ЗВІТ

про виконання лабораторної роботи № 7.03

«Опрацювання динамічних багатовимірних масивів» з дисципліни

«Алгоритмізація та програмування» студента групи РІ-12

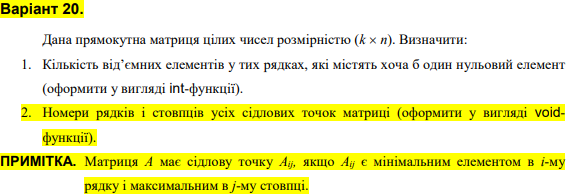
Мілянця Мар`яна

# Мета роботи:

Навчитися опрацьовувати динамічні двовимірні та багатовимірні масиви.

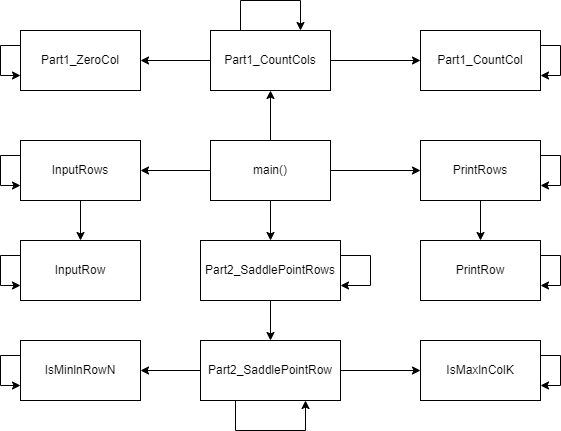
# Умова завдання:

Розмірності масивів слід задати за допомогою змінних а їх значення – вводити з клавіатури під час виконання програми (масиви мають бути динамічними). Усі необхідні дані мають передаватися функціям як параметри; всі величини, які використовуються лише всередині функцій, мають бути описані як локальні. Використання глобальних змінних у функціях не допускається. Виведення результатів роботи функцій має виконуватися в головній функції. Кожна функція має виконувати лише одну роль, і ця роль має бути відображена у назві функції. Рекурсивний та ітераційний способи – це 2 різні проекти, для яких потрібно 2 різних unit-тести і 2 різних звіти. «Функція, яка повертає / обчислює / шукає ...» – має не виводити ці значення, а повернути їх у місце виклику як результат функції або як відповідний вихідний параметр. Дана прямокутна матриця цілих чисел розмірністю (k  n). Визначити: 1. Кількість додатних елементів в тих стовпцях, які містять хоч б один нульовий елемент (оформити у вигляді int-функції). 2. Номери рядків та стовпців всіх сідлових точок матриці (оформити у вигляді voidфункції). ПРИМІТКА. Матриця А має сідлову точку Аij, якщо Аij є максимальним елементом в i-му рядку та мінімальним в j-му стовпці.



# Структурна схема програми:

**Рекурсивний спосіб:**



# Текст програми:

#include <iostream> #include <iomanip> using namespace std;

void PrintRow(int\*\* a, const int rowNo, const int colCount, int colNo)

{

cout << setw(4) << a[rowNo][colNo]; if (colNo < colCount - 1)

PrintRow(a, rowNo, colCount, colNo + 1);

else

}

cout << endl;

void PrintRows(int\*\* a, const int rowCount, const int colCount, int rowNo)

{

PrintRow(a, rowNo, colCount, 0); if (rowNo < rowCount - 1)

PrintRows(a, rowCount, colCount, rowNo + 1);

else

}

cout << endl;

void InputRow(int\*\* a, const int rowNo, const int colCount, int colNo)

{

cout << "a[" << rowNo << "][" << colNo << "] = "; cin >> a[rowNo][colNo];

if (colNo < colCount - 1)

InputRow(a, rowNo, colCount, colNo + 1);

else

}

cout << endl;

void InputRows(int\*\* a, const int rowCount, const int colCount, int rowNo)

{

InputRow(a, rowNo, colCount, 0); if (rowNo < rowCount - 1)

InputRows(a, rowCount, colCount, rowNo + 1);

else

}

cout << endl;

void Part1\_ZeroCol(int\*\* a, const int colCount, bool& result, int colNo, int rowNo, int& k\_zero)

{

if (a[rowNo][colNo] == 0)

{

}

else

}

result = true; k\_zero++;

if (colNo < colCount - 1)

Part1\_ZeroCol(a, colCount, result, colNo + 1, rowNo, k\_zero);

void Part1\_CountCol(int\*\* a, const int colCount, int& count, int colNo, int rowNo)

{

if (a[rowNo][colNo] < 0) count++;

if (colNo < colCount - 1)

Part1\_CountCol(a, colCount, count, colNo + 1, rowNo);

}

int Part1\_CountCols(int\*\* a, const int rowCount, const int colCount, int& count, bool& result, int rowNo)

{

int k\_zero = 0;

Part1\_ZeroCol(a, colCount, result, 0, rowNo, k\_zero); if (k\_zero > 0)

Part1\_CountCol(a, colCount, count, 0, rowNo); if (rowNo < rowCount - 1)

return Part1\_CountCols(a, rowCount, colCount, count, result, rowNo + 1);

}

bool IsMinInRowN(int\*\* a, const int rowNo, const int min, const int colCount, int colNo)

{

if (a[rowNo][colNo] < min) return false;

if (colNo < colCount - 1)

return IsMinInRowN(a, rowNo, min, colCount, colNo + 1);

else

}

return true;

bool IsMaxInColK(int\*\* a, const int colNo, const int max, const int rowCount, int rowNo)

{

if (a[rowNo][colNo] > max) return false;

if (rowNo < rowCount - 1)

return IsMaxInColK(a, colNo, max, rowCount, rowNo + 1);

else

}

return true;

void Part2\_SaddlePointRow(int\*\* a, const int rowCount, const int colCount, int& No, const int rowNo, int colNo)

{

if (IsMinInRowN(a, rowNo, a[rowNo][colNo], colCount, 0) && IsMaxInColK(a, colNo, a[rowNo][colNo], rowCount, 0))

{

cout << setw(4) << ++No << setw(6) << rowNo << setw(6) << colNo << endl;

}

if (colNo < colCount - 1)

Part2\_SaddlePointRow(a, rowCount, colCount, No, rowNo, colNo + 1);

}

void Part2\_SaddlePointRows(int\*\* a, const int rowCount, const int colCount, int& No, int rowNo)

{

Part2\_SaddlePointRow(a, rowCount, colCount, No, rowNo, 0); if (rowNo < rowCount - 1)

Part2\_SaddlePointRows(a, rowCount, colCount, No, rowNo + 1);

}

int main()

{

int rowCount, colCount;

cout << "rowCount = "; cin >> rowCount; cout << "colCount = "; cin >> colCount; int\*\* a = new int\* [rowCount];

for (int i = 0; i < rowCount; i++) a[i] = new int[colCount];

InputRows(a, rowCount, colCount, 0); PrintRows(a, rowCount, colCount, 0);

int count = 0;

bool result = false;

Part1\_CountCols(a, rowCount, colCount, count, result, 0); if (result)

cout << "count = " << count << endl;

else

cout << "there are no zero elements" << endl;

cout << endl;

cout << "Saddle points: min in row & max in col" << endl;

cout << setw(4) << "No" << setw(6) << "Row" << setw(6) << "Col" << endl; int No = 0;

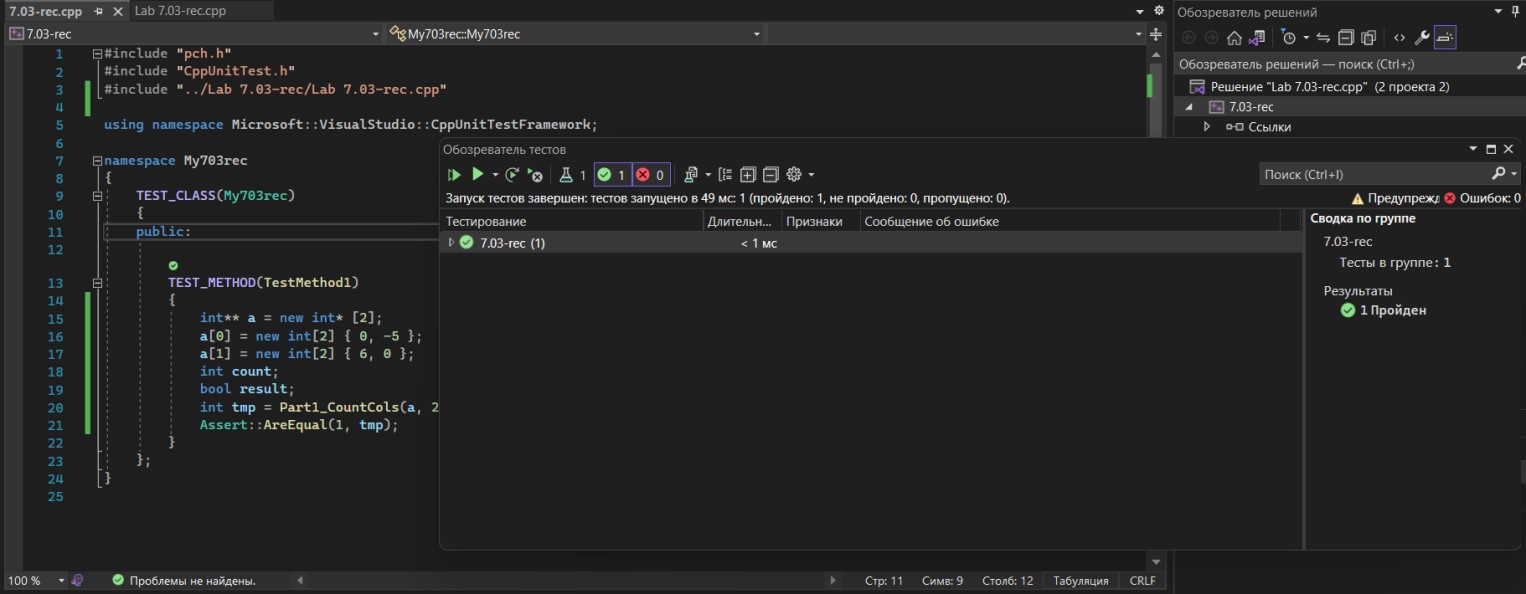
Part2\_SaddlePointRows(a, rowCount, colCount, No, 0); cout << endl;

for (int i = 0; i < rowCount; i++) delete[] a[i];

delete[] a; return 0;

}

# Результати unit-тесту:



**Текст unit-тесту:**

#include "pch.h" #include "CppUnitTest.h"

#include "../Lab 7.03-rec/Lab 7.03-rec.cpp"

using namespace Microsoft::VisualStudio::CppUnitTestFramework; namespace My703rec

{

TEST\_CLASS(My703rec)

{

public:

TEST\_METHOD(TestMethod1)

{

int\*\* a = new int\* [2];

a[0] = new int[2] { 0, -5 };

a[1] = new int[2] { 6, 0 }; int count;

bool result;

int tmp = Part1\_CountCols(a, 2, 2, count, result, 0); Assert::AreEqual(1, tmp);

}

};

}

**Посилання**: <https://github.com/milianets20/Lab-7.2-7.3>

**Висновок**: виконуючи лабораторну роботу я навчився опрацьовувати динамічні двовимірні та багатовимірні масиви.