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 10

Виконав студент ІП-11, Друзенко Олександра Юріївна
(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.

**Варіант 10**

| | | | |
|---|---|---|---|
| **10** | 4 x 8 | Дійсний | Із мінімальних значень елементів стовпців двовимірного масиву. Відсортувати методом Шела за спаданням. |

## 1.Постановка задачі

Дано матрицю 4x8 заповнену дійсними числами. Створити масив заповнений мінімальними значеннями стовпців матриці. Відсортувати створений масив методом Шела за спаданням. Матрицю та масив вивести в консоль.

## 2.Математична модель

| Змінна | Тип | Ім’я | Призначення |
|---|---|---|---|
| Кількість стовпців | int | n | початкове дане |
| Кількість рядків | int | m | початкове дане |
| Матриця | matrix of float | matrixA | проміжне дане |
| Масив | array of float | arrayB | результат |
| Лічильник і для рядків | int | i | проміжне дане |
| Лічильник j для стовпців | int | j | проміжне дане |
| Змінна для збереження тимчасових даних | float | x | проміжне дане |
| Крок сортування | int | d | проміжне дане |
| Перевірка умови сортування | bool | k | проміжне дане |
| Функція виведення матриці | function | outputMatrix | функція виведення |
| Функція виведення масиву | function | outputArray | функція виведення |
| Функція ініціалізації масиву | function | initArray | функція ініціалізації |
| Функція ініціалізації матриці | function | initMatrix | функція ініціалізації |
| Функція сортування методом Шела | function | ShellSort | функція сортування |

Функція print() – виведення на екран

Функція rand(a,b) – генерація рандомних чисел від a до b

**Кроки алгоритму**

Крок 1. Визначимо основні дії

Крок 2. Деталізуємо підпрограму ініціалізації матриці

Крок 3. Деталізуємо підпрограму ініціалізації масиву

Крок 4. Деталізуємо підпрограму сортування масиву

Крок 5. Деталізуємо підпрограму виведення матриці в консоль

Крок 6. Деталізуємо підпрограму виведення масиву в консоль

**3.Псевдокод**

*Крок 1. Основна програма*

**Початок**

m=4; n=8;
initMatrix(matrixA);
outputMatrix(matrixA);
initArray(arrayB, matrixA);
ShellSort(arrayB);
outputArray(arrayB);

**Кінець**

*Крок 2. Підпрограма ініціалізації матриці*

**Початок**

**Функція** initMatrix (*matrixA*)**:**
**Повторити для *i* від 0 до m**
**Повторити для j від 0 до n**
matrixA[i][j] = rand(-15,15);
**Все повторити**
**Все повторити**

**Кінець**

*Крок 3. Підпрограма ініціалізації масиву*

**Початок**

**Функція** initArray (*arrayB, matrixA*)**:**
**Повторити для *j* від 0 до m**
x = matrixA[0][j];
**Повторити для *i* від 0 до n**
**Якщо** *x > matrix[i][j]* **то**
x = matrix[i][j];
**все якщо**
**Все повторити**
arrayB[j] = x;
**Все повторити**

**Кінець**

*Крок 4. Підпрограма сортування масиву*

```
Початок
    Функція ShellSort (arrayA):
      d = n;
      повторити
          d = d/2;
          k = 1;
          поки k
              k=0;
              Повторити для i від 0 до n-d
                  Якщо arrayA[i] < arr[i + d] то
                      x = arrayA [i];
                      arrayA [i] = arrayA [i + d];
                      arrayA [i + d] = x;
                      k = 1;
                  все якщо
              все повторити
      повторити
      все повторити
      поки d>0
      все повторити
Кінець
```

*Крок 5. Підпрограма виведення матриці в консоль*

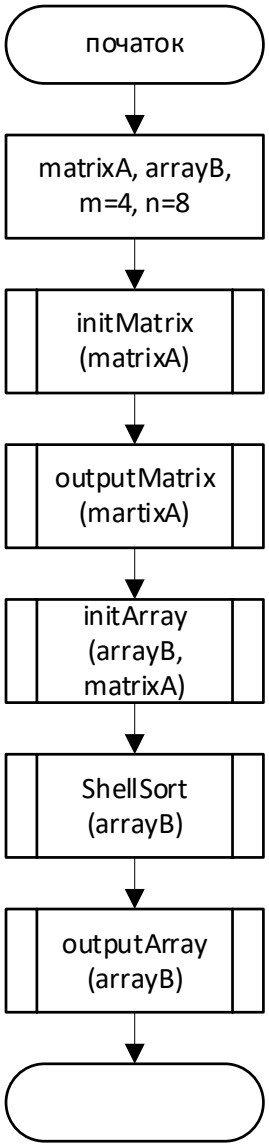
```
Початок
    Функція outputMatrix(matrixA):
          Повторити для i від 0 до 10
              Повторити для j від 0 до m
                  print(matrixA[i][j])
              Все повторити
          Все повторити
Кінець
```

*Крок 6. Підпрограма виведення масиву в консоль*

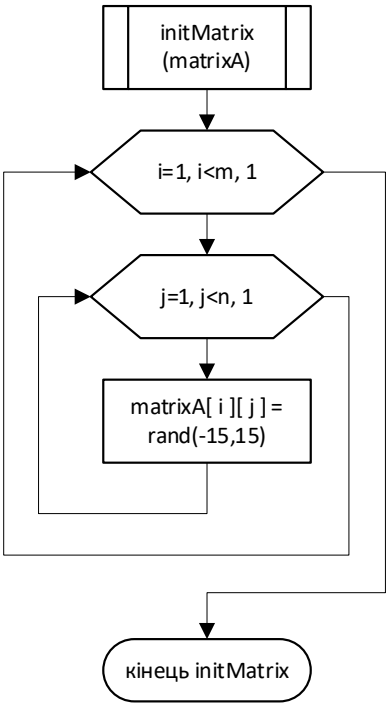
```
Початок
    Функція outputArray(arrayB):
          Повторити для i від 0 до 10
              print(arrayB[i])
          Все повторити
Кінець
```

## 4.Блок-схема

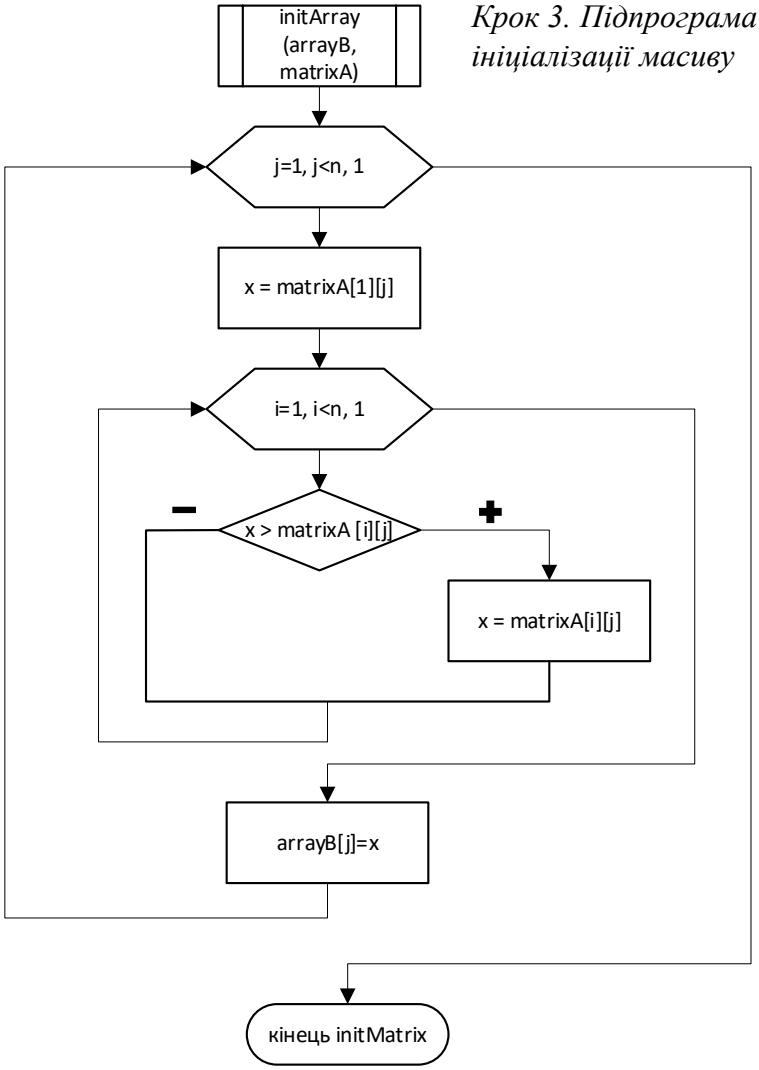
*Крок 1*

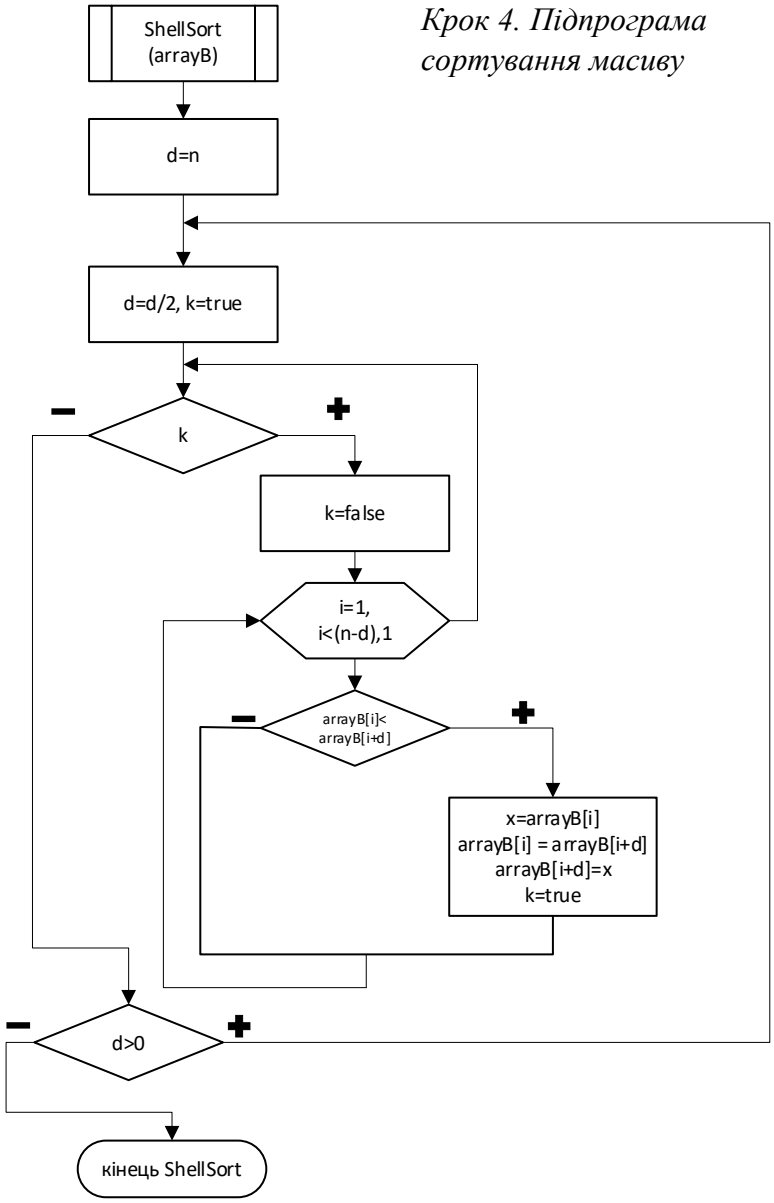



*Крок 2. Підпрограма ініціалізації матриці*



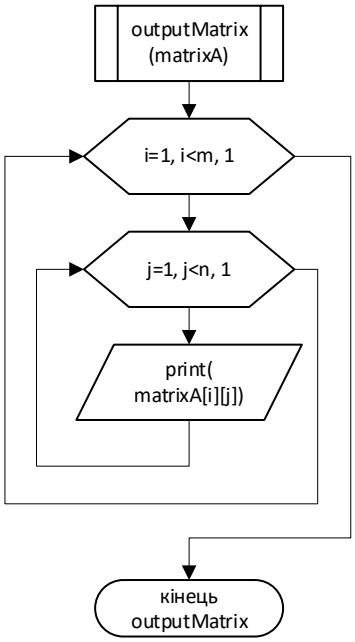

*Крок 3. Підпрограма ініціалізації масиву*



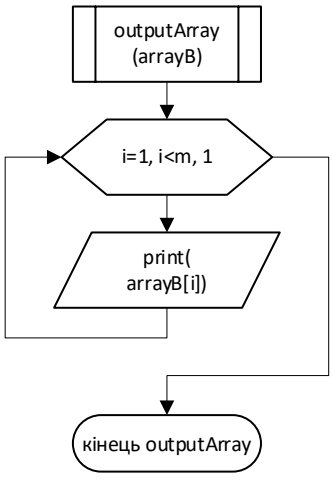



*Крок 4. Підпрограма сортування масиву*

*Крок 5. Підпрограма виведення матриці в консоль*




*Крок 6. Підпрограма виведення масиву в консоль*



## 5. Код програми (C++)

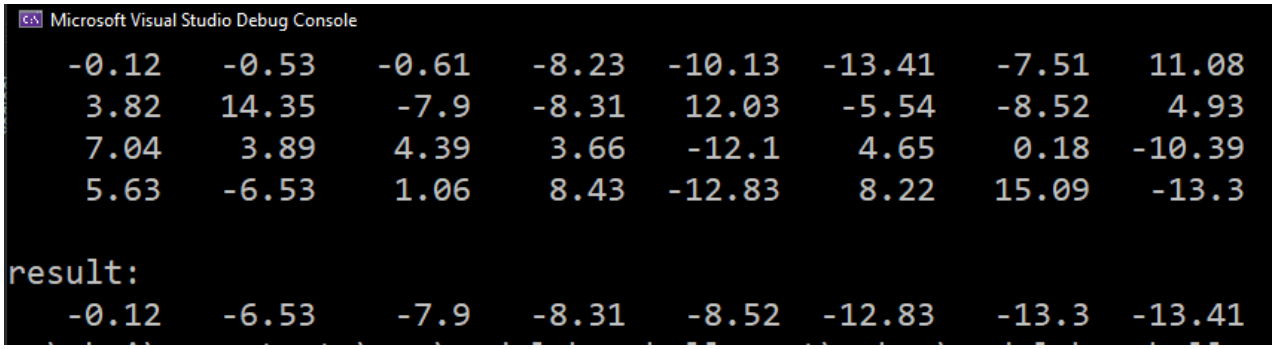
```
#include <iostream>
#include <ctime>
#include <stdlib.h>
#include <iomanip>
using namespace std;

int const m=4, n=8;
typedef float Matrix[m][n];
typedef float Array[n];
Matrix matrixA;
Array arrayB;
void initMatrix(Matrix);
void outputMatrix(Matrix);
void initArray(Array,Matrix);
void ShellSort(Array);
void outputArray(Array);

int main()
{
    srand(time(NULL));
    initMatrix(matrixA);
    outputMatrix(matrixA);
    initArray(arrayB, matrixA);
    ShellSort(arrayB);
    outputArray(arrayB);
}
void initMatrix(Matrix matr) {
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            matr[i][j] = rand() % 31 - 15 + (rand()%100/(float)100);
        }
    }
}
void initArray(Array arr, Matrix matr) {
    float x;
    for (int j = 0; j < n; j++) {
        x = matr[0][j];
        for (int i = 0; i < m; i++) {
            if (x > matr[i][j]) x = matr[i][j];
        }
        arr[j] = x;
    }
}
void outputMatrix(Matrix matr) {
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            cout<<setw(8)<<matr[i][j];
        }
        cout << endl;
    }
    cout << endl;
}
void outputArray(Array arr) {
    cout << "result:" << endl;
    for (int i = 0; i < n; i++) {
        cout<< setw(8)<<arr[i];
    }
}
void ShellSort(Array arr) {
    int d=n; float x; bool k=1;
        do {
```

```
            d /= 2;
            k = 1;
            while (k) {
                k = 0;
                for (int i = 0; i < (n - d); i++) {
                    if (arr[i] < arr[i + d]) {
                        x = arr[i];
                        arr[i] = arr[i + d];
                        arr[i + d] = x;
                        k = 1;
                    }
                }
            }
            } while (d>0);
}
```

## 6. Тестування програми




| крок | дія |
|---|---|
| | початок |
| 1 | матриця A:<br>-0.12 -0.53 -0.61 -8.23 -10.13 -13.41 -7.51 11.08<br>3.82 14.35 -7.9 -8.31 12.03 -5.54 -8.52 4.93<br>7.04 3.89 4.39 3.66 -12.1 4.65 0.18 -10.39<br>5.63 -6.53 1.06 8.43 -12.83 8.22 15.09 -13.3 |
| 2 | створення масиву з мінімальних значень<br>-0.12 -6.53 -7.9 -8.31 -12.83 -13.41 -8.52 -10.39 |
| 3 | сортування масиву за спаданням<br>-0.12 -6.53 -7.9 -8.31 -8.52 -12.83 -13.3 -13.41 |
| | кінець |

## 7. Висновок

Отже, сьогодні я дослідила алгоритм пошуку в матрицях та алгоритм сортування методом Шела.

В результаті лабораторної роботи я розробила алгоритм який заповнює матрицю, шукає найменші елементи в стовбцях і записує їх в масив, після чого методом Шела масив сортується від найбільшого до найменшого.

Метод Шела виконує декілька впорядкувань включенням, кожен раз порівнюючи і переставляючи елементи, що розташовані на різній відстані один від одного. Елементи порівнюються через $d_1$ до dm = 1, де d- це довжина кроку (через скільки елементів порівнюємо; для кожної нової ітерації ділиться на 2; початкове значення – кількість елементів поділена на 2).