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-15 Поліщук Валерій Олександрович

(шифр, прізвище, ім'я, по батькові)

Перевірила: Вечерковська Анастасія Сергіївна

(прізвище, ім'я, по батькові)

Київ 2021

Лабораторна робота №9

Дослідження алгоритмів обходу масивів

Варіант 26

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

Постановка задачі

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

1. Опису змінної індексованого типу 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.

Математична модель

Змінна	Тип	Ім'я	Призначення
Двовимірний масив	Дійсний	a	Проміжні дані
Значення j	Цілий	j	Проміжні дані
Значення і	Цілий	і	Проміжні дані
Значення x	Дійсний	x	Проміжні дані

Значення m	Цілий	m	Проміжні дані
Значення n	Цілий	n	Проміжні дані
Значення x_i	Цілий	x_i	Вихідні дані
Значення x_j	Цілий	x_j	Вихідні дані
Значення dir	Цілий	dir	Проміжні дані
Значення count	Цілий	count	Вихідні дані
Функція, що обчислює кількість елементів, що менші за x, над побічною діагоналлю	Функція	Search_Count	Вихідні дані
Функція, що знаходить останній додатній елемент при обході стовпцями	Функція	SearchMax	Вихідні дані
Функція, що заповнює матрицю	Функція	Create_Matrix	Проміжні дані

Random(a,b) – повертає випадкове дійсне число в проміжку від a до b

Вивести: вивід у консоль

Ми створюємо матрицю потрібної розмірності, заповнюємо її за допомогою функції Create_Matrix з використанням функції Random(), потім обходом стовпцями знаходимо останній додатній елемент x та його розташування, після чого знаходимо кількість елементів масиву над побічною діагоналлю, що більші за x.

Розв'язання

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

Крок 1. Визначимо основні дії.

Крок 2. Деталізуємо процес заповнення матриці випадковими значеннями.

Крок 3. Деталізуємо процес знаходження останнього додатного елементу матриці x за допомогою обходу стовпцями.

Крок 3. Деталізуємо процес знаходження кількості елементів матриці над побічною діагоналлю, що більші за x.

Псевдокод

Основна програма

Крок 1

початок

```
    double a[m][n]
    Create_Matrix(a,m,n)
    x = Last_Positive(a, m, n)
    SearchCount(a, m, n, x)
```

кінець

Підпрограма

Create_Matrix(a,m,n)

Початок

повторити

для і від 0 до m-1

повторити

для j від 0 до n-1

a[i][j] := Random(-10,10)

все повторити

все повторити

Кінець

SearchCount (a, m, n, x)

початок

count = 0

повторити

для і від 0 до m-1

повторити

для j від 0 до n-1

якщо ((i + j - 1)< n-2) && a[i, j] > x

то

count++

все якщо

все повторити

все повторити

вивести : "Кількість єлементів над побічною діагоналлю що більші за X - " +count

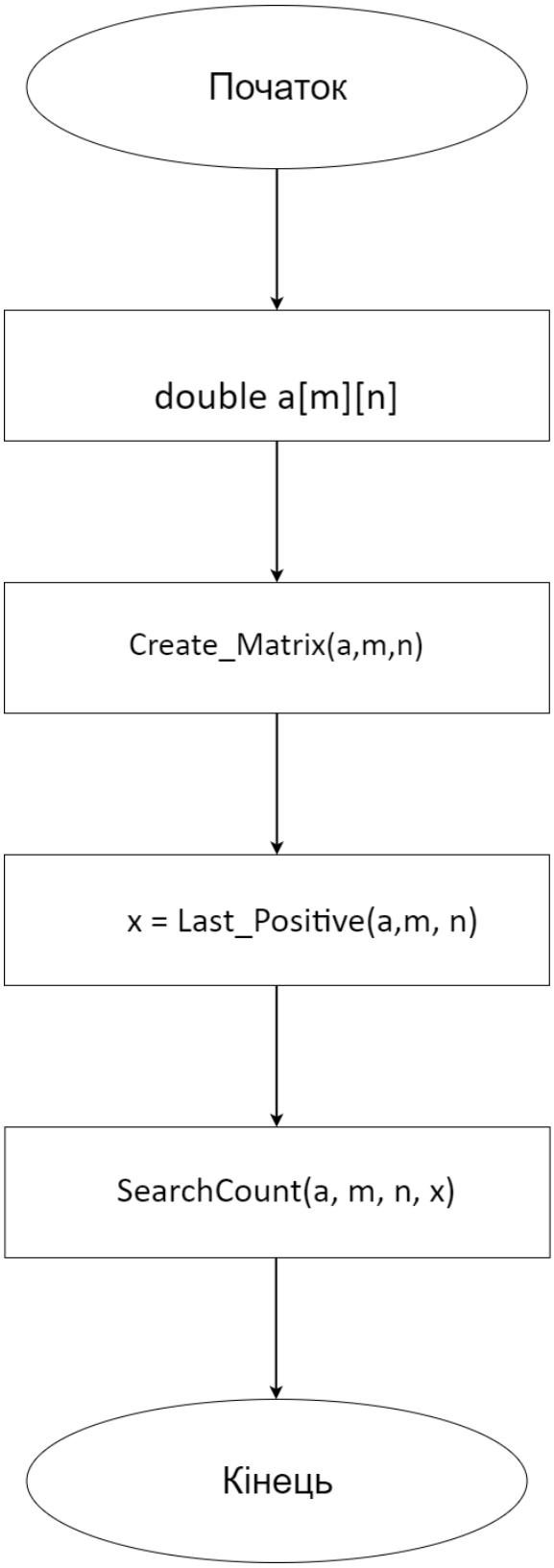
кінець

```
Last_Positive(a,m,n)
початок
  dir = -1
  повторити
      для j від 0 до n-1
          якщо dir < 0
              то
                  повторити
                      для i від 0 до m-1
                      якщо a[i,j]>0
                          то
                              x = a[i, j]
                              x_i = i
                              x_j = j
                      все якщо
                  все повторити
              dir = -dir
              інакше
                  повторити
                      для i від m-1 до 0 включно
                      якщо a[i,j]>0
                          то
                              x = a[i, j]
                              x_i = i
                              x_j = j
                      все якщо
                  все повторити
              dir = -dir
          все якщо
  все повторити
  вивести : "x = " + x
  вивести : "рядок № " + (x_i + 1)
  вивести : " стовпець № " + (x_j + 1)
  повернути x
Кінець
```

Блок-схема

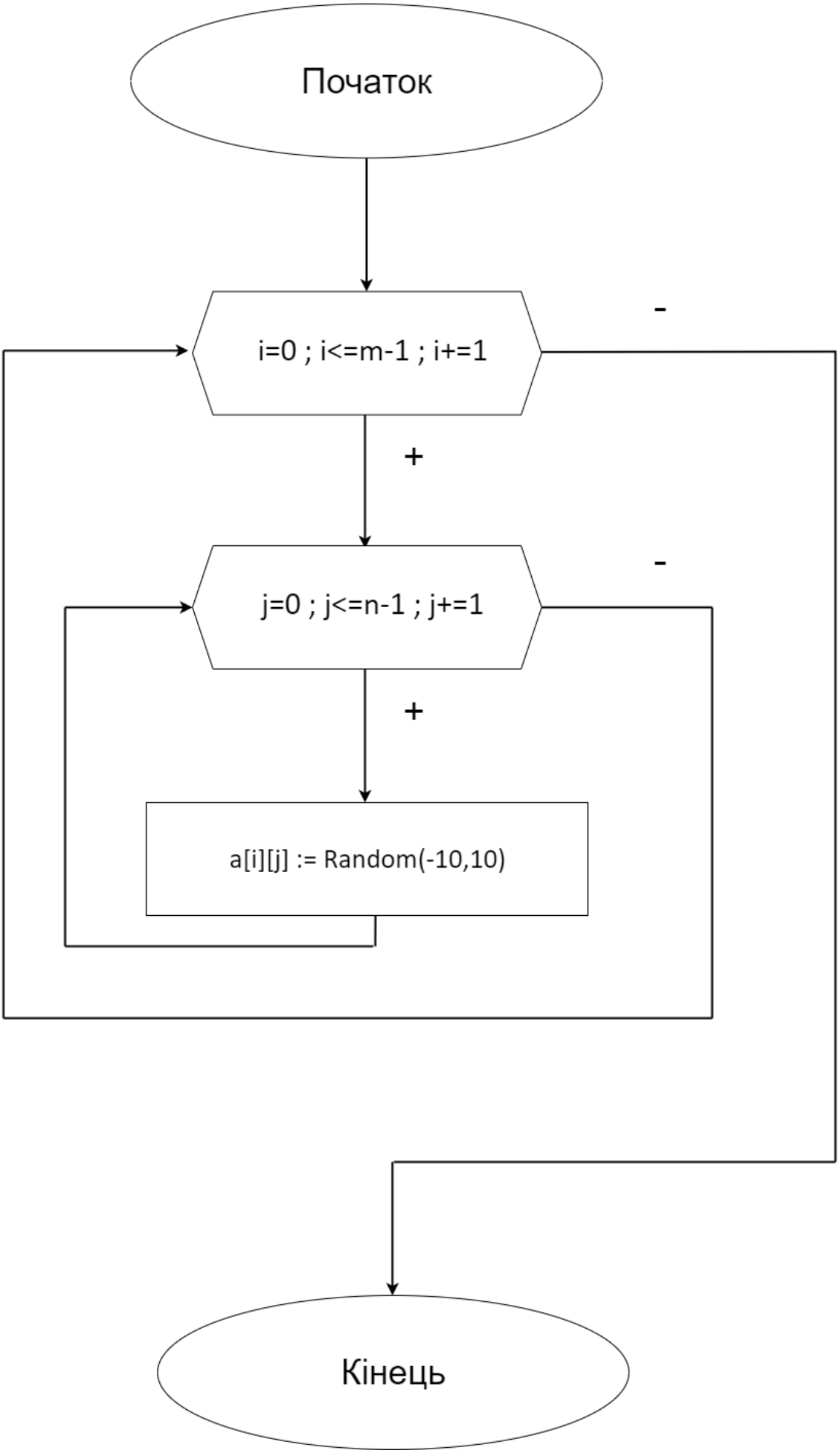
Основна програма

Крок 1

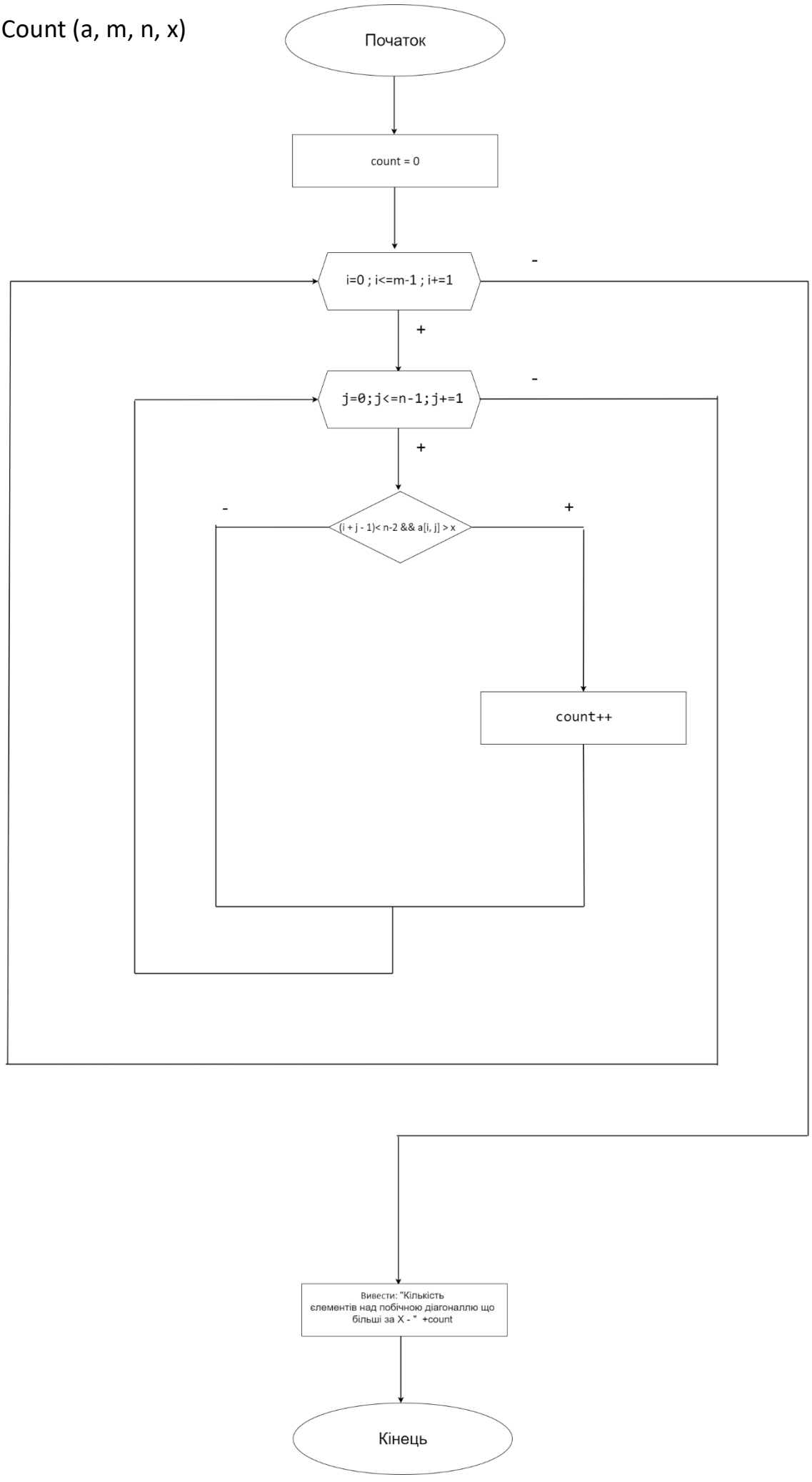


Підпрограми:

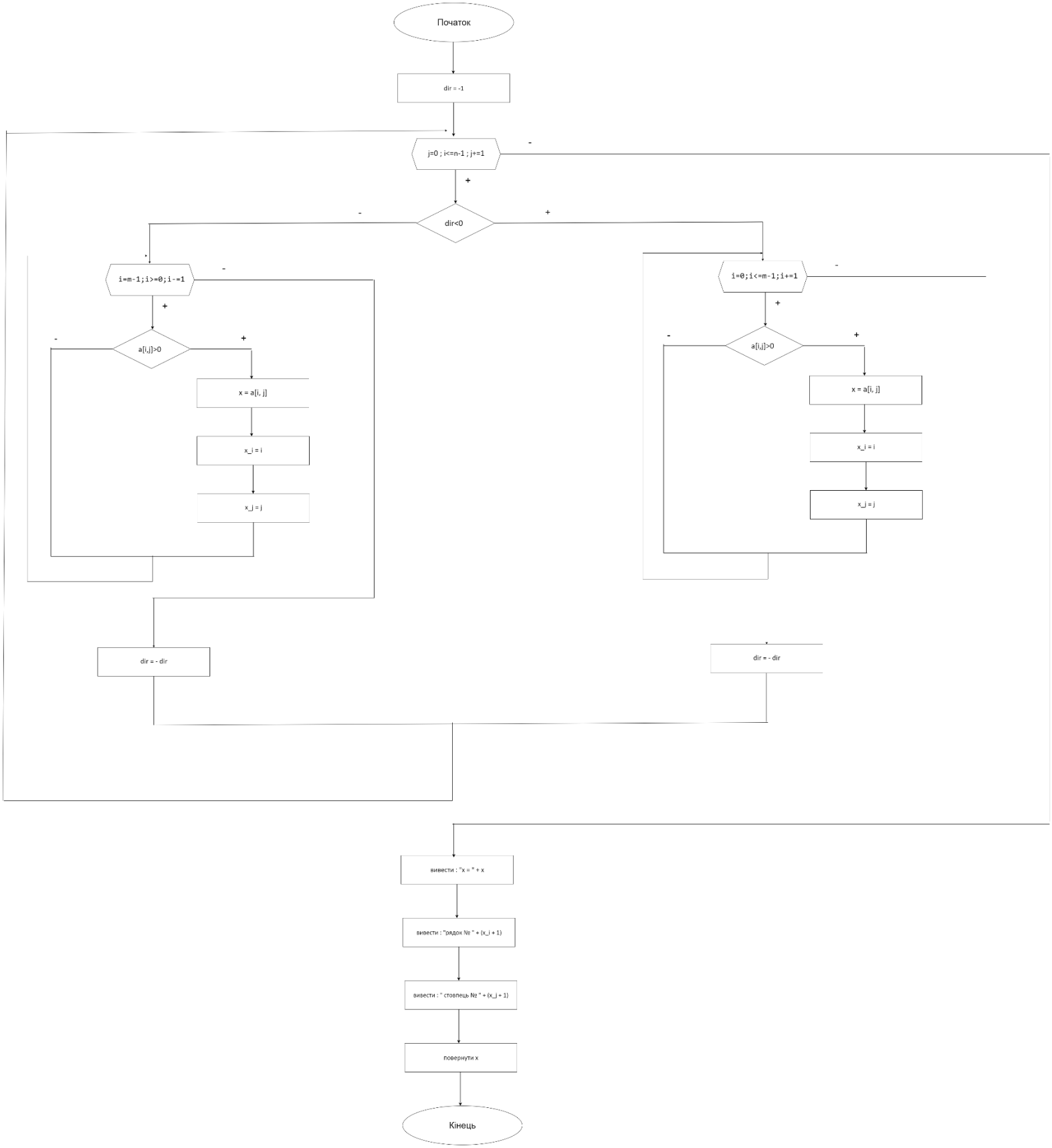
Create_Matrix(a,m,n)



SearchCount (a, m, n, x)



Last_Positive(a,m,n)



Код програми

```
ссылка: 1
private static void Crere_Matrix(double[,] a, int m, int n, Random rnd)
{
    for (int j = 0; j <= n - 1; j++)
    {
        for (int i = 0; i <= m - 1; i++)
        {
            a[i, j] = Math.Round((rnd.NextDouble() + rnd.Next(-10, 10)), 3);
        }
    }

}
ссылка: 1
private static int SearchCount(double[,] a, int m, int n, double x)
{
    int count = 0;
    for (int i = 0; i <= m - 1; i++)
    {
        for (int j = 0; j <= n - 1; j++)
        {
            if (((i + j - 1)< n-2) && a[i, j] > x)
            {
                count++;
            }
        }

    }
    Console.WriteLine("Кількість елементів над побічною діагоналлю, що більші за X - " +count);
    return count;

}
```

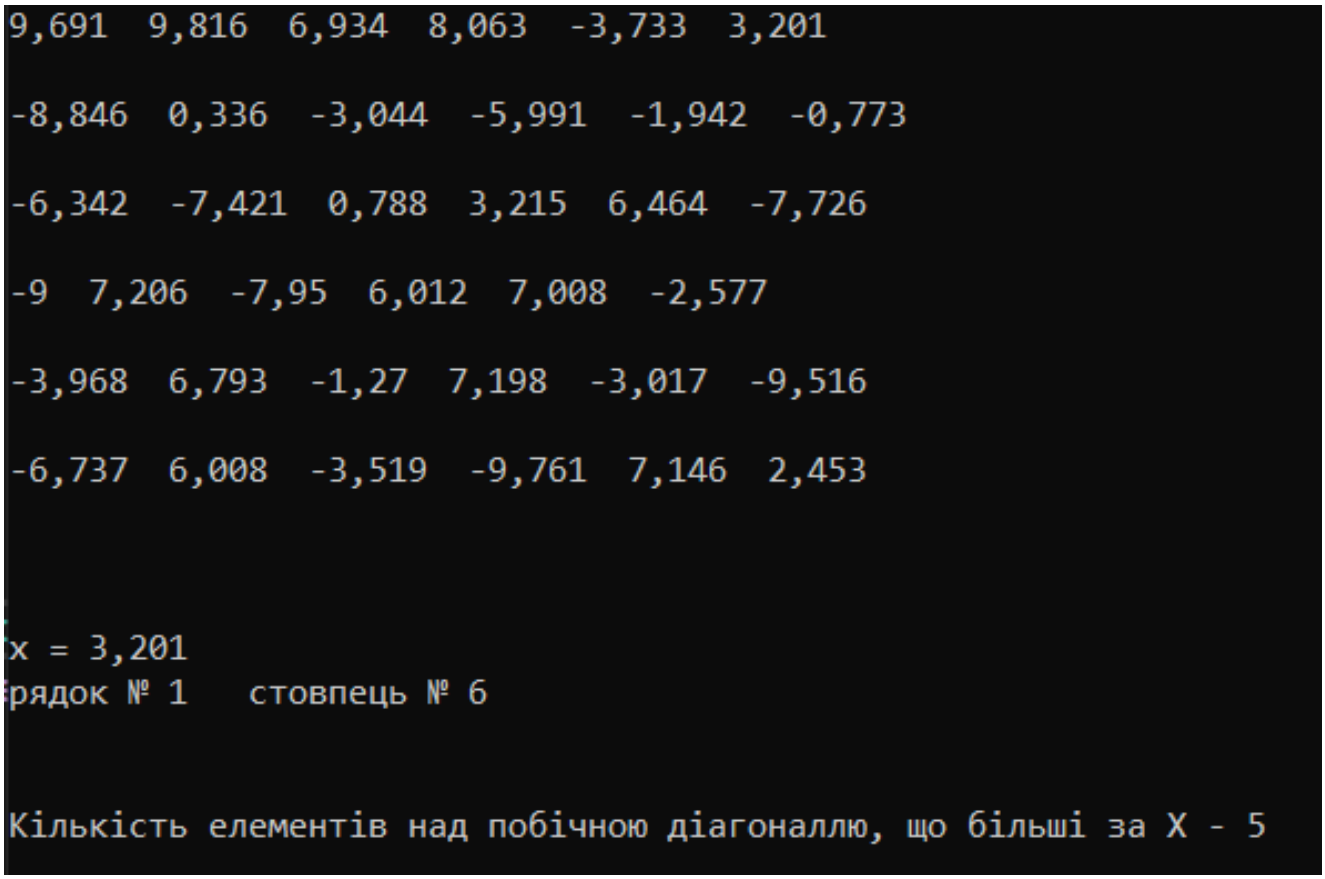
```
ссылка: 1
private static double Last_Positive(double[,] a, int m, int n)
{
    double x = 0;
    int x_i = 0;
    int x_j = 0;
    int dir = -1;
    for (int j = 0; j <= n - 1; j++)
    {
        if (dir < 0)
        {
            for (int i = 0; i <= m - 1; i++)
            {
                if (a[i, j] > 0)
                {
                    x = a[i, j];
                    x_i = i;
                    x_j = j;
                }
            }
            dir = -dir;
        }
        else
        {
            for (int i = m - 1; i >= 0; i--)
            {
                if (a[i, j] > 0)
                {
                    x = a[i, j];
                    x_i = i;
                    x_j = j;
                }
            }
            dir = -dir;
        }

    }
    Console.WriteLine("x = " + x);
    Console.WriteLine("рядок № " + (x_i + 1) + "   стовпець № " + (x_j + 1));
    return (x);
}
Ссылок: 0
```

```
static void Main(string[] args)
{

    const int m = 6; int n = 6;
    double[,] a = new double[m, n];
    double x;
    Random rnd = new Random();
    Crere_Matrix(a, m, n, rnd);
    x = Last_Positive(a, m, n);
    Console.WriteLine();
    Console.WriteLine();
    SearchCount(a, m, n, x);
}
```

Випробування алгоритму



Висновки

Я дослідив алгоритми обходу масивів, набув практичних навичок використання цих алгоритмів під час складання програмних специфікацій.