МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Рекурсия и головоломки

(наименование темы проекта или работы)

ОТЧЕТ

к лабораторной работе №4

по дисциплине

Технологии программирования

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Капранов С.Н..

(подпись) (фамилия, и.,о.)

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Толстова М.Г.

(подпись) (фамилия, и.,о.)

18-ИСТ-4

(шифр группы)

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород 2020

Вариант 19

Задание:

Реализовать игру «Небоскрёбы».

**Небоскрёбы**  – вид логической головоломки. В ней требуется заполнить числами квадрат так, чтобы в каждой строке и в каждом столбце каждое число использовалось лишь единожды. Заполнение нужно делать с помощью ключевых чисел, считая, что в каждой клетке внутри квадрата стоит дом с количеством этажей, равном стоящему в клетке числу, а числа внутри поля означают, сколько домов видно в соответствующей строке или столбце, если смотреть от этого числа.

Листинг

Laba4.cpp

#include "stdafx.h"

#include <iostream>

#include <ctime>

#define N 4

using namespace std;

void show(int\*\*);

void show(int\*);

void changeColumn(int\*\*, int, int);

void changeRow(int\*\*, int, int);

void generateMap(int\*\*);

void setIndex(int\*\*, int\*, int\*, int\*, int\*); //цифры по краям

void setRow(int\*, int\*, int\*\*);

void setColumn(int\*, int\*, int\*\*);

void startGame();

void showMap(int\*\*, int\*, int\*, int\*, int\*);

int check(int\*\*);

int check(int\*\*, int\*\*);

int main()

{

srand(time(NULL));

setlocale(LC\_ALL, "Russian");

startGame();

system("pause");

return 0;

}

//создаем карту

void generateMap(int\*\* arr)

{

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

arr[i] = new int[N];

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

{

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

{

arr[i][j] = ((i + j) % N) + 1;

}

}

int repeat = 10;

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

changeRow(arr, rand() % N, rand() % N);

changeColumn(arr, rand() % N, rand() % N);

}

}

//Игровое поле

void startGame()

{

int\*\* startMap = new int\*[N];

generateMap(startMap);

int\*\* userMap = new int\*[N];

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

userMap[i] = new int[N];

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

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

userMap[i][j] = 0;

int\* left = new int[N];

int\* top = new int[N];

int\* right = new int[N];

int\* bottom = new int[N];

setIndex(startMap, left, top, right, bottom);

int x, y, value;

while (true)

{

showMap(userMap, left, top, right, bottom);

cout << endl << "(Чтобы сдаться нажмите 0)" << endl;

cout << "Введите координаты ячейки: (максимум " << N << ")" << endl;

while (!(cin >> x) || x < 1 || x > N)

{

cout << "Введен некорректный индекс. Введите верные значения" << endl;

cout << "Введите координаты ячейки: (максимум " << N << ")" << endl;

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

while (!(cin >> y) || y < 1 || y > N)

{

cout << "Введен некорректный индекс. Введите верные значения" << endl;

cout << "Введите координаты ячейки: (максимум " << N << ")" << endl;

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

cout << "Введите значение: ";

while (!(cin >> value) || value < 1 || value > N)

{

cout << "Введено некорректное значение. Введите верное значение" << endl;

cout << "Введите значение: (максимум " << N << ")" << endl;

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

userMap[x - 1][y - 1] = value;

if (check(userMap))

{

if (check(userMap, startMap))

{

cout << "Поздравляем. Вы прошли игру!" << endl;

break;

}

else

{

cout << "Результат неверен. Подумайте еще." << endl;

system("pause");

}

}

system("cls");

}

}

//выбор позиции на карте

void changeRow(int\*\* arr, int m, int n)

{

int buf;

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

{

buf = arr[m][i];

arr[m][i] = arr[n][i];

arr[n][i] = buf;

}

}

void changeColumn(int\*\* arr, int m, int n)

{

int buf;

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

{

buf = arr[i][m];

arr[i][m] = arr[i][n];

arr[i][n] = buf;

}

}

// индекс - цифры по краям, setRow и setColumn для того, чтобы получить место цифры

void setIndex(int\*\* arr, int\* left, int\* top, int\* right, int\* bottom)

{

setRow(left, right, arr);

setColumn(top, bottom, arr);

}

void setRow(int\* vec, int\* vecRev, int\*\* arr)

{

int max;

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

{

max = arr[i][0];

vec[i] = 1;

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

{

if (arr[i][j] > max)

{

vec[i]++;

max = arr[i][j];

}

}

max = arr[i][N - 1];

vecRev[i] = 1;

for (int j = N - 1; j >= 0; j--)

{

if (arr[i][j] > max)

{

vecRev[i]++;

max = arr[i][j];

}

}

}

}

void setColumn(int\* vec, int\* vecRev, int\*\* arr)

{

int max;

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

{

max = arr[0][i];

vec[i] = 1;

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

{

if (arr[j][i] > max)

{

vec[i]++;

max = arr[j][i];

}

}

max = arr[N - 1][i];

vecRev[i] = 1;

for (int j = N - 1; j >= 0; j--)

{

if (arr[j][i] > max)

{

vecRev[i]++;

max = arr[j][i];

}

}

}

}

//вывод карты

void showMap(int\*\* userMap, int\* left, int\* top, int\* right, int\* bottom)

{

cout << "\t";

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

cout << top[i] << "\t";

cout << endl << endl;

for (int k = 0; k < N; k++)

{

cout << left[k] << "\t";

show(userMap[k]);

cout << right[k] << endl << endl;

}

cout << endl << "\t";

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

cout << bottom[i] << "\t";

cout << endl;

}

// 2 проверки, одна проверяет на заполненность матрицы, другая сравнивает с оригиналом

int check(int\*\* userMap)

{

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

{

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

{

if (userMap[i][j] == 0)

return 0;

}

}

return 1;

}

int check(int\*\* userMap, int\*\* startMap)

{

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

{

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

{

if (userMap[i][j] != startMap[i][j])

return 0;

}

}

return 1;

}

void show(int\*\* arr)

{

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

{

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

{

cout << arr[i][j] << "\t";

if (j == N - 1)

cout << endl;

}

}

}

// 2 show, один для карты, другой для цифр по краям

void show(int\* vec) {

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

if (vec[i] != 0)

cout << vec[i] << "\t";

else

cout << "\*\t";

}