МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

Лабораторная работа №4 Рекурсия и головоломки

(наименование работы)

ОТЧЕТ

по лабораторной работе

по дисциплине

Технологии программирования

(наименование дисциплины)

РУКОВОДИТЕЛЬ:

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

(подпись) (фамилия, и., о.)

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Савченков Н.В.

(подпись) (фамилия, и., о.)

18-ИСТ-4

(шифр группы)

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

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

Нижний Новгород, 2020

Оглавление

[Задача 3](#_Toc35875137)

[Среда разработки 3](#_Toc35875138)

[Код программы 3](#_Toc35875139)

[Результат работы программы 4](#_Toc35875140)

# Задача

Вариант 17 – Нумератор

# Среда разработки

Программа написана в среде разработки Qt Creator при помощи фреймворка Qt

## Код программы

**matrix.h**

#ifndef MATRIX\_H

#define MATRIX\_H

#include <vector>

struct **Size**

{

int width,height;

};

class **Matrix**

{

public:

**Matrix**(int \_size);

int \***take**(int i, int j);

void **resize**(int \_m, int \_n);

int **size**();

Matrix &operator=(const Matrix&);

private:

std::vector<std::vector<int>> matrix;

int m,n;

};

#endif // MATRIX\_H

**matrix.cpp**

#include "matrix.h"

#include <vector>

//Конструктор матрицы

Matrix::**Matrix**(int \_size)

{

resize(\_size,\_size);

}

//Метод получения значения по индексам

int \*Matrix::**take**(int i, int j)

{

//Если индексы в пределах матрицы возвращаем результат

if((i > -1 && j > -1) && (i < m && j < n))

return &matrix[i][j];

//Если нет, возвращаем пустой указатель

return nullptr;

}

//Метод изменяет размер векторов

//в соответствии с размером матрицы

void Matrix::**resize**(int \_m, int \_n)

{

//Запоминаем размеры

n = \_n;

m = \_m;

//Изменяем вектора

matrix.resize(\_m);

for (int i(0); i < \_n;i++)

{

matrix[i].resize(\_n);

}

}

//Метод возращающий размер матрицы

int Matrix::**size**()

{

return m;

}

//Перегрузка оператора присваивания

Matrix &Matrix::operator=(const Matrix& \_original)

{

n = \_original.n;

m = \_original.m;

resize(m,n);

matrix = \_original.matrix;

return \*this;

}

**main.cpp**

#include "matrix.h"

#include <iostream>

#include <list>

#include <string>

#include <fstream>

#include <QStringList>

#define SIZE 6

using namespace std;

//Метод рекурсионого решения задачи

bool **recurse**(Matrix &m, int x, int y, int deph)

{

//Если вышли за поле

if(x < 0 || y < 0 || x >= m.size() || y >= m.size())

return false;

bool flagIsNotDelete = false;

//Если клетка не пустая

if(\*m.take(x,y) > 0)

{

flagIsNotDelete = true;

if(\*m.take(x,y) != deph)

return false;

}

//Если это последний элемент

if(deph == m.size() \* m.size())

{

\*m.take(x,y) = deph;

return true;

}

//Заполняем клетку

\*m.take(x,y) = deph;

//Рекурсивно идем далее

if(recurse(*m*,x+1,y,deph + 1))

return true;

if(recurse(*m*,x-1,y,deph + 1))

return true;

if(recurse(*m*,x,y+1,deph + 1))

return true;

if(recurse(*m*,x,y-1,deph + 1))

return true;

//Если все безуспешно

if(!flagIsNotDelete)

\*m.take(x,y) = 0;

return false;

}

//Метод печати матрицы

void **printMatrix**(Matrix &m)

{

for (int x(0); x < m.size(); x++)

{

cout << "-|";

for (int y(0); y < m.size(); y++)

{

if(\*m.take(x,y) < 10)

cout << " " << \*m.take(x,y) << " |";

else

cout << " " << \*m.take(x,y) << " |";

}

cout << endl;

for(int line(0); line < m.size(); line++) cout << "------";

cout << endl;

}

cout << endl << endl;

}

//Метод чтения первичных элементов из файла

bool **readListFromFile**(Matrix &m)

{

ifstream startDate("Start.txt");

string line;

if(startDate.is\_open())

{

cout << "File open, loading" << endl;

while(getline(startDate,*line*))

{

QString str;

str = str.fromStdString(line);

QStringList list = str.split(',');

if(list.size() < 3 || list.size() > 3)

{

cout << "Problem with the number of items in a row." << endl;

return false;

}

bool ok = true;

\*m.take(list[0].toInt(&ok) - 1,list[1].toInt(&ok) - 1) = list[2].toInt(&ok);

if(!ok)

{

cout << "Probltm with the data." << endl;

return false;

}

}

startDate.close();

}

else

{

cout << "Problem with file" << endl;

}

return true;

}

int **main**(int argc, char \*argv[])

{

Matrix m(SIZE);

for (int x(0); x < m.size(); x++)

for (int y(0); y < m.size(); y++)

\*m.take(x,y) = 0;

//printMatrix(m);

if(!readListFromFile(*m*))

return 1;

printMatrix(*m*);

for (int x(0); x < SIZE;x++)

{

for(int y(0); y < SIZE; y++)

{

if(\*m.take(x,y) == 0 || \*m.take(x,y) == 1)

recurse(*m*,x,y,1);

}

}

cout<<"------RESULT--------"<<endl;

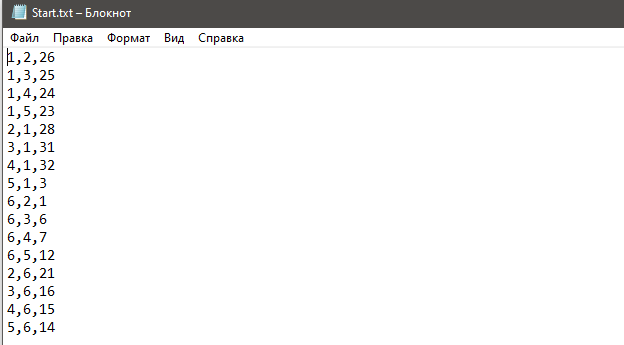
printMatrix(*m*);

return 0;

}

## Результат работы программы

Рис. 1. Результат работы программы

Рис. 2. Файл стартовых значений, где первая цифра – X, вторая – Y, третье - значение