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**

Дисциплина: «Основы алгоритмизации и программирования»

Тема: “Графы”

Вариант 15

Выполнил:

Студент группы ИВТ-20-2Б Чувашев Максим

Проверила:

Доцент кафедры ИТАС

Полякова О.А.

Пермь, 2021

**Постановка задачи**

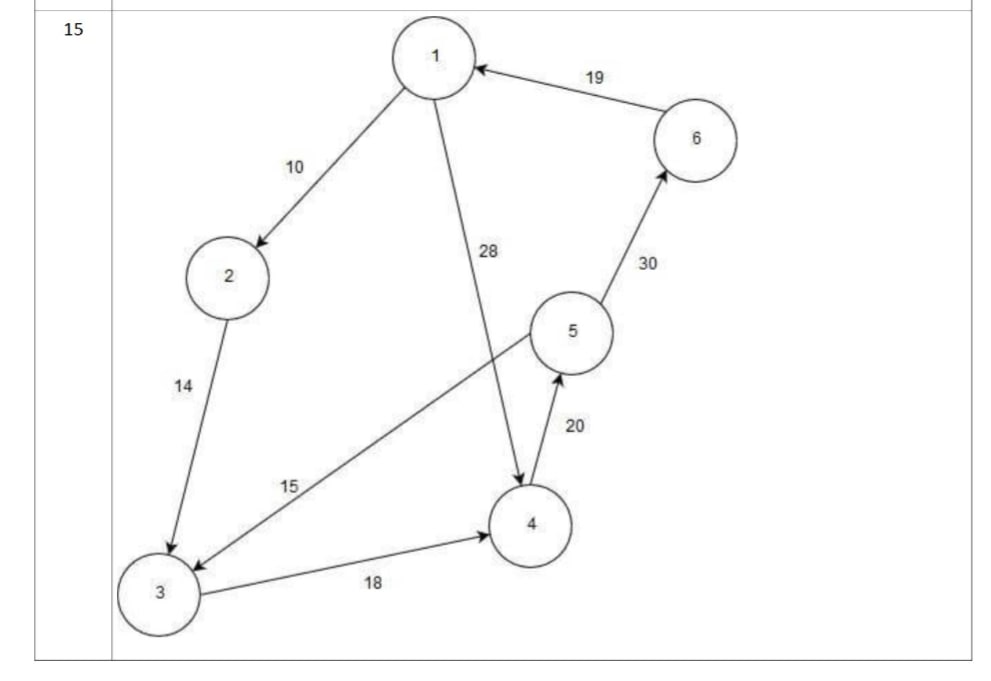
Реализовать алгоритм Дейкстры для графа, соответствующего выбранному варианту.

1. Без использования Qt.

2. Интерфейс на усмотрение разработчика.

3. Визуализация графа в OpenGL.

4. Реализуется только метод Дейкстры



**Анализ задачи**

1. Для решения задач необходимо:
   1. Организовать класс myGLWidget, который наследуется от класса QOpenGLWidget
   2. Организовать класс mypainter, которой будет отвечать за визуализацию графа
   3. Организовать функцию Dejkstra(), чтобы вывести кратчайшие пути до вершин в консоли.
2. В ходе работы были использованы следующие типы данных:
   1. Класс myPainterс необходимыми методами для отрисовки линий, узлов и самого графа в форму. В качестве поля используется двумерный вектор matrix.

#ifndef MYPAINTER\_H

#define MYPAINTER\_H

#include <QBrush>

#include <QFont>

#include <QPen>

#include <QWidget>

class myPainter

{

public:

myPainter();

public:

void draw(QPainter \*painter, QPaintEvent \*event);

QBrush front;

private:

QBrush back;

QFont glFont;

QPen textPen;

};

#endif //MYPAINTER\_H

* 1. В главном файле main.cpp используется глобальные вектор mat, массив result, который будет использоваться для вывода ответа, массив visited для используемый для флажков.

vector <vector<int>> 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 } };

int result[6] = { INT\_MAX, INT\_MAX, INT\_MAX, INT\_MAX,INT\_MAX, INT\_MAX };

bool visited[6] = { 0, 0, 0, 0, 0, 0 };

void Dejkstra();

1. Для решения задачи данные были представлены в следующем виде:
   1. Глобальные переменные num, R, r

int num=6; //кол-во узлов в графе

int R=200; //радиус графа

int r=30; //радиус узла

1. Для операций ввода и вывода использовались следующие операторы и функции:
   1. За отрисовку отвечает класс mypainter, а именно метод drow(), данного метода, отвечает за отрисовку линий, кругов, цифр.

void myPainter::draw(QPainter \*painter, QPaintEvent \*event)

{

painter->fillRect(event->rect(),back);

painter->setBrush(front);

painter->setPen(textPen);

painter->setFont(glFont);

textPen.setWidth(2);

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

{

double a=M\_PI\*3/2+M\_PI\*2/num\*i;

int xc=cos(a)\*R+xa; //координата узла по x

int yc=sin(a)\*R+ya; //координата узла по y

int const1=xc-20; //координаты текста

int const2=yc-HeightText/2;

int const3=40;

int const4=HeightText;

QString text; text.setNum(i+1);

painter->drawEllipse(QRectF(xc-r,yc-r,2\*r,2\*r));

painter->drawText(QRect(const1,const2,const3,const4),Qt::AlignCenter,text);

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

{

if (matrix[i][k]!=0)

{

//линия

double a2=M\_PI\*3/2+M\_PI\*2/num\*k;

double xc2=cos(a2)\*R+xa;

double yc2=sin(a2)\*R+ya;

double l=sqrt((xc2-xc)\*(xc2-xc)+(yc2-yc)\*(yc2-yc));

int xl1=(xc2-xc)/l\*r+xc;

int yl1=(yc2-yc)/l\*r+yc;

int xl2=xc2-(xc2-xc)/l\*r;

int yl2=yc2-(yc2-yc)/l\*r;

painter->drawLine(xl1,yl1,xl2,yl2);

//текст

const1=(xl2+xl1)/2-20 + 10;

const2=(yl2+yl1)/2-HeightText/2 + 10;

const3=40;

const4=HeightText;

text.setNum(matrix[i][k]);

painter->drawText(QRect(const1,const2,const3,const4),Qt::AlignCenter,text);

}

}

}

}

1. Поставленные задачи будут решены следующими действиями:
   1. Функция Dejkstra() используется для вывода кратчайших путей до вершин – реализация метода Дейкстры.

void Dejkstra()

{

int top = 0; //вершина с номером 1

queue<int> line;

line.push(top);

result[0] = 0;

while (!line.empty()) {

int vertex = line.front();

line.pop();

for (int i = 0; i < mat[vertex].size(); i++) {

if (!visited[i] && mat[vertex][i] && (mat[vertex][i] + result[vertex] < result[i])) {

result[i] = mat[vertex][i] + result[vertex];

line.push(i);

}

}

}

for (int i = 0; i < 6; i++) {

cout << i + 1 << ": " << result[i] << endl;

}

}

5.2. Метод draw класса mypainter отвечает за отрисовку графа

void myPainter::draw(QPainter \*painter, QPaintEvent \*event)

{

painter->fillRect(event->rect(),back);

painter->setBrush(front);

painter->setPen(textPen);

painter->setFont(glFont);

textPen.setWidth(2);

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

{

double a=M\_PI\*3/2+M\_PI\*2/num\*i;

int xc=cos(a)\*R+xa; //координата узла по x

int yc=sin(a)\*R+ya; //координата узла по y

int const1=xc-20; //координаты текста

int const2=yc-HeightText/2;

int const3=40;

int const4=HeightText;

QString text; text.setNum(i+1);

painter->drawEllipse(QRectF(xc-r,yc-r,2\*r,2\*r));

painter->drawText(QRect(const1,const2,const3,const4),Qt::AlignCenter,text);

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

{

if (matrix[i][k]!=0)

{

//линия

double a2=M\_PI\*3/2+M\_PI\*2/num\*k;

double xc2=cos(a2)\*R+xa;

double yc2=sin(a2)\*R+ya;

double l=sqrt((xc2-xc)\*(xc2-xc)+(yc2-yc)\*(yc2-yc));

int xl1=(xc2-xc)/l\*r+xc;

int yl1=(yc2-yc)/l\*r+yc;

int xl2=xc2-(xc2-xc)/l\*r;

int yl2=yc2-(yc2-yc)/l\*r;

painter->drawLine(xl1,yl1,xl2,yl2);

//текст

const1=(xl2+xl1)/2-20 + 10;

const2=(yl2+yl1)/2-HeightText/2 + 10;

const3=40;

const4=HeightText;

text.setNum(matrix[i][k]);

painter->drawText(QRect(const1,const2,const3,const4),Qt::AlignCenter,text);

}

}

}

}

**Код**

**Mainwindow.h**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

private:

Ui::MainWindow \*ui;

};

#endif // MAINWINDOW\_H

**Myglwidget.h**

#ifndef MYGLWIDGET\_H

#define MYGLWIDGET\_H

#include "mypainter.h"

#include <QOpenGLWidget>

class myGLWidget : public QOpenGLWidget

{

Q\_OBJECT

public:

myGLWidget( QWidget \*parent);

public slots:

protected:

void paintEvent(QPaintEvent \*event) override;

private:

myPainter \*mypainter;

};

#endif // MYGLWIDGET\_H

**Mypainter.h**

#ifndef MYPAINTER\_H

#define MYPAINTER\_H

#include <QBrush>

#include <QFont>

#include <QPen>

#include <QWidget>

class myPainter

{

public:

myPainter();

public:

void draw(QPainter \*painter, QPaintEvent \*event);

QBrush front;

private:

QBrush back;

QFont glFont;

QPen textPen;

};

#endif //MYPAINTER\_H

**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.cpp**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "myglwidget.h"

#include <QPainter>

#include <QGridLayout>

#include <QPushButton>

#include <queue>

#include <iostream>

using namespace std;

vector <vector<int>> 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 } };

int result[6] = { INT\_MAX, INT\_MAX, INT\_MAX, INT\_MAX,INT\_MAX, INT\_MAX };

bool visited[6] = { 0, 0, 0, 0, 0, 0 };

void Dejkstra();

MainWindow::MainWindow(QWidget \*parent)

:QMainWindow(parent),ui(new Ui::MainWindow)

{

ui->setupUi(this);

Dejkstra();

myGLWidget \*openGLW=new myGLWidget(this);

QGridLayout \*grid=new QGridLayout();

grid->addWidget(openGLW,0,0);

ui->centralwidget->setLayout(grid);

}

MainWindow::~MainWindow()

{

delete ui;

}

void Dejkstra()

{

int top = 0; //вершина с номером 1

queue<int> line;

line.push(top);

result[0] = 0;

while (!line.empty()) {

int vertex = line.front();

line.pop();

for (int i = 0; i < mat[vertex].size(); i++) {

if (!visited[i] && mat[vertex][i] && (mat[vertex][i] + result[vertex] < result[i])) {

result[i] = mat[vertex][i] + result[vertex];

line.push(i);

}

}

}

for (int i = 0; i < 6; i++) {

cout << i + 1 << ": " << result[i] << endl;

}

}

**Myglwidget.cpp**

#include "myglwidget.h"

#include "mypainter.h"

#include <QPainter>

myGLWidget::myGLWidget(QWidget \*parent)

:QOpenGLWidget(parent)

{

mypainter=new myPainter;

}

void myGLWidget::paintEvent(QPaintEvent \*event)

{

QPainter painter;

painter.begin(this);

mypainter->draw(&painter,event);

painter.end();

}

**Mypainter.cpp**

#include "mypainter.h"

#include <QPaintEvent>

#include <QPainter>

#include <QWidget>

#include <QtCore/qmath.h>

#include <QtMath>

int num=6; //кол-во узлов в графе

int R=200; //радиус графа

int r=30; //радиус узла

int HeightText=20;

int xa=400; //отступ по вертикали

int ya=300; //отступ по горизонтали

int matrix[6][6]={ {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 } };

myPainter::myPainter()

{

// цвет для нод

front=QBrush(QColor(255,200,44));

// цвет для фона

back=QBrush(QColor(255,255,255));

// цвет для текста

textPen=QPen(QColor(0,0,0));

glFont.setPixelSize(HeightText);

}

void myPainter::draw(QPainter \*painter, QPaintEvent \*event)

{

painter->fillRect(event->rect(),back);

painter->setBrush(front);

painter->setPen(textPen);

painter->setFont(glFont);

textPen.setWidth(2);

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

{

double a=M\_PI\*3/2+M\_PI\*2/num\*i;

int xc=cos(a)\*R+xa; //координата узла по x

int yc=sin(a)\*R+ya; //координата узла по y

int const1=xc-20; //координаты текста

int const2=yc-HeightText/2;

int const3=40;

int const4=HeightText;

QString text; text.setNum(i+1);

painter->drawEllipse(QRectF(xc-r,yc-r,2\*r,2\*r));

painter->drawText(QRect(const1,const2,const3,const4),Qt::AlignCenter,text);

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

{

if (matrix[i][k]!=0)

{

//линия

double a2=M\_PI\*3/2+M\_PI\*2/num\*k;

double xc2=cos(a2)\*R+xa;

double yc2=sin(a2)\*R+ya;

double l=sqrt((xc2-xc)\*(xc2-xc)+(yc2-yc)\*(yc2-yc));

int xl1=(xc2-xc)/l\*r+xc;

int yl1=(yc2-yc)/l\*r+yc;

int xl2=xc2-(xc2-xc)/l\*r;

int yl2=yc2-(yc2-yc)/l\*r;

painter->drawLine(xl1,yl1,xl2,yl2);

//текст

const1=(xl2+xl1)/2-20 + 10;

const2=(yl2+yl1)/2-HeightText/2 + 10;

const3=40;

const4=HeightText;

text.setNum(matrix[i][k]);

painter->drawText(QRect(const1,const2,const3,const4),Qt::AlignCenter,text);

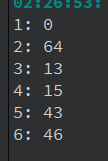
}

}

}

}

**Скриншоты**

1. 
2. 