*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 № 1 з дисципліни
«Алгоритми та структури даних-1.
Основи алгоритмізації»

«Дослідження лінійних алгоритмів»

Варіант 12

Виконав студент Дойчев Костянтин Миколайович
(шифр, прізвище, ім'я, по батькові)

Перевірила Вєчерковська Анастасія Сергіївна
( прізвище, ім'я, по батькові)

Київ 2021

# Лабораторна робота №8

Тема

**Мета** – дослідити алгоритми пошуку та сортування, набути практичних навичок використання цих алгоритмів під час складання програмних специфікацій

## Варіант №

### 1) Постановка задачі:

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. Створення нової змінної індексованого типу (одновимірний масив) та її ініціювання значеннями, що обчислюються згідно з варіантом

| Розмірність | Тип даних | Обчислення значень елементів одновимірного масиву |
|---|---|---|
| 6 x 4 | Цілий | Із максимальних значень елементів рядків двовимірного масиву. Відсортувати методом вставки за спаданням. |

*Розв'язання*

Крок 1. Визначимо основні дії.

Крок 2. Заповнимо матрицю випадково сгенерованими числами

Крок 3. Заповнимо масив з максимальними значеннями кожного рядка

Крок 4: Відсортуємо масив

2) Побудова математичної моделі:

Таблиця імен змінних

| Змінна | Тип | Ім’я | Призначення |
|---|---|---|---|
| Кл-сть рядків | Цілий | ROWS | Проміжні дані |
| Кл-сть стовпців | Цілий | COLUMNS | Проміжні дані |
| Мінімальне значення випадково згенерованого числа | Цілий | MIN_VALUE | Проміжні дані |
| Максимальне значення випадково згенерованого числа | Цілий | MAX_VALUE | Проміжні дані |
| Матриця | Цілий | matrix | Проміжні дані |
| Результуючий масив | Цілий | result | Вихідні дані |
| Розмір результуючого масиву | Цілий | size | Проміжні дані |
| Максимальне значення в певному рядку матриці | Цілий | max | Проміжні дані |
| Елемент масиву в ф-ції сортування | Цілий | currentValue | Проміжні дані |
| Ф-ція заповнення матриці | void | fillMatrix(matrix, callback) | Проміжні дані |
| Ф-ція генерування випадкового числа | Цілий | randomizeValue(min, max) | Проміжні дані |

| | | | |
|---|---|---|---|
| Колбек для заповнення матриці | Цілий | fillMatrixCallback(item) | Проміжні дані |
| Ф-ція для заповнення результуючого масиву | void | fillArray(array, matrix) | Проміжні дані |
| Ф-ція сортування | void | insertionSort | Проміжні дані |

Таким чином, математичне формулювання задачі зводиться до створення ф-ції **randomizeValue** для генерації випадкових чисел в діапазоні від **MIN_VALUE** до **MAX_VALUE.** В підпрограмі **randomizeValue** будемо використовувати вбудовану ф-цію **rand()** (rand() в C++, random() в Python).Завдяки цій ф-ції ми можемо заповнити матрицю **matrix** певними числами. Це заповнення буде виконано у підпрограмі **fillMatrix.** Наступним кроком ми будемо заповнювати масив **result** максимальними значеннями кожного рядка. Цей функціонал буде реалізований у підпрограмі **fillArray**. Далі буде реалізоване . Сортування вставками (Insertion sort) у ф-ції **insertionSort**. Тут ми будемо записувати обране значення у змінну **currentValue** і ітерувати масив для знаходження потрібного місця(щоб елементи, які більше **currentValue**, були лівіше, а елементи, які менше - справа).

## 3) Псевдокод алгоритму

Крок 1:

**Початок**

Заповнення матриці

Заповнення результуючого масиву

Сортування результуючого масиву

**Кінець**

Крок 2:

**Підпрограма**

randomizeValue(min, max)

**повернути** rand(min, max)

**Все підпрограма**

**Підпрограма**
fillMatrix(matrix, callback)
**для** i **від** 0 **до** ROWS **з кроком** 1 **повторити**
**для** j **від** 0 **до** COLUMNS **з кроком 1 повторити**
matrix[i][j] := callback(matrix[i][j]);
**все повторити**
**все повторити**
**Все підпрограма**

**Підпрограма**
fillMatrixCallback(item)
**повернути** randomizeValue(MIN_VALUE, MAX_VALUE);
**Все підпрограма**

**Підпрограма**
fillArray(array, matrix)
**для** i **від** 0 **до** ROWS **з кроком** 1 **повторити**
max: = matrix[i][0]
**для** j **від** 0 **до** COLUMNS **з кроком 1 повторити**
**якщо** matrix[i][j] > max
**то**
max: = matrix[i][j];
**все якщо**
**все повторити**
array[i] := max;
**все повторити**
**Все підпрограма**

**Підпрограма**
insertionSort(array, length)
**для** i **від** 0 **до** length **з кроком** 1 **повторити**
currentValue := array[i];
j = i - 1;
**Поки** j >= 0 && array[j] < currentValue **повторити**

```
                array[j + 1] = array[j];
                j = j - 1;
            все повторити
            array[j +1] := currentValue;
        все повторити
Все підпрограма



Початок
    fillMatrix(matrix, fillMatrixCalllback);
    Заповнення результуючого масиву
    Сортування результуючого масиву
Кінець
```

Крок 3:

```
Підпрограма
    randomizeValue(min, max)
        повернути rand(min, max)
Все підпрограма

Підпрограма
    fillMatrix(matrix, callback)
        для і від 0 до ROWS з кроком 1 повторити
            для j від 0 до COLUMNS з кроком 1 повторити
                matrix[i][j] := callback(matrix[i][j]);
            все повторити
        все повторити
Все підпрограма

Підпрограма
    fillMatrixCallback(item)
        повернути randomizeValue(MIN_VALUE, MAX_VALUE);
Все підпрограма
```

**Підпрограма**

fillArray(array, matrix)

**для** і **від** 0 **до** ROWS **з кроком** 1 **повторити**

max: = matrix[i][0]

**для** j **від** 0 **до** COLUMNS **з кроком 1 повторити**

**якщо** matrix[i][j] > max

**то**

max: = matrix[i][j];

**все якщо**

**все повторити**

array[i] := max;

**все повторити**

**Все підпрограма**

**Підпрограма**

insertionSort(array, length)

**для** i **від** 0 **до** length **з кроком** 1 **повторити**

currentValue := array[i];

j = i - 1;

**Поки** j >= 0 && array[j] < currentValue **повторити**

array[j + 1] = array[j];

j = j - 1;

**все повторити**

array[j +1] := currentValue;

**все повторити**

**Все підпрограма**

**Початок**

fillMatrix(matrix, fillMatrixCalllback);

fillArray(result, matrix)

Сортування результуючого масиву

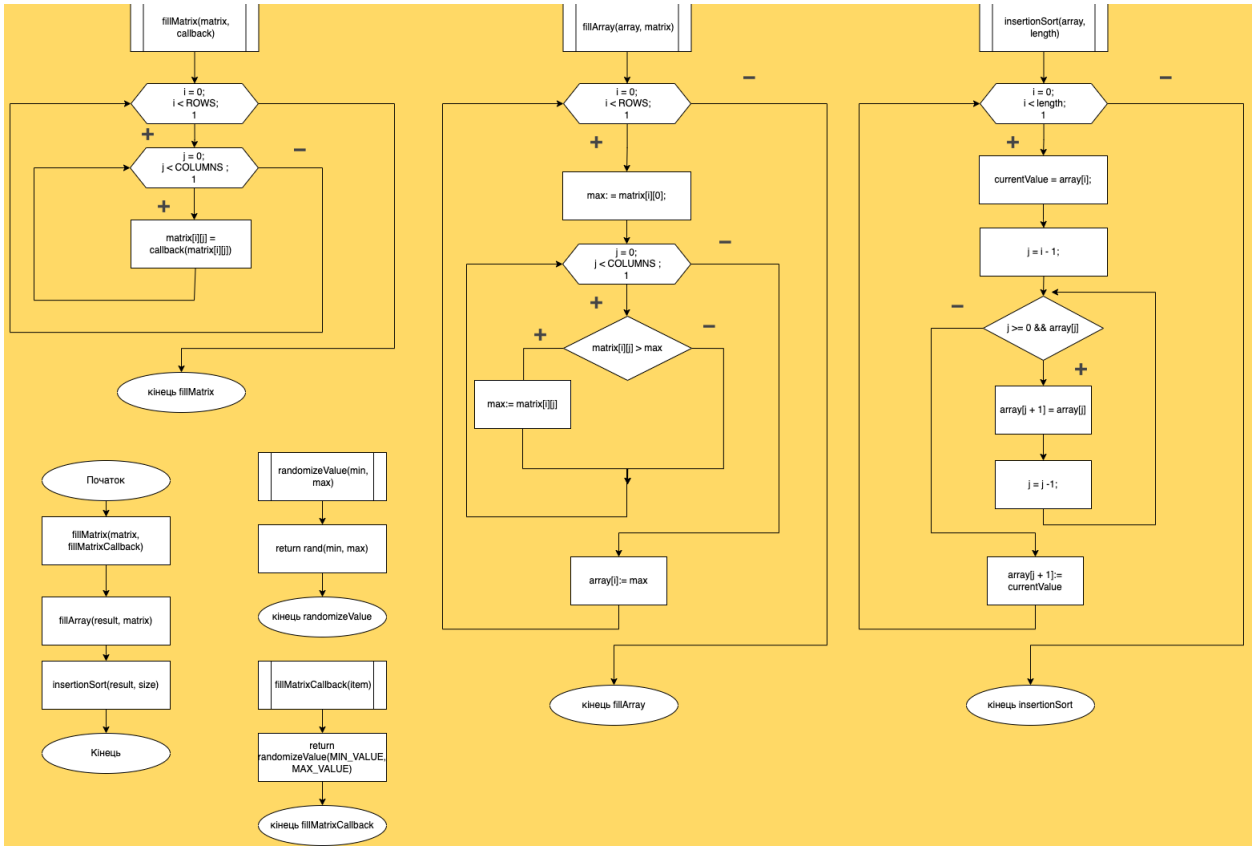
**Кінець**

Крок 4:

```
Підпрограма
    randomizeValue(min, max)
        повернути rand(min, max)
Все підпрограма

Підпрограма
    fillMatrix(matrix, callback)
        для і від 0 до ROWS з кроком 1 повторити
            для j від 0 до COLUMNS з кроком 1 повторити
                matrix[i][j] := callback(matrix[i][j]);
            все повторити
        все повторити
Все підпрограма

Підпрограма
    fillMatrixCallback(item)
        повернути randomizeValue(MIN_VALUE, MAX_VALUE);
Все підпрограма


Підпрограма
    fillArray(array, matrix)
        для і від 0 до ROWS з кроком 1 повторити
            max: = matrix[i][0]
            для j від 0 до COLUMNS з кроком 1 повторити
                якщо matrix[i][j] > max
                    то
                        max: = matrix[i][j];
                все якщо
            все повторити
            array[i] := max;
        все повторити
Все підпрограма

Підпрограма
```

```
        insertionSort(array, length)
            для i від 0 до length з кроком 1 повторити
                currentValue := array[i];
                j = i - 1;
                Поки j >= 0 && array[j] < currentValue повторити
                    array[j + 1] = array[j];
                    j = j - 1;
                все повторити
                array[j +1] := currentValue;
            все повторити
Все підпрограма




Початок
        fillMatrix(matrix, fillMatrixCallback);
        fillArray(result, matrix)
        insertionSort(result, size)
Кінець
```

4) Блок схема алгоритму


fillMatrix(matrix, callback)
i = 0; i < ROWS; 1
j = 0; j < COLUMNS ; 1
matrix[i][j] = callback(matrix[i][j])
кінець fillMatrix
fillArray(array, matrix)
i = 0; i < ROWS; 1
max: = matrix[i][0];
j = 0; j < COLUMNS ; 1
matrix[i][j] > max
max:= matrix[i][j]
array[i]:= max
кінець fillArray
insertionSort(array, length)
i = 0; i < length; 1
currentValue = array[i];
j = i - 1;
j >= 0 && array[j]
array[j + 1] = array[j]
j = j -1;
array[j + 1]:= currentValue
кінець insertionSort
Початок
fillMatrix(matrix, fillMatrixCallback)
fillArray(result, matrix)
insertionSort(result, size)
Кінець
randomizeValue(min, max)
return rand(min, max)
кінець randomizeValue
fillMatrixCallback(item)
return randomizeValue(MIN_VALUE, MAX_VALUE)
кінець fillMatrixCallback

5) Код алгоритму

CMakeLists.txt | main.cpp

```cpp
#include <iostream>
#include <iomanip>
#include <cstdlib>
#include <ctime>
#include <cmath>

const int ROWS = 6;
const int COLUMNS = 4;
const int MIN_VALUE = -100;
const int MAX_VALUE = 100;
typedef int Matrix[ROWS][COLUMNS];

int randomizeValue(int min, int max);

void fillMatrix(Matrix matrix, int (*callback)(int));

int fillMatrixCallback(int item);

void fillArray(int array[], Matrix matrix);

void insertionSort(int array[], int length);

template<typename T>
void printArray(T array[], int size);

using namespace std;

int main() {
    srand(time(NULL));
```

main

CMakeLists.txt | main.cpp

```
int main() {
    srand(time(NULL));

    Matrix matrix;
    int result[ROWS];

    fillMatrix(matrix, fillMatrixCallback);
    fillArray(result, matrix);
    size_t size = sizeof(result) / sizeof(result[0]);
    insertionSort(result, (int) size);

    printArray(result, size);
    return 0;
}

int randomizeValue(int min, int max) {
    return (rand() % (max - min + 1) + min);
}


void fillMatrix(Matrix matrix, int (*callback)(int)) {
    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLUMNS; j++) {
            matrix[i][j] = callback(matrix[i][j]);
        }
    }
}

void fillArray(int array[], Matrix matrix) {
```

main

CMakeLists.txt | main.cpp

```
void fillArray(int array[], Matrix matrix) {
    for (int i = 0; i < ROWS; i++) {
        int max = matrix[i][0];
        for (int j = 0; j < COLUMNS; j++) {
            if (matrix[i][j] > max) {
                max = matrix[i][j];
            }
        }
        array[i] = max;
    }
}

void insertionSort(int array[], int length) {
    int i, j, currentValue;
    for (i = 0; i < length; i++) {
        currentValue = array[i];
        j = i - 1;
        while (j >= 0 && array[j] < currentValue) {
            array[j + 1] = array[j];
            j--;
        }
        array[j + 1] = currentValue;
    }
}

int fillMatrixCallback(int item) {
    return randomizeValue(MIN_VALUE, MAX_VALUE);
}

```

```
int fillMatrixCallback(int item) {
    return randomizeValue(MIN_VALUE, MAX_VALUE);
}

template<typename T>
void printArray(T *array, int size) {
    int maxValueLength = (int) log10(MAX_VALUE) + 1;
    int minValueLength = (int) log10(MAX_VALUE) + 1;
    int longest = maxValueLength > minValueLength ? maxValueLength : minValueLength;

    cout << "Result: [";
    for (int i = 0; i < size; i++) {
        bool isLast = i + 1 == size;
        string separator = isLast ? "" : ", ";
        cout << setw( n: longest + 1) << array[i] << separator;
    }
    cout << "]" << endl;
}

```

## 6) Виновки:

Дослідив алгоритми пошуку та сортування, набув практичних навичок використання цих алгоритмів під час складання програмних специфікацій. Розв’язав задачу, побудував мат. модель, блок схему, написав псевдокод і код на мові С++