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 Творческой работе №2**

Дисциплина: основы алгоритмизации и программирования

Тема: Графы, задача Коммивояжёра

Выполнил работу

студент группы РИС-20-1б

Кукшинова А.А.

Проверила

Доцент кафедры ИТАС

ПоляковаО.А.

Пермь, 2021

Постановка задачи

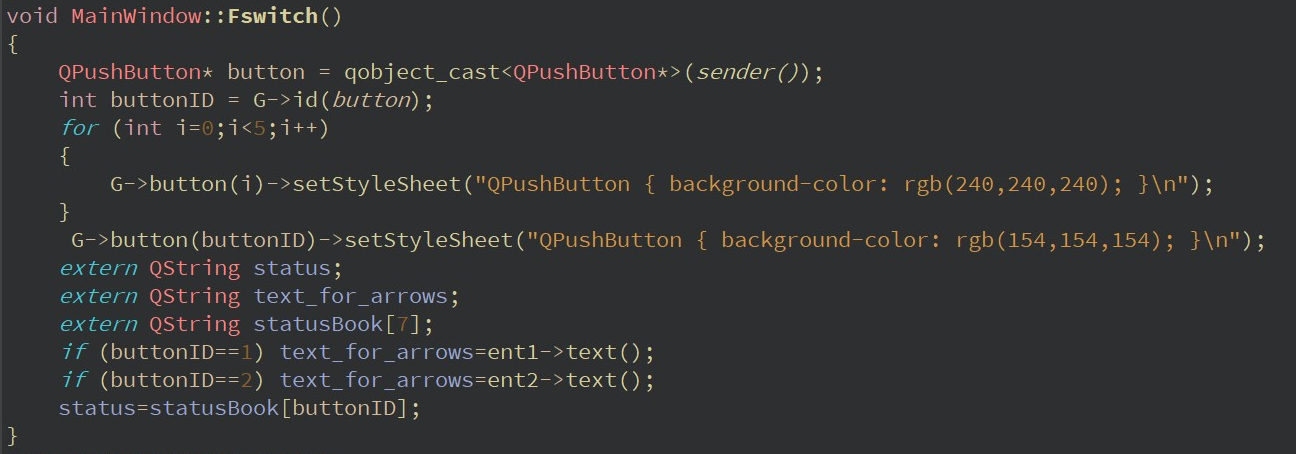
Разработать алгоритм решения задачи Коммивояжёра соответствии с вариантом. Адаптировать граф для решения задачи.

Реализовать алгоритм в виде программы на алгоритмическом языке С++.

Разработать интерфейс средствами QT.

Анализ задачи

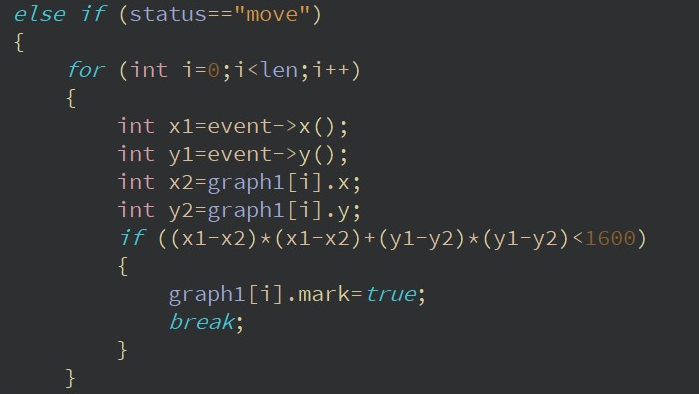
Весь интерфейс программы завязан на кнопках вызова операций, но в данной программе нажатие на кнопку не означает действие. Нажав на кнопку, пользователь говорит программе, что он планирует сделать с opengl виджетом. За смену таких рабочих состояний и отвечает данная функция.



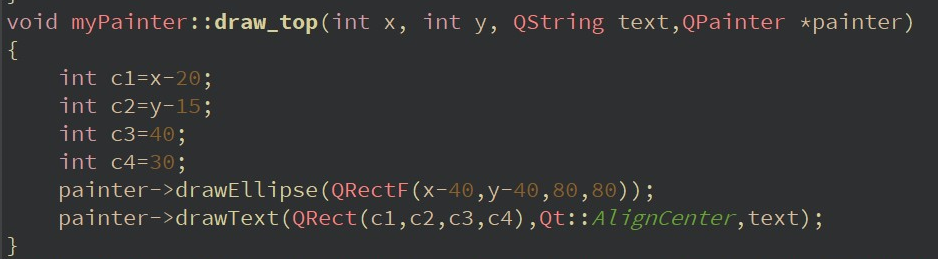
Задача решается методом ветвей и границ. За поиск ответа отвечает отдельный метод:



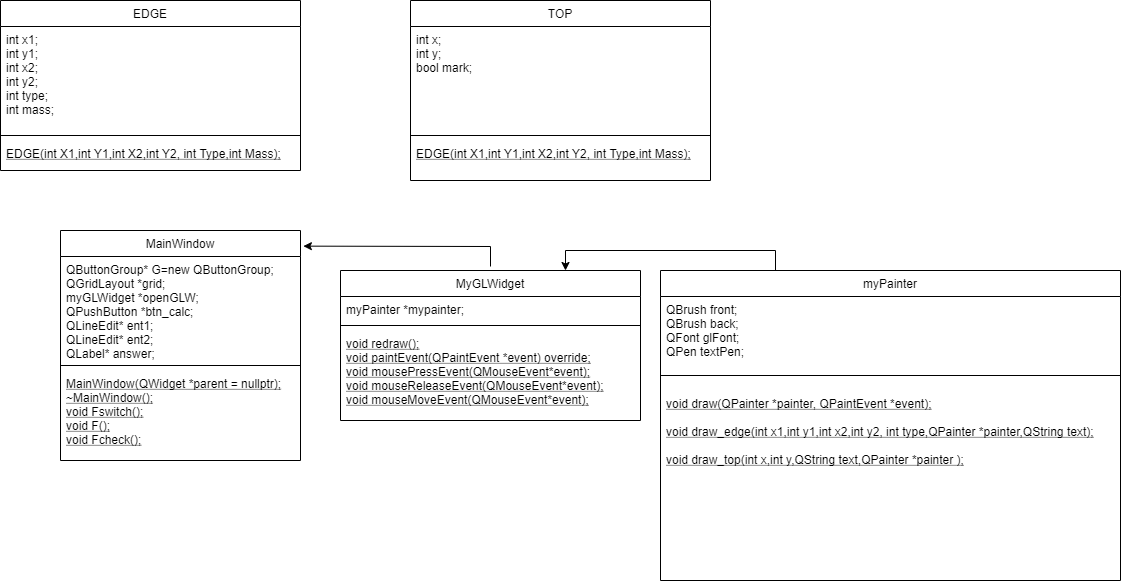
При нажатии на ту или иную кнопку, переменной status присваивается определенное значение, например, выбирая действие движения узлов, status получает значение “move”, и вызывается метод для движения:



Также существует отдельный класс MyPainter, который отвечает за визуализацию различных частей графа. Например, вершины:



UML диаграмма:



Код основных функций:  
mainwindow.cpp:  
#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "myglwidget.h"

#include <QPainter>

#include <QGridLayout>

#include <QPushButton>

#include <QLineEdit>

#include <QMessageBox>

#include <QLabel>

#include <iostream>

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

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

{

ui->setupUi(*this*);

ent1=*new* QLineEdit;

ent2=*new* QLineEdit;

grid=*new* QGridLayout();

openGLW=*new* myGLWidget(*this*);

btn\_calc=*new* QPushButton;

*for* (int i=0;i<5;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*

answer = *new* QLabel;

grid->addWidget(*answer*,7,1,1,2);

G->button(0)->setIcon(QIcon("C:/KOMIVOYA/VISUALPIC/1"));

G->button(0)->setIconSize(maximumSize());

G->button(1)->setIcon(QIcon("C:/KOMIVOYA/VISUALPIC/2"));

G->button(1)->setIconSize(maximumSize());

G->button(2)->setIcon(QIcon("C:/KOMIVOYA/VISUALPIC/3"));

G->button(2)->setIconSize(maximumSize());

G->button(3)->setIcon(QIcon("C:/KOMIVOYA/VISUALPIC/4"));

G->button(3)->setIconSize(maximumSize());

G->button(4)->setIcon(QIcon("C:/KOMIVOYA/VISUALPIC/5"));

G->button(4)->setIconSize(maximumSize());

btn\_calc->setIcon(QIcon("C:/KOMIVOYA/VISUALPIC/6"));

btn\_calc->setIconSize(maximumSize());

*//sizes*

G->button(1)->setFixedSize(80,50);

G->button(2)->setFixedSize(80,50);

btn\_calc->setFixedSize(100,50);

ent1->setFixedSize(100,50);

ent1->setStyleSheet("background-color: rgb(214,200,255)");

ent2->setFixedSize(100,50);

ent2->setStyleSheet("background-color: rgb(214,200,255)");

*//locations*

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

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*,5,1,1,2);

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<5;i++)

{

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

}

G->button(buttonID)->setStyleSheet("QPushButton { background-color: rgb(154,154,154); }\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*;

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]=INT\_MAX;

*while* (flag)

{

QVector <int> row(len);

row.fill(INT\_MAX);

QVector <int> column(len);

column.fill(INT\_MAX);

*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]!=INT\_MAX && data[i][k]!=INT\_MAX) 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]!=INT\_MAX && data[i][k]!=INT\_MAX) 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=INT\_MAX;

int min2=INT\_MAX;

*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==INT\_MAX || min2==INT\_MAX || 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]=INT\_MAX;

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

{

data[max\_mark\_x][i]=INT\_MAX;

data[i][max\_mark\_y]=INT\_MAX;

}

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]!=INT\_MAX) flag=*true*;

}

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

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

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]==x)

{

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

x=result\_arr\_2[i];

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

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

*break*;

}

}

result[0]=' ';

result[1]=' ';

result+="\nОтвет: "+QString::number(sum);

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

answer->setText(result);

}

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

{

*delete* ui;

}

Myglwiget.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**()

{

*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;

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

}

*else*

{

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

graph2.push\_back(a);

*//matrix*

matrix[mem\_num][i]=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(168,142,229));

back=QBrush(QColor(214,200,255));

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);

}

*//текст*

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+1),

*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));

}

}

Работа кода:  
