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 творческому заданию (часть 2)**

Дисциплина: основы алгоритмизации и программирования

Тема: решение задачи Коммивояжёра

Выполнил работу

студент группы РИС-22-2б

Шальнев А.В.

Проверил

Доцент кафедры ИТАС

Полякова О.А.

Пермь, 2023

**Постановка задачи**

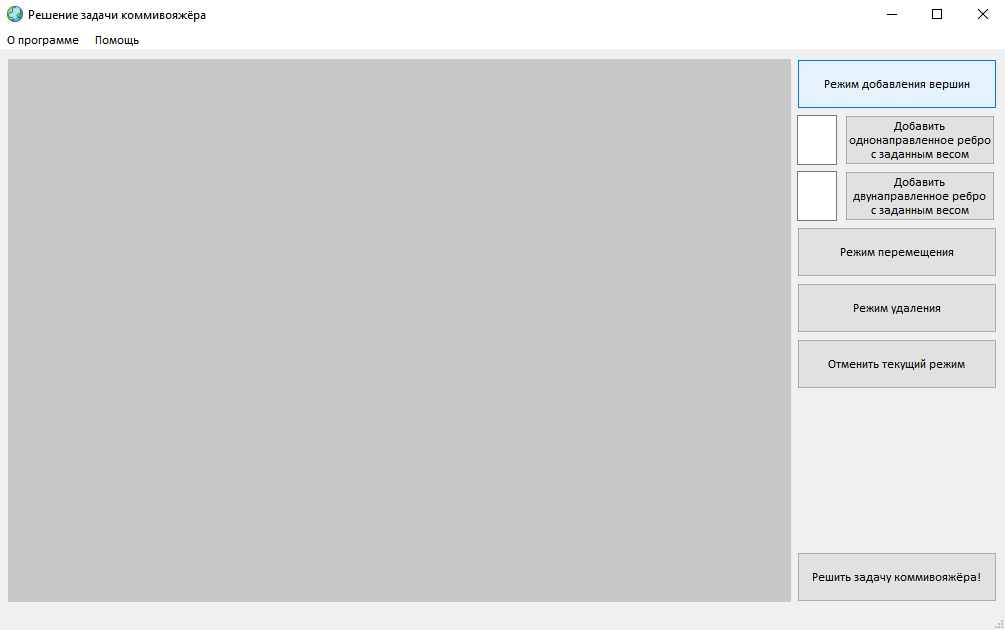
1. Взять свой вариант из лабораторной работы по графам. Модифицировать граф таким образом, чтобы для этого графа можно было решить задачу коммивояжера. Можно придумать альтернативную задачу, которую можно решить методом ветвей и границ.
2. Проработать визуальную часть, а именно—построение графа, средствами QT.
3. Исходные данные должны приниматься с консоли и через графический интерфейс.
4. Задокументировать программу диаграммой классов UML.
5. Записать видеоролик с экрана монитора при помощи программ Bandicam, OBS, Camtasia или любой другой и загрузить на Youtube.

В видеоролике должны быть отражены:

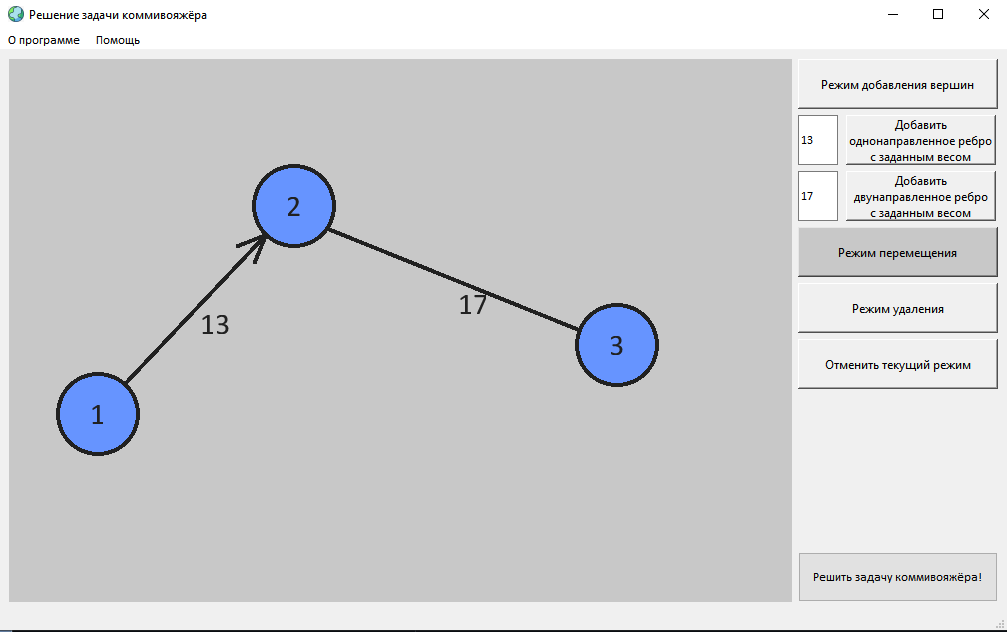
* Продемонстрирована работа программы с тем вариантом, который был установлен в пункте 1, от задания исходных данных до получения результата;
* Уделено внимание дизайнерским и конструкторским решениям;
* Показана и объяснена UML-диаграмма;
* Представлены реализации ключевых классов и функций в коде;
* Рассказано про инструменты, технологии и программы, которые были использованы для создания программы и видеоролика;
* Подчеркнуты те достижения, которыми гордится автор в своей разработке;
* Продолжительность видеоролика не должна превышать 6 минут.

1. Подготовить отчет который будет включать, кроме основных разделов (введение, заключение и других), UML-диаграмму с разъяснением реализованных классов, описание используемых инструментов: среды разработки, программы записи видео с экрана монитора, библиотеки, API и другие инструменты, а также—ссылки на используемые источники.

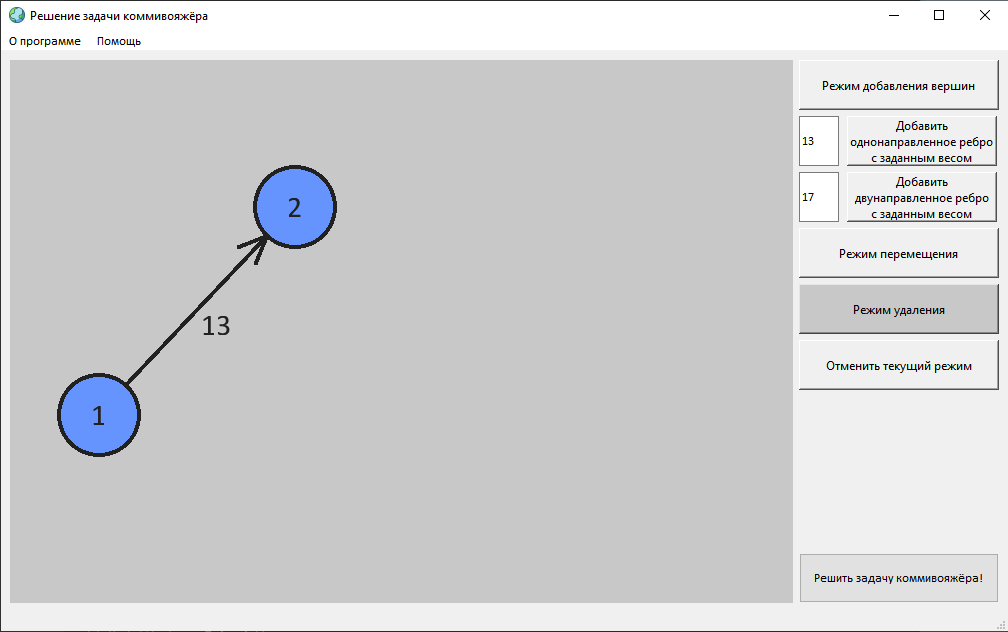
**Анализ задачи**

1. Для решения задачи необходимо…
   1. Проработать интерфейс для пользователя. Интерфейс должен быть понятным и удобным.  
      

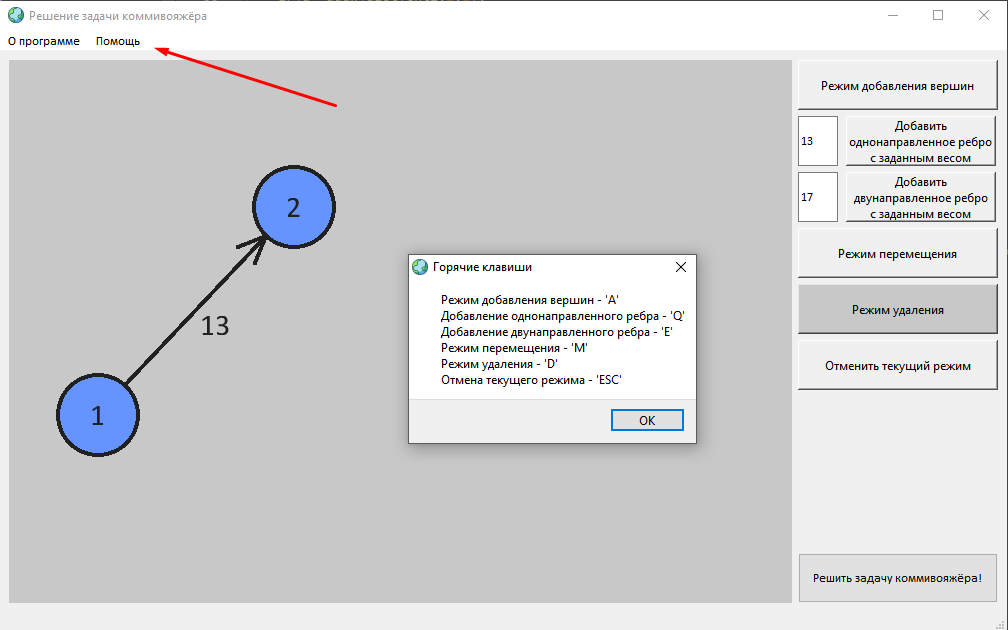
Как видно на данной картинке, интерфейс достаточно удобен, чтобы быть понятным любому пользователю. Нужно всего лишь переключаться между главными режимами, чтобы добавлять вершины, ребра, задать веса. Также можно перемещать вершины для более удобного вида графа.



Есть две кнопки с прилегающими к ним полями, для того, чтобы задавать ребра однонаправленного вида или двунаправленного, а также задавать веса к ним. Однонаправленное ребро имеет одну стрелку, двунаправленное ребро не имеет стрелок, как показано на рисунке.

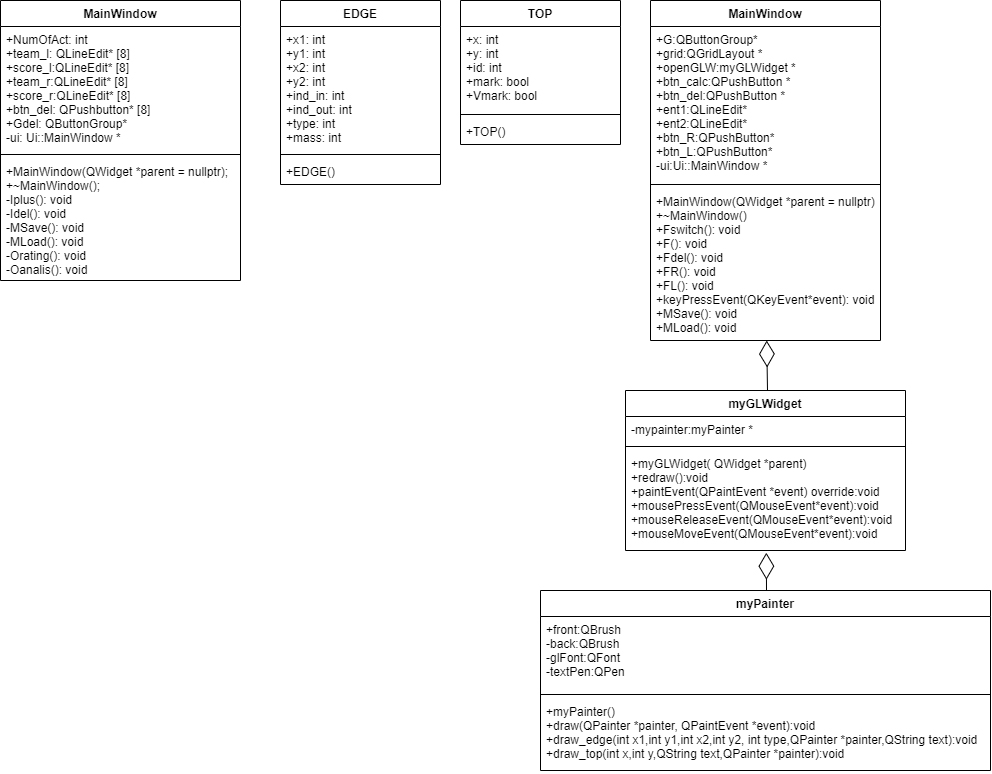


Режим удаления может удалять вершины. Когда вершина исчезает, то исчезают и все связи к нему. Сначала нажимается сама кнопка удаления, затем нужная вершина.



Отдельно хотелось бы сказать о горячих клавишах, они позволяют быстро ориентироваться в функциональных компонентах программы, практически не используя мышь.

**Диаграмма классов**



**Код программы**

**Main.cpp**

#include "mainwindow.h"

#include "top.h"

#include "edge.h"

#include <QApplication>

#include <QLineEdit>

#include <QSplashScreen>

#include <QTimer>

QString status="none";

int len=0;

int mem\_x;

int mem\_y;

int mem\_num;

QVector <TOP> graph1;

QVector <EDGE> graph2;

QVector <QVector<int>> matrix;

QString statusBook[]={"top","edge1.1","edge2.1","move","del","none"};

QString text\_for\_arrows;

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

{

QApplication a(argc, argv);

a.setWindowIcon(QIcon("C:/Users/Lexa/Downloads/commivoyager-main/commivoyager-main/earth.ico"));

QPixmap pixmap("C:/Users/Lexa/Downloads/commivoyager-main/commivoyager-main/cosmo.png");

QSplashScreen splash(pixmap);

splash.setMask(pixmap.mask());

splash.show();

MainWindow w;

QTimer::singleShot(3000, &splash, SLOT(close()));

QTimer::singleShot(3000, &w, SLOT(show()));

//w.show();

return a.exec();

}  
  
  
**edge.cpp**

#include "edge.h"

EDGE::**EDGE**(int X1,int Y1,int X2,int Y2, int Type,int Mass)

{

x1=X1;

y1=Y1;

x2=X2;

y2=Y2;

type=Type;

mass=Mass;

}

**mainwindow.cpp**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "myglwidget.h"

#include <QPainter>

#include <QGridLayout>

#include <QPushButton>

#include <QLineEdit>

#include <QMessageBox>

#include <QLabel>

MainWindow::MainWindow(QWidget \*parent)

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

{

ui->setupUi(this); // Растянуть окно вправо на 200 пикселей при запуске

//widgets

//extern QLineEdit\* ent1;

//extern QLineEdit\* ent2;

ent1=new QLineEdit;

ent2=new QLineEdit;

grid=new QGridLayout();

openGLW=new myGLWidget(this);

btn\_calc=new QPushButton;

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

{

QPushButton\* btn=new QPushButton;

btn->setFixedSize(200,50);

grid->addWidget(btn,i,1,1,2);

connect(btn,&QPushButton::clicked,this,&MainWindow::Fswitch);

G->addButton(btn,i);

}

//texts

G->button(0)->setText("Режим добавления вершин");

G->button(1)->setText("Добавить\nоднонаправленное ребро\nс заданным весом");

G->button(2)->setText("Добавить\nдвунаправленное ребро\nс заданным весом");

G->button(3)->setText("Режим перемещения");

G->button(4)->setText("Режим удаления");

G->button(5)->setText("Отменить текущий режим");

btn\_calc->setText("Решить задачу коммивояжёра!");

G->button(0)->setShortcut(Qt::Key\_A);

G->button(1)->setShortcut(Qt::Key\_Q);

G->button(2)->setShortcut(Qt::Key\_E);

G->button(3)->setShortcut(Qt::Key\_M);

G->button(4)->setShortcut(Qt::Key\_D);

G->button(5)->setShortcut(Qt::Key\_Escape);

//sizes

G->button(1)->setFixedSize(150,50);

G->button(2)->setFixedSize(150,50);

btn\_calc->setFixedSize(200,50);

ent1->setFixedSize(40,50);

ent2->setFixedSize(40,50);

//locations

grid->addWidget(openGLW,0,0,8,1);

grid->addWidget(G->button(1),1,2);

grid->addWidget(G->button(2),2,2);

grid->addWidget(ent1,1,1);

grid->addWidget(ent2,2,1);

grid->addWidget(G->button(1),1,2);

grid->addWidget(G->button(2),2,2);

grid->addWidget(btn\_calc,7,1,1,2);

connect(btn\_calc,&QPushButton::clicked,this,&MainWindow::Fcheck);

ui->centralwidget->setLayout(grid);

}

void MainWindow::Fswitch()

{

QPushButton\* button = qobject\_cast<QPushButton\*>(sender());

int buttonID = G->id(button);

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

{

G->button(i)->setStyleSheet("QPushButton { background-color: rgb(240,240,240); }\n");

}

if (buttonID!=5) G->button(buttonID)->setStyleSheet("QPushButton { background-color: rgb(200,200,200); }\n");

extern QString status;

extern QString text\_for\_arrows;

extern QString statusBook[7];

if (buttonID==1) text\_for\_arrows=ent1->text();

if (buttonID==2) text\_for\_arrows=ent2->text();

status=statusBook[buttonID];

}

void MainWindow::Fcheck()

{

try

{

F();

}

catch(...)

{

QMessageBox::warning(this, "Внимание","невозможно вычислить");

}

}

void MainWindow::F()

{

extern int len;

extern QVector <QVector<int>> matrix;

QVector <int> result\_arr\_1;

QVector <int> result\_arr\_2;

int sum=0;

bool flag=true;

int M=1000;

QVector <QVector<int>> data=matrix;

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+=matrix[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;

}

result\_arr\_1.push\_back(max\_mark\_x+1);

result\_arr\_2.push\_back(max\_mark\_y+1);

flag=false;

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

}

QString result = "->" + QString::number(result\_arr\_1[0] - 1) + "->" + QString::number(result\_arr\_2[0] - 1);

int x = result\_arr\_2[0] - 1;

result\_arr\_1.erase(result\_arr\_1.begin());

result\_arr\_2.erase(result\_arr\_2.begin());

while (result\_arr\_1.size() != 0) {

for (int i = 0; i < result\_arr\_1.size(); i++) {

if (result\_arr\_1[i] - 1 == x) {

result += "->" + QString::number(result\_arr\_2[i] - 1);

x = result\_arr\_2[i] - 1;

result\_arr\_1.erase(result\_arr\_1.begin() + i);

result\_arr\_2.erase(result\_arr\_2.begin() + i);

break;

}

}

}

result[0]=' ';

result[1]=' ';

result+=" | "+QString::number(sum);

QMessageBox::information(this, "Результат",result);

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

{

*delete* ui;

}

void MainWindow::on\_action\_triggered()

{

QMessageBox msgBox;

QLabel \*label = new QLabel(&msgBox);

label->setText("<a href=\"https://github.com/tyloong1337\">GitHub</a>");

label->setAlignment(Qt::AlignHCenter | Qt::AlignVCenter);

label->setOpenExternalLinks(true);

msgBox.setText("Симулятор решения задачи коммивояжёра\nВыполнил Шальнев Алексей\nСтудент РИС-22-2Б");

msgBox.setWindowTitle("О программе");

msgBox.layout()->addWidget(label);

msgBox.exec();

}

void MainWindow::on\_action\_2\_triggered()

{

QMessageBox msgBox;

msgBox.setText("Режим добавления вершин - 'A'\nДобавление однонаправленного ребра - 'Q'\nДобавление двунаправленного ребра - 'E'\nРежим перемещения - 'M'\nРежим удаления - 'D'\nОтмена текущего режима - 'ESC'");

msgBox.setWindowTitle("Горячие клавиши");

msgBox.exec();

}

**myglwidget.cpp**

#include "myglwidget.h"

#include "mypainter.h"

#include <QPainter>

#include <top.h>

#include <edge.h>

#include <QMouseEvent>

myGLWidget::**myGLWidget**(QWidget \*parent)

:QOpenGLWidget(*parent*)

{

mypainter=*new* myPainter;

}

void myGLWidget::**redraw**()

{

*//mypainter->back.setColor((QColor(255,0,0)));*

*this*->update();

}

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

{

QPainter painter;

painter.begin(*this*);

mypainter->draw(*&painter*,*event*);

painter.end();

}

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

{

*extern* QVector<TOP> graph1;

*extern* QVector<EDGE> graph2;

*extern* QVector <QVector<int>> matrix;

*extern* int len;

*extern* int mem\_x;

*extern* int mem\_y;

*extern* int mem\_num;

*extern* QString status;

*extern* QString text\_for\_arrows;

int m=text\_for\_arrows.toInt();

*if* (status=="top")

{

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

graph1.push\_back(a);

len++;

*//matrix*

QVector <int> row(len);

row.fill(0);

matrix.append(row);

*for* (int i=0;i<len-1;i++) matrix[i].append(0);

}

*else* *if* (status=="edge1.1" || status=="edge2.1")

{

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

{

int x1=event->x();

int y1=event->y();

int x2=graph1[i].x;

int y2=graph1[i].y;

*if* ((x1-x2)\*(x1-x2)+(y1-y2)\*(y1-y2)<1600)

{

mem\_x=x2;

mem\_y=y2;

mem\_num=i;

status=(status=="edge1.1")? "edge1.2":"edge2.2";

*break*;

}

}

}

*else* *if* (status=="edge1.2" || status=="edge2.2")

{

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

{

int x1=event->x();

int y1=event->y();

int x2=graph1[i].x;

int y2=graph1[i].y;

*if* ((x1-x2)\*(x1-x2)+(y1-y2)\*(y1-y2)<1600)

{

*if* (status=="edge1.2")

{

EDGE a(mem\_x,mem\_y,x2,y2,1,m);

graph2.push\_back(a);

*//matrix*

matrix[mem\_num][i]=m;

}

*else*

{

EDGE a(mem\_x,mem\_y,x2,y2,2,m);

graph2.push\_back(a);

*//matrix*

matrix[mem\_num][i]=m;

matrix[i][mem\_num]=m;

}

status=(status=="edge1.2")? "edge1.1":"edge2.1";

*break*;

}

}

}

*else* *if* (status=="move")

{

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

{

int x1=event->x();

int y1=event->y();

int x2=graph1[i].x;

int y2=graph1[i].y;

*if* ((x1-x2)\*(x1-x2)+(y1-y2)\*(y1-y2)<1600)

{

graph1[i].mark=*true*;

*break*;

}

}

}

*else* *if* (status=="del")

{

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

{

int x1=event->x();

int y1=event->y();

int x2=graph1[i].x;

int y2=graph1[i].y;

*if* ((x1-x2)\*(x1-x2)+(y1-y2)\*(y1-y2)<1600)

{

QVector<EDGE> graph2\_new;

*for* (int j=0;j<graph2.size();j++)

{

*if* (!(graph2[j].x1==graph1[i].x && graph2[j].y1==graph1[i].y)

&& !(graph2[j].x2==graph1[i].x && graph2[j].y2==graph1[i].y))

{

graph2\_new.push\_back(graph2[j]);

}

}

graph2=graph2\_new;

graph1.erase(graph1.begin()+i);

len--;

*//matrix*

*for* (int j=0;j<len;j++) matrix[j].erase(matrix[j].begin()+i);

matrix.erase(matrix.begin()+i);

*break*;

}

}

}

update();

}

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

{

*extern* QVector<TOP> graph1;

*extern* int len;

*extern* QString status;

*if* (status=="move") *for* (int i=0;i<len;i++) graph1[i].mark=*false*;

update();

}

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

{

*extern* QVector<TOP> graph1;

*extern* QVector<EDGE> graph2;

*extern* int len;

*extern* QString status;

*if* (status=="move")

{

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

{

*if* (graph1[i].mark)

{

*for* (int j=0;j<graph2.size();j++)

{

*if* (graph2[j].x1==graph1[i].x && graph2[j].y1==graph1[i].y)

{

graph2[j].x1=event->x();

graph2[j].y1=event->y();

}

*if* (graph2[j].x2==graph1[i].x && graph2[j].y2==graph1[i].y)

{

graph2[j].x2=event->x();

graph2[j].y2=event->y();

}

}

graph1[i].x=event->x();

graph1[i].y=event->y();

*break*;

}

}

}

update();

}

**mypainter.cpp**

#include "mypainter.h"

#include <QPaintEvent>

#include <QPainter>

#include <QWidget>

#include <QtCore/qmath.h>

#include "top.h"

#include "edge.h"

double pi=3.1415;

double angle=pi/8;*//угол* *раствора* *стрелки*

int ls=30; *//длина* *кончика* *стрелки*

myPainter::**myPainter**()

{

front=QBrush(QColor(255,193,199));

back=QBrush(QColor(200,200,200));

textPen=QPen(QColor(33,33,33));

glFont.setPixelSize(30);

}

void myPainter::**draw\_top**(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-40,y-40,80,80));

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

}

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

{

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

int xl1=(x2-x1)/l\*40+x1;

int yl1=(y2-y1)/l\*40+y1;

int xl2=x2-(x2-x1)/l\*40;

int yl2=y2-(y2-y1)/l\*40;

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

*if* (type==1)

{

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)\*ls+xl2;

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

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

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

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

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

}

int x = (xl1 + xl2) / 2;

int y = (yl1 + yl2) / 2;

// Calculate the position of the rectangle for drawing the text

int dx = (xl2 - xl1);

int dy = (yl2 - yl1);

int margin = 5;

if (dx != 0) {

dx /= abs(dx);

}

if (dy != 0) {

dy /= abs(dy);

}

int c1 = x + margin \* dx;

int c2 = y + margin \* dy;

int c1=(xl2+xl1)/2-20;

int c2=(yl2+yl1)/2-15;

int c3=40;

int c4=30;

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

}

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

{

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

painter->setBrush(front);

painter->setPen(textPen);

painter->setFont(glFont);

textPen.setWidth(4);

*extern* QVector<TOP> graph1;

*extern* QVector<EDGE> graph2;

*for* (int i=0;i<graph1.size();i++)

{

draw\_top(graph1[i].x,

graph1[i].y,

QString::number(i),

*painter*);

}

*for* (int i=0;i<graph2.size();i++)

{

draw\_edge(graph2[i].x1,

graph2[i].y1,

graph2[i].x2,

graph2[i].y2,

graph2[i].type,

*painter*,

QString::number(graph2[i].mass));

}

}

**top.cpp**

#include "top.h"

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

{

x=X;

y=Y;

mark=*false*;

}

**edge.h**

#ifndef EDGE\_H

#define EDGE\_H

*class* **EDGE**

{

*public*:

**EDGE**(int X1,int Y1,int X2,int Y2, int Type,int Mass);

*public*:

int x1;

int y1;

int x2;

int y2;

int type;

int mass;

};

#endif *//* *EDGE\_H*

**mainwindow.h**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include "myglwidget.h"

#include <QPainter>

#include <QGridLayout>

#include <QPushButton>

#include <QVector>

#include <QButtonGroup>

#include <QLineEdit>

QT\_BEGIN\_NAMESPACE

*namespace* Ui { *class* MainWindow; }

QT\_END\_NAMESPACE

*class* MainWindow : *public* QMainWindow

{

Q\_OBJECT

*public*:

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

~MainWindow();

void Fswitch();

void F();

void Fcheck();

*public*:

QButtonGroup\* G=*new* QButtonGroup;

QGridLayout \*grid;

myGLWidget \*openGLW;

QPushButton \*btn\_calc;

QLineEdit\* ent1;

QLineEdit\* ent2;

*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:

void **redraw**();

*protected*:

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

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

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

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

*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);

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

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

QBrush front;

*private*:

QBrush back;

QFont glFont;

QPen textPen;

};

#endif *//MYPAINTER\_H*

**top.h**

#ifndef TOP\_H

#define TOP\_H

*class* **TOP**

{

*public*:

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

*public*:

int x;

int y;

bool mark;

};

#endif *//* *TOP\_H*

**Работа программы**

