Министерство образования и науки Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

«**Пермский национальный исследовательский политехнический университет**»

Кафедра «Информационные технологии и автоматизированные системы»

ОТЧЁТ

по лабораторной работе №21

Тема: Графы. Алгоритм Дейкстры.

Вариант 15

Выполнил:

Студент группы ИВТ-20-2б

Карелов Вадим Андреевич

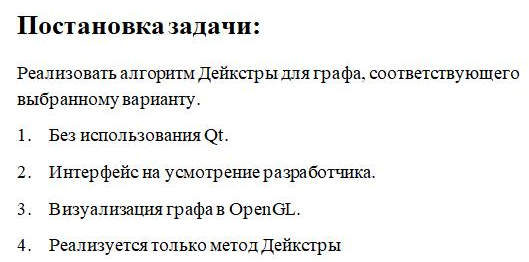
Проверила:

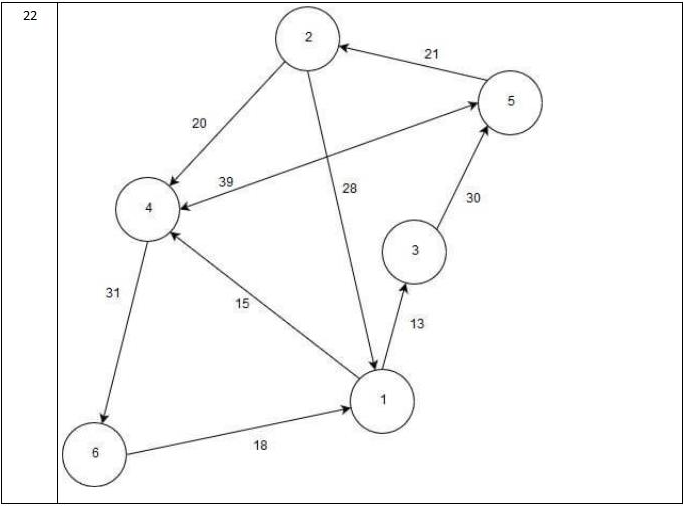
Доцент кафедры ИТАС

Полякова О. А.

Пермь, 2021

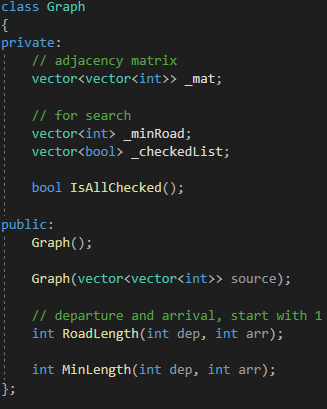
**Постановка задачи**



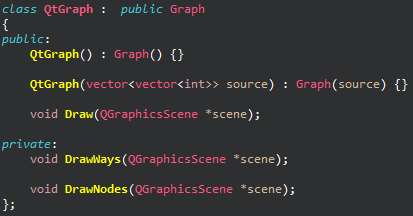


**Анализ задачи**

1. Используемые типы данных
   1. double
   2. Int
   3. Graph
   4. Vector<T>
2. Действия над используемыми данными
   1. Int, double – арифметические операции.
3. Вид данных.
   1. Vector<T> – динамический одномерный массив.
   2. Graph – граф, содержащий матрицу смежности.
4. Структуры
   1. Graph – класс реализации графа и алгоритма дейкстры для поиска минимального пути.



* 1. QtGraph – класс, наследник Graph. Содержит методы отрисовки класса в qt.



1. Организация ввода-вывода
   1. Cin
   2. Cout
   3. Отрисовка экземпляра класса QtGraph в qt на виджете graphicsView.
2. Действия для решения задачи.
   1. Для создания графического интерфейса использована IDE QtCreator.

**Код программы**

**Консольная программа**

**Main.cpp**

#include <iostream>

#include "Graph.h"

using namespace std;

int main()

{

Graph g;

cout << g.MinLength(5, 1);

}

**Graph.h**

#pragma once

#include <vector>

using namespace std;

class Graph

{

private:

// adjacency matrix

vector<vector<int>> \_mat;

// for search

vector<int> \_minRoad;

vector<bool> \_checkedList;

bool IsAllChecked();

public:

Graph();

Graph(vector<vector<int>> source);

// departure and arrival, start with 1

int RoadLength(int dep, int arr);

int MinLength(int dep, int arr);

};

**Graph.cpp**

#include "Graph.h"

Graph::Graph()

{

\_mat = { {0, 0, 13, 15, 0, 0},

{28, 0, 0, 20, 0, 0},

{0, 0, 0, 0, 30, 0},

{0, 0, 0, 0, 39, 31},

{0, 21, 0, 39, 0, 0},

{18, 0, 0, 0, 0, 0} };

}

Graph::Graph(vector<vector<int>> source)

{

if (source.size() > 0 && source.size() == source[0].size())

\_mat = source;

}

int Graph::RoadLength(int dep, int arr)

{

dep--;

arr--;

if (arr >= 0 && arr < \_mat.size() && dep >= 0 && dep < \_mat.size())

{

return \_mat[dep][arr];

}

else

{

return 0;

}

}

int Graph::MinLength(int dep, int arr)

{

if (dep == arr) return 0;

dep--;

arr--;

// preparing

\_minRoad.clear();

\_checkedList.clear();

for (int i = 0; i < \_mat.size(); i++)

{

\_minRoad.push\_back(10000);

\_checkedList.push\_back(false);

}

// starting from departure node

// road from dep to dep will be infinity

for (int i = 0; i < \_mat.size(); i++)

{

if (RoadLength(dep + 1, i + 1) > 0)

{

\_minRoad[i] = min(\_minRoad[i], RoadLength(dep + 1, i + 1));

}

}

\_checkedList[dep] = true;

// for avoid looping

int k = \_mat.size() \* 2;

// now check other elements

while (!IsAllChecked() && k > 0)

{

k--;

for (int i = 0; i < \_mat.size(); i++)

{

if (!\_checkedList[i])

{

for (int j = 0; j < \_mat.size(); j++)

{

if (RoadLength(i + 1, j + 1) > 0)

{

\_minRoad[j] = min(\_minRoad[j], \_minRoad[i] + RoadLength(i + 1, j + 1));

}

}

\_checkedList[i] = true;

}

}

}

return \_minRoad[arr];

}

bool Graph::IsAllChecked()

{

for (int i = 0; i < \_checkedList.size(); i++)

{

if (!\_checkedList[i]) return false;

}

return true;

}

**Код из QtCreator**

**Main.cpp**

#include "mainwindow.h"

#include <QApplication>

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

{

QApplication a(*argc*, argv);

MainWindow w;

w.show();

*return* a.exec();

}

**Mainwindow.h**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QMessageBox>

#include "QtGraph.h"

*using* *namespace* std;

QT\_BEGIN\_NAMESPACE

*namespace* **Ui** { *class* **MainWindow**; }

QT\_END\_NAMESPACE

*class* **MainWindow** : *public* QMainWindow

{

Q\_OBJECT

QtGraph \*myGraph;

QGraphicsScene \*scene = *new* QGraphicsScene(*this*);

*public*:

**MainWindow**(QWidget \*parent = *nullptr*);

~***MainWindow***();

*private* slots:

void **on\_pushButton\_Compute\_clicked**();

*private*:

Ui::MainWindow \*ui;

bool **CheckFields**();

int **GetInt**(string text);

};

#endif *//* *MAINWINDOW\_H*

**Mainwindow.cpp**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

MainWindow::**MainWindow**(QWidget \*parent)

: QMainWindow(parent)

, ui(*new* Ui::MainWindow)

{

ui->setupUi(*this*);

*//* *create* *scene*

scene = *new* QGraphicsScene();

scene->setSceneRect(0, 0, ui->graphicsView->width(), ui->graphicsView->height());

ui->graphicsView->setScene(scene);

scene->clear();

*//* *create* *graph*

myGraph = *new* QtGraph();

myGraph->Draw(scene);

}

MainWindow::~***MainWindow***()

{

*delete* ui;

}

*//* *slots*

void MainWindow::**on\_pushButton\_Compute\_clicked**()

{

*if* (CheckFields())

{

*//* *drawing*

scene->clear();

myGraph->Draw(scene);

*//* *show* *message*

QMessageBox::information(*this*, "Вычисления", "Путь = " +

QString::number(myGraph->MinLength(GetInt(ui->lineEdit\_dep->text().toStdString()),

GetInt(ui->lineEdit\_arr->text().toStdString()))));

}

}

*//* *methods* *to* *checking* *fields*

bool MainWindow::**CheckFields**()

{

*if* (ui->lineEdit\_arr->text() == "" && GetInt(ui->lineEdit\_arr->text().toStdString()) <= 0

&& GetInt(ui->lineEdit\_arr->text().toStdString()) >= myGraph->Size())

{

QMessageBox::warning(*this*, "Ошибка","Ошибка пункта прибытия");

*return* *false*;

}

*if* (ui->lineEdit\_dep->text() == "" && GetInt(ui->lineEdit\_dep->text().toStdString()) <= 0

&& GetInt(ui->lineEdit\_dep->text().toStdString()) >= myGraph->Size())

{

QMessageBox::warning(*this*, "Ошибка","Ошибка пункта отправления");

*return* *false*;

}

*return* *true*;

}

int MainWindow::**GetInt**(string text)

{

int res = -1;

*try*

{

res = stoi(text);

}

*catch* (...)

{

}

*return* res;

}

#ifndef QTGRAPH\_H

#define QTGRAPH\_H

#include <QColor>

#include <QFile>

#include <QFont>

#include <QGraphicsItem>

#include <QGraphicsScene>

#include <QPainter>

#include <QString>

#include <QtCore/qmath.h>

#include <QTextStream>

#include "Graph.h"

*class* **QtGraph** : *public* Graph

{

*public*:

**QtGraph**() : Graph() {}

**QtGraph**(vector<vector<int>> source) : Graph(source) {}

void **Draw**(QGraphicsScene \*scene);

*private*:

void **DrawWays**(QGraphicsScene \*scene);

void **DrawNodes**(QGraphicsScene \*scene);

};

#endif *//* *QTGRAPH\_H*

**QtGraph.cpp**

#include "QtGraph.h"

void QtGraph::**Draw**(QGraphicsScene \*scene)

{

DrawWays(scene);

DrawNodes(scene);

}

void QtGraph::**DrawWays**(QGraphicsScene \*scene)

{

double step = 2 \* 3.14 / \_mat.size();

int centerX = scene->width() / 2;

int centerY = scene->height() / 2;

int r = 15;

int l = 170;

*for* (int i = 0; i < \_mat.size(); i++)

{

*for* (int j = 0; j < \_mat.size(); j++)

{

*if* (RoadLength(i + 1, j + 1) > 0)

{

int x1 = centerX + sin(step \* i) \* l;

int x2 = centerX + sin(step \* j) \* l;

int y1 = centerY + cos(step \* i) \* l;

int y2 = centerY + cos(step \* j) \* l;

QGraphicsItem \*ell = scene->addLine(x1, y1, x2, y2);

scene->addItem(ell);

*//* *draw* *text*

QFont font("Times");

QGraphicsTextItem \*txt = scene->addText(QString::number(RoadLength(i + 1, j + 1)), font);

txt->setPos(x2 + (x1 - x2) / 8, y2 + (y1 - y2) / 8);

}

}

}

}

void QtGraph::**DrawNodes**(QGraphicsScene \*scene)

{

double step = 2 \* 3.14 / \_mat.size();

int centerX = scene->width() / 2;

int centerY = scene->height() / 2;

int r = 15;

int l = 170;

*for* (int i = 0; i < \_mat.size(); i++)

{

QGraphicsItem \*ell = scene->addEllipse(centerX + sin(step \* i) \* l - r, centerY + cos(step \* i) \* l - r,

2 \* r, 2 \* r,

QPen(Qt::*black*), QBrush(Qt::*green*));

scene->addItem(ell);

*//* *draw* *text*

QFont font("Times");

QGraphicsTextItem \*txt = scene->addText(QString::number(i + 1), font);

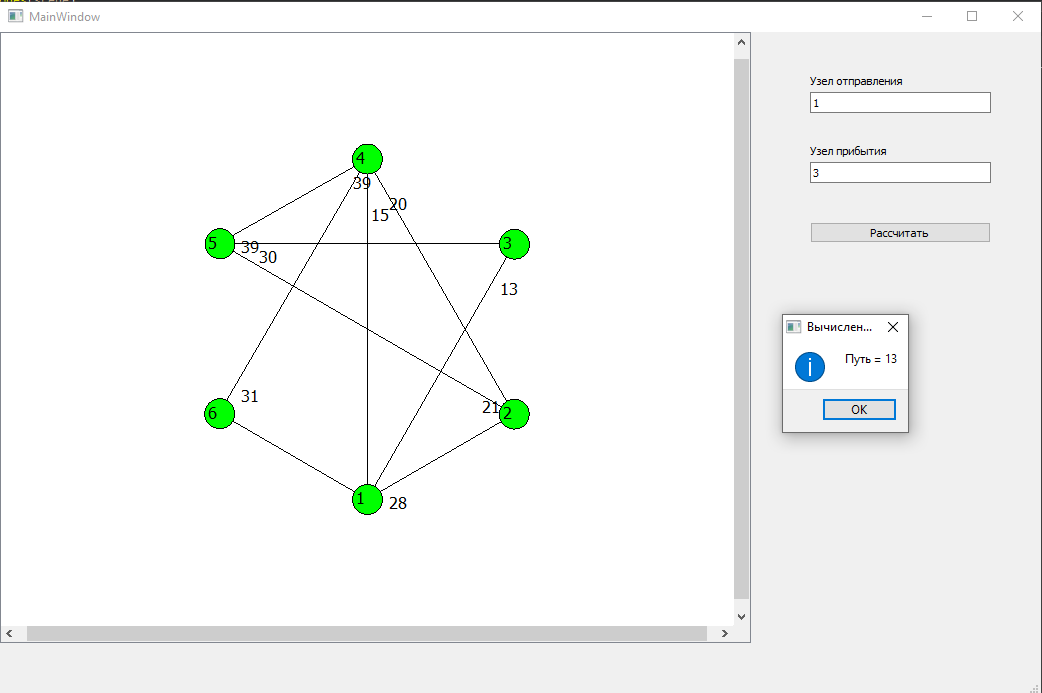
txt->setPos(centerX + sin(step \* i) \* l - r, centerY + cos(step \* i) \* l - r);

}

}

**Скриншоты**





**UML-диаграмма**

