Министерство образования и науки Российской Федерации  
Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский  
политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

**по лабораторной работе (ТЗ)**

Тема: задача коммивояжёра

Выполнил:

Студент группы РИС-20-1б

Поспелов А.А

Проверила:

Доцент кафедры ИТАС

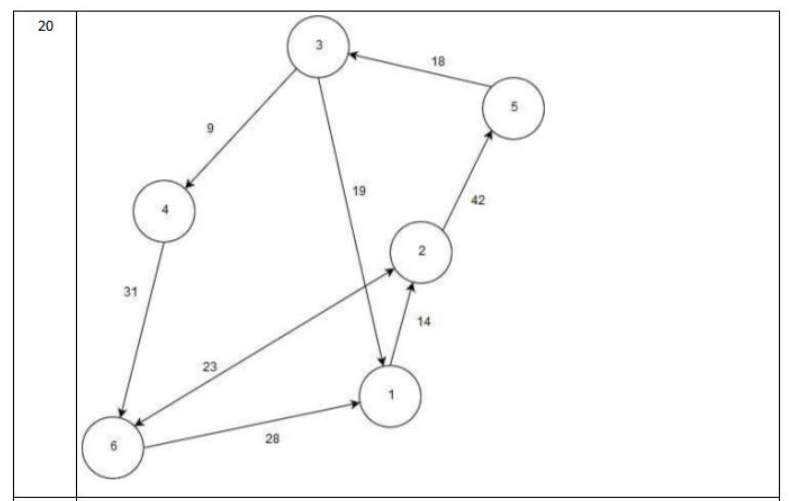
Полякова О. А.

Пермь, 2021

Постановка задачи

1. Разработать алгоритм решения задачи коммивояжера.
2. Реализовать алгоритм в виде программы на алгоритмическом языке С++.
3. Разработать интерфейс средствами Qt.
4. Использование анимации, автоматизации, креативной визуализации повышает рейтинг автора проекта и влияет на экзаменационную оценку.

Вариант 20



Анализ задачи

1. Для решения задачи необходимо:

1.1. Создать класс Qt Designer для окна программы и диалогового окна QDialog, в которых будет происходить ввод условий задачи коммивояжера(длины путей), расчет и вывод результата.

mainwindow.h, mainwindow.cpp, mainwindow.ui

textedge.h, textedge.cpp, textedge.ui

1.2 Создать классы для узлов, ребер графа , их отрисовки , ввода условий задачи коммивояжера(городов и путей между ними), формирования матрицы смежности.

edge.h, edge.cpp

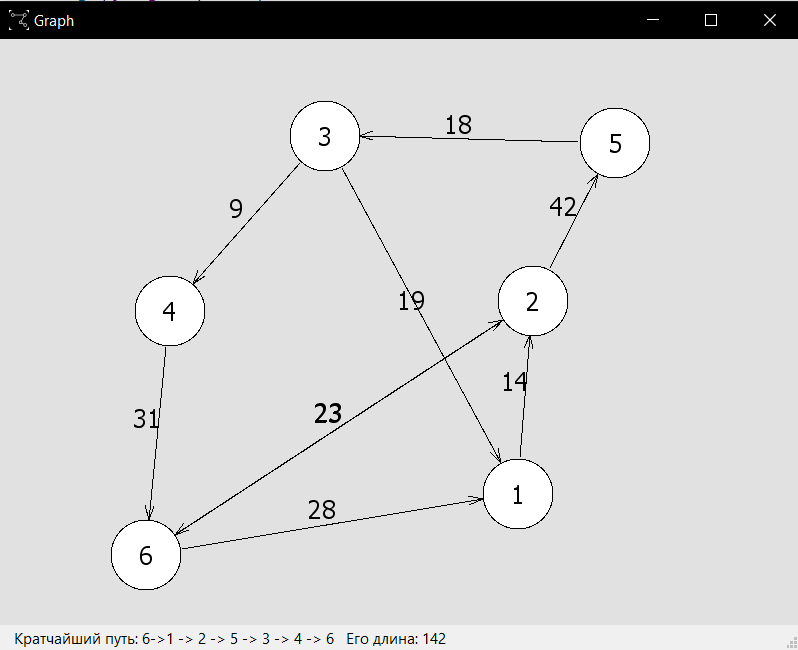
vertex.h, vertex.cpp

myOpenGLWidget.h, myOpenGLWidget.cpp

1.2. Разработать дизайн и функционал для решения задачи коммивояжера.

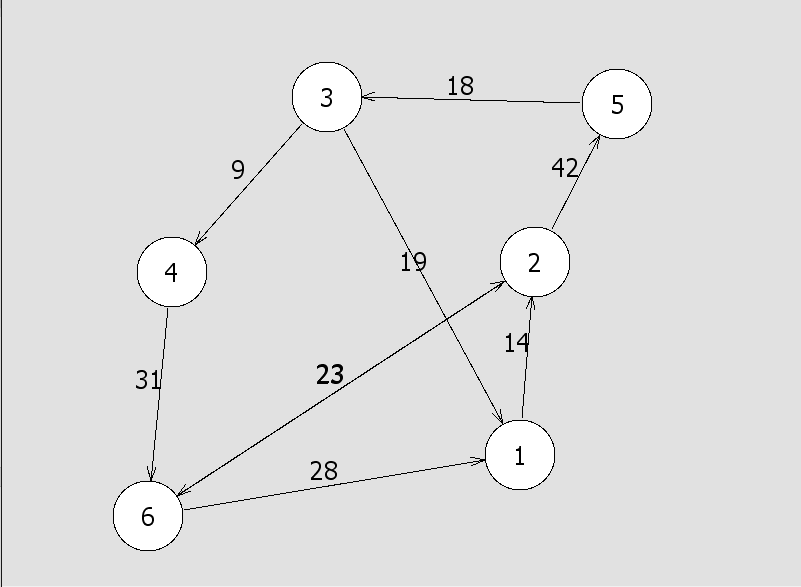
2. Поставленные задачи были решены следующим образом:

2.1 mainwindow.ui – основное окно и строка вывода результата и их дизайн.



Отображение графа реализовано с помощью переопределенного в

myOpenGLWidget виджета QOpenGLWidget.



Вывод результата реализован c помощью QStatusBar.



2.2 mainwindow.h, mainwindow.cpp – реализация расчета и вывода результата.

Расчет и вывод результата разработан в методе keyPressEvent.

Метод вызывается по нажатию клавиши “ = “ на клавиатуре.

void **draw\_vertex**(int x, int y, QString text,QPainter \*painter);

void **draw\_edge**(int x1, int y1, int x2, int y2,QPainter \*painter,QString text);

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

void ***paintEvent***(QPaintEvent \*event) override;

void ***mouseDoubleClickEvent***(QMouseEvent \*event) override;

void ***mousePressEvent***(QMouseEvent\*event)override;

void ***mouseReleaseEvent***(QMouseEvent\*event)override;

void ***mouseMoveEvent***(QMouseEvent\*event)override; void MainWindow::***keyPressEvent***(QKeyEvent \*event)

{

if (event->key()==Qt::Key\_Equal)

{

extern QVector <QVector<int>> arr\_lens;

QVector <int> arr\_res1;

QVector <int> arr\_res2;

int len=arr\_lens.size();

int sum=0;

int M=10000;

bool flag=true;

QVector <QVector<int>> data=arr\_lens;

for (int i=0;i<len;i++) for (int k=0;k<len;k++) if (i==k || data[i][k]==0) data[i][k]=M;

while (flag)

{

QVector <int> row(len);

row.fill(M);

QVector <int> column(len);

column.fill(M);

for (int i=0;i<len;i++) for (int k=0;k<len;k++) row[i]=qMin(row[i],data[i][k]);

for (int i=0;i<len;i++) for (int k=0;k<len;k++) if (row[i]!=M && data[i][k]!=M) data[i][k]-=row[i];

for (int i=0;i<len;i++) for (int k=0;k<len;k++) column[k]=qMin(column[k],data[i][k]);

for (int i=0;i<len;i++) for (int k=0;k<len;k++) if (column[k]!=M && data[i][k]!=M) data[i][k]-=column[k];

int max\_mark\_x;

int max\_mark\_y;

int max\_mark=-1;

for (int i=0;i<len;i++) for (int k=0;k<len;k++) if (data[i][k]==0)

{

int min1=M;

int min2=M;

for (int j=0;j<len;j++) if (j!=k) min1=qMin(min1,data[i][j]);

for (int j=0;j<len;j++) if (j!=i) min2=qMin(min2,data[j][k]);

if (min1+min2>max\_mark)

{

max\_mark=min1+min2;

max\_mark\_x=i;

max\_mark\_y=k;

}

}

sum+=arr\_lens[max\_mark\_x][max\_mark\_y];

data[max\_mark\_y][max\_mark\_x]=M;

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

{

data[max\_mark\_x][i]=M;

data[i][max\_mark\_y]=M;

}

arr\_res1.push\_back(max\_mark\_x);

arr\_res2.push\_back(max\_mark\_y);

flag=false;

for (int i=0;i<len;i++) for (int k=0;k<len;k++) if (data[i][k]!=M) flag=true;

}

if (arr\_res1[0]!=arr\_res2[arr\_res2.size()-1])

{

ui->statusbar->showMessage("Ошибка");

return;

}

QString r=" Кратчайший путь: "+QString::number(arr\_res1[0]+1)+"->"+QString::number(arr\_res2[0]+1);

int x=arr\_res2[0];

arr\_res1.erase(arr\_res1.begin());

arr\_res2.erase(arr\_res2.begin());

while (arr\_res1.size()!=0)

{

for (int i=0;i<arr\_res1.size();i++) if (arr\_res1[i]==x)

{

r+=" -> "+QString::number(arr\_res2[i]+1);

x=arr\_res2[i];

arr\_res1.erase(arr\_res1.begin()+i);

arr\_res2.erase(arr\_res2.begin()+i);

break;

}

}

r[0]=' ';

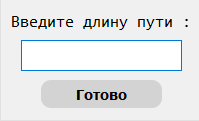
r+=" Его длина: "+QString::number(sum);

ui->statusbar->showMessage(r);

}

}

2.3 textedge.ui – дизайн диалогового окна QDialog для ввода длины пути.



2.4 textedge.h, textedge.cpp – реализация ввода длины пути.

Для ввода длины пути реализован метод on\_ok\_clicked. Он вызывается по нажатию клавиши Enter на клавиатуре или по нажатию кнопки “ Готово ” в диалоговом окне.

void TextEdge::**on\_Ok\_clicked**()

{

extern QString edge\_text;

edge\_text = ui->Val->text();

this->close();

}

2.5 edge.h, edge.cpp – пользовательский класс для узлов графа.

Edge::**Edge**(int X1,int Y1,int X2,int Y2,int Length)

{

x1=X1;

y1=Y1;

x2=X2;

y2=Y2;

length=Length;

}

2.6 vertex.h, vertex.cpp – пользовательский класс для ребер графа.

Vertex::**Vertex**(int X,int Y)

{

x=X;

y=Y;

flag=false;

}

2.7 myOpenGLWidget.h, myOpenGLWidget.cpp – отрисовка графа, взаимодействие с пользователем, формирование матрицы смежности.

Для отрисовки графа реализованы методы draw, draw\_vertex, draw\_edge, использован метод paintEvent и слот redraw.

void myOpenGLWidget::**draw\_vertex**(int x, int y, QString text,QPainter \*painter)

{

int c1=x-20;

int c2=y-15;

int c3=40;

int c4=30;

painter->drawEllipse(QRectF(x-35,y-35,70,70));

painter->drawText(QRect(c1,c2,c3,c4),Qt::AlignCenter,text);

}

void myOpenGLWidget::**draw\_edge**(int x1, int y1, int x2, int y2,QPainter \*painter,QString text)

{

double l=sqrt((x2-x1)\*(x2-x1)+(y2-y1)\*(y2-y1));

int xl1=(x2-x1)/l\*37+x1;

int yl1=(y2-y1)/l\*37+y1;

int xl2=x2-(x2-x1)/l\*35;

int yl2=y2-(y2-y1)/l\*35;

int c1=(xl2+xl1)/2-30;

int c2=(yl2+yl1)/2-30;

int c3=40;

int c4=30;

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

double b;

if (x2==xl2 && yl2<y2) b=pi\*3/2;

else if (y2==yl2 && xl2<x2) b=pi;

else if (x2==xl2 && yl2>y2) b=pi/2;

else if (y2==yl2 && xl2>x2) b=0;

else if (xl2>x2 && yl2<y2) b=2\*pi+atan(1.0\*(y2-yl2)/(x2-xl2));

else if (xl2<x2 && yl2<y2) b=atan(1.0\*(y2-yl2)/(x2-xl2))+pi;

else if (xl2<x2 && yl2>y2) b=atan(1.0\*(y2-yl2)/(x2-xl2))+pi;

else if (xl2>x2 && yl2>y2) b=atan(1.0\*(y2-yl2)/(x2-xl2));

int xs1=cos(b+angle)\*HeadEdge+xl2;

int ys1=sin(b+angle)\*HeadEdge+yl2;

int xs2=cos(b-angle)\*HeadEdge+xl2;

int ys2=sin(b-angle)\*HeadEdge+yl2;

painter->drawLine(xl2,yl2,xs1,ys1);

painter->drawLine(xl2,yl2,xs2,ys2);

painter->drawText(QRect(c1,c2,c3,c4),Qt::AlignCenter,text);

}

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

{

painter->fillRect(event->rect(),QBrush(QColor(225, 225, 225)));

painter->setBrush(QBrush(QColor(255, 255, 255)));

painter->setPen(QPen(QColor(0,0,0)));

glFont.setPixelSize(25);

painter->setFont(glFont);

extern QVector<Vertex> arr\_vertex;

extern QVector<Edge> arr\_edge;

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

{

draw\_vertex(arr\_vertex[i].x,

arr\_vertex[i].y,

QString::number(i+1),

*painter*);

}

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

{

draw\_edge(arr\_edge[i].x1,

arr\_edge[i].y1,

arr\_edge[i].x2,

arr\_edge[i].y2,

*painter*,

QString::number(arr\_edge[i].length));

}

}

void myOpenGLWidget::**redraw**()

{

this->update();

}

void myOpenGLWidget::***paintEvent***(QPaintEvent \*event)

{

QPainter painter;

painter.begin(this);

draw(*&painter*,*event*);

painter.end();

}

Для взаимодействия с пользователем использованы методы mousePressEvent, mouseDoubleClickEvent, mouseMoveEvent, mouseReleaseEvent. В них же происходит формирование матрицы смежности. Эти методы вызываются по одинарному, двойному нажатию или удержанию(в случае левой кнопки мыши) кнопок мыши, при движении и при остановке этих действий соответственно.

void myOpenGLWidget::***mousePressEvent***(QMouseEvent \*event)

{

extern QVector<Vertex> arr\_vertex;

extern QVector<Edge> arr\_edge;

extern QVector <QVector<int>> arr\_lens;

extern int len;

extern int x\_2;

extern int y\_2;

extern int num\_2;

extern QString edge\_text;

int l=edge\_text.toInt();

if (event->button() == Qt::RightButton)

{

if (!fl)

{

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

{

int x1=event->x();

int y1=event->y();

int x2=arr\_vertex[i].x;

int y2=arr\_vertex[i].y;

if ((x1-x2)\*(x1-x2)+(y1-y2)\*(y1-y2)<1400)

{

x\_2=x2;

y\_2=y2;

num\_2=i;

fl=true;

break;

}

}

}

else if (fl)

{

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

{

int x1=event->x();

int y1=event->y();

int x2=arr\_vertex[i].x;

int y2=arr\_vertex[i].y;

if ((x1-x2)\*(x1-x2)+(y1-y2)\*(y1-y2)<1400)

{

TextEdge wind;

wind.setModal(true);

wind.*exec*();

l=edge\_text.toInt();

Edge a(x\_2,y\_2,x2,y2,l);

arr\_lens[num\_2][i]=l;

arr\_edge.push\_back(a);

}

fl=false;

}

}

}

else if((event->button() == Qt::LeftButton))

{

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

{

int x1=event->x();

int y1=event->y();

int x2=arr\_vertex[i].x;

int y2=arr\_vertex[i].y;

if ((x1-x2)\*(x1-x2)+(y1-y2)\*(y1-y2)<1400)

{

arr\_vertex[i].flag=true;

break;

}

}

}

update();

}

void myOpenGLWidget::***mouseDoubleClickEvent***(QMouseEvent \*event){

extern QVector<Vertex> arr\_vertex;

extern QVector<Edge> arr\_edge;

extern QVector <QVector<int>> arr\_lens;

extern int len;

if((event->button() == Qt::LeftButton))

{

Vertex a(event->x(),event->y());

arr\_vertex.push\_back(a);

len++;

QVector <int> row(len);

row.fill(0);

arr\_lens.append(row);

for (int i=0;i<len-1;i++) arr\_lens[i].append(0);

}

else if(event->button() == Qt::RightButton)

{

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

{

int x1=event->x();

int y1=event->y();

int x2=arr\_vertex[i].x;

int y2=arr\_vertex[i].y;

if ((x1-x2)\*(x1-x2)+(y1-y2)\*(y1-y2)<1400)

{

QVector<Edge> arr\_edge\_new;

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

{

if (!(arr\_edge[j].x1==arr\_vertex[i].x && arr\_edge[j].y1==arr\_vertex[i].y)

&& !(arr\_edge[j].x2==arr\_vertex[i].x && arr\_edge[j].y2==arr\_vertex[i].y))

{

arr\_edge\_new.push\_back(arr\_edge[j]);

}

}

arr\_edge=arr\_edge\_new;

arr\_vertex.erase(arr\_vertex.begin()+i);

len--;

for (int j=0;j<len;j++) arr\_lens[j].erase(arr\_lens[j].begin()+i);

arr\_lens.erase(arr\_lens.begin()+i);

fl=false;

break;

}

}

}

update();

}

void myOpenGLWidget::***mouseReleaseEvent***(QMouseEvent \*event)

{

extern QVector<Vertex> arr\_vertex;

extern int len;

for (int i=0;i<len;i++) arr\_vertex[i].flag=false;

update();

}

void myOpenGLWidget::***mouseMoveEvent***(QMouseEvent \*event)

{

extern QVector<Vertex> arr\_vertex;

extern QVector<Edge> arr\_edge;

extern int len;

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

{

if (arr\_vertex[i].flag)

{

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

{

if (arr\_edge[j].x1==arr\_vertex[i].x && arr\_edge[j].y1==arr\_vertex[i].y)

{

arr\_edge[j].x1=event->x();

arr\_edge[j].y1=event->y();

}

if (arr\_edge[j].x2==arr\_vertex[i].x && arr\_edge[j].y2==arr\_vertex[i].y)

{

arr\_edge[j].x2=event->x();

arr\_edge[j].y2=event->y();

}

}

arr\_vertex[i].x=event->x();

arr\_vertex[i].y=event->y();

break;

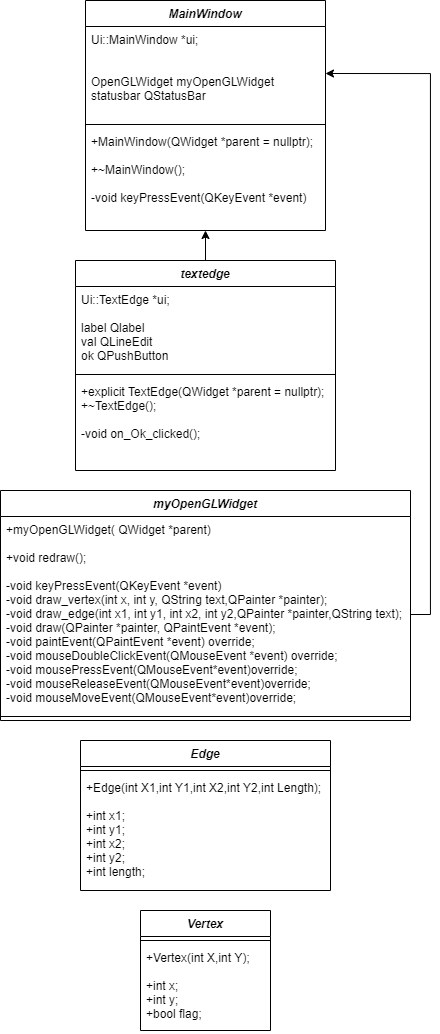
}

}

update();

}

UML-диаграмма



Код программы

Mainwindow.h

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include "myOpenGLWidget.h"

#include <QPainter>

#include <QPushButton>

#include <QVector>

#include <QKeyEvent>

QT\_BEGIN\_NAMESPACE

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

QT\_END\_NAMESPACE

class **MainWindow** : public QMainWindow

{

Q\_OBJECT

public:

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

~***MainWindow***();

protected:

void ***keyPressEvent***(QKeyEvent \*event);

private:

Ui::MainWindow \*ui;

};

#endif // MAINWINDOW\_H

Mainwindow.cpp

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "myOpenGLWidget.h"

#include <QPainter>

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

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

{

ui->setupUi(this);

ui->statusbar->showMessage(" 2xЛКМ - добавление , 2xПКМ - удаление , ПКМ - добавление стрелки , '=' - показать результат" );

}

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

{

delete ui;

}

void MainWindow::***keyPressEvent***(QKeyEvent \*event)

{

if (event->key()==Qt::Key\_Equal)

{

extern QVector <QVector<int>> arr\_lens;

QVector <int> arr\_res1;

QVector <int> arr\_res2;

int len=arr\_lens.size();

int sum=0;

int M=10000;

bool flag=true;

QVector <QVector<int>> data=arr\_lens;

for (int i=0;i<len;i++) for (int k=0;k<len;k++) if (i==k || data[i][k]==0) data[i][k]=M;

while (flag)

{

QVector <int> row(len);

row.fill(M);

QVector <int> column(len);

column.fill(M);

for (int i=0;i<len;i++) for (int k=0;k<len;k++) row[i]=qMin(row[i],data[i][k]);

for (int i=0;i<len;i++) for (int k=0;k<len;k++) if (row[i]!=M && data[i][k]!=M) data[i][k]-=row[i];

for (int i=0;i<len;i++) for (int k=0;k<len;k++) column[k]=qMin(column[k],data[i][k]);

for (int i=0;i<len;i++) for (int k=0;k<len;k++) if (column[k]!=M && data[i][k]!=M) data[i][k]-=column[k];

int max\_mark\_x;

int max\_mark\_y;

int max\_mark=-1;

for (int i=0;i<len;i++) for (int k=0;k<len;k++) if (data[i][k]==0)

{

int min1=M;

int min2=M;

for (int j=0;j<len;j++) if (j!=k) min1=qMin(min1,data[i][j]);

for (int j=0;j<len;j++) if (j!=i) min2=qMin(min2,data[j][k]);

if (min1+min2>max\_mark)

{

max\_mark=min1+min2;

max\_mark\_x=i;

max\_mark\_y=k;

}

}

sum+=arr\_lens[max\_mark\_x][max\_mark\_y];

data[max\_mark\_y][max\_mark\_x]=M;

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

{

data[max\_mark\_x][i]=M;

data[i][max\_mark\_y]=M;

}

arr\_res1.push\_back(max\_mark\_x);

arr\_res2.push\_back(max\_mark\_y);

flag=false;

for (int i=0;i<len;i++) for (int k=0;k<len;k++) if (data[i][k]!=M) flag=true;

}

if (arr\_res1[0]!=arr\_res2[arr\_res2.size()-1])

{

ui->statusbar->showMessage("Ошибка");

return;

}

QString r=" Кратчайший путь: "+QString::number(arr\_res1[0]+1)+"->"+QString::number(arr\_res2[0]+1);

int x=arr\_res2[0];

arr\_res1.erase(arr\_res1.begin());

arr\_res2.erase(arr\_res2.begin());

while (arr\_res1.size()!=0)

{

for (int i=0;i<arr\_res1.size();i++) if (arr\_res1[i]==x)

{

r+=" -> "+QString::number(arr\_res2[i]+1);

x=arr\_res2[i];

arr\_res1.erase(arr\_res1.begin()+i);

arr\_res2.erase(arr\_res2.begin()+i);

break;

}

}

r[0]=' ';

r+=" Его длина: "+QString::number(sum);

ui->statusbar->showMessage(r);

}

}

Main.cpp

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

{

QApplication a(*argc*, *argv*);

MainWindow w;

w.setWindowIcon(QIcon(":/1.ico"));

w.setWindowTitle("Graph");

w.show();

return a.exec();

}

Textedge.h

#ifndef TEXTEDGE\_H

#define TEXTEDGE\_H

#include <QDialog>

namespace **Ui** {

class **TextEdge**;

}

class **TextEdge** : public QDialog

{

Q\_OBJECT

public:

explicit **TextEdge**(QWidget \*parent = nullptr);

~***TextEdge***();

private slots:

void **on\_Ok\_clicked**();

private:

Ui::TextEdge \*ui;

};

#endif // TEXTEDGE\_H

Textedge.cpp

#include "textedge.h"

#include "ui\_textedge.h"

TextEdge::**TextEdge**(QWidget \*parent) :

QDialog(*parent*),

ui(new Ui::TextEdge)

{

ui->setupUi(this);

setWindowFlags(Qt::Window | Qt::FramelessWindowHint);

}

TextEdge::~***TextEdge***()

{

delete ui;

}

void TextEdge::**on\_Ok\_clicked**()

{

extern QString edge\_text;

edge\_text = ui->Val->text();

this->close();

}

Edge.h

#ifndef EDGE\_H

#define EDGE\_H

class **Edge**

{

public:

**Edge**(int X1,int Y1,int X2,int Y2,int Length);

public:

int x1;

int y1;

int x2;

int y2;

int length;

};

#endif // EDGE\_H

Edge.cpp

#include "edge.h"

Edge::**Edge**(int X1,int Y1,int X2,int Y2,int Length)

{

x1=X1;

y1=Y1;

x2=X2;

y2=Y2;

length=Length;

}

Vertex.h

#ifndef VERTEX\_H

#define VERTEX\_H

class **Vertex**

{

public:

**Vertex**(int X,int Y);

public:

int x;

int y;

bool flag;

};

#endif // VERTEX\_H

Vertex.cpp

Vertex::**Vertex**(int X,int Y)

{

x=X;

y=Y;

flag=false;

}

MyOpenGLWidget.h

#ifndef MYOPENGLWIDGET\_H

#define MYOPENGLWIDGET\_H

#include <QOpenGLWidget>

#include <QKeyEvent>

#include <QMouseEvent>

#include <QPaintEvent>

#include <QPainter>

class **myOpenGLWidget** : public QOpenGLWidget

{

Q\_OBJECT

public:

**myOpenGLWidget**( QWidget \*parent);

public slots:

void **redraw**();

protected:

void **draw\_vertex**(int x, int y, QString text,QPainter \*painter);

void **draw\_edge**(int x1, int y1, int x2, int y2,QPainter \*painter,QString text);

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

void ***paintEvent***(QPaintEvent \*event) override;

void ***mouseDoubleClickEvent***(QMouseEvent \*event) override;

void ***mousePressEvent***(QMouseEvent\*event)override;

void ***mouseReleaseEvent***(QMouseEvent\*event)override;

void ***mouseMoveEvent***(QMouseEvent\*event)override;

};

#endif // MYOPENGLWIDGET\_H

MyOpenGLWidget.cpp

#include "myOpenGLWidget.h"

#include "textedge.h"

#include "vertex.h"

#include "edge.h"

#include <QBrush>

#include <QFont>

#include <QPushButton>

double pi=3.1415;

double angle=pi/10;

int HeadEdge=15;

bool fl=false;

QFont glFont;

myOpenGLWidget::**myOpenGLWidget**(QWidget \*parent)

:QOpenGLWidget(*parent*)

{

}

void myOpenGLWidget::**draw\_vertex**(int x, int y, QString text,QPainter \*painter)

{

int c1=x-20;

int c2=y-15;

int c3=40;

int c4=30;

painter->drawEllipse(QRectF(x-35,y-35,70,70));

painter->drawText(QRect(c1,c2,c3,c4),Qt::AlignCenter,text);

}

void myOpenGLWidget::**draw\_edge**(int x1, int y1, int x2, int y2,QPainter \*painter,QString text)

{

double l=sqrt((x2-x1)\*(x2-x1)+(y2-y1)\*(y2-y1));

int xl1=(x2-x1)/l\*37+x1;

int yl1=(y2-y1)/l\*37+y1;

int xl2=x2-(x2-x1)/l\*35;

int yl2=y2-(y2-y1)/l\*35;

int c1=(xl2+xl1)/2-30;

int c2=(yl2+yl1)/2-30;

int c3=40;

int c4=30;

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

double b;

if (x2==xl2 && yl2<y2) b=pi\*3/2;

else if (y2==yl2 && xl2<x2) b=pi;

else if (x2==xl2 && yl2>y2) b=pi/2;

else if (y2==yl2 && xl2>x2) b=0;

else if (xl2>x2 && yl2<y2) b=2\*pi+atan(1.0\*(y2-yl2)/(x2-xl2));

else if (xl2<x2 && yl2<y2) b=atan(1.0\*(y2-yl2)/(x2-xl2))+pi;

else if (xl2<x2 && yl2>y2) b=atan(1.0\*(y2-yl2)/(x2-xl2))+pi;

else if (xl2>x2 && yl2>y2) b=atan(1.0\*(y2-yl2)/(x2-xl2));

int xs1=cos(b+angle)\*HeadEdge+xl2;

int ys1=sin(b+angle)\*HeadEdge+yl2;

int xs2=cos(b-angle)\*HeadEdge+xl2;

int ys2=sin(b-angle)\*HeadEdge+yl2;

painter->drawLine(xl2,yl2,xs1,ys1);

painter->drawLine(xl2,yl2,xs2,ys2);

painter->drawText(QRect(c1,c2,c3,c4),Qt::AlignCenter,text);

}

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

{

painter->fillRect(event->rect(),QBrush(QColor(225, 225, 225)));

painter->setBrush(QBrush(QColor(255, 255, 255)));

painter->setPen(QPen(QColor(0,0,0)));

glFont.setPixelSize(25);

painter->setFont(glFont);

extern QVector<Vertex> arr\_vertex;

extern QVector<Edge> arr\_edge;

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

{

draw\_vertex(arr\_vertex[i].x,

arr\_vertex[i].y,

QString::number(i+1),

*painter*);

}

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

{

draw\_edge(arr\_edge[i].x1,

arr\_edge[i].y1,

arr\_edge[i].x2,

arr\_edge[i].y2,

*painter*,

QString::number(arr\_edge[i].length));

}

}

void myOpenGLWidget::**redraw**()

{

this->update();

}

void myOpenGLWidget::***paintEvent***(QPaintEvent \*event)

{

QPainter painter;

painter.begin(this);

draw(*&painter*,*event*);

painter.end();

}

void myOpenGLWidget::***mousePressEvent***(QMouseEvent \*event)

{

extern QVector<Vertex> arr\_vertex;

extern QVector<Edge> arr\_edge;

extern QVector <QVector<int>> arr\_lens;

extern int len;

extern int x\_2;

extern int y\_2;

extern int num\_2;

extern QString edge\_text;

int l=edge\_text.toInt();

if (event->button() == Qt::RightButton)

{

if (!fl)

{

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

{

int x1=event->x();

int y1=event->y();

int x2=arr\_vertex[i].x;

int y2=arr\_vertex[i].y;

if ((x1-x2)\*(x1-x2)+(y1-y2)\*(y1-y2)<1400)

{

x\_2=x2;

y\_2=y2;

num\_2=i;

fl=true;

break;

}

}

}

else if (fl)

{

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

{

int x1=event->x();

int y1=event->y();

int x2=arr\_vertex[i].x;

int y2=arr\_vertex[i].y;

if ((x1-x2)\*(x1-x2)+(y1-y2)\*(y1-y2)<1400)

{

TextEdge wind;

wind.setModal(true);

wind.*exec*();

l=edge\_text.toInt();

Edge a(x\_2,y\_2,x2,y2,l);

arr\_lens[num\_2][i]=l;

arr\_edge.push\_back(a);

}

fl=false;

}

}

}

else if((event->button() == Qt::LeftButton))

{

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

{

int x1=event->x();

int y1=event->y();

int x2=arr\_vertex[i].x;

int y2=arr\_vertex[i].y;

if ((x1-x2)\*(x1-x2)+(y1-y2)\*(y1-y2)<1400)

{

arr\_vertex[i].flag=true;

break;

}

}

}

update();

}

void myOpenGLWidget::***mouseDoubleClickEvent***(QMouseEvent \*event){

extern QVector<Vertex> arr\_vertex;

extern QVector<Edge> arr\_edge;

extern QVector <QVector<int>> arr\_lens;

extern int len;

if((event->button() == Qt::LeftButton))

{

Vertex a(event->x(),event->y());

arr\_vertex.push\_back(a);

len++;

QVector <int> row(len);

row.fill(0);

arr\_lens.append(row);

for (int i=0;i<len-1;i++) arr\_lens[i].append(0);

}

else if(event->button() == Qt::RightButton)

{

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

{

int x1=event->x();

int y1=event->y();

int x2=arr\_vertex[i].x;

int y2=arr\_vertex[i].y;

if ((x1-x2)\*(x1-x2)+(y1-y2)\*(y1-y2)<1400)

{

QVector<Edge> arr\_edge\_new;

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

{

if (!(arr\_edge[j].x1==arr\_vertex[i].x && arr\_edge[j].y1==arr\_vertex[i].y)

&& !(arr\_edge[j].x2==arr\_vertex[i].x && arr\_edge[j].y2==arr\_vertex[i].y))

{

arr\_edge\_new.push\_back(arr\_edge[j]);

}

}

arr\_edge=arr\_edge\_new;

arr\_vertex.erase(arr\_vertex.begin()+i);

len--;

for (int j=0;j<len;j++) arr\_lens[j].erase(arr\_lens[j].begin()+i);

arr\_lens.erase(arr\_lens.begin()+i);

fl=false;

break;

}

}

}

update();

}

void myOpenGLWidget::***mouseReleaseEvent***(QMouseEvent \*event)

{

extern QVector<Vertex> arr\_vertex;

extern int len;

for (int i=0;i<len;i++) arr\_vertex[i].flag=false;

update();

}

void myOpenGLWidget::***mouseMoveEvent***(QMouseEvent \*event)

{

extern QVector<Vertex> arr\_vertex;

extern QVector<Edge> arr\_edge;

extern int len;

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

{

if (arr\_vertex[i].flag)

{

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

{

if (arr\_edge[j].x1==arr\_vertex[i].x && arr\_edge[j].y1==arr\_vertex[i].y)

{

arr\_edge[j].x1=event->x();

arr\_edge[j].y1=event->y();

}

if (arr\_edge[j].x2==arr\_vertex[i].x && arr\_edge[j].y2==arr\_vertex[i].y)

{

arr\_edge[j].x2=event->x();

arr\_edge[j].y2=event->y();

}

}

arr\_vertex[i].x=event->x();

arr\_vertex[i].y=event->y();

break;

}

}

update();

}