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 № 8 з дисципліни

«Алгоритми та структури даних

1. Основи алгоритмізації»

«Дослідження алгоритмів пошуку та сортування»

Варіант 22

Виконав студент ІП-14 Нікулін Павло Юрійович
(шифр, прізвище, ім'я, по батькові)

Перевірив Мартинова Оксана Петрівна
(прізвище, ім'я, по батькові)

Київ 2021

Лабораторна робота 8

Дослідження алгоритмів пошуку та сортування

Мета: дослідити алгоритми пошуку та сортування, 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. Створення нової змінної індексованого типу (одновимірний масив) та її ініціювання значеннями, що обчислюються згідно з варіантом

Знайти: Створити дійсний двомірний масив *5 x 7*. Обчислення значень елементів одновимірного масиву із середнього арифметичного додатних значень елементів стовпців двовимірного масиву. Відсортувати обміном за спаданням.

Розв'язання

1. **Постановка задачі**: заповнимо матрицю *matrix* випадковими дійсними числами, створимо масив *average* та запонимо його середніми арифметичними додатніх значень стовпців *matrix*. Відсортуємо матрицю методом обміну («бульбашки»). Виведемо матрицю.
2. Побудова **математичної моделі**. Складемо таблицю імен змінних.

Змінна	Тип	Ім'я	Призначення
Кількість рядків	Ціле	N	Початкове дане
Кількість стовпціві	Ціле	M	Початкове дане
Матриця для сортування	Дійсний	MATRIX	Початкове/Результат
Масив середніх арифметичних	Дійсний	AVERAGE	Проміжне дане
Задання матриці	Універсальний	INPUT	Функція
Середні арифметичні стовпців	Універсальний	AVERAGE_COL	Функція
Сортування матриці	Універсальний	SORT	Функція
Виведення матриці	Універсальний	OUTPUT	Функція

Формальні масиви	Дійсний	ARR/AVE	Проміжне дане
Тимчасове значення	Дійсний	TEMP	Проміжне дане
Лічильник	Ціле	COUNT	Проміжне дане
Формальні змінні циклів	Ціле	K, I, J	Проміжне дане

Крок 1. Визначимо основні дії.

Крок 2. Заповнити матрицю.

Крок 3. Знайти середні арифметичні.

Крок 4. Відсортувати матрицю.

Крок 5. Вивести матрицю.

.

Псевдокод

крок 1

початок

Заповнити матрицю

Знайти середні арифметичні

Відсортувати матрицю

Вивести матрицю

кінець

крок 2

початок

поки i < n **повторити**

matrix[i][j] = (rand() % 200 - 100) / 10

Вивести matrix[i][j]

i++

все повторити

Знайти середні арифметичні

Відсортувати матрицю

Вивести матрицю

кінець

крок 3

початок

поки i < n **повторити**

matrix[i][j] = (rand() % 200 - 100) / 10

Вивести matrix[i][j]

i++

все повторити

поки j < m **повторити**

average[j] = 0

поки i < n **повторити**

якщо arr[i][j] > 0 **то**

ave[j] += arr[i][j]

count++

все якщо

i++

все повторити

j++

все повторити

Відсортувати матрицю

Вивести матрицю

кінець

крок 4

початок

поки i < n **повторити**

matrix[i][j] = (rand() % 200 - 100) / 10

Вивести matrix[i][j]

i++

все повторити

поки j < m **повторити**

average[j] = 0

поки i < n **повторити**

якщо matrix [i][j] > 0 **то**

average [j] += matrix [i][j]

count++

все якщо

i++

все повторити

average [j] /= count

j++

все повторити

поки k = 1; k < m **повторити**

поки j < m - k **повторити**

якщо average [j + 1] > average [j] **то**

поки i < n **повторити**

temp = matrix [i][j + 1]
matrix [i][j + 1] = matrix [i][j]
matrix [i][j] = temp

i++

все повторити

temp = average [j + 1]
average [j + 1] = average [j]
average [j] = temp

все якщо

j++

все повторити

k++

все повторити

Вивести матрицю

кінець

крок 5

початок

поки i < n **повторити**

matrix[i][j] = (rand() % 200 - 100) / 10

Вивести matrix[i][j]

i++

все повторити

поки j < m **повторити**

average[j] = 0

поки i < n **повторити**

```
            якщо matrix [i][j] > 0 то

                average [j] += matrix [i][j]

                count++

            все якщо

            i++

        все повторити

        average [j] /= count

        j++

    все повторити

    поки k = 1; k < m повторити

        поки j < m - k повторити

            якщо average [j + 1] > average [j] то

                поки i < n повторити

                    temp = matrix [i][j + 1]
                    matrix [i][j + 1] = matrix [i][j]
                    matrix [i][j] = temp

                    i++

                все повторити

                temp = average [j + 1]
                average [j + 1] = average [j]
                average [j] = temp

            все якщо

            j++

        все повторити

        k++

    все повторити
```

поки повторити

поки повторити

Вивести matrix[i][j]

j++

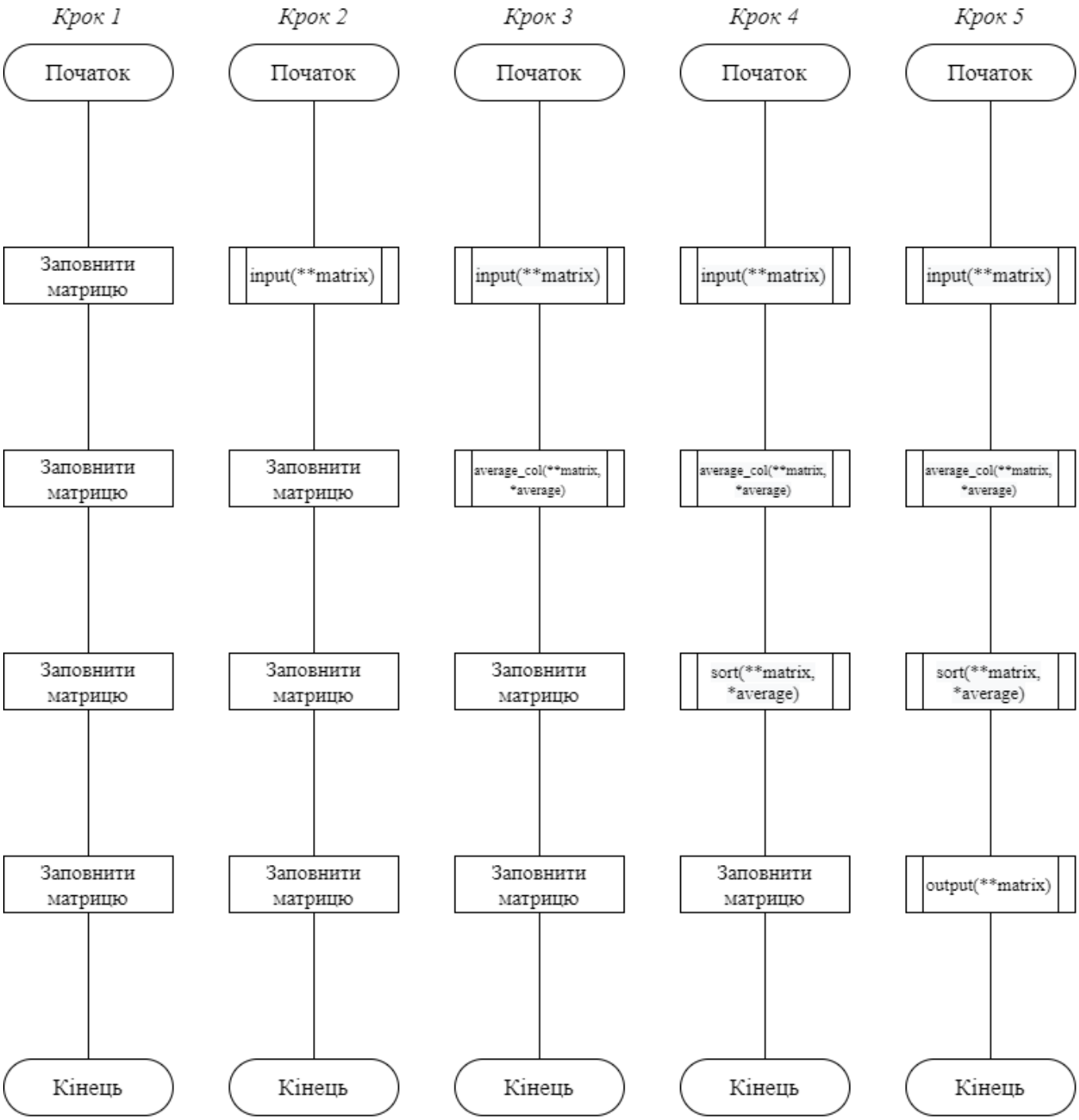
все повторити

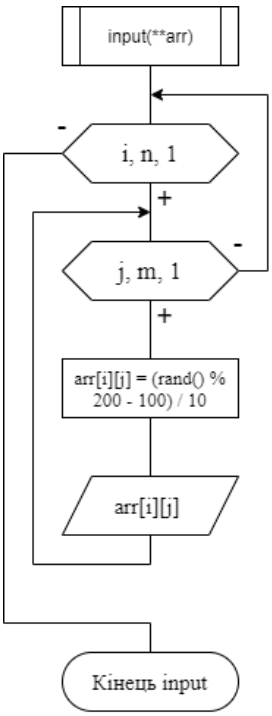
i++

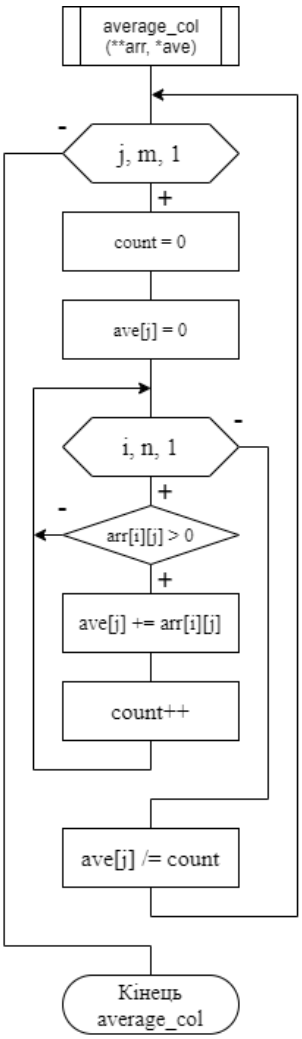
все повторити

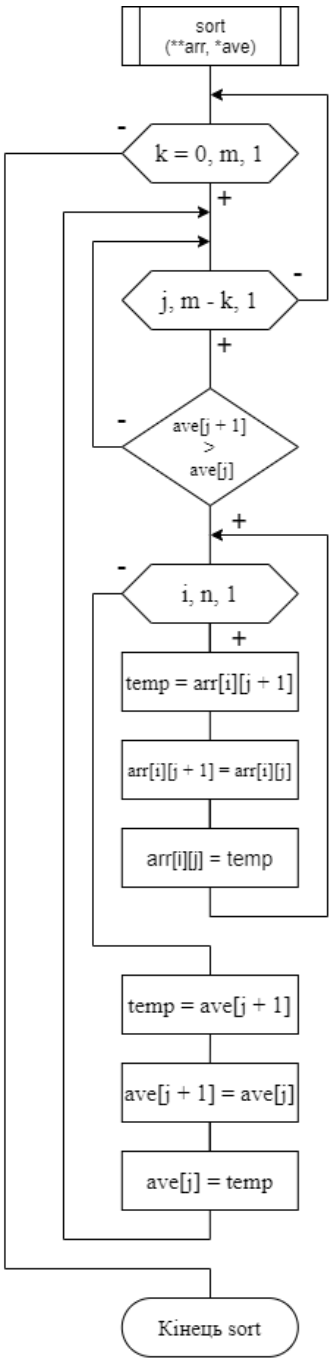
кінець

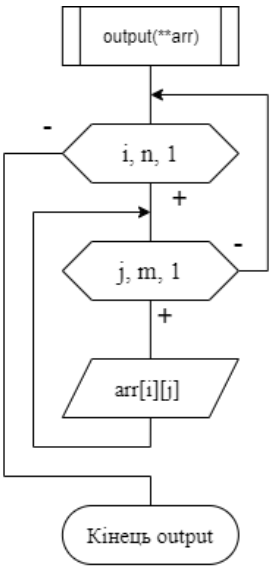
Блок-схема


Крок 1
Початок
Заповнити матрицю
Заповнити матрицю
Заповнити матрицю
Заповнити матрицю
Кінець
Крок 2
Початок
input(**matrix)
Заповнити матрицю
Заповнити матрицю
Заповнити матрицю
Кінець
Крок 3
Початок
input(**matrix)
average_col(**matrix, *average)
Заповнити матрицю
Заповнити матрицю
Кінець
Крок 4
Початок
input(**matrix)
average_col(**matrix, *average)
sort(**matrix, *average)
Заповнити матрицю
Кінець
Крок 5
Початок
input(**matrix)
average_col(**matrix, *average)
sort(**matrix, *average)
output(**matrix)
Кінець


input(**arr)
i, n, 1
j, m, 1
arr[i][j] = (rand() % 200 - 100) / 10
arr[i][j]
Кінець input


average_col (**arr, *ave)
j, m, 1
count = 0
ave[j] = 0
i, n, 1
arr[i][j] > 0
ave[j] += arr[i][j]
count++
ave[j] /= count
Кінець average_col


sort (**arr, *ave)
k = 0, m, 1
j, m - k, 1
ave[j + 1] > ave[j]
i, n, 1
temp = arr[i][j + 1]
arr[i][j + 1] = arr[i][j]
arr[i][j] = temp
temp = ave[j + 1]
ave[j + 1] = ave[j]
ave[j] = temp
Кінець sort


output(**arr)
i, n, 1
j, m, 1
arr[i][j]
Кінець output

Код програми

```cpp
#include <iostream>
#include "windows.h"

using namespace std;

/*Протипи функції*/
void input(float**);
void average_col(float**, float*);
void sort(float**, float*);
void output(float**);

const int n = 5, m = 7;

int main()
{
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    /*Ініціалізація матриці*/
    float** matrix = new float* [n];
    for (int i = 0; i < n; i++)
    {
        matrix[i] = new float[m];
    }

    float* average = new float[m];

    /*Задаємо матрицю*/
    input(matrix);

    /*Середнє арифметичне додатних елементів стовпця*/
    average_col(matrix, average);

    /*Сортування обміном*/
    sort(matrix, average);

    /*Виведення матриці*/
    output(matrix);
}

void input(float** arr)
{
    cout << "Початкова матриця:\n";

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            arr[i][j] = (float)(rand() % 200 - 100) / 10;
            cout << arr[i][j] << "\t";
        }
        cout << "\n";
    }
}

```

```
56  void average_col(float** arr, float* ave)
57  {
58      for (int j = 0; j < m; j++)
59      {
60          int count = 0;
61          ave[j] = 0;
62
63          for (int i = 0; i < n; i++)
64          {
65              if (arr[i][j] > 0)
66              {
67                  ave[j] += arr[i][j];
68                  count++;
69              }
70          }
71
72          ave[j] /= count;
73          cout << "\nСередня арифметичне стовпця №" << j << ": " << ave[j];
74      }
75  }
76
77  void sort(float** arr, float* ave)
78  {
79      float temp;
80
81      for (int k = 1; k < m; k++)
82      {
83          for (int j = 0; j < m - k; j++)
84          {
85              if (ave[j + 1] > ave[j])
86              {
87                  for (int i = 0; i < n; i++)
88                  {
89                      temp = arr[i][j + 1];
90                      arr[i][j + 1] = arr[i][j];
91                      arr[i][j] = temp;
92                  }
93
94                  temp = ave[j + 1];
95                  ave[j + 1] = ave[j];
96                  ave[j] = temp;
97              }
98          }
99      }
100 }
101
102 void output(float** arr)
103 {
104     cout << "\n\nВідсортована матриця:\n";
105
106     for (int i = 0; i < n; i++)
107     {
108         for (int j = 0; j < m; j++)
109         {
110             cout << arr[i][j] << "\t";
111         }
112         cout << "\n";
113     }
114 }
```

Тестування програми

```
Початкова матриця:
-5.9    -3.3    3.4     0       6.9     2.4     -2.2
5.8     6.2     -3.6    0.5     4.5     -1.9    -7.3
6.1     -0.9    9.5     4.2     -7.3    -6.4    9.1
-9.6    0.2     5.3     -0.8    8.2     -7.9    1.6
1.8     -0.5    -5.3    2.6     7.1     3.8     -3.1

Середнє арифметичне стовпця №0: 4.56667
Середнє арифметичне стовпця №1: 3.2
Середнє арифметичне стовпця №2: 6.06667
Середнє арифметичне стовпця №3: 2.43333
Середнє арифметичне стовпця №4: 6.675
Середнє арифметичне стовпця №5: 3.1
Середнє арифметичне стовпця №6: 5.35

Відсортована матриця:
6.9     3.4     -2.2    -5.9    -3.3    2.4     0
4.5     -3.6    -7.3    5.8     6.2     -1.9    0.5
-7.3    9.5     9.1     6.1     -0.9    -6.4    4.2
8.2     5.3     1.6     -9.6    0.2     -7.9    -0.8
7.1     -5.3    -3.1    1.8     -0.5    3.8     2.6
```

Висновок

Під час виконання лабораторної роботи було досліджено алгоритми пошуку та сортування, набуто практичних навичок використання цих алгоритмів під час складання програмних специфікацій. У роботі використано 5 функцій: *main, input, average_col, sort* та *output*. У функції *input* ми задаємо випадкові дійсні значення матриці, у *average_col* ми шукаємо середнє арифметичне додатніх значень для кожного стовпця. У функції *sort* ми сортуємо матрицю алгоритмом обміну («бульбашки»), а саме розташовуємо стовпці у порядку спадання їх середніх арифметичних. У функції *output* ми виводимо відсортовану матрицю на екран. Було використано 6 звичаних і 3 вкладених цикли *for*, де у ролі інкремент виступали *i, j* та *k*. Код написаний на мові програмування C++. Результат роботи є вірним, виконано згідно з алгоритмом.