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.

Графы.

Вариант 21.

Выполнил:

Студент группы ИВТ-20-2б

Галинов Олег Юрьевич

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

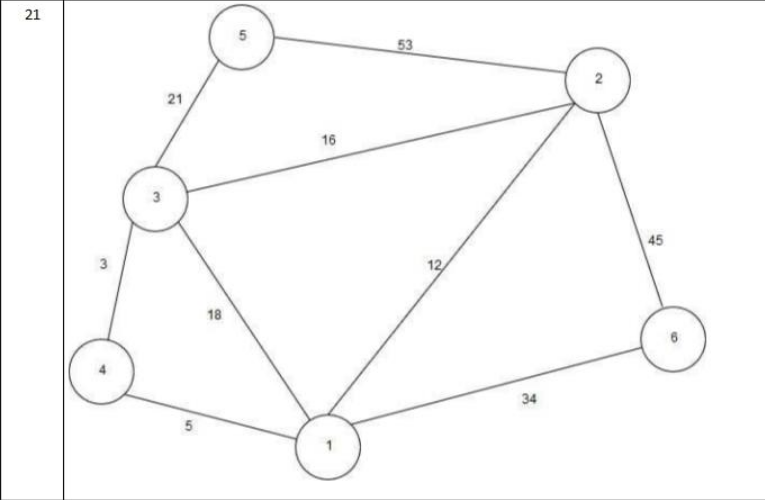
Пермь, 2021

**Введение**

Задача состоит в том, чтобы на заданном графе найти самый короткий путь от вводимого узла. Также требуется отрисовать граф из варианта. Используемый язык программирования – C++.

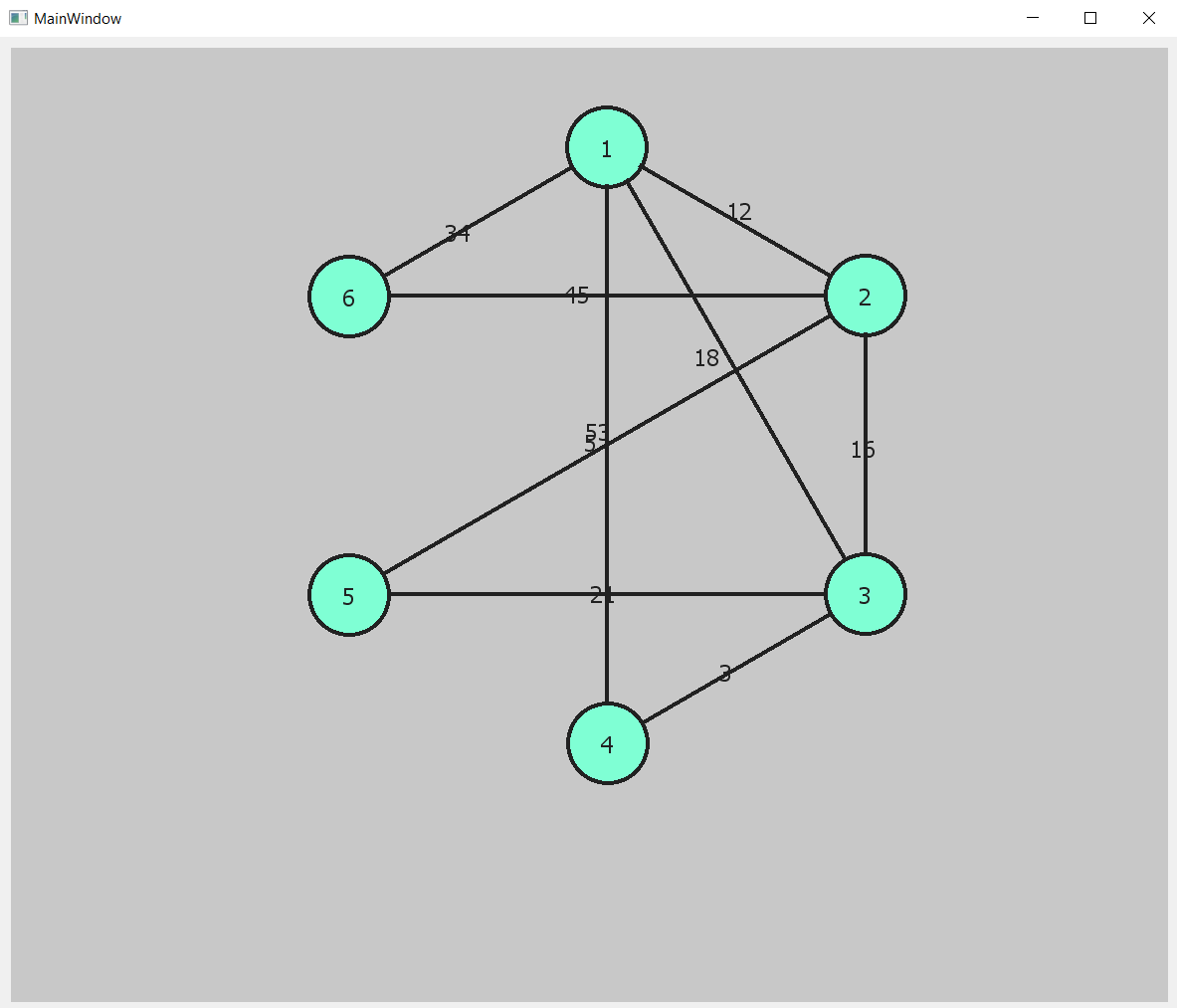
**Постановка задачи**

Задан следующий граф.

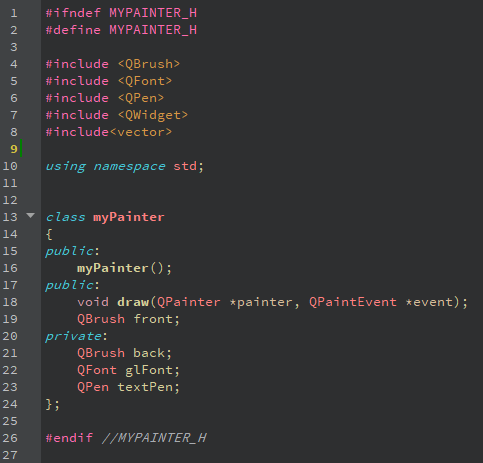
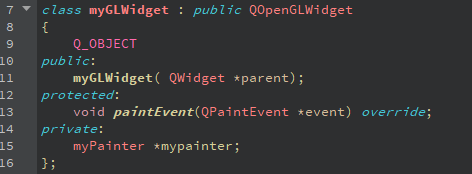


**Анализ задачи**

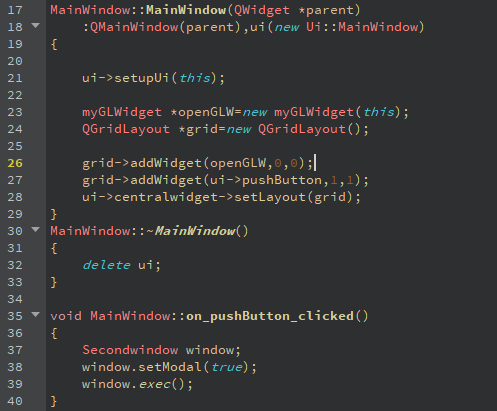
Интерфейс:



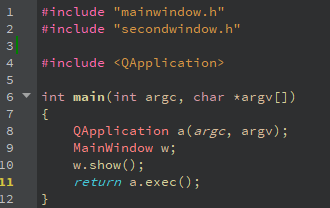
1. Описание алгоритма работы.
   1. Отрисовка. Узлы графа располагаются на окружности. Между узлами рисуются дороги.
2. Описание структуры программы.
   1. Класс mypainter. Наследуется от класса myglwidget. В нем описаны методы для отрисовки графа.

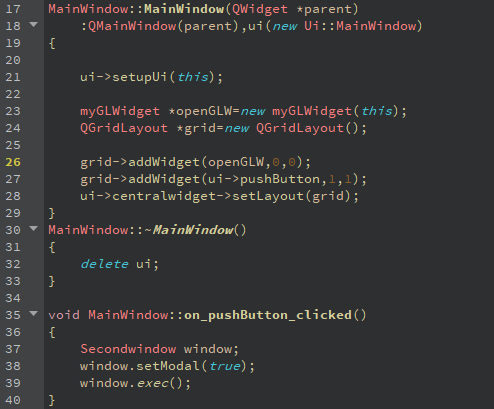
* 1. Класс MainWindow описывает окно. Одним из полей является ссылка на экземпляр класса myglwidget.



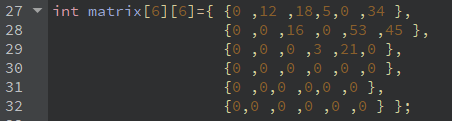
1. Функция main(). Нет отличий от того, что генерируется по умолчанию.



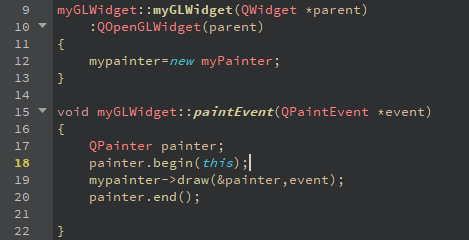
1. Объяснения результатов работы программы.
   1. В конструкторе MainWindow происходит создание экземпляра класса myglwidget и последующий вызов методов для отрисовки графа.



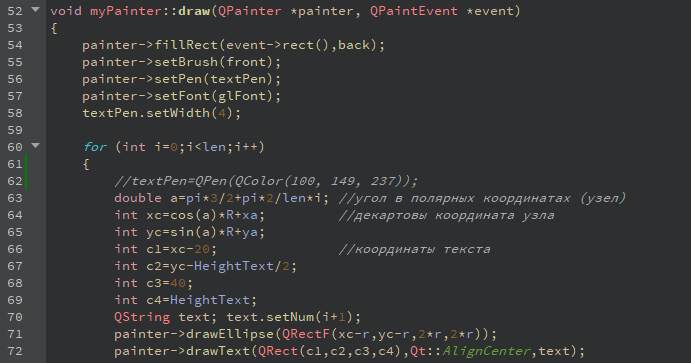
* 1. В файле mypainter.cpp по умолчанию прописывается матрица – расстояния между узлами. Для корректной работы пришлось заменить дубли нулями, то есть если в матрице уже есть путь 1-4, то путь 4-1 сделать равным 0.



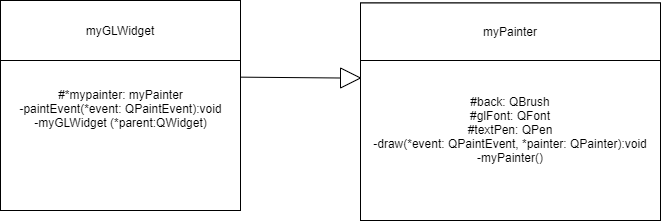
* 1. Вызов метода отрисовки.



* 1. Пример отрисовки узлов, вызванный в draw.



**UML-диаграмма**



**Код**

#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*

#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;

}

#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*

#include "myglwidget.h"

#include "mypainter.h"

#include <QPainter>

#include<vector>

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();

}

#ifndef MYPAINTER\_H

#define MYPAINTER\_H

#include <QBrush>

#include <QFont>

#include <QPen>

#include <QWidget>

#include<vector>

*using* *namespace* std;

*class* **myPainter**

{

*public*:

**myPainter**();

*public*:

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

QBrush front;

*private*:

QBrush back;

QFont glFont;

QPen textPen;

};

#endif *//MYPAINTER\_H*

#include "mypainter.h"

#include <QPaintEvent>

#include <QPainter>

#include <QWidget>

#include <QtCore/qmath.h>

#include<vector>

#include <QFile>

#include <QTextStream>

#include <QProcess>

#include<algorithm>

#include<fstream>

*using* *namespace* std;

int len = 6;

int R=300; *//радиус* *графа*

int r=40; *//радиус* *узла*

double pi=3.1415;

int HeightText=23;

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=40; *//разброс* *подписей* *ребер*

int amountOfLinesInFile = 0;

vector<string>str;

int matrix[6][6]={ {0 ,12 ,18,5,0 ,34 },

{0 ,0 ,16 ,0 ,53 ,45 },

{0 ,0 ,0 ,3 ,21,0 },

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

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

{0,0 ,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);

painter->drawLine(0,0,0,0);

painter->drawLine(0,0,0,0);

*//текст*

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

}

}

}

}

#include "mainwindow.h"

#include <QApplication>

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

{

QApplication a(*argc*, argv);

MainWindow w;

w.show();

*return* a.exec();

}