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**

Тема: «Разработка калькулятора и решение задачи коммивояжера»

Вариант 10

Выполнил:

студент группы РИС-20-2б

Назаров Р.Р.

Проверила:

доцент кафедры ИТАС

Полякова О. А.

Пермь, 2021

**Введение**

На основе фреймворка Qt и среды разработки Qt creator были реализованы приложение калькулятор, вычисляющий логарифмы, и приложение, которое строит графы, и по построенному графу решает задачу коммивояжера.

**Творческая часть 1,**

**Разработка калькулятора.**

**Постановка задачи**

1. Разработать алгоритм калькулятора в соответствии с заданием.

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.

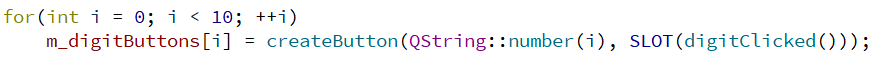
|  |  |  |
| --- | --- | --- |
| 10 | Логарифмы чисел при различных основаниях. | Сложение, вычитание, умножение, деление, возведение в степень, переход к другому основанию. |

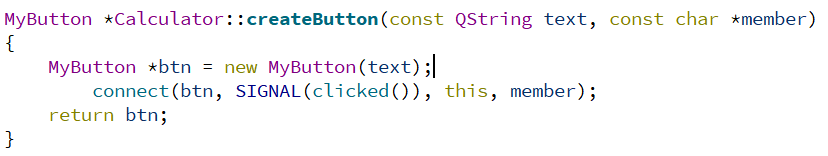
**Анализ задачи**

1. Для ввода и вывода числовых значений необходимо:

1.1 Создать кнопки для цифр от 0 до 9 – для ввода.

Для этого задается сигнал кнопок clicked(), который активирует функцию digitClicked()





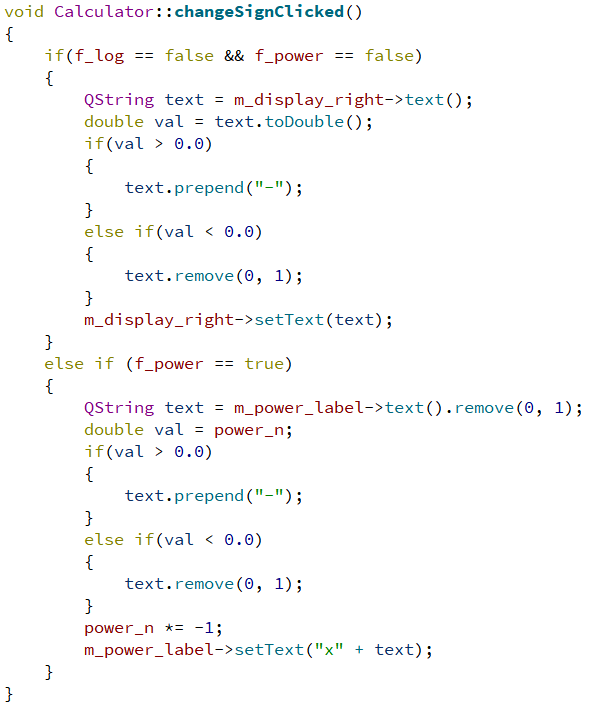
* 1. Создать объект QLineEdit – для вывода введенных значений.
  2. Разработать функцию digitClicked() внутри которой в объект Qlabel или QLineEdit вставляется текст нажатой кнопки, в зависимости от активности строки. Так же имеется проверка на активную строку.



* 1. Разработать функцию pointClicked() для ввода вещественных чисел, а именно добавление точки и проверка на ее наличие в строке.



1.5 Реализовать функцию changeSignClicked() для изменения знака числа в строке.



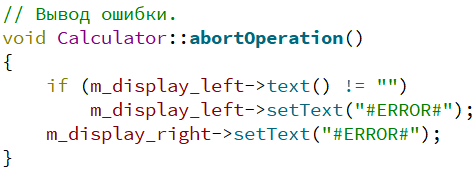
1.6Разработать функцию doubleOperatorClicked() для операций «+» «-» «\*» «/»



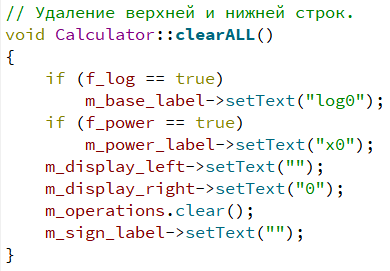
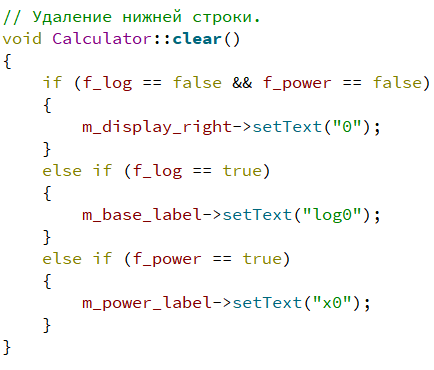
1.7 Реализовать функцию equalClicked() для вычисления значения операции

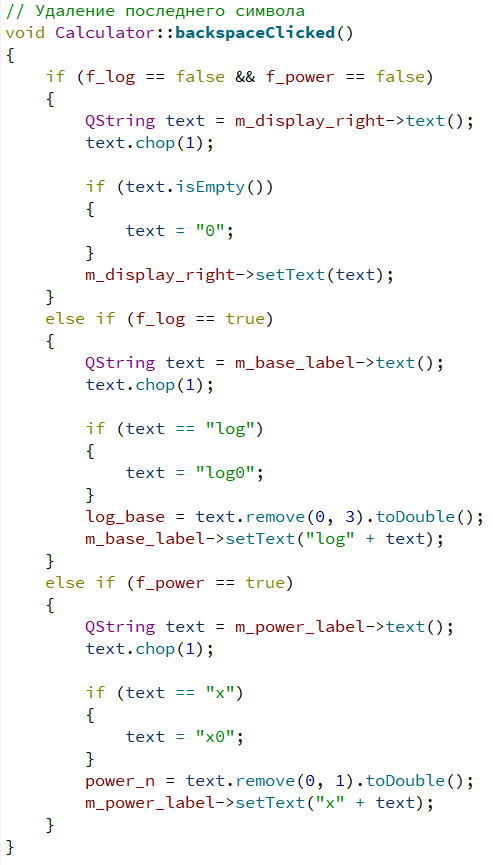


Где функция abortOperation(), реагирует на ошибку, т.е. при делении на ноль



1.8 Реализовать функции backspaceClicked() clear() clearAll() для очищения строк

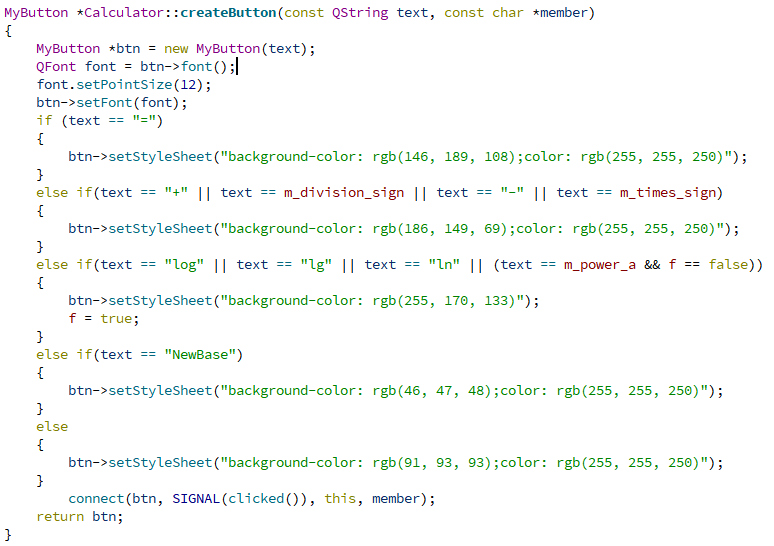
 



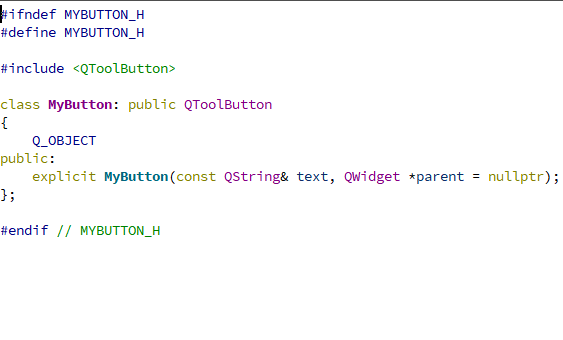
1.9 Реализовать функцию unaryOperatorClicked() для вычисления натурального и десятичного логарифма, логарифма n-го основания, и числа n-ой степени.



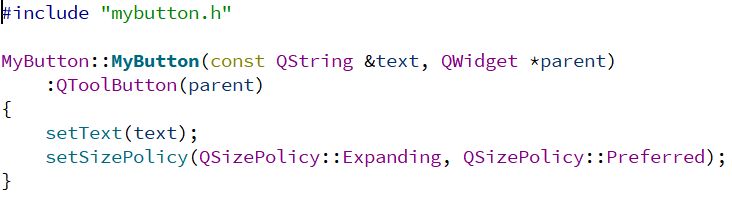
2. Для соединения кнопок использовался класс MyButton, для его использования была реализована функция createButton(), в которую заносился текст, который будет выводить кнопка, и указатель на функцию которую необходимо выполнить.



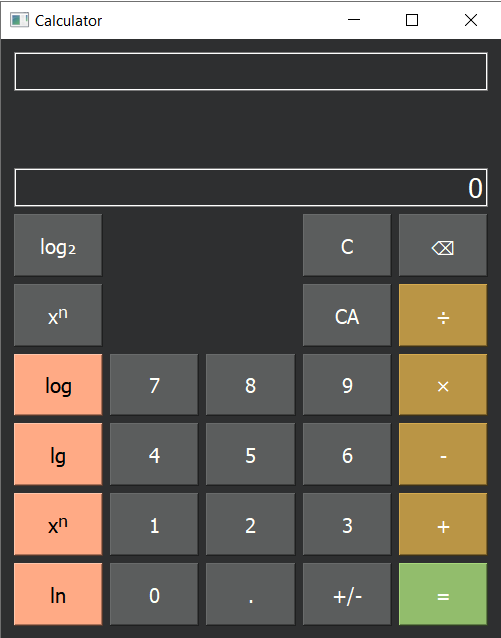
---------------------------------------



---------------------------------------

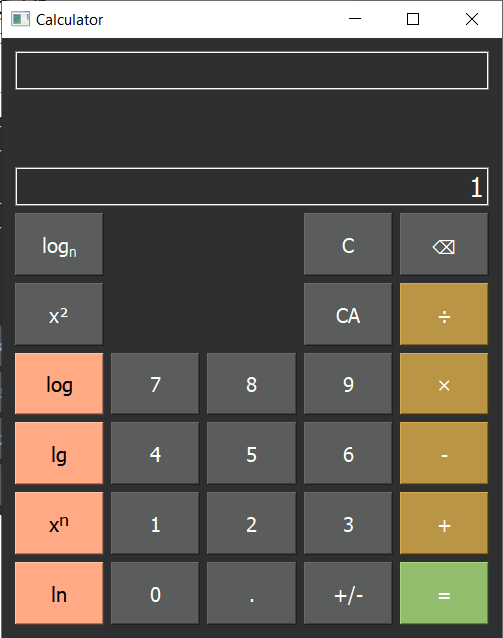
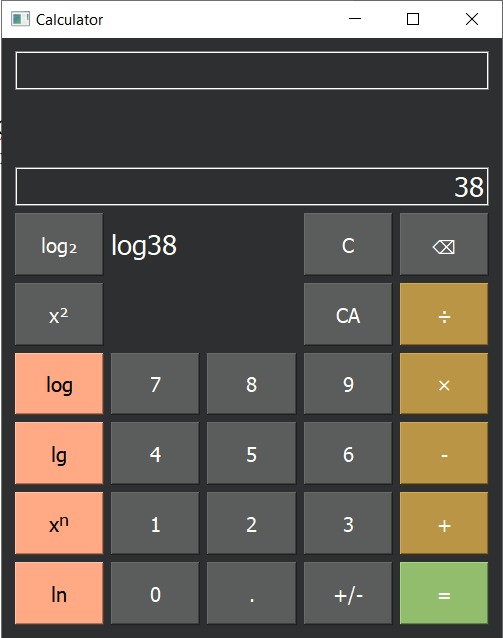
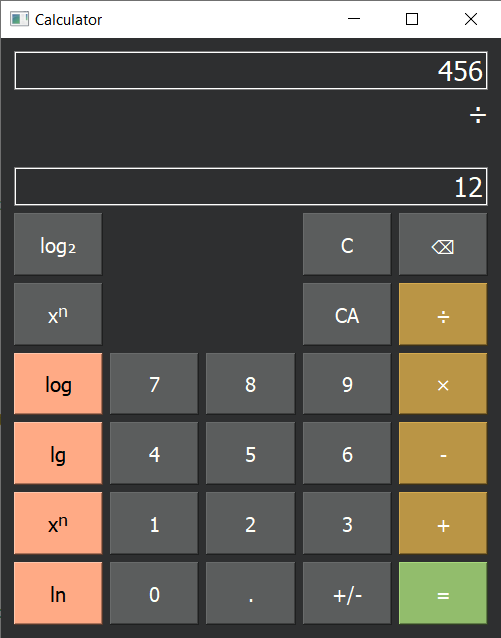


**Интерфейс приложения**

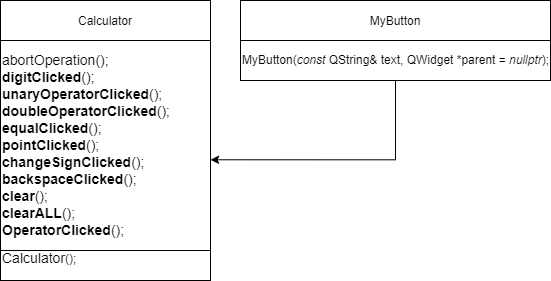
****

Интерфейс приложения был сделан на объекте QGridLayout, и выравнивание всех объектов было автоматическим. Каждой кнопке стиль был задан с помощью функции setStyleSheet().

**Скриншоты приложения**



**UML-Диаграмма**



**Творческая работа часть 2**

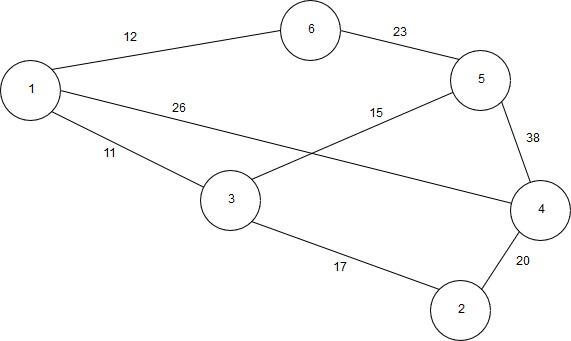
**Разработка приложения для визуализации и решения задачи коммивояжера**

**Постановка задачи**

1.Визуализировать граф.

2.Решить задачу коммивояжера.

**Анализ задачи**



1.Для создания графа необходимо:

1.1 Реализовать таблицу смежности. Для создания таблицы используется объект tableWidget класса QTableWidget

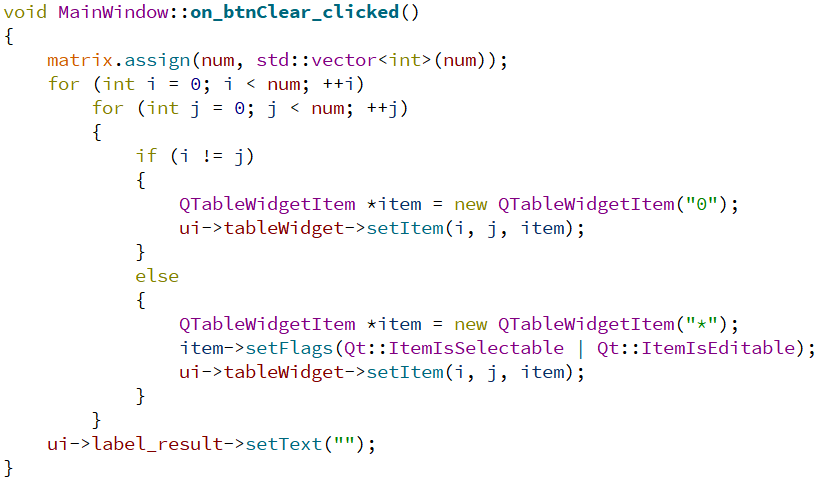


1.2 Разработать функции on\_btnAdd\_clicked() on\_btnDel\_clicked() для изменения таблицы смежности





1.3 Разработать функцию on\_btn\_clicked() для очистки таблицы смености



1.4 Разработать функцию paintEvent(), которая рисует граф с помощью объекта painter класса QPainter, на созданном объекте openGLW класса QOpenGLWidget

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

{

if(ris==true)

{

ris = false;

QPainter painter(ui->openGLW);

front=QBrush(QColor(186, 149, 69));

back=QBrush(QColor(66, 67, 68));

textPen=QPen(QColor(255, 255, 250));

painter.setBrush(front);

painter.setFont(glFont);

painter.setPen(textPen);

textPen.setWidth(2);

painter.begin(ui->openGLW);

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

{

double p = (2 \* M\_PI) / num\*i + M\_PI \* 3.0/2.0; // Расположение вершин от точки (pointOx;pointOy) в полярной системе координат

double xc = pointOx + R \* cos(p); //Координаты вершины по x

double yc = pointOy + R \* sin(p); //Координаты вершины по y

QString text; text.setNum(i+1); //Текст в вершине

textPen.setColor(QColor(155, 155, 250));

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

textPen.setColor(QColor(255, 255, 250));

painter.drawText(QRect(xc+20, yc+20, 40, 30),Qt::AlignCenter, text);

for (int j = 0; j < num; ++j)

{

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

{

textPen.setColor(QColor(255, 255, 250));

double p2 = (2 \* M\_PI) / num\*j + M\_PI \* 3.0/2.0;

double xc2 = pointOx + R \* cos(p2);

double yc2 = pointOx + R \* sin(p2);

double l = sqrt(pow((xc2 - xc), 2) + pow((yc2 - yc), 2));

int xl = (xc2 - xc)/l\*r + xc; // Координаты х по первой точке

int yl = (yc2 - yc)/l\*r + yc; // Координаты y по первой точке

int xl2 = (xc - xc2)/l\*r + xc2; // Координаты x по второй точке

int yl2 = (yc - yc2)/l\*r + yc2; // Координаты y по второй точке

painter.drawLine(xl + r, yl + r, xl2 + r, yl2 + r);

double b = 0;

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

else if (yc2==yl2 && xl2<xc2) b=M\_PI;

else if (xc2==xl2 && yl2>yc2) b=M\_PI\_2;

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

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

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

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

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

int xs1=cos(b + M\_PI/8)\*ls+xl2;

int ys1=sin(b + M\_PI/8)\*ls+yl2;

int xs2=cos(b - M\_PI/8)\*ls+xl2;

int ys2=sin(b - M\_PI/8)\*ls+yl2;

painter.drawLine(xl2 + r, yl2 + r, xs1 + r, ys1 + r);

painter.drawLine(xl2 + r, yl2 + r, xs2 + r, ys2 + r);

double cx = (xl2 + xl) / 2;

double cy = (yl2 + yl) / 2;

cx = (cx + xl)/2;

cy = (cy + yl)/2;

cx = (cx + xl2)/2;

cy = (cy + yl2)/2;

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

textPen.setColor(QColor(155, 155, 250));

if (matrix[i][j] == matrix[j][i])

{

painter.drawText(QRectF((xl2 + xl) / 2 + 18, (yl2 + yl) / 2 + 27, 20, 15), Qt::AlignCenter, text);

}

else

{

painter.drawText(QRectF(cx + 25, cy + 25, 20, 15), Qt::AlignCenter, text);

}

}

}

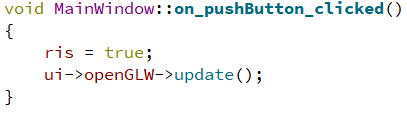
}

painter.end();

}

}

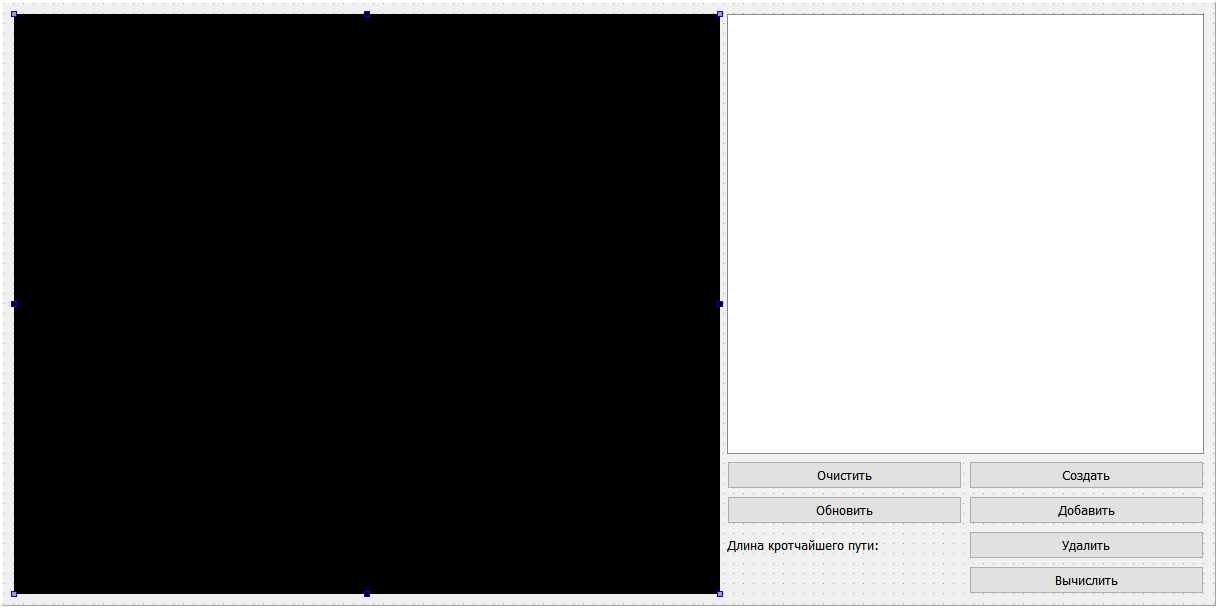
1.5 Разработать функцию для обновления дисплея OpenGLW



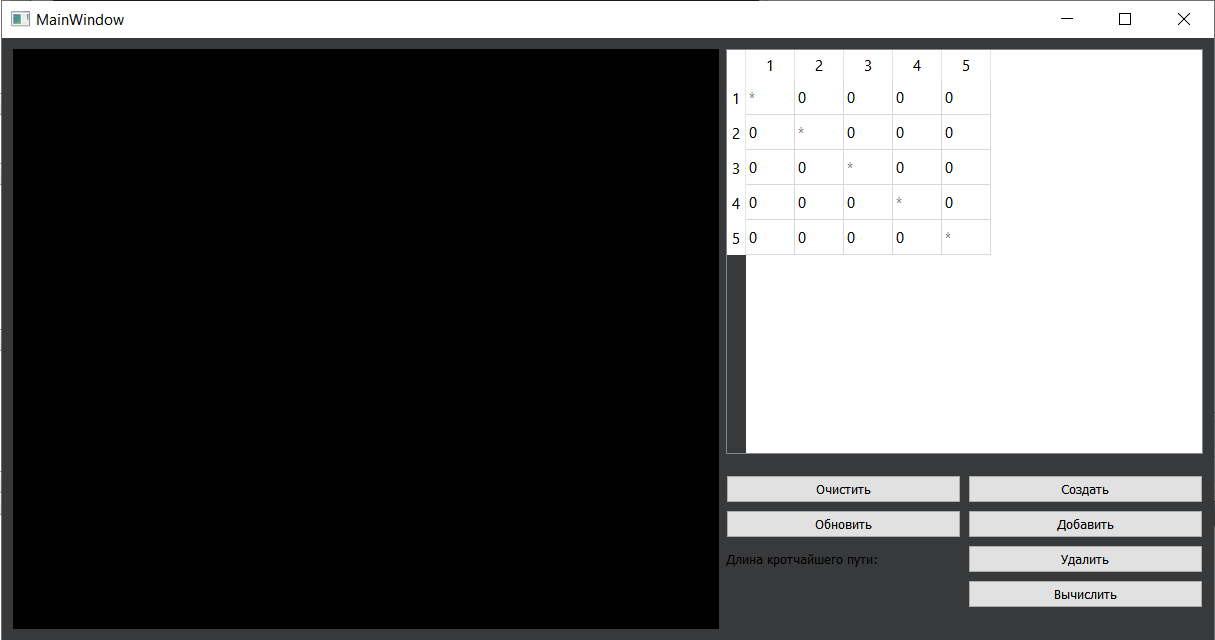
1.6 Разработать функцию on\_btnResult\_clicked() для нахождения кротчайшего пути.

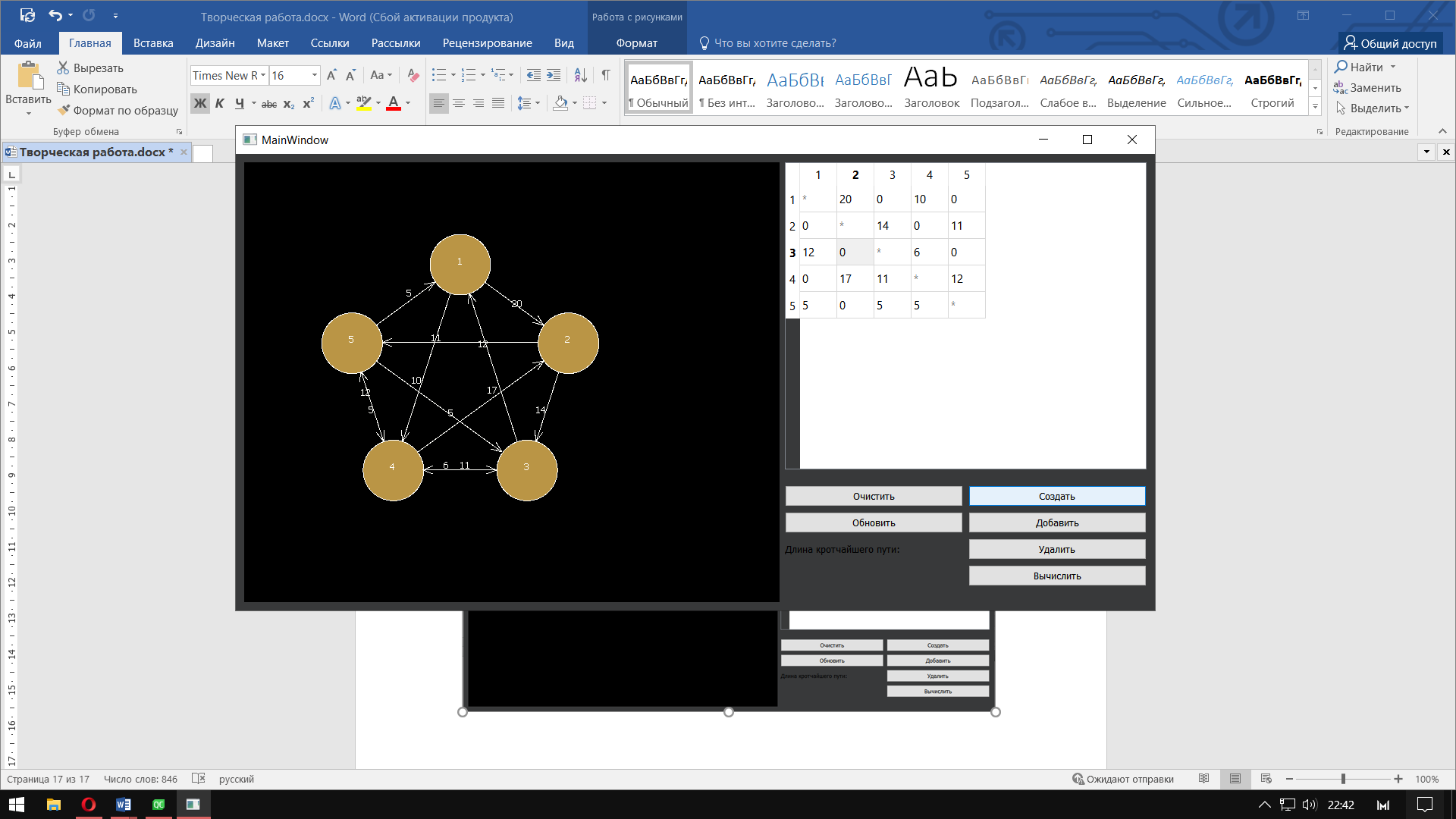


**Интерфейс приложения**

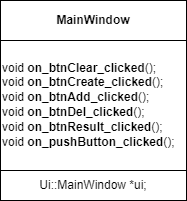
****

**Скриншоты программы**

****



**UML-Диаграмма**



**Ссылки:**

[**https://github.com/Rush-IT/Study**](https://github.com/Rush-IT/Study)

**Qt-документация:**

https://doc.qt.io