Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій



Звіт

про виконання лабораторної роботи №7.3

Опрацювання динамічних багатовимірних масивів

з дисципліни «Алгоритмізація та програмування»

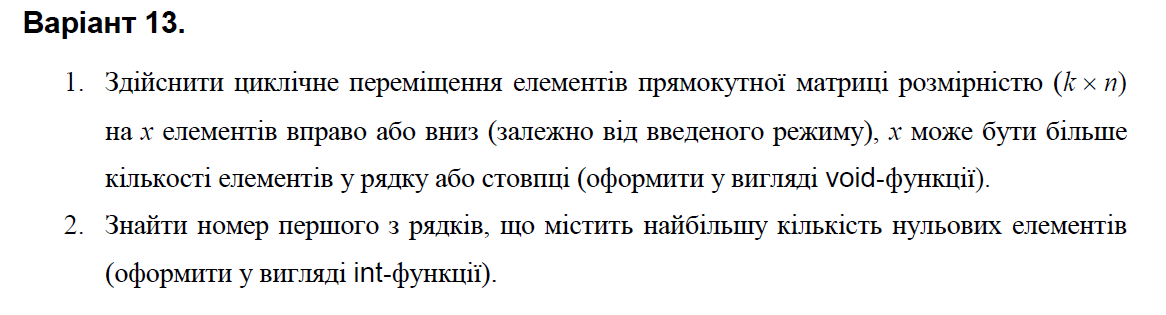
студента IT-11 Лісничука Арсена

Прийняв доцент Григорович В.Г.

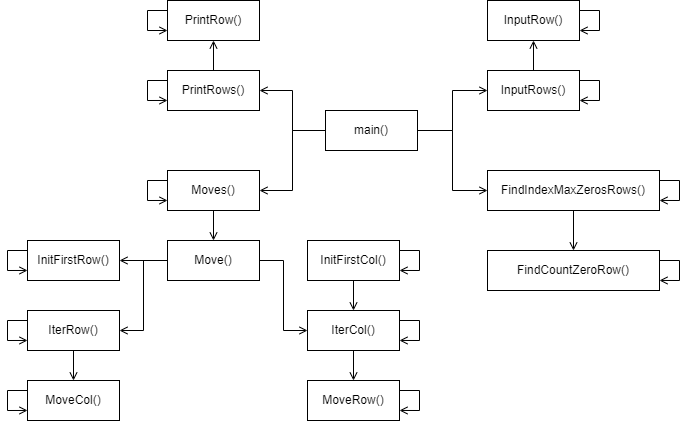
2021

Мета: Навчитися опрацьовувати динамічні двовимірні та багатовимірні масиви.

Умова завдання:



Структурна схема програми:



Текст програми:

// Main.cpp

// Лісничук Арсен

// Лабораторна робота № 7.3.2

// Опрацювання динамічних багатовимірних масивів

// Варіант 13(Рекурсивний спосіб)

#include <iostream>

#include <iomanip>

using namespace std;

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

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

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

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

void InitFirstRow(int\*\* arr, int\* tmp, const int colCount, int colNo);

void IterRow(int\*\* arr, const int rowCount, int rowNo, const int colCount, int\* tmp);

void MoveCol(int\*\* arr, const int rowCount, const int rowNo, const int colCount, int colNo, int\* tmp);

void InitFirstCol(int\*\* arr, int\* tmp, const int rowCount, int rowNo);

void IterCol(int\*\* arr, const int rowCount, const int colCount, int colNo, int\* tmp);

void MoveRow(int\*\* arr, const int rowCount, const int rowNo, const int colCount, int colNo, int\* tmp);

void Move(int\*\* arr, const int rowCount, const int colCount, int mode);

// mode - напрям: 1 - вправо, 2 - вниз

// steps - крок(на скільки елементів перемістити)

void Moves(int\*\* arr, const int rowCount, const int colCount, const int mode, const int steps, int s);

int FindCountZeroRow(int\*\* arr, const int rowNo, const int colCount, int colNo);

int FindIndexMaxZerosRows(int\*\* arr, const int rowCount, const int colCount, int max\_zeros, int index\_max\_zeros, int rowNo);

int main()

{

srand((unsigned)time(NULL));

int n, k;

cout << "k(width) = "; cin >> k;

cout << "n(height) = "; cin >> n;

int\*\* arr = new int\* [n];

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

arr[i] = new int[k];

InputRows(arr, n, k, 0);

PrintRows(arr, n, k, 0);

while (true) {

int steps, mode;

cout << "mode(1=down, 2=right, ?=down, 0=exit) = "; cin >> mode;

if (mode == 0) break;

cout << "steps = "; cin >> steps;

Moves(arr, n, k, mode, steps, 0);

PrintRows(arr, n, k, 0);

}

int index\_max\_null = FindIndexMaxZerosRows(arr, n, k, 0, -1, 0);

if (index\_max\_null != -1)

cout << "index row with maximum count of zeros = " << index\_max\_null << endl;

else

cout << "there are no zeros " << endl;

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

delete[] arr[i];

delete[] arr;

return 0;

}

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

{

cout << "arr[" << rowNo << "][" << colNo << "] = ";

cin >> arr[rowNo][colNo];

if (colNo < colCount - 1)

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

else

cout << endl;

}

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

{

InputRow(arr, rowNo, colCount, 0);

if (rowNo < rowCount - 1)

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

else

cout << endl;

}

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

{

cout << setw(4) << arr[rowNo][colNo];

if (colNo < colCount - 1)

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

else

cout << endl;

}

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

{

PrintRow(arr, rowNo, colCount, 0);

if (rowNo < rowCount - 1)

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

else

cout << endl;

}

void InitFirstRow(int\*\* arr, int\* tmp, const int colCount, int colNo)

{

if (colNo < colCount) {

tmp[colNo] = arr[0][colNo];

InitFirstRow(arr, tmp, colCount, colNo + 1);

}

}

void IterRow(int\*\* arr, const int rowCount, int rowNo, const int colCount, int\*tmp)

{

if (rowNo < rowCount) {

MoveCol(arr, rowCount, rowNo, colCount, 0, tmp);

IterRow(arr, rowCount, rowNo + 1, colCount, tmp);

}

}

void MoveCol(int\*\* arr, const int rowCount, const int rowNo, const int colCount, int colNo, int\* tmp)

{

if (colNo < colCount) {

if (rowNo == rowCount - 1)

arr[0][colNo] = tmp[colNo];

else {

int tmp1 = arr[rowNo + 1][colNo];

arr[rowNo + 1][colNo] = tmp[colNo];

tmp[colNo] = tmp1;

}

MoveCol(arr, rowCount, rowNo, colCount, colNo + 1, tmp);

}

}

void InitFirstCol(int\*\* arr, int\* tmp, const int rowCount, int rowNo)

{

if (rowNo < rowCount) {

tmp[rowNo] = arr[rowNo][0];

InitFirstCol(arr, tmp, rowCount, rowNo + 1);

}

}

void IterCol(int\*\* arr, const int rowCount, const int colCount, int colNo, int\* tmp)

{

if (colNo < colCount) {

MoveRow(arr, rowCount, 0, colCount, colNo, tmp);

IterCol(arr, rowCount, colCount, colNo + 1, tmp);

}

}

void MoveRow(int\*\* arr, const int rowCount, const int rowNo, const int colCount, int colNo, int\* tmp)

{

if (rowNo < rowCount)

{

if (colNo == colCount - 1)

arr[rowNo][0] = tmp[rowNo];

else {

int tmp1 = arr[rowNo][colNo + 1];

arr[rowNo][colNo + 1] = tmp[rowNo];

tmp[rowNo] = tmp1;

}

MoveRow(arr, rowCount, rowNo+1, colCount, colNo, tmp);

}

}

void Move(int\*\* arr, const int rowCount, const int colCount, int mode)

{

if (mode != 1 && mode != 2)

mode = 1;

if (mode == 1)

{

int\* tmp = new int[colCount];

InitFirstRow(arr, tmp, colCount, 0);

IterRow(arr, rowCount, 0, colCount, tmp);

delete[] tmp;

}

if (mode == 2)

{

int\* tmp = new int[rowCount];

InitFirstCol(arr, tmp, rowCount, 0);

IterCol(arr, rowCount, colCount, 0, tmp);

delete[] tmp;

}

}

void Moves(int\*\* arr, const int rowCount, const int colCount, const int mode, const int steps, int s)

{

if (s < steps) {

Move(arr, rowCount, colCount, mode);

Moves(arr, rowCount, colCount, mode, steps, s + 1);

}

}

int FindCountZeroRow(int\*\* arr, const int rowNo, const int colCount, int colNo)

{

if (colNo < colCount) {

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

return 1 + FindCountZeroRow(arr, rowNo, colCount, colNo + 1);

else

return FindCountZeroRow(arr, rowNo, colCount, colNo + 1);

}

else return 0;

}

int FindIndexMaxZerosRows(int\*\* arr, const int rowCount, const int colCount, int max\_zeros, int index\_max\_zeros, int rowNo)

{

if (rowNo < rowCount) {

int zeros = FindCountZeroRow(arr, rowNo, colCount, 0);

if (zeros > max\_zeros)

{

max\_zeros = zeros;

index\_max\_zeros = rowNo;

}

return FindIndexMaxZerosRows(arr, rowCount, colCount, max\_zeros, index\_max\_zeros, rowNo+1);

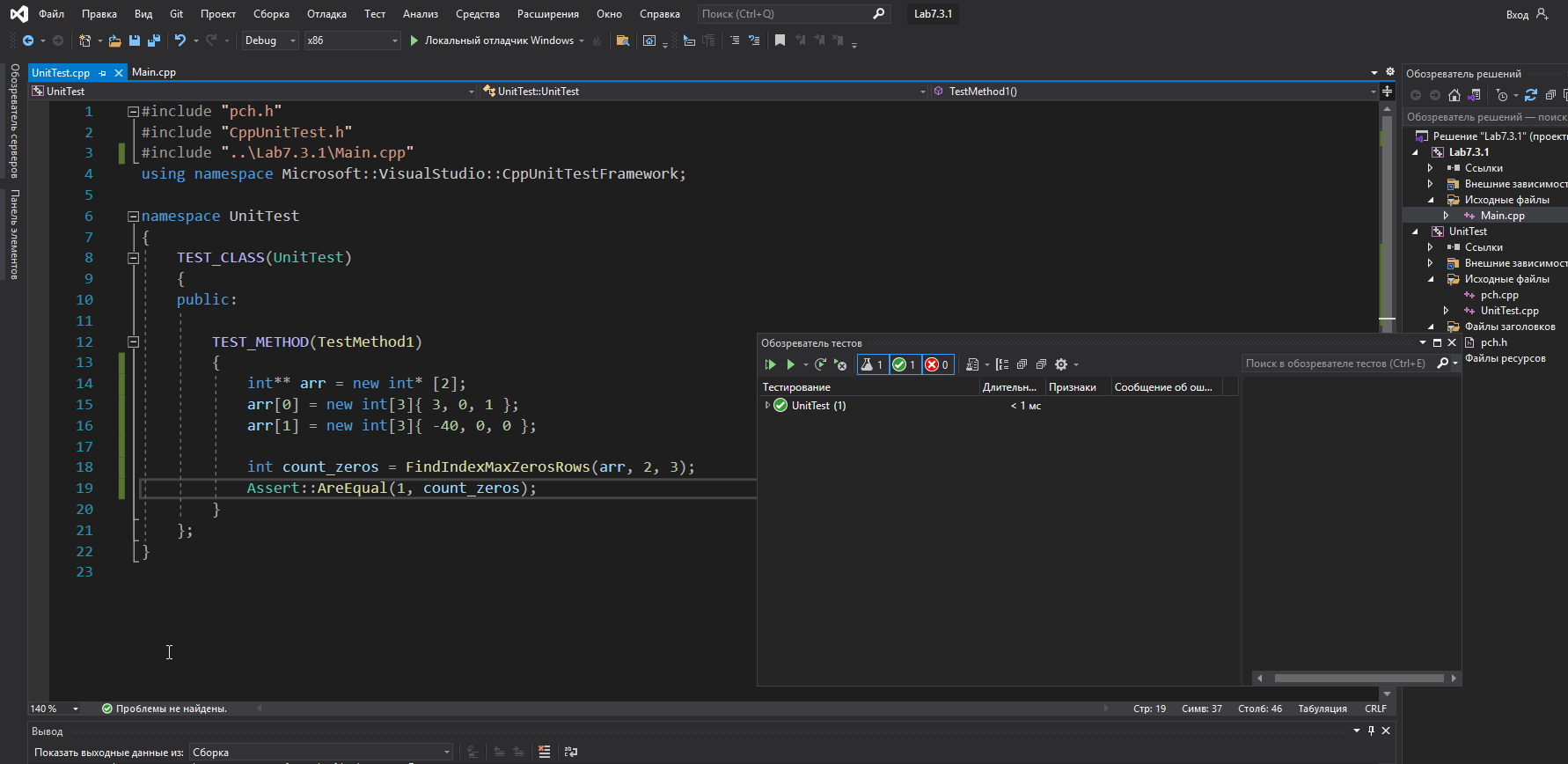
}

else return index\_max\_zeros;

}

Git-посилання: <https://github.com/fhctyw/Lab7.3.git>

Юніт тест:



Висновок: В цій лабораторній я навчитися опрацьовувати динамічні двовимірні та багатовимірні масиви.