Додаток 1

Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

Звіт

з лабораторної роботи №8
з дисципліни «Алгоритми та структури даних-1. Основи алгоритмізації»
«Дослідження алгоритмів пошуку та сортування»
Варіант 6

Виконав студент ІП-15, Волинець Кирило Михайлович
(шифр, прізвище, ім'я, по батькові)

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

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.

Індивідуальне завдання

Варіант 6

6	7 x 5	Дійсний	Із максимальних значень елементів рядків двовимірного масиву. Відсортувати методом бульбашки за спаданням.

Побудова математичної моделі.

Складемо таблицю імен змінних.

Змінна	***Тип***	***Ім'я***	***Призначення***
Кількість рядків	Цілочисельний	base_y_size (=7)	Константа
Кількість стовбців	Цілочисельний	base_x_size (=5)	Константа
Вхідна матриця	Дійсний[base_y_size][base_x_size]	matrix	Вхідні
Масив максимальних елементів рядків	Дійсний[base_y_size]	result_array	Результат
Ітератор	Цілочисельний	i	Проміжні
Максимальний елемент масиву	Дійсний (функція)	max(double*, int)	Обчислення
Максимум для розглянутих елементів	Дійсний	maximum	Проміжні
Сортування бульбашкою (перевернуте)	Пустий (функція)	inverted_bubble_sort(double*, int)	Обчислення
Тимчасова змінна для перестановки елементів	Дійсний	temp	Проміжні
Кількість операцій на проході	Цілочисельний	num_of_operations	Проміжні
Номер проходу	Цілочисельний	pass	Проміжні

За умовою створимо матрицю розміром 7 на 5 та заповнимо її числами (у програмі генеруються випадкові числа). Потім знаходимо максимум кожного з рядків і записуємо у новий масив. Далі йде сортування бульбашкою. Замість двох вкладених арифметичних циклів я використовую цикл з постумовою на відсутність перестановок за прохід, що сигналізує про відсортованість масиву.

Розв'язання

Програмні специфікації запишемо у псевдокоді та графічній формі у вигляді блок-схеми.

Псевдокод

Основна програма:

```
початок
    ввести matrix
    повторити для i від 0 до i < base_y_size:
        result_array[i] = max(matrix[i], base_x_size)
    все повторити
    inverted_bubble_sort(result_array, base_y_size)
    вивести result_array
кінець
```

Підпрограми:

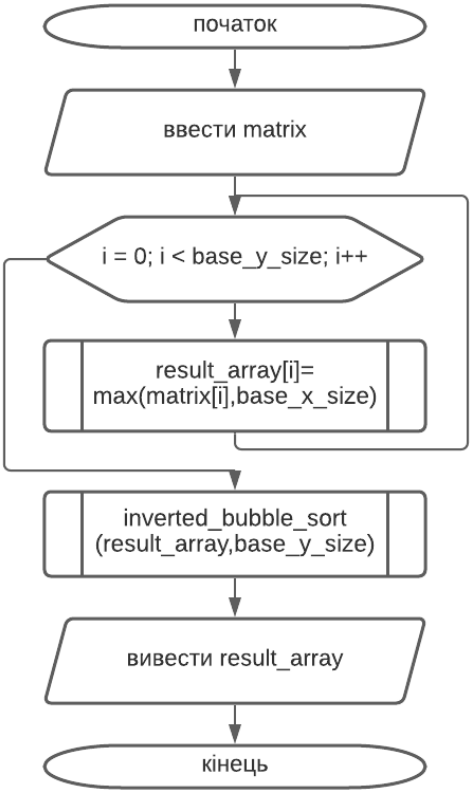
```
max(double* array, int size)
    maximum = array[1];
    повторити для i від 0 до i < size:
        якщо array[i] > maximum то maximum = array[i]
    все повторити
    повернути maximum
кінець max

inverted_bubble_sort(double* array, int size)
    num_of_operations = 1
    pass = 1
    повторити:
        num_of_operations = 0
        повторити для i від 0 до i < size - pass:
            якщо array[i] < array[i + 1] то:
                temp = array[i]
                array[i] = array[i + 1]
                array[i + 1] = temp
                num_of_operations = num_of_operations + 1
            все якщо
        все повторити
```

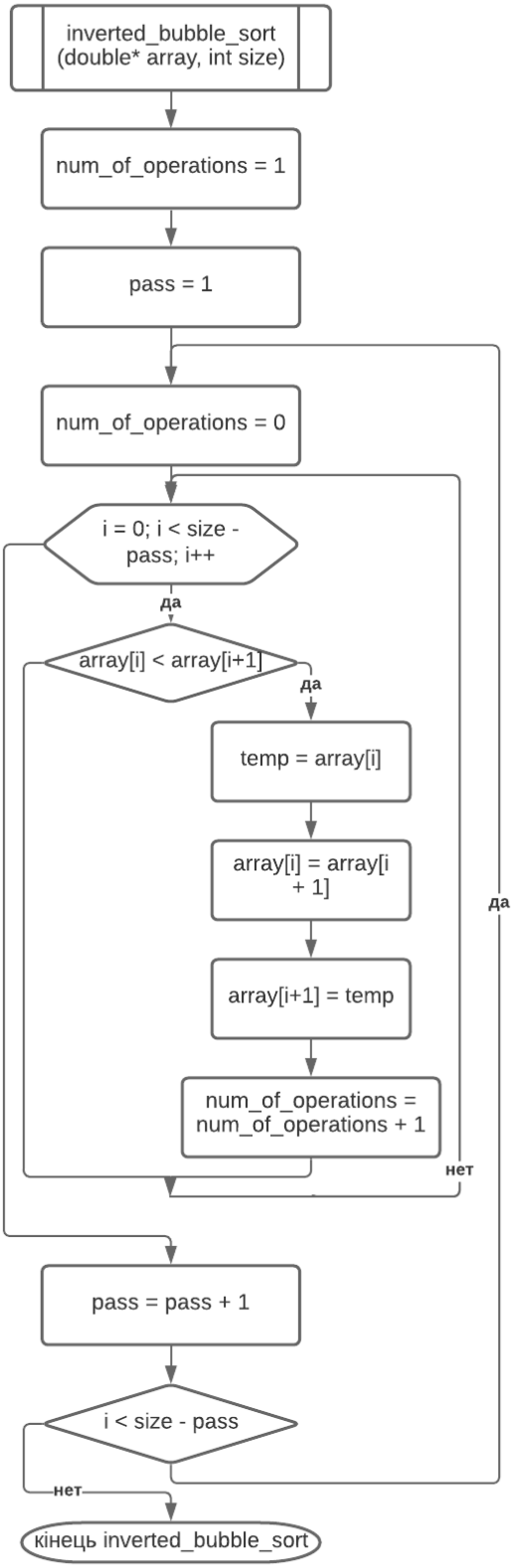
pass = pass + 1

поки num_of_operations != 0 **все повторити**

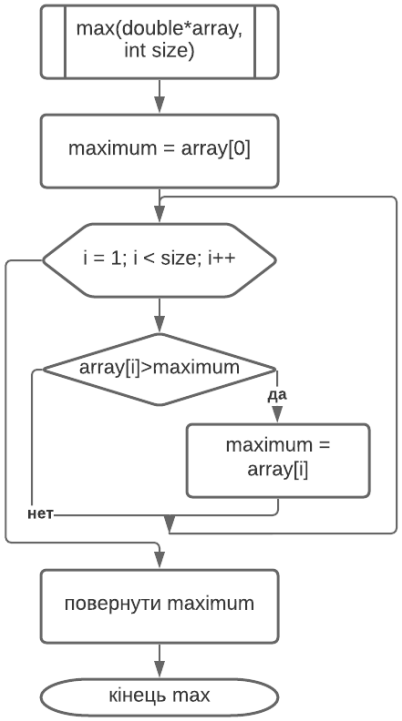
кінець inverted_bubble_sort

Блок-схема:





Підпрограми:



Код

```
#include <iostream>
#include <random>
#include <ctime>
#include <cmath>
#include <iomanip>

using namespace std;

const int base_y_size = 7, base_x_size = 5;

double max(double* array, int size);
void inverted_bubble_sort(double* array, int size);

void generate_random_values(double** matrix, int y_size, int x_size, double min_value, double
max_value);
void print(double** matrix, int y_size, int x_size);
void print(double* array, int size);

int main()
{
    double** matrix = new double* [base_y_size];
    for (int i = 0; i < base_y_size; i++) {
        matrix[i] = new double[base_x_size];
    }
    generate_random_values(matrix, base_y_size, base_x_size, -50, 110);
    print(matrix, base_y_size, base_x_size);
    double* result_array = new double[base_y_size];
    for (int i = 0; i < base_y_size; i++) {
        result_array[i] = max(matrix[i], base_x_size);
    }
    print(result_array, base_y_size);
    inverted_bubble_sort(result_array, base_y_size);
    print(result_array, base_y_size);
}

void generate_random_values(double** matrix, int y_size, int x_size, double min_value, double
max_value) {
    srand(time(NULL) * 100);
    //srand(229);
    double random_zeroone_number;
    for (int i = 0; i < y_size; i++) {
        for (int j = 0; j < x_size; j++) {
            random_zeroone_number = double(rand()) / RAND_MAX;
            matrix[i][j] = random_zeroone_number * (max_value - min_value) + min_value;
        }
    }
}

double max(double* array, int size) {
    double maximum = array[0];
    for (int i = 1; i < size; i++) {
        if (array[i] > maximum) maximum = array[i];
    }
    return maximum;
}

void inverted_bubble_sort(double* array, int size) {
    double temp;
    int num_of_operations, pass = 1;
    do{
        num_of_operations = 0;
        for (int i = 0; i < size - pass; i++) {
            if (array[i] < array[i + 1]) {
                temp = array[i];
```

```
                array[i] = array[i + 1];
                array[i + 1] = temp;
                num_of_operations++;
            }
        }
        pass++;
    } while (num_of_operations != 0);
}

void print(double* array, int size) {
    for (int i = 0; i < size; i++) {
        cout << round(array[i]*10)/10 << " ";
    }
    cout << endl;
}

void print(double** matrix, int y_size, int x_size) {
    int max_column_length = 0, curr_len;
    int* column_lengths = new int[x_size];
    for (int i = 0; i < x_size; i++) {
        max_column_length = 0;
        for (int j = 0; j < y_size; j++) {
            curr_len = ceil(log10(fabs(matrix[j][i])));
            if (curr_len > max_column_length) max_column_length = curr_len;
        }
        column_lengths[i] = max_column_length;
    }
    for (int i = 0; i < y_size; i++) {
        for (int j = 0; j < x_size; j++) {
            cout<< setw(column_lengths[j]+3) << round(matrix[i][j]*10)/10 << " ";
        }
        cout << endl;
    }
    cout << endl;
}
```

Висновки

Ми дослідили алгоритми пошуку та сортування, набули практичних навичок використання цих алгоритмів під час складання програмних специфікацій. В результаті виконання лабораторної роботи ми отримали алгоритм для знаходження масиву максимальних значень у рядку матриці та сортування цього масиву.