0; j < columns; j++) {
         if (i == 0 || i == -1){
               i = 0;
               temp = 1;
         }
         else if (i == rows) {
               temp = -1;
               i--;
         }
         for (int k = 0; k < rows; k++) {
               if (A[i][j] > 0) {
                     x = A[i][j];
                     Xm = i + 1;
                     Xn = j + 1;
               }
               i += temp;
         }
   }
   printf("\nLast positive number is %5.2f . Position:\nrow: %d \ncolumn: %d", x, Xm,
   Xn);
}
```

```
void count_elements(float** A, int rows, int columns, float x) {
   int kilk = 0;
   for (int i = 0; i < rows; i++) {
         for (int j = 0; j < columns; j++) {
               if ((i + j < rows - 1) && (A[i][j] > x)) {
                     kilk++;
               }
         }
   }
   printf("\nThe amount of numbers situated above the side diagonal that are bigger than
   X is: %d\n", kilk);
}
```

Тестування:


C:\Users\Lera\source\repos\ASD9\Debug\ASD9.exe
Enter m: 3
Enter n: 2
Initial array:
58.02 27.30
22.89 50.39
51.58 -82.49
Last positive number is 27.30 . Position:
row: 1
column: 2
The amount of numbers situated above the side diagonal that are bigger than X is: 0


C:\Users\Lera\source\repos\ASD9\Debug\ASD9.exe
Enter m: 4
Enter n: 5
Initial array:
59.22 -36.52 -35.06 62.64 28.89
-44.47 -10.81 78.20 48.84 -99.20
-77.28 15.18 88.05 44.06 76.36
-93.95 54.26 -53.50 13.03 22.24
Last positive number is 22.24 . Position:
row: 4
column: 5
The amount of numbers situated above the side diagonal that are bigger than X is: 0


C:\Users\Lera\source\repos\ASD9\Debug\ASD9.exe
Enter m: 7
Enter n: 7
Initial array:
59.79 65.98 -73.07 -78.11 54.59 50.57 -59.90
-39.93 -28.58 12.58 96.55 22.98 6.67 -27.21
4.42 -25.07 1.10 7.80 -62.93 -13.85 -26.99
-24.30 81.27 -86.69 1.74 -9.47 -61.18 19.43
92.86 -48.30 -36.96 -10.46 -43.83 -65.04 11.24
98.30 -72.04 -49.29 94.45 85.82 -31.47 76.34
5.62 -2.16 -33.55 68.49 -92.27 -53.22 80.68
Last positive number is 80.68 . Position:
row: 7
column: 7
The amount of numbers situated above the side diagonal that are bigger than X is: 5

6. ***Висновки.***

На цій лабораторній роботі було досліджено алгоритми обходу масивів, набуто практичних навичок використання цих алгоритмів під час складання програмних специфікацій. Для розв’язання поставленої задачі було використано метод обходу масиву стовпчиками.