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 19**

Тема: Графы. Алгоритм Дейкстры

Выполнил работу

студент группы РИС-20-1б

Томилов В.А.

Проверила

Доцент кафедры ИТАС

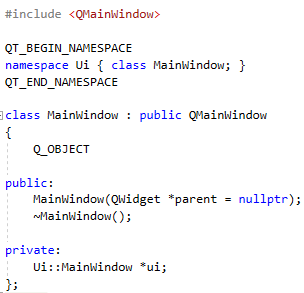
к.т.н ПоляковаО.А.

Пермь, 2021

Постановка задачи

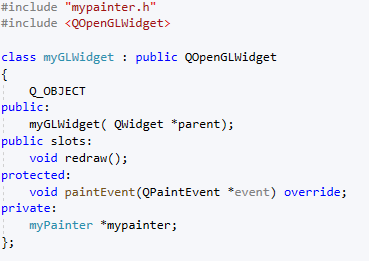
Задача: найти кратчайшие расстояния от вершины Х до всех остальных вершин графа.

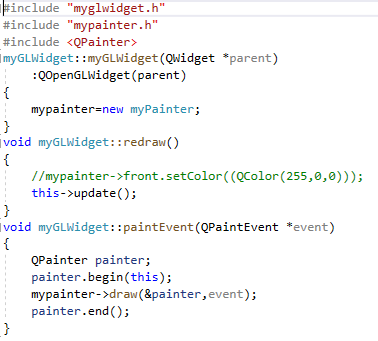
Анализ задачи

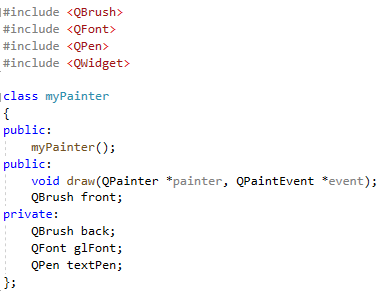
* 1. Для решения задачи необходимо
  2. Добавить в проект файл mainwindow.h, содержащий описание класса.
  3. Добавим в проект файл mainwindow.cpp, в котором задаются параметры фона приложения, описания методов.

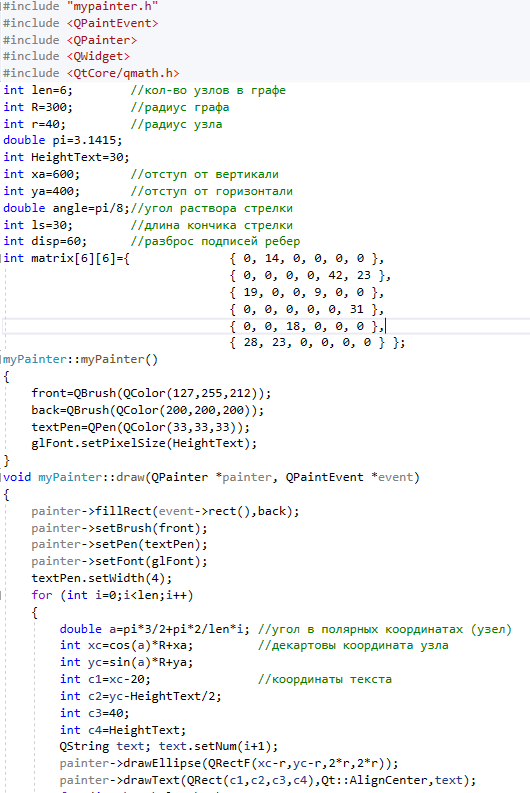


* 1. Добавить в проект файлы myglwidget.h/.cpp и mypainter.h/.cpp, в которых содержится описание классов форм, отвечающих за отрисовку и перерисовку графа.



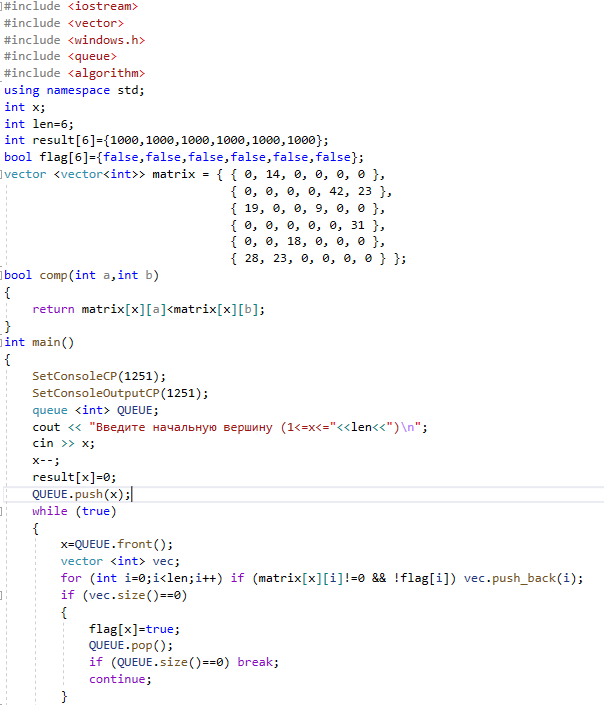


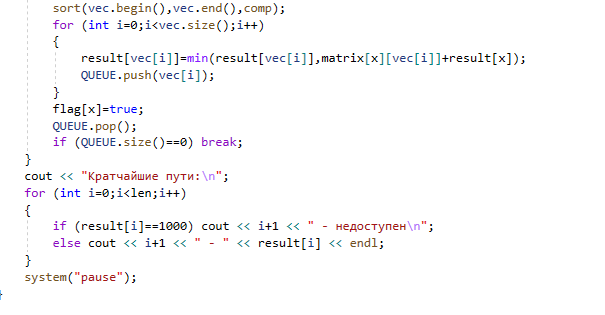


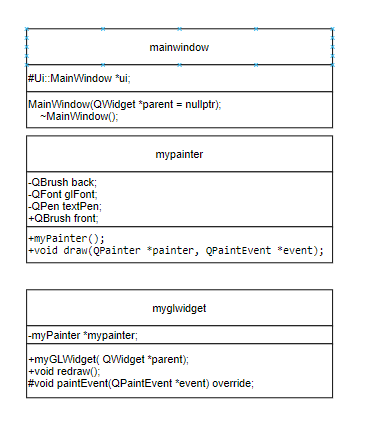




* 1. Добавить в проект файл Lab21\_main.cpp, содержащий основную программу, а именно за основной рассчет кратчайших путей от заданной вершины по алгоритму Дейкстры.





UML – диаграмма

Код программы

Main.cpp

#include <iostream>

#include <vector>

#include <windows.h>

#include <queue>

#include <algorithm>

using namespace std;

int x;

int len=6;

int result[6]={1000,1000,1000,1000,1000,1000};

bool flag[6]={false,false,false,false,false,false};

vector <vector<int>> matrix = { { 0, 14, 0, 0, 0, 0 },

{ 0, 0, 0, 0, 42, 23 },

{ 19, 0, 0, 9, 0, 0 },

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

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

{ 28, 23, 0, 0, 0, 0 } };

bool comp(int a,int b)

{

return matrix[x][a]<matrix[x][b];

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

queue <int> QUEUE;

cout << "Введите начальную вершину (1<=x<="<<len<<")\n";

cin >> x;

x--;

result[x]=0;

QUEUE.push(x);

while (true)

{

x=QUEUE.front();

vector <int> vec;

for (int i=0;i<len;i++) if (matrix[x][i]!=0 && !flag[i]) vec.push\_back(i);

if (vec.size()==0)

{

flag[x]=true;

QUEUE.pop();

if (QUEUE.size()==0) break;

continue;

}

sort(vec.begin(),vec.end(),comp);

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

{

result[vec[i]]=min(result[vec[i]],matrix[x][vec[i]]+result[x]);

QUEUE.push(vec[i]);

}

flag[x]=true;

QUEUE.pop();

if (QUEUE.size()==0) break;

}

cout << "Кратчайшие пути:\n";

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

{

if (result[i]==1000) cout << i+1 << " - недоступен\n";

else cout << i+1 << " - " << result[i] << endl;

}

system("pause");

}

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

Mainwindow.cpp

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "myglwidget.h"

#include <QPainter>

#include <QGridLayout>

#include <QPushButton>

MainWindow::MainWindow(QWidget \*parent)

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

{

ui->setupUi(this);

myGLWidget \*openGLW=new myGLWidget(this);

QGridLayout \*grid=new QGridLayout();

QPushButton \*btn=new QPushButton();

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

grid->addWidget(btn,1,0);

ui->centralwidget->setLayout(grid);

connect(btn,&QPushButton::clicked,openGLW,&myGLWidget::redraw);

}

MainWindow::~MainWindow()

{

delete ui;

}

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;

private:

myPainter \*mypainter;

};

#endif // MYGLWIDGET\_H

Myglwidget.cpp

#include "myglwidget.h"

#include "mypainter.h"

#include <QPainter>

myGLWidget::myGLWidget(QWidget \*parent)

:QOpenGLWidget(parent)

{

mypainter=new myPainter;

}

void myGLWidget::redraw()

{

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

this->update();

}

void myGLWidget::paintEvent(QPaintEvent \*event)

{

QPainter painter;

painter.begin(this);

mypainter->draw(&painter,event);

painter.end();

}

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

Mypainter.cpp

#include "mypainter.h"

#include <QPaintEvent>

#include <QPainter>

#include <QWidget>

#include <QtCore/qmath.h>

int len=6; //кол-во узлов в графе

int R=300; //радиус графа

int r=40; //радиус узла

double pi=3.1415;

int HeightText=30;

int xa=600; //отступ от вертикали

int ya=400; //отступ от горизонтали

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

int ls=30; //длина кончика стрелки

int disp=60; //разброс подписей ребер

int matrix[6][6]={ { 0, 14, 0, 0, 0, 0 },

{ 0, 0, 0, 0, 42, 23 },

{ 19, 0, 0, 9, 0, 0 },

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

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

{ 28, 23, 0, 0, 0, 0 } };

myPainter::myPainter()

{

front=QBrush(QColor(127,255,212));

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

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

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(4);

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

{

double a=pi\*3/2+pi\*2/len\*i; //угол в полярных координатах (узел)

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

int yc=sin(a)\*R+ya;

int c1=xc-20; //координаты текста

int c2=yc-HeightText/2;

int c3=40;

int c4=HeightText;

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

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

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

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

{

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

{

//линия

double a2=pi\*3/2+pi\*2/len\*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);

//стрелка

double b;

if (xc2==xl2 && yl2<yc2) b=pi\*3/2;

else if (yc2==yl2 && xl2<xc2) b=pi;

else if (xc2==xl2 && yl2>yc2) b=pi/2;

else if (yc2==yl2 && xl2>xc2) b=0;

else if (xl2>xc2 && yl2<yc2) b=atan((yc2-yl2)/(xc2-xl2))+pi\*2;

else if (xl2<xc2 && yl2<yc2) b=atan((yc2-yl2)/(xc2-xl2))+pi;

else if (xl2<xc2 && yl2>yc2) b=atan((yc2-yl2)/(xc2-xl2))+pi;

else if (xl2>xc2 && yl2>yc2) b=atan((yc2-yl2)/(xc2-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

c1=(xl2+xl1)/2-20+qrand()%disp-disp/2;

c2=(yl2+yl1)/2-HeightText/2+qrand()%disp-disp/2;

c3=40;

c4=HeightText;

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

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

}

}

}

}