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

і

Звіт

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

«Дослідження алгоритмів обходу массивів»

Варіант 23

Виконав студент ІП-15, Мочалов Дмитро Юрійович
Перевірив Вєчєрковська Анастасія Сергіївна

Київ 2021

Лабораторна робота 9

Дослідження алгоритмів обходу массивів

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

Варіант 23

Задача.

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. Обчислення змінної, що описана в п.1, згідно з варіантом (табл. 1).

23	Задано матрицю дійсних чисел A[n,n], ініціалізувати матрицю обходом по стовбцям. Знайти середньоарифметичне значення P елементів побічної діагоналі матриці. Елементи, розташовані вище головної діагоналі і є меншими за P, обнулити.

Мат. модель

Змінна	Тип	Ім’я	Призначення
Розмірність массиву	цілий	n	Проміжні данні
Початковий массив	дійсний	arr	Проміжні данні, результат
лічильник	цілий	i	Проміжні данні
лічильник	цілий	j	Проміжні данні
Серєднє арифметичне елементів побічної діагоналі	дійсний	P	Проміжні данні
Напрям обходу массиву	цілий	dir	Проміжні данні
Заповнювач массиву	дійсний	a	Проміжні данні
Сєрєднє арифметичне у функції	дійсний	result	Проміжні

Таким чином, математичне модулювання зводится до заповнення массиву по стовпцям у функції sett_arr, у якій за допомогою змінної dir ми визначаємо напрям обходу. Якшо меньше нуля, то вниз по стовпцю, якщо більше нуля – вверх по стовпцю.Заповнюємо масив значеннями змінної a. У функції average визначаємо сєрєднє арифметичне елементів побічної діагоналі массиву arr. У функції zeroing обнуляємо значення елементів массиву які знаходятся вище головної діагоналі і меньші за сєрєднє значення елементів побічної діагоналі.

В розгалуженні “і” позначимо як &&. ++ означає збільшення змінної на 1.

Крок1: визначитись з алгоритмом

Крок2: деталізуємо алгоритм задання значень двовимірного массиву за допомогою підпрограми

Крок2: деталізуємо алгоритм обнуленнямя елементів массиву за допомогою підпрограми.

Псевдокод

Початок

n = 5

set_arr(arr)

P = average(arr)

zeroing(arr,P)

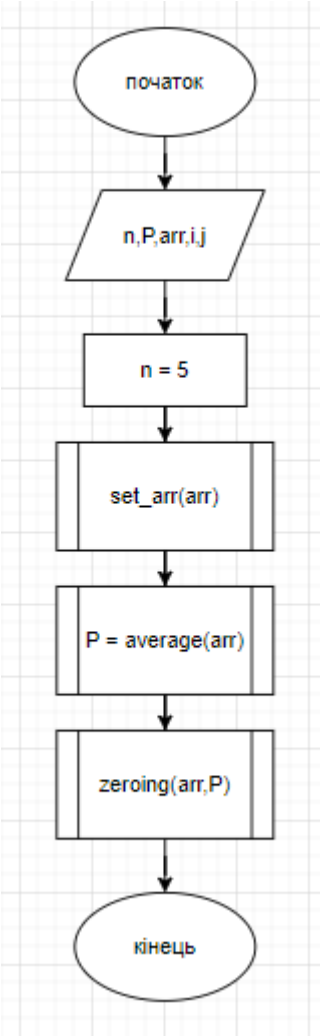
кінець

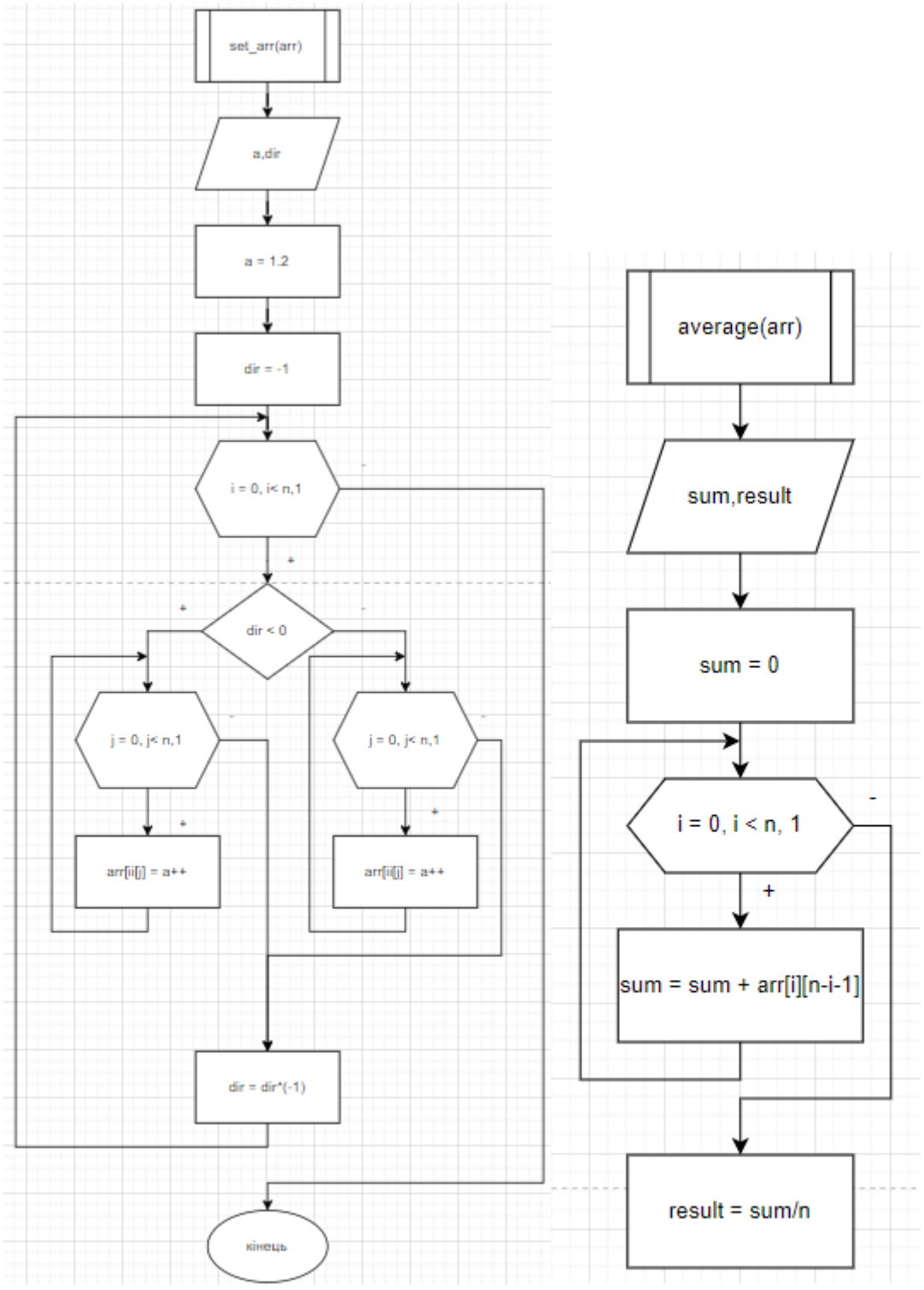
Підпрограми

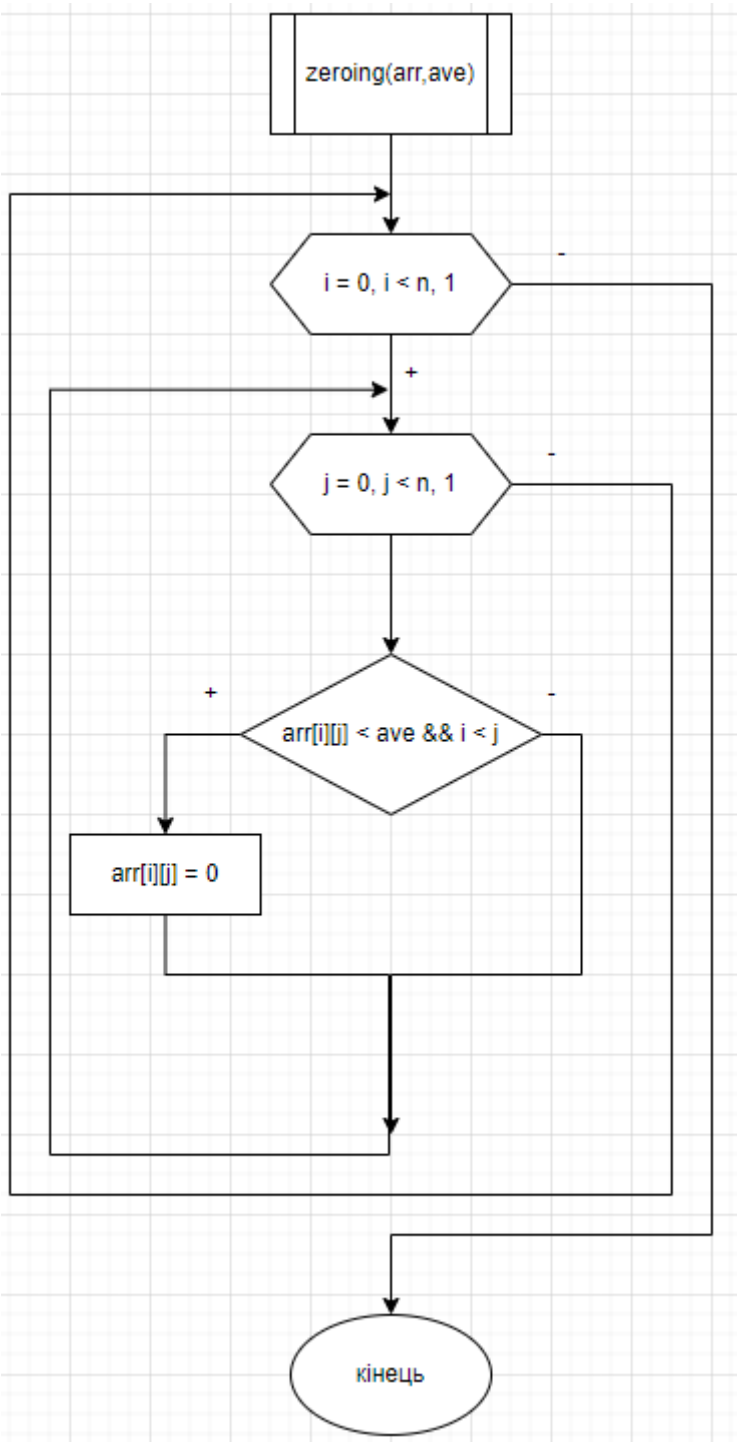
```
set_arr(arr)
початок
    a = 1.2
    dir = -1
    повторити для i від 0 до n
        якщо dir < 0
            повторити для j від 0 до n
                arr[i][j] = a++
            все повторити
            інакше
                повторити для j від 0 до n
                arr[i][j] = a++
                все повторити
            все інакше
        все якщо
        dir = dir * (-1)
    все повторити
кінець


average(arr)
початок
    sum = 0
    повторити для i від 0 до n
            sum = sum + arr[i,n-1-i]
    все повторити
    result = sum/n
    повернути result
кінець
```

```
zeroing(arr,ave)
початок
   повторити для i від 0 до n
      повторити для j від 0 до n
         якщо arr[i,j] < ave && i<j то
            arr[i,j] = 0
         все якщо
      все повторити
   все повторити
кінець
```

Блок-схема


початок
n,P,arr,i,j
n = 5
set_arr(arr)
P = average(arr)
zeroing(arr,P)
кінець


set_arr(arr)
a,dir
a = 1.2
dir = -1
i = 0, i< n,1
+
dir < 0
+
-
j = 0, j< n,1
+
arr[ii[j] = a++
j = 0, j< n,1
+
arr[ii[j] = a++
dir = dir*(-1)
кінець
average(arr)
sum,result
sum = 0
i = 0, i < n, 1
-
+
sum = sum + arr[i][n-i-1]
result = sum/n


zeroing(arr,ave)
i = 0, i < n, 1
-
+
j = 0, j < n, 1
-
+
arr[i][j] < ave && i < j
-
arr[i][j] = 0
кінець

Код

```
using System;

namespace Lab9
{
    internal class Program
    {
        public static void Main(string[] args)
        {
            int n = 5;
            double[,] arr = new double[n, n];
            set_arr(arr);
            Console.WriteLine("Вхідний массив:");
            for (int i = 0; i < n; i++)
            {
                for (int j = 0; j < n; j++)
                {
                    Console.Write(arr[i,j] + "\t");
                }
                Console.WriteLine();
            }

            double P = average(arr);
            Console.WriteLine("\nСереднє арифметичне: " + P);
            zeroing(arr,P);
            Console.WriteLine("\nРезультуючий массив:");
            for (int i = 0; i < n; i++)
            {
                for (int j = 0; j < n; j++)
                {
                    Console.Write(arr[i,j] + "\t");
                }
                Console.WriteLine();
            }

        }
```

```
static void set_arr(double[,] arr)
{
    double a = 1.2;
    int dir = -1;
    for (int i = 0; i < arr.GetLength( dimension: 0); i++)
    {
        if (dir < 0)
        {
            for (int j = 0; j < arr.GetLength( dimension: 1); j++)
            {
                arr[j, i] = a++;
            }
        }
        else
        {
            for (int j = arr.GetLength( dimension: 1) - 1; j >=0 ; j--)
            {
                arr[j, i] = a++;
            }
        }

        dir *= (-1);
    }
}

1 usage
static double average(double[,] arr)
{
    double sum = 0;
    double result;
    int n = arr.GetLength( dimension: 0);
    for (int i = 0; i < n; i++)
    {
        sum += arr[i, n - 1 - i];
    }

    result = sum / n;
    return result;
}
```

```
static void zeroing(double[,] arr, double ave)
{
    for (int i = 0; i < arr.GetLength( dimension: 0); i++)
    {
        for (int j = 0; j < arr.GetLength( dimension: 1); j++)
        {
            if (arr[i, j] < ave && i < j)
            {
                arr[i, j] = 0;
            }
        }
    }
}
```

Тестування

```
Вхідний массив:
1,2     10,2    11,2    20,2    21,2
2,2     9,2     12,2    19,2    22,2
3,2     8,2     13,2    18,2    23,2
4,2     7,2     14,2    17,2    24,2
5,2     6,2     15,2    16,2    25,2

Сєрєднє арифметичне: 13,2

Результуючий массив:
1,2     0       0       20,2    21,2
2,2     9,2     0       19,2    22,2
3,2     8,2     13,2    18,2    23,2
4,2     7,2     14,2    17,2    24,2
5,2     6,2     15,2    16,2    25,2
```

Висновок:Ми дослідили алгоритми обходу масивів, набули практичних навичок використання цих алгоритмів під час складання програмних специфікацій.