Додаток 1

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 34

Виконав студент ІП-13 Шиманська Ганна Артурівна
(шифр, прізвище, ім'я, по батькові)

Перевірила Вечерковська Анастасія Сергіївна
(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.

Варіант 34

34	Задано матрицю дійсних чисел A[m,n]. При обході матриці по рядках знайти в ній останній мінімальний елемент X і його місцезнаходження. Порівняти значення X із середньоарифметичним значенням елементів під побічною діагоналлю.

- **Постановка задачі**

  Необхідно розробити алгоритм та написати програму, яка складається з наступних дій:

  1. Опису змінної індексованого типу (двовимірний масив).
  2. Ініціювання даної змінної.
  3. Обчислення даної змінної.

Побудова математичної моделі

Складемо таблицю змінних

Змінна	***Тип***	***Ім'я***	***Призначення***
Двовимірний масив	double	matrix	Проміжні дані
Кількість рядків і стовпців матриці	int	m	Проміжні дані
Останній мінімальний елемент матриці	double	minElement	Вихідні дані
Індекс рядка матриці	int	x	Вихідні дані
Індекс стовпця матриці	int	y	Вихідні дані

Середньоарифметичне значення елементів під побічною діагоналлю	double	arithmetic	Вихідні дані
Знак для порівняння значень	char	sign	Вихідні дані
Лічильник i	int	i	Проміжні дані
Лічильник j	int	j	Проміжні дані
Напрямок обходу по рядках матриці	int	direction	Проміжні дані
Індекс рядка або стовпця матриці	int	index	Проміжні дані
Кількість елементів матриці під побічною діагоналлю	int	counter	Проміжні дані
Сума елементів матриці під побічною діагоналлю	double	sum	Проміжні дані

Складемо таблицю функцій

Назва	Синтаксис	Призначення
Округлення до певної кількості знаків після коми в залежності від другого параметра	Round(a, b)	Округлює до b знаків після коми числа a
Генерація випадкового цілочисельного	Next(a,b)	Генерує ціле число з проміжку [a, b)

значення у певному діапазоні		
Генерація випадкової дробової частини числа	NextDouble()	Генерує правильний десятковий дріб з проміжку (0, 1)
Повернення довжини потрібного виміру багатовимірного масиву	GetLength(a)	Повертає довжину a-го виміру масиву

Отже, ми будемо заповнювати двовимірний масив за допомогою підпрограми **GenerateMatrix,** яка за допомогою двох арифметичних циклів заповнить **matrix** із заданими користувачем вимірами випадково згенерованими дійсними числами.

За допомогою підпрограми **GetMinElement** знайдемо останній мінімальний елемент матриці. Для реалізації обходу змійкою по рядках використаємо for- цикл для проходження по першому виміру матриці, за яким слідує конструкція if … else, яка, в залежності від напрямку direction виконує наступний for- цикл для проходження по другому виміру матриці та if- цикл для пошуку мінімального значення елементів на кожній ітерації.

Далі, використовуючи підпрограму **GetIndex,** робота якої реалізується за допомогою двох for- та одного if- циклу, знаходимо потрібний нам індекс необхідного елемента.

Викликавши підпрограму **FindArithmetic,** робота якої базується на двох for- та одному if- циклах, знайдемо середнє арифметичне значення елементів під побічною діагоналлю матриці.

Розв’язання

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

Крок 1. Визначимо основні дії.

Крок 2. Деталізуємо підпрограму заповнення матриці випадково згенерованими значеннями.

Крок 3. Деталізуємо підпрограму пошуку останнього мінімального елемента матриці.

Крок 4. Деталізуємо підпрограму пошуку індексів останнього мінімального елемента.

Крок 5. Деталізуємо підпрограму пошуку середнього арифметичного значення елементів під побічною діагоналлю матриці.

Крок 6. Деталізуємо визначення знаку для порівняння останнього мінімального елемента матриці та середнього арифметичного значення елементів під побічною діагоналлю матриці.

- **Псевдокод алгоритму**

початок

Введення m

matrix = FillMatrix(m, m)

minElement = GetMinElement(matrix)

x = GetIndex(matrix, minElement)

y = GetIndex (matrix, minElement, false)

arithmetic = FindArithmetic(matrix)

Виведення x, y, minElement

якщо minElement > arithmetic

то

```
                sign = '>'
            інакше
                sign = '<'
        все якщо
        Виведення minElement, sign, arithmetic
кінець

підпрограма GenerateMatrix(int m, int n)
        double[][] matrix = new double [m, n]

        для і від 0 до m повторити

            для j від 0 до n повторити

                matrix[i][j] = Math.Round(rand.Next(-100, 101) +
            rand.NextDouble(), 2)

            все повторити

        все повторити
        return matrix
все підпрограма

підпрограма GetMinElement (double[][] matrix)
        direction = 1
        minValue = matrix[0][0]

        для і від 0 до matrix.GetLength(0)  повторити

            якщо direction > 0

                то
```

```
                для j від 1 до matrix.GetLength(1) + 1 повторити
                    якщо  matrix[i, j-1] <= minValue
                        minValue = matrix[i, j-1]
                    все якщо
                все повторити
            інакше
                для j від matrix.GetLength(1)  до 0 повторити
                    якщо matrix[i, j-1] <= minValue
                        minValue = matrix[i, j-1]
                    все якщо
                все повторити
            все якщо
            direction = direction * -1
        все повторити
        return minValue
все підпрограма

підпрограма  GetIndex(double [][] matrix, double element, bool x = true)
int index = 0
```

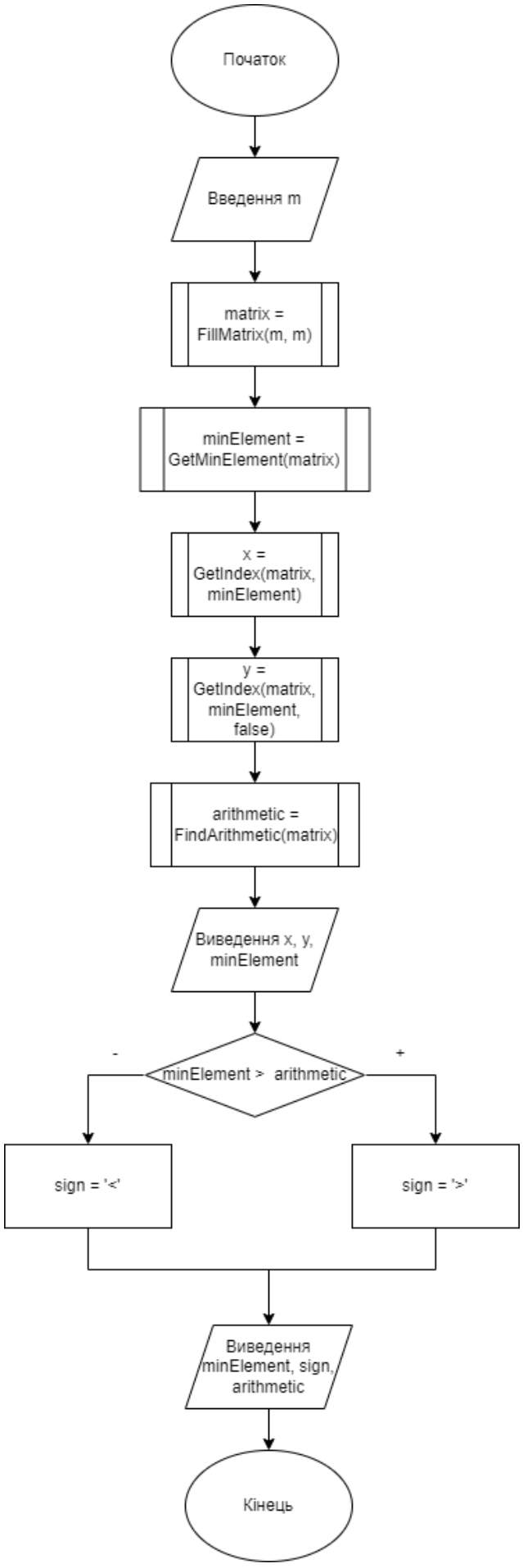
```
        для j від 0 до matrix.GetLength(1)
            для i від 0 до matrix.GetLength(0)
                якщо matrix[i,j] == element
                    якщо x
                        то
                        index = i + 1
                    інакше
                        index = j + 1
                    все якщо
                все якщо
            все повторити
        все повторити
        return index
все підпрограма

підпрограма FindArithmetic(double[][] matrix)
        int counter = 0
        double sum = 0
        double arithmResult
        для j від 0 до matrix.GetLength(1)
            для i від 0 до matrix.GetLength(0)
                якщо i+j >= matrix.GetLength(0)
                    counter = counter + 1
```

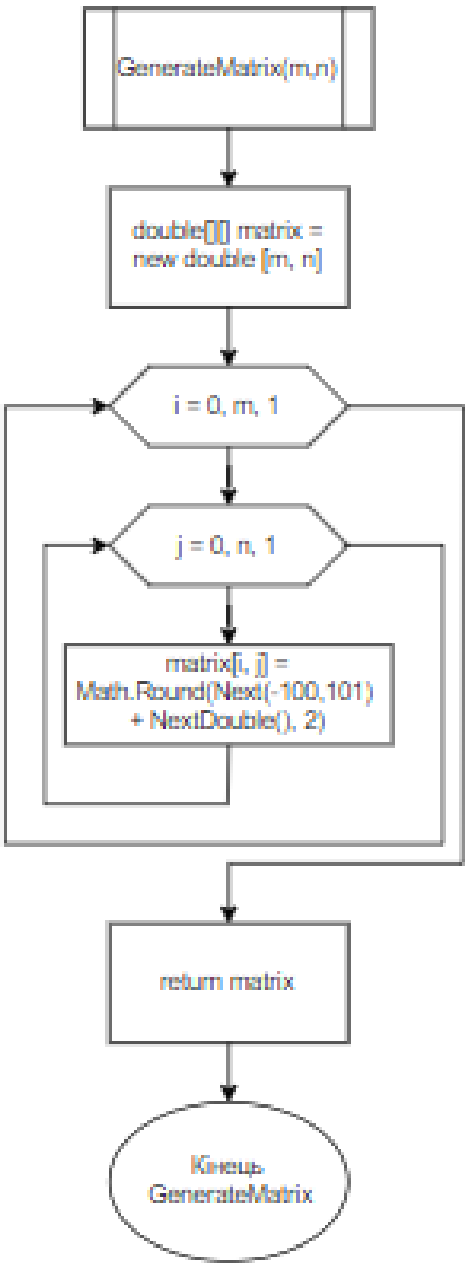
```
                    sum = sum + matrix[i,j]
                все якщо
            все повторити
        все повторити
        arithmResult = Math.Round((sum/counter),2)
        return arithmResult
    все підпрограма
```

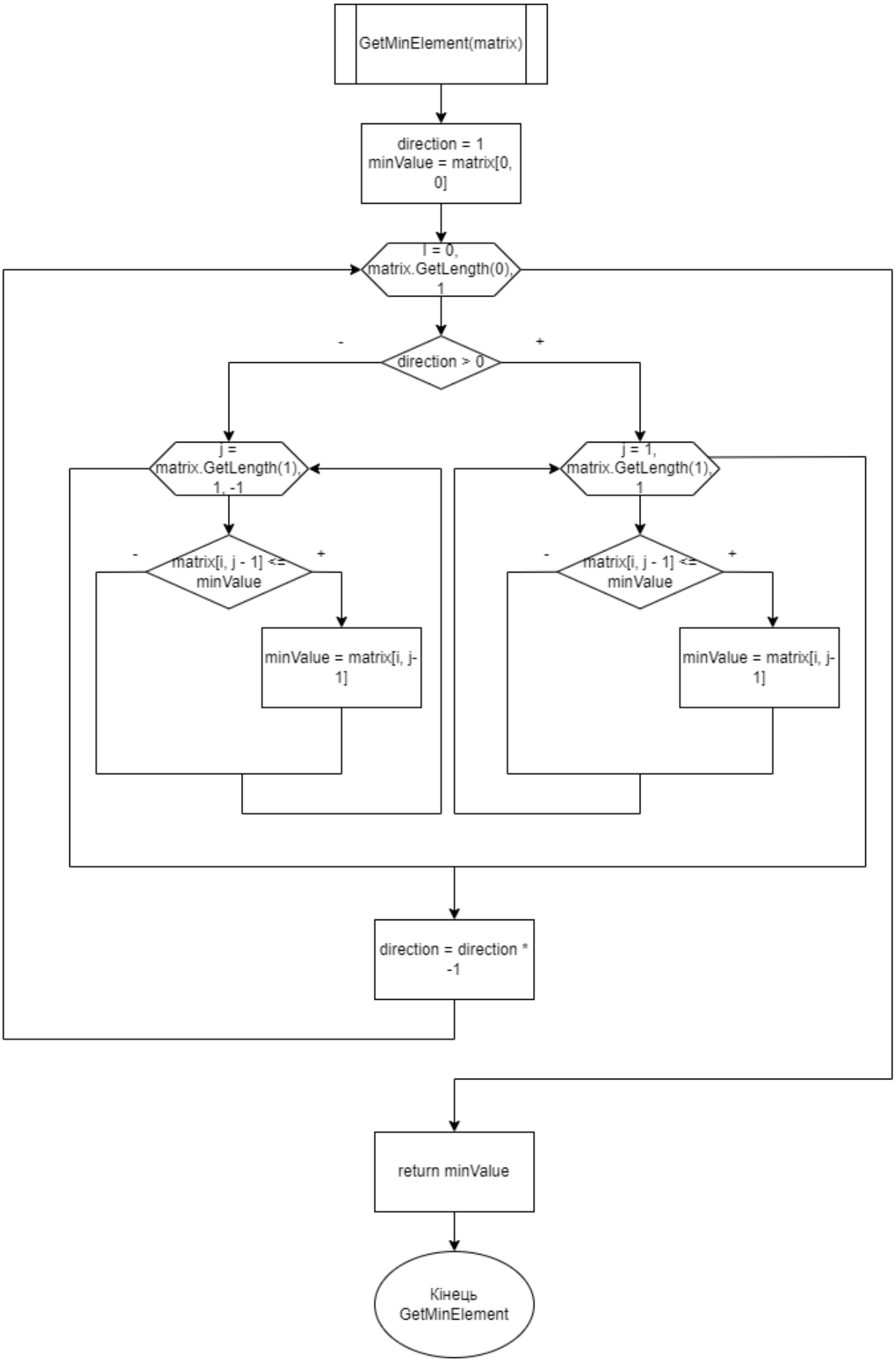
- **Блок-схема**

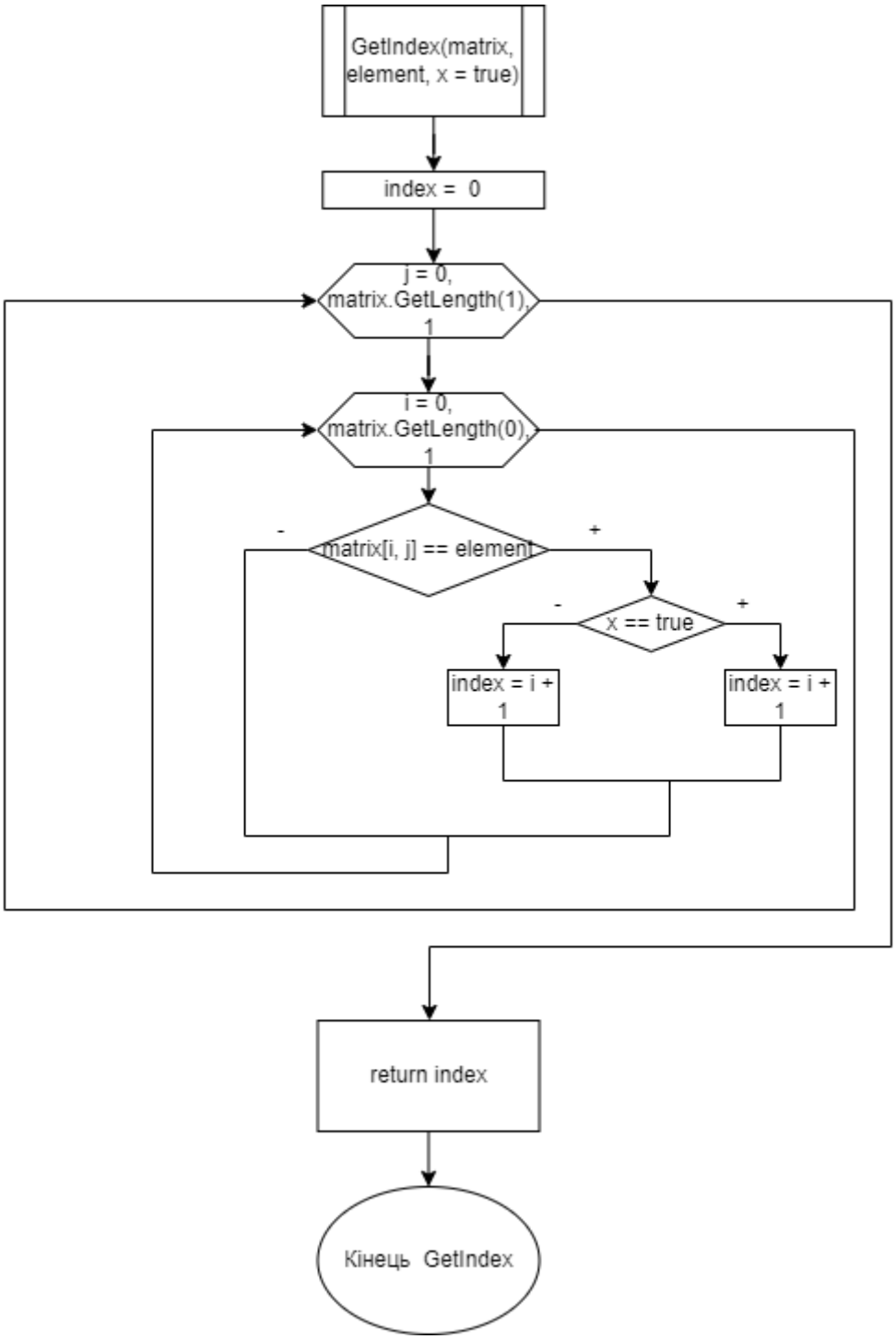
Основна програма

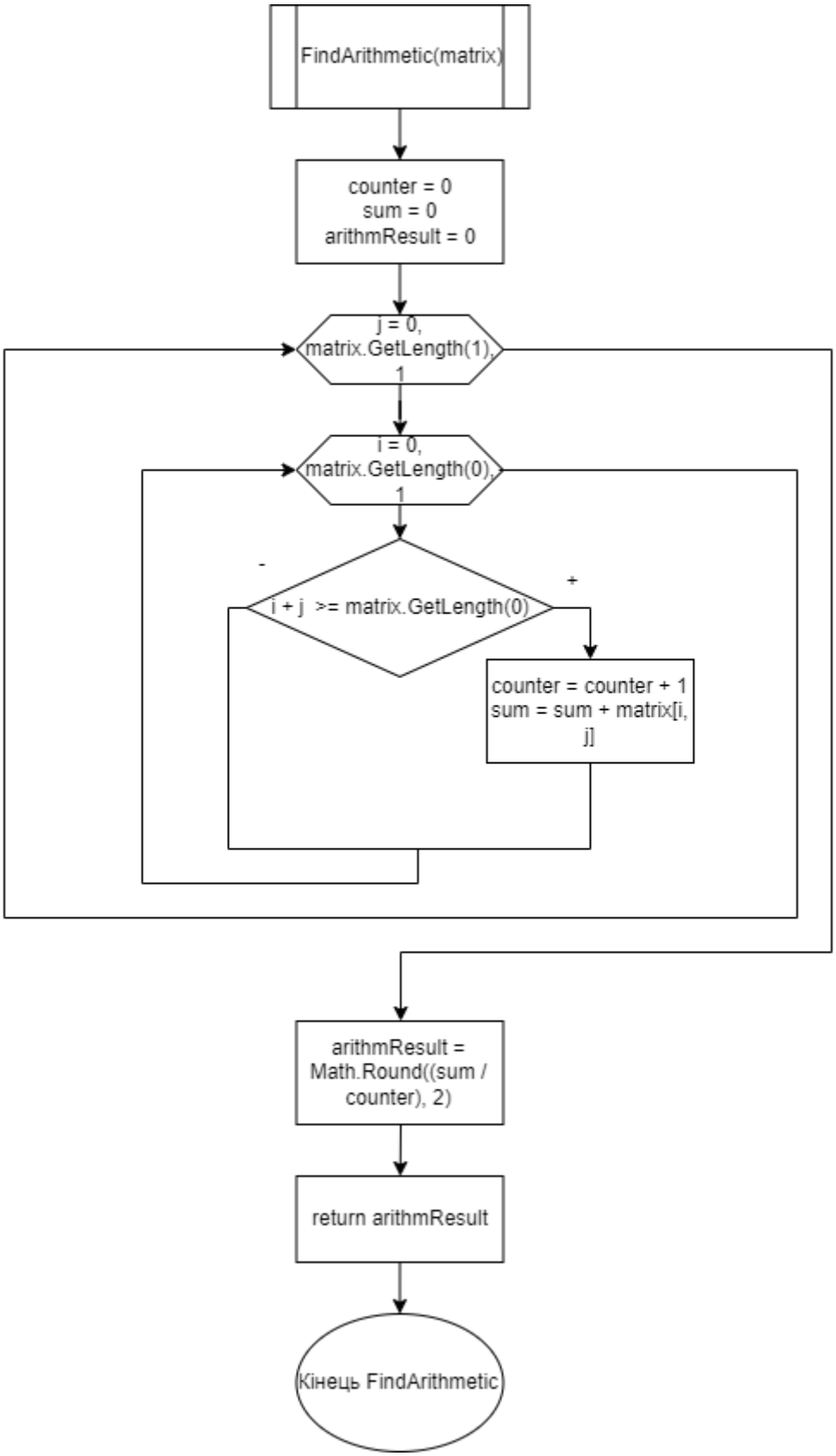

Початок
Введення m
matrix = FillMatrix(m, m)
minElement = GetMinElement(matrix)
x = GetIndex(matrix, minElement)
y = GetIndex(matrix, minElement, false)
arithmetic = FindArithmetic(matrix)
Виведення x, y, minElement
minElement > arithmetic
-
+
sign = '<'
sign = '>'
Виведення minElement, sign, arithmetic
Кінець

Підпрограми




GetMinElement(matrix)
direction = 1
minValue = matrix[0, 0]
i = 0, matrix.GetLength(0), 1
direction > 0
-
+
j = matrix.GetLength(1), 1, -1
j = 1, matrix.GetLength(1), 1
matrix[i, j - 1] <= minValue
-
+
minValue = matrix[i, j-1]
matrix[i, j - 1] <= minValue
-
+
minValue = matrix[i, j-1]
direction = direction * -1
return minValue
Кінець GetMinElement


GetIndex(matrix, element, x = true)
index = 0
j = 0, matrix.GetLength(1), 1
i = 0, matrix.GetLength(0), 1
matrix[i, j] == element
-
+
x == true
-
+
index = i + 1
index = i + 1
return index
Кінець GetIndex


FindArithmetic(matrix)
counter = 0
sum = 0
arithmResult = 0
j = 0,
matrix.GetLength(1),
1
i = 0,
matrix.GetLength(0),
1
-
+
i + j >= matrix.GetLength(0)
counter = counter + 1
sum = sum + matrix[i,
j]
arithmResult =
Math.Round((sum /
counter), 2)
return arithmResult
Кінець FindArithmetic

- **Код програми**

Основна програма

```
using System;

namespace Lab9
{
    Ссылок: 0
    class Program
    {
        Ссылок: 0
        static void Main(string[] args)
        {
            int m;
            double[,] matrix;
            double minElement;
            int x;
            int y;
            double arithmetic;
            char sign;

            m = Convert.ToInt32(Console.ReadLine());
            matrix = GenerateMatrix(m, m);
            minElement = GetMinElement(matrix);
            x = GetIndex(matrix, minElement);
            y = GetIndex(matrix, minElement, false);
            arithmetic = FindArithmetic(matrix);

            Console.WriteLine($" minElement position {x}, {y} \n minElement {minElement} \n arithmeticValue under the secondary diagonal {arithmetic}");
            sign = minElement > arithmetic ? '>' : '<';
            Console.WriteLine($" {minElement} {sign} {arithmetic}");
        }

```

Підпрограми

```
        ссылка: 1
        static double[,] GenerateMatrix(int m, int n)
        {
            double[,] matrix = new double[m, n];
            Random rand = new Random();
            for (int i = 0; i < m; i++)
            {
                for (int j = 0; j < n; j++)
                {
                    matrix[i, j] = Math.Round(rand.Next(-100, 101) + rand.NextDouble(), 2);
                    Console.Write("{0,7}", matrix[i, j] + " ");
                }

                Console.WriteLine();
            }
            return matrix;
        }
```

```
ссылка: 1
static double GetMinElement(double[,] matrix)
{
    int direction = 1;
    double minValue = matrix[0, 0];
    for (int i = 0; i < matrix.GetLength(0); i++)
    {
        if (direction > 0)
        {
            for (int j = 1; j <= matrix.GetLength(1); j++)
            {
                if (matrix[i, j - 1] <= minValue)
                {
                    minValue = matrix[i, j - 1];
                }
            }
        }
        else
        {
            for (int j = matrix.GetLength(1); j >= 1; j--)
            {
                if (matrix[i, j - 1] <= minValue)
                {
                    minValue = matrix[i, j - 1];
                }
            }
        }
        direction *= -1;
    }
    return minValue;
}
```

```
static int GetIndex(double[,] matrix, double element, bool x = true)
{
    int index = 0;
    for (int j = 0; j < matrix.GetLength(1); j++)
    {
        for (int i = 0; i < matrix.GetLength(0); i++)
        {
            if (matrix[i, j] == element)
            {
                index = x ? i + 1 : j + 1;
            }
        }
    }
    return index;
}
```

```
ссылка: 1
static double FindArithmetic(double[,] matrix)
{
    int counter = 0;
    double sum = 0;
    double arithmResult;
    for (int j = 0; j < matrix.GetLength(1); j++)
    {
        for (int i = 0; i < matrix.GetLength(0); i++)
        {
            if (i + j >= matrix.GetLength(0))
            {
                counter++;
                sum += matrix[i, j];
            }
        }
    }
    arithmResult = Math.Round((sum / counter), 2);
    return arithmResult;
}
}
}
```

```
5
 97,62  95,48 -32,47  -56,3  -52,4
 60,02 -16,86  -20,5  47,94   70,7
 15,23  -2,98 -24,42  32,38  74,72
-83,59 -66,04   83,9 -28,13  -19,9
-49,79   56,2  67,21   6,21  44,47
 minElement position 4, 1
 minElement -83,59
 arithmeticValue under the secondary diagonal 38,78
 -83,59 < 38,78
```

```
4
 -1,84 -60,06 -29,87 -15,95
-95,04 -62,71 -93,16 -10,93
 58,22  -2,73  55,97 -30,55
 15,44  -93,5  33,29 -17,42
 minElement position 2, 1
 minElement -95,04
 arithmeticValue under the secondary diagonal -10,52
 -95,04 < -10,52
```

```
2
 29,11  -2,23
 48,77  42,54
 minElement position 1, 2
 minElement -2,23
 arithmeticValue under the secondary diagonal 42,54
 -2,23 < 42,54
```

- **Висновки:**

Виконуючи лабораторну роботу, я дослідила алгоритми обходу масивів та набула практичних навичок використання цих алгоритмів під час складання програмних специфікацій. Зокрема, результатом мого дослідження стала програма для знаходження останнього мінімального елемента матриці обходом по рядках та його індексів, порівняння цього елемента з середнім арифметичним значенням під побічною діагоналлю матриці.