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 12

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	Цілий	currentValue	Проміжні дані

ф-ції сортування			
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
Виведення матриці та результуючого масиву

Кінець

Крок 2:

```
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);

Заповнення результуючого масиву

Сортування результуючого масиву

Виведення матриці та 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

Виведення матриці та результуючого масиву

Кінець

Крок 4:

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]

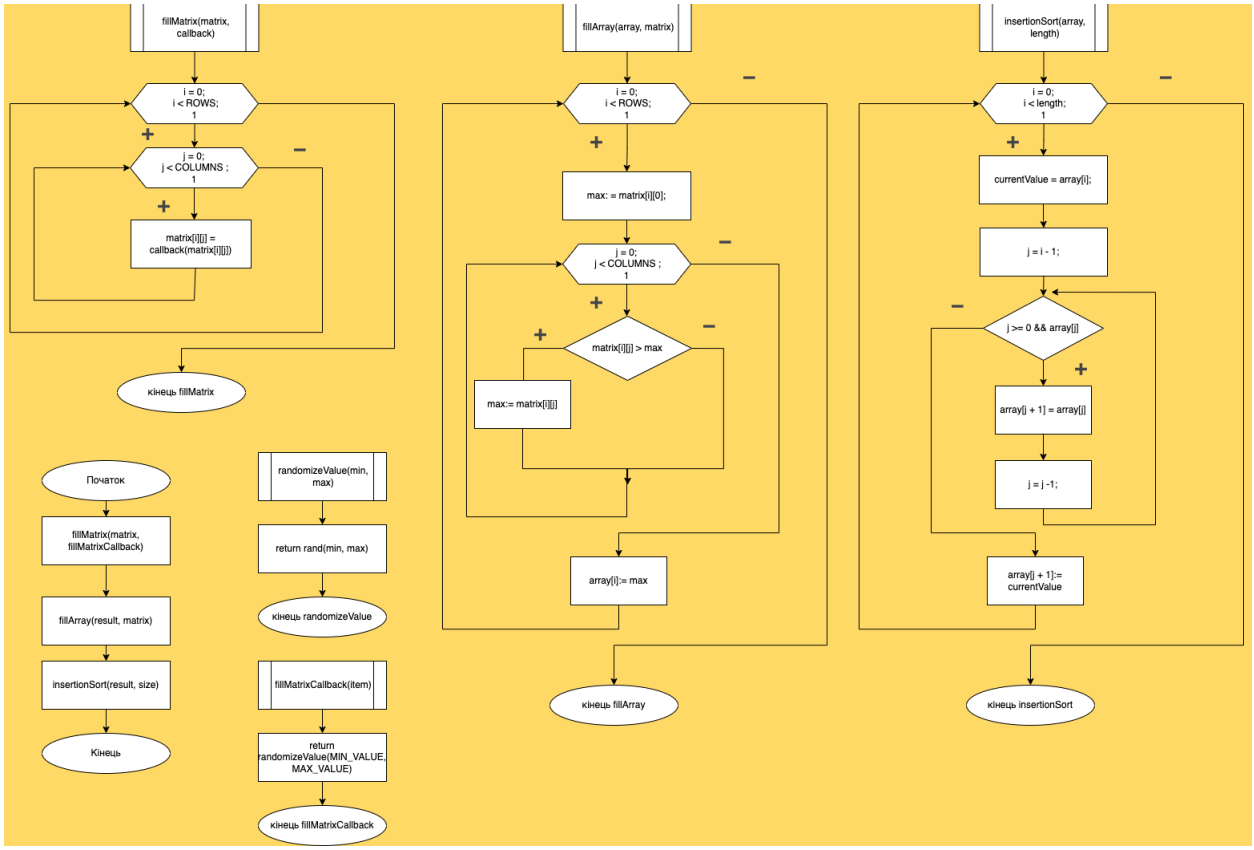
для j **від** 0 **до** COLUMNS **з кроком 1 повторити**

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, fillMatrixCallback);
fillArray(result, matrix)
insertionSort(result, size)
Виведення матриці та результуючого масиву
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

main.cpp README.md

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

using namespace std;
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);

string generateMatrixRow(int width, int columns, string content);

```

randomizeValue

main.cpp README.md

```cpp

void printMatrix(Matrix matrix, int columns, int rows);


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);

    cout << "Matrix: " << endl;
    printMatrix(matrix, COLUMNS, ROWS);
    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++) {
```

randomizeValue

main.cpp README.md

```
53
54  void fillMatrix(Matrix matrix, int (*callback)(int)) {
55      for (int i = 0; i < ROWS; i++) {
56          for (int j = 0; j < COLUMNS; j++) {
57              matrix[i][j] = callback(matrix[i][j]);
58          }
59      }
60  }
61
62  void fillArray(int array[], Matrix matrix) {
63      for (int i = 0; i < ROWS; i++) {
64          int max = matrix[i][0];
65          for (int j = 0; j < COLUMNS; j++) {
66              if (matrix[i][j] > max) {
67                  max = matrix[i][j];
68              }
69          }
70          array[i] = max;
71      }
72  }
73
74  void insertionSort(int array[], int length) {
75      int i, j, currentValue;
76      for (i = 0; i < length; i++) {
77          currentValue = array[i];
78          j = i - 1;
79          while (j >= 0 && array[j] < currentValue) {
80              array[j + 1] = array[j];
```

f randomizeValue

main.cpp README.md

```
73
74  void insertionSort(int array[], int length) {
75      int i, j, currentValue;
76      for (i = 0; i < length; i++) {
77          currentValue = array[i];
78          j = i - 1;
79          while (j >= 0 && array[j] < currentValue) {
80              array[j + 1] = array[j];
81              j--;
82          }
83          array[j + 1] = currentValue;
84      }
85  }
86
87  int fillMatrixCallback(int item) {
88      return randomizeValue(MIN_VALUE, MAX_VALUE);
89  }
90
91  template<typename T>
92  void printArray(T *array, int size) {
93      int maxValueLength = (int) log10(MAX_VALUE) + 1;
94      int minValueLength = (int) log10(MAX_VALUE) + 1;
95      int longest = maxValueLength > minValueLength ? maxValueLength : minValueLength;
96
97      cout << "Result: [";
98      for (int i = 0; i < size; i++) {
99          bool isLast = i + 1 == size;
100         string separator = isLast ? "" : ", ";
```

f randomizeValue

main.cpp README.md

```cpp
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;
}

void printMatrix(Matrix matrix, int columns, int rows) {
    int maxValueLength = (int) log10(MAX_VALUE) + 1;
    int minValueLength = (int) log10(MAX_VALUE) + 1;
    int longest = maxValueLength > minValueLength ? maxValueLength : minValueLength;

    const int CELL_WIDTH = longest + 1;
    cout << generateMatrixRow(CELL_WIDTH, columns, content: "-") << endl;
    for (int i = 0; i < rows; i++) {
        cout << "|";
        for (int j = 0; j < columns; j++) {
            cout << setw(CELL_WIDTH) << matrix[i][j] << "|";
        }
        cout << endl;
```

f randomizeValue

main.cpp README.md

```cpp
105
106 void printMatrix(Matrix matrix, int columns, int rows) {
107     int maxValueLength = (int) log10(MAX_VALUE) + 1;
108     int minValueLength = (int) log10(MAX_VALUE) + 1;
109     int longest = maxValueLength > minValueLength ? maxValueLength : minValueLength;
110
111     const int CELL_WIDTH = longest + 1;
112     cout << generateMatrixRow(CELL_WIDTH, columns, content: "-") << endl;
113     for (int i = 0; i < rows; i++) {
114         cout << "|";
115         for (int j = 0; j < columns; j++) {
116             cout << setw(CELL_WIDTH) << matrix[i][j] << "|";
117         }
118         cout << endl;
119         cout << generateMatrixRow(CELL_WIDTH, columns, content: "-") << endl;
120     }
121 }
122
123 string generateMatrixRow(int width, int columns, string content) {
124     string res = "|";
125     for (int i = 0; i < columns; i++) {
126         for (int j = 0; j < width; j++) {
127             res += content;
128         }
129         res += "|";
130     }
131     return res;
132 }
```

f randomizeValue

Debug: Lab_8

Debugger Console

```
/Users/Kostia/Documents/Programming/kpi/algorithms-and-data-structures/Lab-8/cmake-build-debug/Lab_8
Matrix:
|----|----|----|----|
| -27| -34|  11| -12|
|----|----|----|----|
| -71|  16|  81|  10|
|----|----|----|----|
| -62| -10| -67|  39|
|----|----|----|----|
|  60| 100| -66|  65|
|----|----|----|----|
|   8| -67|  41|  75|
|----|----|----|----|
|  39|  36|  95| -80|
|----|----|----|----|
Result: [ 100,   95,   81,   75,   39,   11]

Process finished with exit code 0
```

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++