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 30

Виконав студент ІП-12 Тарасюк Євгеній Сергійович

Перевірив _____

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.

Задача 30.

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).

30	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 з середньоарифметичним значенням елементів під головною діагоналлю.

Розв'язок.

1. Постановка задачі.

Початкові дані - це дійсні числа, додаткових змінних для розв'язку не потрібно. Додаткові умови: n = m (масив повинен мати головну діагональ). Для обчислення використовуватимемо обхід масиву змійкою. Результатом розв'язку є двовимірний масив та порівняння двох дійсних чисел. Використовуватимемо стандартні логічні та арифметичні операції, цикли з передумовою, функції. Функція randFloat() використовується для знайдення випадкового дійсного числа.

2. Побудова математичної моделі

Таблиця змінних та функцій:

Змінні та функції	Тип	Ім’я	Призначення
Вертикальна позиція мінімального	**Ціле число**	**minVertPos**	**Зберігає перший індекс елемента з найменшим значенням у масиві**
Горизонтальна позиція мінімального	**Ціле число**	**minHorPos**	**Зберігає другий індекс елемента з найменшим значенням у масиві**
Мінімальний елемент	**Дійсне число**	**minElem**	**Найменше значення, знайдене в масиві**
Середнє арифметичне	**Дійсне число**	**average**	**Середнє арифметичне чисел під головною діагоналлю**
n (= m)	**Натуральне число**	**n**	**Розмір масиву**
Двовимірний масив	**Двовимірний масив дійсних чисел**	**array2D**	**Двовимірний масив**
i	**Ціле число**	**i**	**Змінна для перебору значень**
j	**Ціле число**	**j**	**Змінна для перебору значень**
Заповнення масиву	**Функція, що не повертає значення**	**fillArray2D (array2D, n)**	**Заповнює масив випадковими значеннями**
Виведення масиву	**Функція, що не повертає значення**	**print2D (array2D, n)**	**Виводить у консоль елементи масиву**
Середнє арифметичне та мінімум	**Функція, що не повертає значення**	**minAndAvg (array2D, minVertPos, minHorPos, minElem, average, n)**	**Знаходить середнє арифметичне елементів під головною діагоналлю та мінімальне значення**

Кількість елементів під діагоналлю	Ціле число	underDiagCount	Лічильник елементів під головною діагоналлю
Випадкове число	Функція, що повертає дійсне число	randFloat(a, b)	Генерує випадкове число на проміжку [a; b]

3. Псевдокод алгоритму

Початок

Введення *n*

minVertPos = 0

minHorPos = 0

minElem = 50

average = 0

fillArray2D(*array2D, n****)***

print2D(*array2D, n****)***

minAndAvg(*array2D, minVertPos, minHorPos, minElem, average, n****)***

Якщо *minElem < average*

Виведення "minimal element < average"

Інакше якщо *minElem == average*

Виведення "minimal element = average"

Інакше

Виведення "minimal element > average"

Все якщо

Кінець.

Функція *fillArray2D(**array2D, n****)***

Початок

Повторити для *i* **на проміжку** *[0; n)*

Повторити для *j* **на проміжку** *[0; n)*

array2D[i][j] **= randFloat**(-50, 50)

Все повторити

Все повторити

Повернення

Кінець

Функція *print2D(**array2D****)***

Початок

Повторити для *i* **на проміжку** *[0; n)*

Повторити для *j* **на проміжку** *[0; n)*

Виведення *array2D[i][j]*

Все повторити

Перейти на наступний рядок

Все повторити

Повернення

Кінець

Функція *minAndAvg(**array2D, minVertPos, minHorPos, minElem, average, n****)***

Початок

i = 0

j = 0

underDiagCount = 0

Повторити поки *i < n*

Повторити поки *j < n*

Якщо *array2D[i][j] <= minElem*

minElem = array2D[i][j]

minVertPos = i

minHorPos = j

Все якщо

Якщо *j < i*

average += array2D[i][j]

underDiagCount += 1

Все якщо

j += 1

Все повторити

i += 1

j -= 1

Якщо *i < n*

Повторити поки *j > -1*

Якщо *array2D[i][j] <= minElem*

minElem = array2D[i][j]

minVertPos = i

minHorPos = j

Все якщо

Якщо *j < i*

average += array2D[i][j]

underDiagCount += 1

Все якщо

j-= 1

Все повторити

j += 1

i += 1

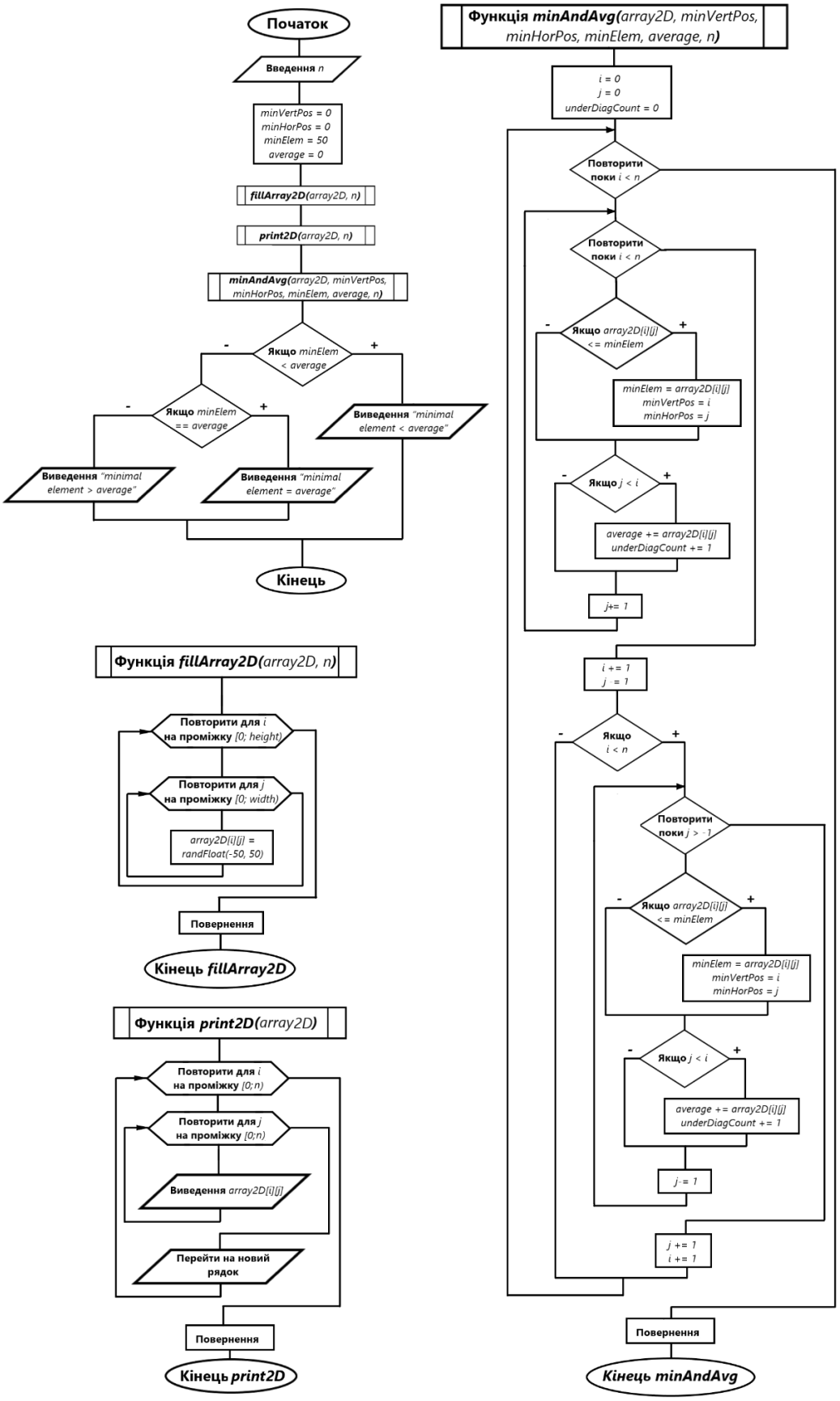
Все якщо

Все повторити

Повернення

Кінець

4. Блок-схема алгоритму



5. Код програми (C++)

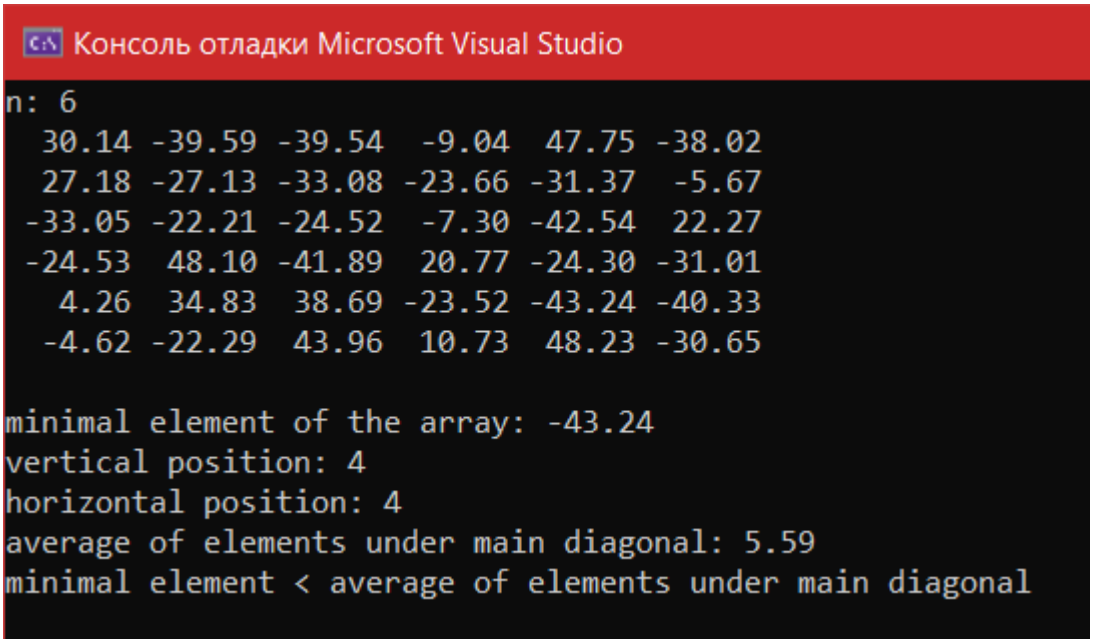
```
#include <iostream>
#include <random>
#define range 100
#define halfRange 50
using namespace std;

void fillArray2D(float**, int);
void minAndAvg(float**, int&, int&, float&, float&, int);
void print2D(float**, int);

int main()
{
   int n;
   int minVertPos = 0;
   int minHorPos = 0;
   float minElem = 50;
   float average = 0;
   float **array2D;
   cout << "n: ";
   cin >> n;
   array2D = new float* [n];
   for (int i = 0; i < n; i++)
   {
      array2D[i] = new float [n];
   }
   fillArray2D(array2D, n);
   print2D(array2D, n);
   minAndAvg(array2D, minVertPos, minHorPos, minElem, average, n);
     printf("minimal element of the array: %.2f \nvertical position: %d \nhorizontal position: %d \n", minElem,
minVertPos, minHorPos);
   printf("average of elements under main diagonal: %.2f \n", average);
   if (minElem < average)
   {
      cout << "minimal element < average of elements under main diagonal\n";
   }
   else if (minElem == average)
   {
      cout << "minimal element = average of elements under main diagonal\n";
   }
   else
   {
      cout << "minimal element > average of elements under main diagonal\n";
   }
   for (int i = 0; i < n; i++)
   {
      delete[] array2D[i];
   }
   delete[] array2D;
}

void fillArray2D(float** array2D, int n)
{
   srand(time(NULL));
   for (int i = 0; i < n; i++)
   {
```

```
        for (int j = 0; j < n; j++)
        {
            array2D[i][j] = -halfRange + float(rand()) / float(RAND_MAX) * range;
        }
    }
}

void minAndAvg(float** array2D, int& minVertPos, int& minHorPos, float& minElem, float& average, int n)
{
    int i = 0;
    int j = 0;
    int underDiagCount = 0;
    average = 0;
    minElem = array2D[0][0];
    while (i < n)
    {
        while (j < n)
        {
            if (array2D[i][j] <= minElem)
            {
                minElem = array2D[i][j];
                minVertPos = i;
                minHorPos = j;
            }
            if (j < i)
            {
                average += array2D[i][j];
                underDiagCount++;
            }
            j++;
        }
        i++;
        j--;
        if (i < n)
        {
            while (j > -1)
            {
                if (array2D[i][j] <= minElem)
                {
                    minElem = array2D[i][j];
                    minVertPos = i;
                    minHorPos = j;
                }
                if (j < i)
                {
                    average += array2D[i][j];
                    underDiagCount++;
                }
                j--;
            }
            j++;
            i++;
        }
    }
    average /= underDiagCount;
}

void print2D(float** array2D, int n)
```

```
{
   for (int i = 0; i < n; i++)
   {
      for (int j = 0; j < n; j++)
      {
         printf("%7.2f", array2D[i][j]);
      }
      cout << endl;
   }
   cout << endl;
}
```

6. Випробування коду

Результати випробування:



7. Висновки

Було досліджено алгоритми обходу масивів, набуто практичних навичок використання цих алгоритмів під час складання програми для пошуку мінімального елемента та середнього значення окремої групи елементів.