# 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 № 9 з дисципліни

«Алгоритми та структури даних.

Основи алгоритмізації»

Варіант 13

Виконав студент ІП-11 Калашніков Андрій Євгенович

(шифр, прізвище, ім'я, по батькові)

Перевірив Мартинова О.П.

( прізвище, ім'я, по батькові)

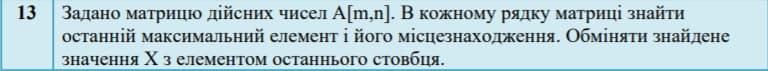
Київ 2021

**Лабораторна робота №9**

**ДОСЛІДЖЕННЯ АЛГОРИТМІВ ОБХОДУ МАСИВІВ**

**Мета –** дослідити алгоритми обходу масивів, набути практичних навичок використання цих алгоритмів під час складання програмних специфікацій.

**Індивідуальне завдання:**



**Постановка задачі**

Необхідно ініціалізувати двовимірний масив розмірності m x n (беручи до уваги, що індексація масивів починається з нульового елементу) випадковими дійсними значеннями. Потім необхідно пройтися по рядкам цього масиву “змійкою” та знайти індекси максимальних елементів кожного рядка й перенести їх до відповідних елементів одновимірного масиву, який доведеться створити для виконання алгоритму. Потім необхідно в кожному рядку замінити місцями останній елемент рядка та знайдений максимальний елемент рядка.

**Побудова математичної моделі**

Складемо таблицю змінних

|  |  |  |  |
| --- | --- | --- | --- |
| Змінна | Тип | Ім’я | Призначення |
| Заданий двовимірний масив | Дійсний | matrix | Проміжні дані |
| Одновимірний масив з індексами мінімальних елементів рядків | Цілий | arr\_max | Проміжні дані |
| Задана розмірність матриці (рядки) | Цілий | m | Вхідні дані |
| Задана розмірність матриці (стовпчики) | Цілий | n | Вхідні дані |
| Лічильник j | Цілий | j | Проміжні дані |
| Лічильник k | Цілий | k | Проміжні дані |
| Поточний індекс максимального елементу рядка | Дісний | max\_index | Проміжні дані |

**Розв’язання**

Програмні специфікації запишемо у псевдокоді та графічній формі у вигляді блок-схеми. Крок 1. Визначимо основні дії;

Крок 2. Деталізуємо підпрограму створення та заповнення матриці випадковими значеннями (fill\_matrix) та її виклик;

Крок 3. Деталізуємо підпрограму виведення матриці (display\_matrix) та її виклик;

Крок 4. Деталізуємо підпрограму пошуку індексів максимальних елементів рядків (find\_max) та її виклик;

Крок 5. Деталізуємо підпрограму виведення одновимірного масиву (display\_arr) та її виклик; Крок 6. Деталізуємо підпрограму обміну значень максимального елементу кожного рядка та останнім елементом того ж рядка (replace\_elements) та її виклик

**Псевдокод**

**Основна програма**

**Початок**

**Введення** m

**Введення** n

arr\_max[m]

matrix[m][n] := fill\_matrix(m, n)

display\_matrix (matrix, m, n)

arr\_max := find\_max(matrix, m, n)

display\_arr(arr\_max, m)

matrix := replace\_elements(matrix, arr\_max, m, n)

**display\_matrix (matrix, m, n)**

**Кінець**

**Підпрограми**

*fill\_matrix(matrix, rows, columns)*

min\_rand := -25

max\_rand := 25

**повторити для** i від 0 до rows

**повторити для** j від 0 до columns

matrix[i][j] := (rand() % (max\_rand \* 10 - min\_rand \* 10 + 1) +

min\_rand \* 10) / 10.0

**все повторити**

**все повторити**

**Кінець** *fill\_matrix(matrix, rows, columns)*

display\_matrix(array, size)

**повторити для** i від 0 до size

**повторити для** j від 0 до size

**Виведення** array[i][j]

**все повторити**

**все повторити**

***Кінець*** *fill\_matrix(matrix, rows, columns)*

підпрограма find\_max(array, rows, columns)

arr[rows]

**повторити для** i від 0 до rows

temp\_max := array[i][0];

max\_index := 0;

**якщо** (i % 2 == 1)

то

**повторити для** j від 0 до columns

**якщо** (array[i][j] >= temp\_max)

то

temp\_max := array[i][j];

max\_index := j;

все **якщо**

**все повторити**

інакше

**повторити для** j від columns до 0 з кроком -1

**якщо** (array[i][j] >= temp\_max)

то

temp\_max := array[i][j];

max\_index := j;

все **якщо**

**все повторити**

все **якщо**

arr[i] := max\_index;

**все повторити**

return arr

**Кінець** find\_max(array, rows, columns)

display\_arr(arr, length)

**повторити для** i від 0 до length

Виведення arr[i]

**все повторити**

**Кінець** display\_arr(arr, length)

replace\_elements(matrix, arr, rows, columns)

**повторити для** i від 0 до rows

max\_index := arr[i]

temp\_max := matrix[i][max\_index]

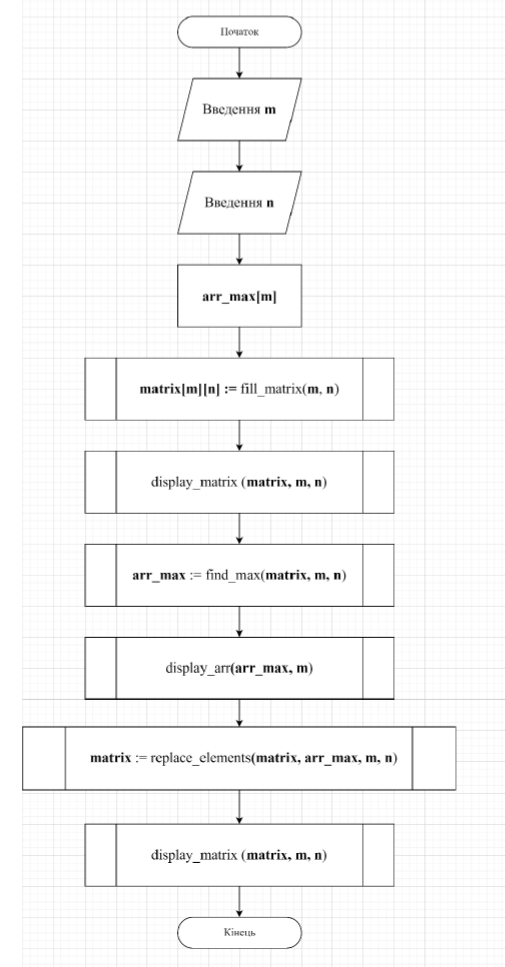
matrix[i][max\_index] := matrix[i][columns - 1]

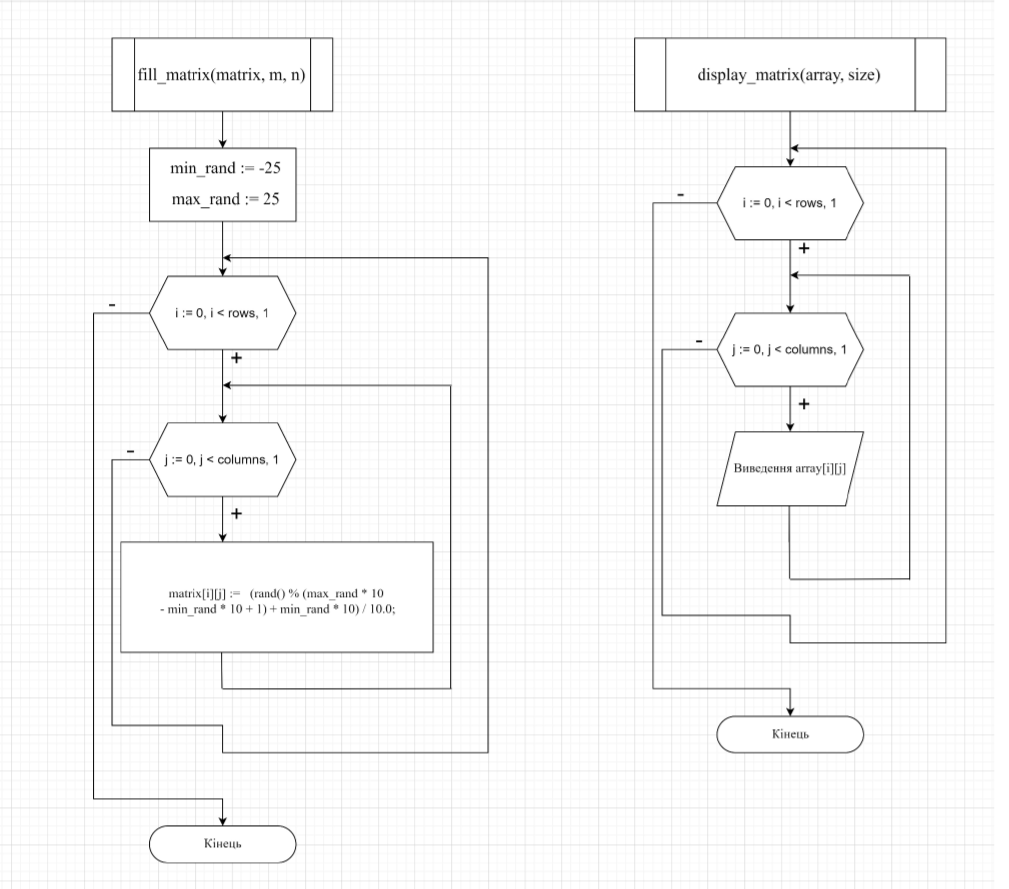
matrix[i][columns - 1] := temp\_max

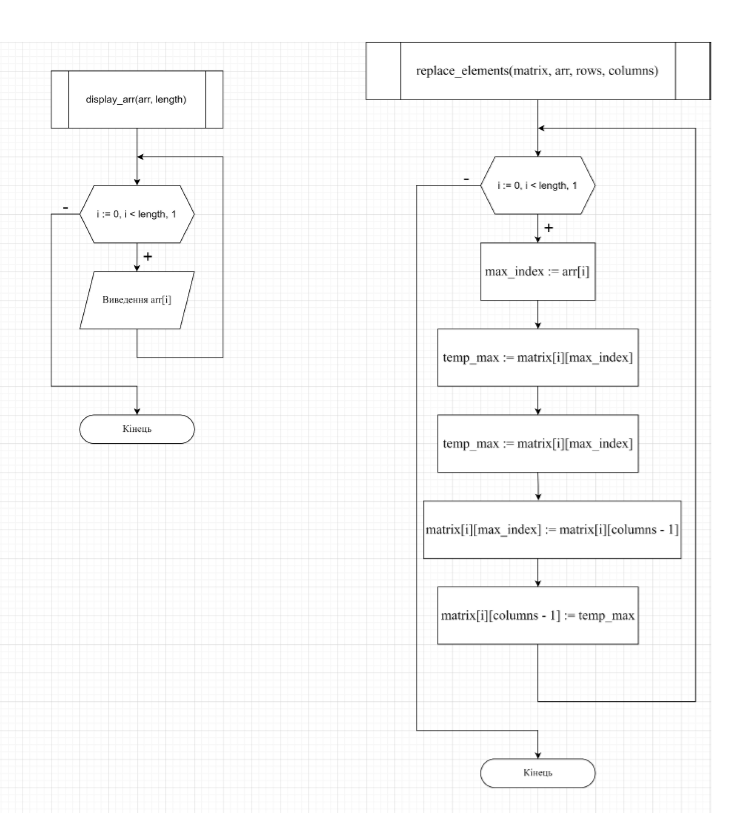
**все повторити**

**Кінець**  replace\_elements(matrix, arr, rows, columns)

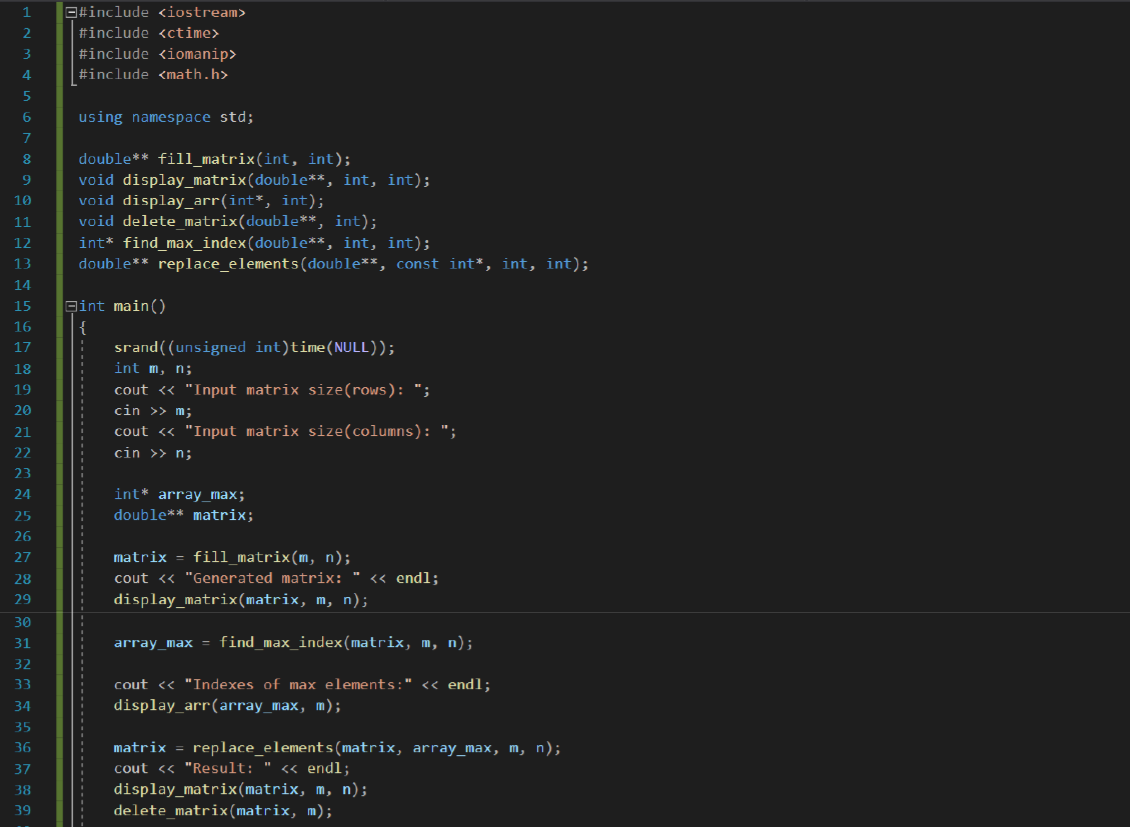
**Блок-схема**

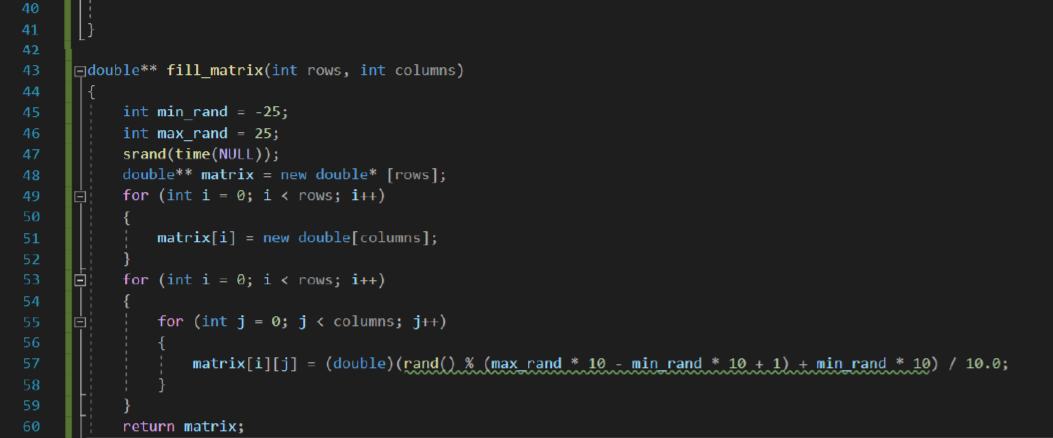
****

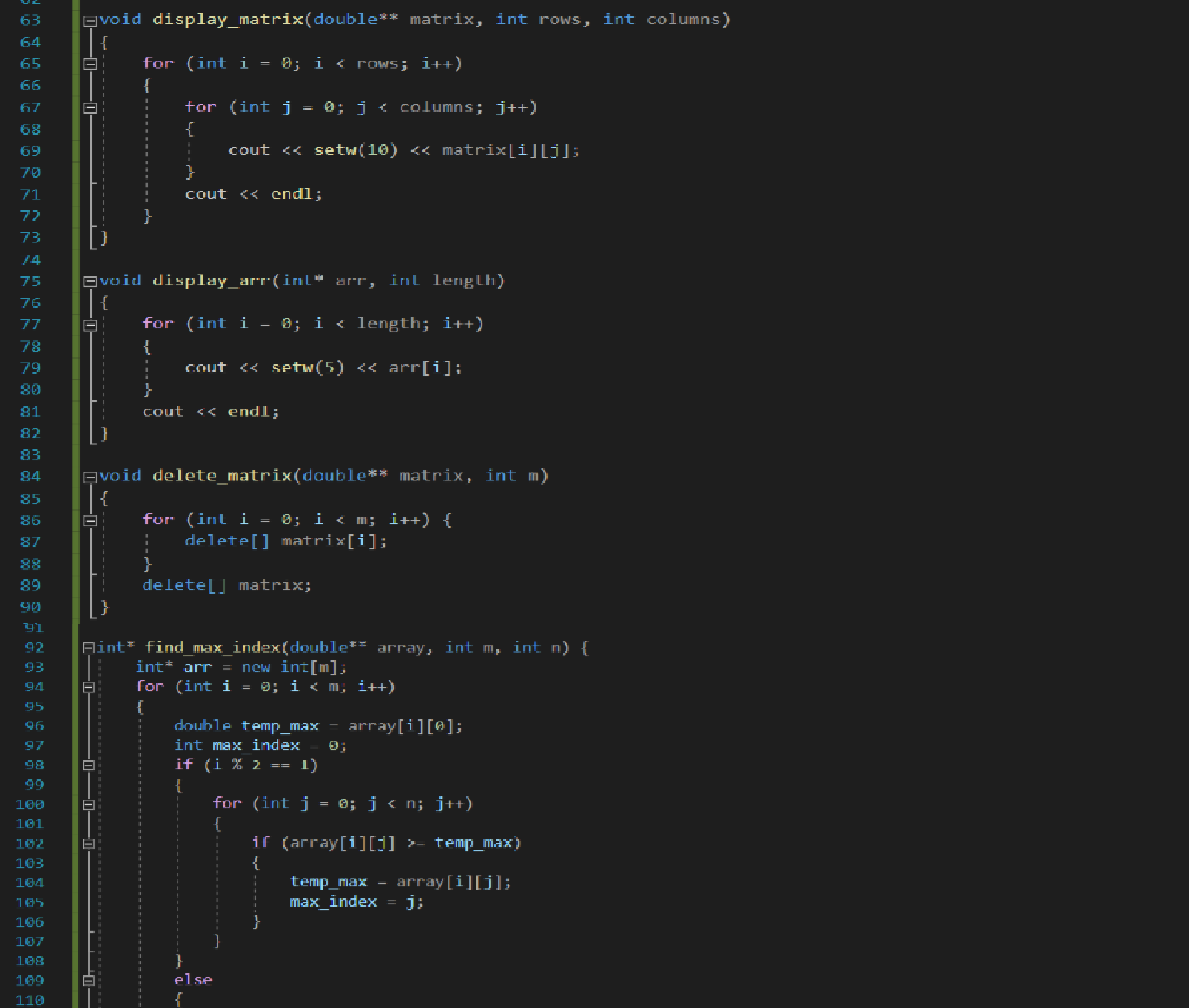
****

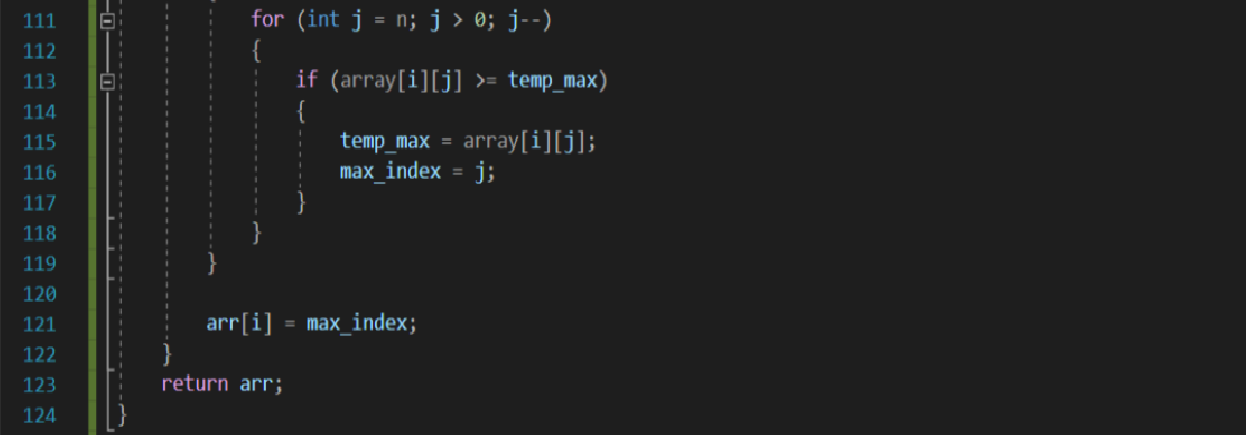
****

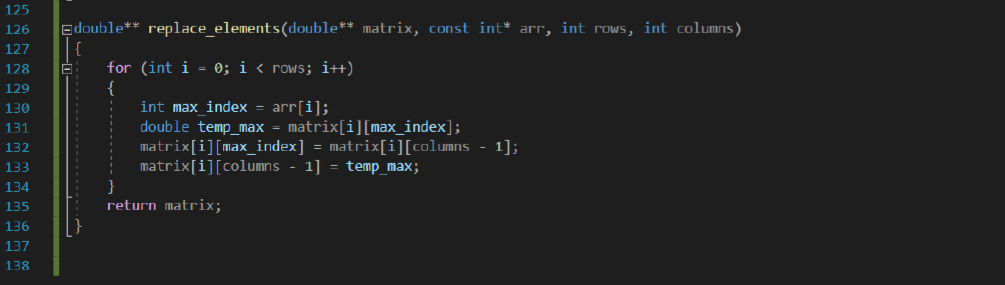
**Код на С++ :**

****

****

****

****

****

#include <iostream>

#include <ctime>

#include <iomanip>

#include <math.h>

using namespace std;

double\*\* fill\_matrix(int, int);

void display\_matrix(double\*\*, int, int);

void display\_arr(int\*, int);

void delete\_matrix(double\*\*, int);

int\* find\_max\_index(double\*\*, int, int);

double\*\* replace\_elements(double\*\*, const int\*, int, int);

int main()

{

srand((unsigned int)time(NULL));

int m, n;

cout << "Input matrix size(rows): ";

cin >> m;

cout << "Input matrix size(columns): ";

cin >> n;

int\* array\_max;

double\*\* matrix;

matrix = fill\_matrix(m, n);

cout << "Generated matrix: " << endl;

display\_matrix(matrix, m, n);

array\_max = find\_max\_index(matrix, m, n);

cout << "Indexes of max elements:" << endl;

display\_arr(array\_max, m);

matrix = replace\_elements(matrix, array\_max, m, n);

cout << "Result: " << endl;

display\_matrix(matrix, m, n);

delete\_matrix(matrix, m);

}

double\*\* fill\_matrix(int rows, int columns)

{

int min\_rand = -25;

int max\_rand = 25;

srand(time(NULL));

double\*\* matrix = new double\* [rows];

for (int i = 0; i < rows; i++)

{

matrix[i] = new double[columns];

}

for (int i = 0; i < rows; i++)

{

for (int j = 0; j < columns; j++)

{

matrix[i][j] = (double)(rand() % (max\_rand \* 10 - min\_rand \* 10 + 1)

+ min\_rand \* 10) / 10.0;

}

}

return matrix;

}

void display\_matrix(double\*\* matrix, int rows, int columns)

{

for (int i = 0; i < rows; i++)

{

for (int j = 0; j < columns; j++)

{

cout << setw(10) << matrix[i][j];

}

cout << endl;

}

}

void display\_arr(int\* arr, int length)

{

for (int i = 0; i < length; i++)

{

cout << setw(5) << arr[i];

}

cout << endl;

}

void delete\_matrix(double\*\* matrix, int m)

{

for (int i = 0; i < m; i++) {

delete[] matrix[i];

}

delete[] matrix;

}

int\* find\_max\_index(double\*\* array, int rows, int columns)

{

int\* arr = new int[rows];

for (int i = 0; i < rows; i++)

{

double temp\_max = array[i][0];

int max\_index = 0;

if (i % 2 == 1)

{

for (int j = 0; j < columns; j++)

{

if (array[i][j] >= temp\_max)

{

temp\_max = array[i][j];

max\_index = j;

}

}

}

else

{

for (int j = columns; j > 0; j--)

{

if (array[i][j] >= temp\_max)

{

temp\_max = array[i][j];

max\_index = j;

}

}

}

arr[i] = max\_index;

}

return arr;

}

double\*\* replace\_elements(double\*\* matrix, const int\* arr, int rows, int

columns)

{

for (int i = 0; i < rows; i++)

{

int max\_index = arr[i];

double temp\_max = matrix[i][max\_index];

matrix[i][max\_index] = matrix[i][columns - 1];

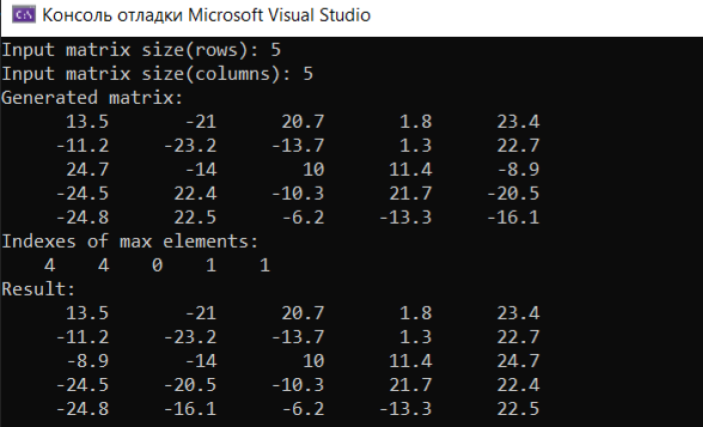
matrix[i][columns - 1] = temp\_max;

}

return matrix;

}

**Скріншот результатів C++ :**

****

**Випробування**

|  |  |
| --- | --- |
| Блок | Дія |
|  | Початок |
| 1 | Введення 5,5 |
| 2 | Виведення: Matrix:  13.5 -21 20.7 1.8 23.4  -11.2 -23.2 -13.7 1.3 22.7  24.7 -14 10 11.4 -8.9  -24.5 22.4 -10.3 21.7 -20.5  -24.8 22.5 -6.2 -13.3 -16.1 |
| 3 | Виведення: Sorted matrix:  13.5 -21 20.7 1.8 23.4  -11.2 -23.2 -13.7 1.3 22.7  -8.9 -14 10 11.4 24.7  -20.5 22.4 -10.3 21.7 22.4  -24.8 -16.1 -6.2 -13.3 22.5 |
|  | Кінець |

**Висновок**

Під час лабораторної роботи я дослідив алгоритми обходу масивів,

й набув практичних навичок їх використання, застосувавши алгоритм обходу для знаходження останнього максимального елементу в кожному рядку та обмін його положення з елементом останнього стовпця. Я отримав коректний результат, отже алгоритм працює правильно.