Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЁТ**

**По творческой работе**

Дисциплина: «Основы теории алгоритмов и структуры данных»

Тема: Разработка калькулятора и Решение задачи Коммивояжёра

Вариант 13

Выполнила:

Студент группы ИВТ-20-2б

Тарасова Алиса Кирилловна

Проверила:

Доцент кафедры ИТАС

Полякова О.А

**Пермь,2021**

1. **Постановка задачи**
2. Разработать алгоритм калькулятора в соответствии с вариантом задания.
3. Разработать алгоритм решения задачи Коммивояжёра.
4. Реализовать алгоритмы в программах.
5. Разработать интерфейс с помощью QT.

**2. Анализ задачи**

**2.1 Реализация калькулятора**

**1.Типы данных**

* 1. Были использованы типы данных int, double и string.

**2. Какие с этими данными были выполнены действия**

**2.1** Глобальная переменная i типа int хранит информацию о том в какой из объектов виджета QLineEdit будет передаваться информация, изначально она равна 1 и указывает на первое окно для ввода.

**2.2** Переменные типа double преобразовывались из типа string для хранения коэффициентов при x, интервале и точности вычисления, для подсчёта корня, смены знака числа и для решения задачи методом половинного деления.

**2.3** Переменные типа string являются вводом по умолчанию в Qt в объект line\_edit.

**3.В каком виде эти данные будут представлены для решения задачи**

**3.1** Данные будут представлены в виде объектов класса.

Count Num

**4.Класс Calculator**

**4.1** Поля и методы класса представлены ниже:

class **Calculator** : public QMainWindow

{

Q\_OBJECT

public:

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

~***Calculator***();

private:

Ui::Calculator \*ui;

private slots:

void **digits\_numbers**();

void **on\_pushButton\_equally\_clicked**();

void **on\_pushButton\_point\_clicked**();

void **on\_pushButton\_replace\_clicked**();

void **on\_pushButton\_next\_clicked**();

void **on\_pushButton\_e\_clicked**();

void **on\_pushButton\_clear\_clicked**();

void **on\_pushButton\_clear\_2\_clicked**();

void **on\_pushButton\_root\_clicked**();

};

**5.Операторы с помощью которых будет организован ввод и вывод**

**5.1** Ввод данных происходит при нажатии кнопок калькулятора объектов класса QPushButton. После этого вызывается соответствующий метод, в котором с помощью метода text() мы получаем информацию из окна в который был произведён ввод.

void **on\_pushButton\_equally\_clicked**();

void **on\_pushButton\_point\_clicked**();

void **on\_pushButton\_replace\_clicked**();

void **on\_pushButton\_next\_clicked**();

void **on\_pushButton\_e\_clicked**();

void **on\_pushButton\_clear\_clicked**();

void **on\_pushButton\_clear\_2\_clicked**();

**void on\_pushButton\_root\_clicked();**

**5.2** Данные выводятся с помощью метода setText().

ui->lineEdit\_show->setText(QString::number(x0));

**6.Действия с помощью которых будут решены поставленные задачи.**

**6.1** Сначала я создаю собственный слот, который в дальнейшем реализую в исходном файле Calculator.cpp, назвала я его digits\_numbers(). В Calculator.cpp. происходит соединение нажатия на цифры с методом с помощью connect.

private slots:

void **digits\_numbers**();

connect(ui->pushButton\_0,SIGNAL(clicked()),this,SLOT(digits\_numbers())) ;

connect(ui->pushButton\_1,SIGNAL(clicked()),this,SLOT(digits\_numbers())) ;

...

**6.2** Создадим переменную указатель button, которая будет наследоваться от класса QPushButton , а метод sender() позволяет получить информацию о том какая кнопка была нажата и будет преобразовываться к классу QPushButton. Т.к. в lineEdit записываются числа, то предусмотрим, чтобы не было в один lineEdit записано число и е(экспонента). Следовательно, с помощью условий мы определяем определенный lineEdit и исключаем запись вместе с е. В переменную lable присваиваем текст с помощью метода text() и потом устанавливаем в lineEdit текст с помощью метода setText.

void Calculator::**digits\_numbers**()

{

QPushButton \*button=(QPushButton\*) sender();

if(i==1 && ui->lineEdit\_1->text()!="e")

{

QString lable=ui->lineEdit\_1->text()+button->text();

ui->lineEdit\_1->setText(lable);

}

else if(i==2 && ui->lineEdit\_2->text()!="e")

{

QString lable=ui->lineEdit\_2->text()+button->text();

ui->lineEdit\_2->setText(lable);

}

...

**6.3** В методе **on\_pushButton\_next\_clicked**() будет изменяться i, следовательно будет передвигаться запись чисел на другой из объектов виджета QLineEdit.

void Calculator::**on\_pushButton\_next\_clicked**()

{

if(i<8)

i=i+1;

}

**6.4** В методе on\_pushButton\_point\_clicked() при нажатии точки реализуется её написание единожды с помощью метода contains, который определяет есть ли данный текст в объекте виджета QLineEdit и исключает её написание с e, а также написание её без цифр. Всё это пропишем в условии с учётом i, если условия выполняются, то производится запись точки в нужный lineEdit.

void Calculator::**on\_pushButton\_point\_clicked**()

{

if (i==1 && !(ui->lineEdit\_1->text().contains('.')) && ui->lineEdit\_1->text()!=0 && ui->lineEdit\_1->text()!="e")

ui->lineEdit\_1->setText(ui->lineEdit\_1->text()+'.');

else if (i==2 && !(ui->lineEdit\_2->text().contains('.')) && ui->lineEdit\_2->text()!=0 && ui->lineEdit\_2->text()!="e")

ui->lineEdit\_2->setText(ui->lineEdit\_2->text()+'.');

...

**6.5** В методе on\_pushButton\_replace\_clicked(), который меняет знак введённых чисел в lineEdit. С помощью условия учитывается i и с помощью дополнительной переменной numbers, в которую присваивается текст из lineEdit приведённый к типу toDouble. Дальше производится смена знака и это записывается, обратно преобразуя к типу QString.

void Calculator::**on\_pushButton\_replace\_clicked**()

{

double numbers;

if(i==1)

{

numbers=(ui->lineEdit\_1->text()).toDouble();

numbers=numbers\*-1;

ui->lineEdit\_1->setText(QString::number(numbers));

}

else if(i==2)

{

numbers=(ui->lineEdit\_2->text()).toDouble();

numbers=numbers\*-1;

ui->lineEdit\_2->setText(QString::number(numbers));

}

...

**6.6** В методе on\_pushButton\_e\_clicked() в условии учитываем i и пустоту нашего lineEdit, следовательно, если все условия соблюдены то производится запись в lineEdit е.

void Calculator::**on\_pushButton\_e\_clicked**()

{

if (i==1 && ui->lineEdit\_1->text()==0)

ui->lineEdit\_1->setText("e");

else if (i==2 && ui->lineEdit\_2->text()==0)

ui->lineEdit\_2->setText("e");

else if (i==3 && ui->lineEdit\_3->text()==0)

ui->lineEdit\_3->setText("e");

else if (i==4 && ui->lineEdit\_4->text()==0)

ui->lineEdit\_4->setText("e");

else if (i==5 && ui->lineEdit\_5->text()==0)

ui->lineEdit\_5->setText("e");

else if (i==8 && ui->lineEdit\_accuracy->text()==0)

ui->lineEdit\_accuracy->setText("e");

}

**6.7** В методе on\_pushButton\_clear\_clicked() отвечает за стирание элементов в lineEdit, в котором производится запись. С помощью условия и i определяем, где нужно произвести стирание чисел.

void Calculator::**on\_pushButton\_clear\_clicked**()

{

if(i==1)

ui->lineEdit\_1->setText("");

if(i==2)

ui->lineEdit\_2->setText("");

…

**6.8** Метод on\_pushButton\_clear\_2\_clicked() стирает все объекты виджета LineEdit и присваевает i=1, чтобы можно заново было произвести вычисления.

void Calculator::**on\_pushButton\_clear\_2\_clicked**()

{

ui->lineEdit\_1->setText("");

ui->lineEdit\_2->setText("");

ui->lineEdit\_3->setText("");

ui->lineEdit\_4->setText("");

ui->lineEdit\_5->setText("");

ui->lineEdit\_from->setText("");

ui->lineEdit\_befor->setText("");

ui->lineEdit\_accuracy->setText("");

ui->lineEdit\_show->setText("");

i=1;

}

**6.9** Метод on\_pushButton\_root\_clicked() с помощью i также определяет в какой LineEdit вписывать результат. Переменной t присваивается значение, которое вписано в LineEdit, преобразуя его в тип toDouble, затем подсчитываем корень и вписываем обратно преобразуя к типу QString.

void Calculator::**on\_pushButton\_root\_clicked**()

{

double t;

if(i==1)

{

t=ui->lineEdit\_1->text().toDouble();

t=sqrt(t);

ui->lineEdit\_1->setText(QString::number(t));

}

if(i==2)

{

t=ui->lineEdit\_2->text().toDouble();

t=sqrt(t);

ui->lineEdit\_2->setText(QString::number(t));

}

...

**6.10** В методе on\_pushButton\_equally\_clicked() будет реализовываться метод половинного деления. Объявляем переменные и записываем в них значения наших объектов класса QLineEdit, преобразуя их из QString в toDouble. Учитываем, что если значение равно е, то переменной мы присвоим M\_E используя синтаксис Qt. С помощью цикла с пост условием решим задачу методом половинного деления. Также используя функцию F(), в которой записано наше уравнение. x0 берем равный середине АВ. Подсчитываем значение произведений функций если оно меньше нуля, то пересекает Ох, следовательно, на нём есть корень. И в зависимости от этого передвигаем х0, x1 присваиваем х0. Находим новый х0 на середине АВ Т.к в Qt не предусмотрена математическая операция модуля, то реализуем её с помощью условия. Цикл будет выполнятся до тех пор, пока разность между двумя соседними корнями не станет меньше заданного eps.

В lineEdit\_show с помощью метода setText записываем ответ преобразуя тип double в QString.

double **F**(double y1, double y2,double y3,double y4,double y5, double x)

{

return y1\*pow(x,3)+y2\*pow(x,2)+y3\*x+y4+y5;

}

void Calculator::**on\_pushButton\_equally\_clicked**()

{

double eps,a,b,x,x0,x1,y1,y2,y3,y4,y5;

if(ui->lineEdit\_1->text()=="e")

y1=M\_E;

else

y1=ui->lineEdit\_1->text().toDouble();

if(ui->lineEdit\_2->text()=="e")

y2=M\_E;

else

y2=ui->lineEdit\_2->text().toDouble();

if(ui->lineEdit\_3->text()=="e")

y3=M\_E;

else

y3=ui->lineEdit\_3->text().toDouble();

if(ui->lineEdit\_4->text()=="e")

y4=M\_E;

else

y4=ui->lineEdit\_4->text().toDouble();

if(ui->lineEdit\_5->text()=="e")

y5=M\_E;

else

y5=ui->lineEdit\_5->text().toDouble();

if(ui->lineEdit\_accuracy->text()=="e")

eps=M\_E;

else

eps=ui->lineEdit\_accuracy->text().toDouble();

a=ui->lineEdit\_from->text().toDouble();

b=ui->lineEdit\_befor->text().toDouble();

x0=(a+b)/2;

do

{

if(F(y1,y2,y3,y4,y5,a)\*F(y1,y2,y3,y4,y5,x0)<0)

b=x0;

else

a=x0;

x1=x0;

x0=(a+b)/2;

if((x1-x0)<0)

x=(x1-x0)\*-1;

else

x=x1-x0;

}while(x>eps);

ui->lineEdit\_show->setText(QString::number(x0));

}

**2.2 Реализация задачи Коммивояжёра**

**1.Типы данных**

* 1. Были использованы типы данных int, double и string.

**2. Какие с этими данными были выполнены действия**

**2.1** Переменные типа int были использованы для хранения данных таблицы со значениями длин дорог между городами, для работы с циклами, а также для вычисления кратчайшего пути.

**2.2** Переменные типа double использовались для решения задачи.

**2.3** Переменные типа string являются вводом по умолчанию в Qt в объект line\_edit. Также переменная strok типа string хранит траекторию кратчайшего пути.

**3.В каком виде эти данные будут представлены для решения задачи**

**3.1** Данные представлены в виде двумерного массива mas.

int mas[6][6]{

{-1,-1,-1,-1,-1,-1},

{-1,-1,-1,-1,-1,-1},

{-1,-1,-1,-1,-1,-1},

{-1,-1,-1,-1,-1,-1},

{-1,-1,-1,-1,-1,-1},

{-1,-1,-1,-1,-1,-1}

};

**4.Класс** Kommivoyazher

**4.1** Поля и методы класса представлены ниже:

class **Kommivoyazher** : public QMainWindow

{

Q\_OBJECT

public:

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

~***Kommivoyazher***();

private:

Ui::Kommivoyazher \*ui;

protected:

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

private slots:

void **on\_pushButton\_clicked**();

};

**5.Операторы с помощью которых будет организован ввод и вывод**

**5.1** Ввод данных производится пользователем с помощью клавиатуры, изначально введены значения длин дорог по умолчанию, но их можно изменять. Все данные таблицы сохраняются в двумерный массив с помощью метода text().

mas[0][1]=ui->lineEdit\_i0j1->text().toInt();

mas[0][2]=ui->lineEdit\_i0j2->text().toInt();

mas[0][3]=ui->lineEdit\_i0j3->text().toInt();

mas[0][4]=ui->lineEdit\_i0j4->text().toInt();

mas[0][5]=ui->lineEdit\_i0j5->text().toInt();

...

**5.2** Данные выводятся в lineEdit\_sum и lineEdit\_show с помощью метода setText().

ui->lineEdit\_show->setText(strok);

ui->lineEdit\_sum->setText(QString::number(sum));

ui->lineEdit\_show->setText("Нет решения");

ui->lineEdit\_sum->setText("Нет");

**6.Действия с помощью которых будут решены поставленные задачи.**

**6.1** Для отрисовки в Qt подключим класс #include <QPainter>. Добавим в класс class **Kommivoyazher** раздел protected: и переопределим функцию paintEvent, слово override необходимо для переопределения. Вся работа по отрисовки происходит внутри этой функции, она вызывается при каждом объявлении окна, когда окно создаётся и когда мы меняем его размер. Также если отрисовка зависит от внешних событий, например нажатие кнопки, мы будем вызывать её функцией repaint().

protected:

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

**6.2** Создаем объект класса QPainter, некая сцена на которой мы будем рисовать. Дальше пишем painter.begin(this) и painter.end(this). Любое обращение к painter должно находится между этими двумя строками.

QPainter painter;

painter.begin(this);

...

painter.end();

**6.3** Начинаем с отрисовки линий между узлами, чтобы узлы находились сверху линий. С помощью вложенного цикла рисуем линии от каждой дороги до остальных дорог с помощью метода drawLine в конструктор прописываем координаты начальной и конечной точки. Высчитаем мы с помощью функций косинуса и синуса. Th это шаг от одного узла до другого. Для изменения линий в Qt отвечает класс Qpen в скобках мы прописываем цвет, толщину и стиль линии. Встраиваем в pаinter методом setPen.

int l = 180;

double th=2\*M\_PI/6;

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

{

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

{

if(i!=j)

{

int x1 = 260 + cos(th \* i) \* l;

int y1 = 270 - sin(th \* i) \* l;

int x2 = 260 + cos(th \* j) \* l;

int y2 = 270 - sin(th \* j) \* l;

painter.setPen(QPen(Qt::darkCyan, 1, Qt::SolidLine));

painter.drawLine(x1, y1, x2, y2);

}

}

}

**6.4** Дальше отрисовываем круги и прописываем номера узлов с помощью методов drawEllips и drawText. В конструкторе drawEllips прописываем координаты верхнего левого угла прямоугольника, его длину и ширину, в который будет вписан эллипс. В конструкторе drawText прописываем координаты начала текста и сам текст. Также изменяем линию. Для заливки используем класс QBrush в скобках прописываем цвет и стиль заливки. Встраиваем в pаinter методом setBrush.

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

{

painter.setPen(QPen(Qt::darkCyan, 3, Qt::SolidLine));

painter.setBrush(QBrush(Qt::cyan, Qt::SolidPattern));

painter.drawEllipse(240 + cos(th \* i) \* l, 250 - sin(th \* i) \* l, 50, 50);

painter.setPen(QPen(Qt::black, 5, Qt::SolidLine));

painter.drawText(260 + cos(th \* i) \* l, 280 - sin(th \* i) \* l, QString::number(i+1));

}

**6.5** Изначально пользовать задаёт длины путей и номер дороги по которой будет производится поиск кратчайшего пути. Нажимая кнопку «Решить» будет вызван метод on\_pushButton\_clicked(), где в двумерный массив (матрица смежности), который объявлен глобальным будут записаны все введённые данные. Дальше создаем ещё один массив, копию ранее заданного массива.

**…**

mas[5][2]=ui->lineEdit\_i5j2->text().toInt();

mas[5][3]=ui->lineEdit\_i5j3->text().toInt();

mas[5][4]=ui->lineEdit\_i5j4->text().toInt();

int mas\_copy[6][6];

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

{

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

{

mas\_copy[i][j]=mas[i][j];

}

}

**6.6** В n записываем номер дороги при этом преобразуя переменную типа string в целочисленную. Дальше вызывается функция Paths(), где методом ветвей и границ будем искать кратчайший путь.

n= ui->lineEdit\_number->text().toInt();

Paths();

**6.7** В функции Paths(). В массив mas\_di[6] будем записывать минимумы строк, а в mas\_dj [6] минимумы столбцов. С помощью вложенных циклов заполняем массивы.

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

{

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

{

if (mas[i][j] < min && mas[i][j] != -1)

min = mas[i][j];

}

mas\_di[i] = min;

if (min == 10000)

mas\_di[i] = 0;

min = 10000;

}

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

{

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

{

if (mas[i][j] != -1)

mas[i][j] = mas[i][j] - mas\_di[i];

}

}

**6.8** В следующих циклах вызывая функцию Rating\_zero, где производим оценку нулевых элементов массива, в который мы записывали длины дорог. В max запоминаем наибольшую из оценок, а в str и st соответствующие ему строку и столбец, это и будет наш первый найденный путь.

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

{

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

{

if (mas[i][j] == 0)

{

if (Rating\_zero(i, j) > max)

{

str = i;

st = j;

max = Rating\_zero(i, j);

}

}

}

}

**6.9** Дальше в этом условие проверяем найден путь или нет. Если его нет, следовательно, решение у задачи нет, и мы выйдем из функции, а n в данном случае является неким флажком.

if (str == 0 && st == 0)

{

n=0;

return;

}

**6.10** Производим редукцию матрицы вызывая функцию Reduction, с помощью которой исключаем из рассмотрения найденные строку и столбец обозначая их -1.

void **Reduction**(int k, int l)

{

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

{

mas[k][i] = -1;

}

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

{

mas[i][l] = -1;

}

}

**6.11** Прибавляем к str и st один чтобы они соответствовали номерам узлов, т.к. до этого нумерация шла с нуля. В массив, в котором будет находится итоговая траектория, он также является глобальным, записываем строку и столбец. В нём значение каждого четного индекса является начало дороги, а нечетный её конец.

str++;

st++;

mas\_result[k\*2]=str;

mas\_result[k\*2+1]=st;

**6.12** Рекурсивно вызываем функцию Paths() до тех пор пока не будут найдены все пути, за это будет отвечать глобальная переменная k. Вернёмся туда, где мы вызывали рекурсивную функцию. Если n соответствует диапазону, который равен количеству дорог или же наш флажок не обратился в ноль, то мы выполняем функцию repaint() которая переносит в *paintEvent* для отрисовки узла с которого начинается наш путь и самого пути. В функции с помощью циклов и условий мы меняем цвет узла. И для найденного пути увеличиваем толщину линий.

if(n>0 && n<=6)

{

repaint();

…

**6.13** Дальше в переменную strok, также она является глобальной, записываем первый узел. И вызываем функцию Answer(n), в которой из массива mas\_result будем составлять нужную нам траекторию в переменную strok. В цикле проходимся по четным индексам массива (то есть начало дорог) и если оно соответствует n, то записываем его в переменную. Вызываем функцию Answer(n) рекурсивно используя условие и переменную k, чтобы количество вызываний соответствовало количеству дорог.

void **Answer**(int n)

{

int i=0;

int current = 0;

while (i<12)

{

if(n==mas\_result[i])

{

strok=strok+" → "+QString::number(mas\_result[i+1]);

current=i+1;

}

i += 2;

}

n=mas\_result[current];

k++;

if (k != 6)

Answer(n);

}

**6.14** Когда отработала функция Answer() мы подсчитываем сумму c помощью цикла и нашего скопированного массива mas\_copy, индексами которого будут значения дорог в массиве mas\_result.

while(i<12)

{

sum = sum + mas\_copy[mas\_result[i]-1][mas\_result[i+1]-1];

i+=2;

}

**6.15** Записываем ответы в виджеты, при этом переменную strok преобразуем к типу QString. Если ранее озвученное условие не сработало, то выводим текст что дороги нет.

ui->lineEdit\_show->setText(strok);

ui->lineEdit\_sum->setText(QString::number(sum));

}

else

{

ui->lineEdit\_show->setText("Нет решения");

ui->lineEdit\_sum->setText("Нет");

}

**3. Код программы**

**3.1 Калькулятор**

Main.cpp

#include "calculator.h"

#include <QApplication>

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

{

QApplication a(*argc*, argv);

Calculator w;

w.show();

return a.exec();

}

Calculator.h

#ifndef CALCULATOR\_H

#define CALCULATOR\_H

#include <QMainWindow>

#include <QtMath>

QT\_BEGIN\_NAMESPACE

namespace **Ui** { class **Calculator**; }

QT\_END\_NAMESPACE

class **Calculator** : public QMainWindow

{

Q\_OBJECT

public:

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

~***Calculator***();

private:

Ui::Calculator \*ui;

private slots:

void **digits\_numbers**();

void **on\_pushButton\_equally\_clicked**();

void **on\_pushButton\_point\_clicked**();

void **on\_pushButton\_replace\_clicked**();

void **on\_pushButton\_next\_clicked**();

void **on\_pushButton\_e\_clicked**();

void **on\_pushButton\_clear\_clicked**();

void **on\_pushButton\_clear\_2\_clicked**();

void **on\_pushButton\_root\_clicked**();

};

#endif // CALCULATOR\_H

Calculator.cpp

#include "calculator.h"

#include "ui\_calculator.h"

#include <QtMath>

int i=1;

Calculator::**Calculator**(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::Calculator)

{

ui->setupUi(this);

connect(ui->pushButton\_0,SIGNAL(clicked()),this,SLOT(digits\_numbers())) ;

connect(ui->pushButton\_1,SIGNAL(clicked()),this,SLOT(digits\_numbers())) ;

connect(ui->pushButton\_2,SIGNAL(clicked()),this,SLOT(digits\_numbers())) ;

connect(ui->pushButton\_3,SIGNAL(clicked()),this,SLOT(digits\_numbers())) ;

connect(ui->pushButton\_4,SIGNAL(clicked()),this,SLOT(digits\_numbers())) ;

connect(ui->pushButton\_5,SIGNAL(clicked()),this,SLOT(digits\_numbers())) ;

connect(ui->pushButton\_6,SIGNAL(clicked()),this,SLOT(digits\_numbers())) ;

connect(ui->pushButton\_7,SIGNAL(clicked()),this,SLOT(digits\_numbers())) ;

connect(ui->pushButton\_8,SIGNAL(clicked()),this,SLOT(digits\_numbers())) ;

connect(ui->pushButton\_9,SIGNAL(clicked()),this,SLOT(digits\_numbers())) ;

}

Calculator::~***Calculator***()

{

delete ui;

}

void Calculator::**digits\_numbers**()

{

QPushButton \*button=(QPushButton\*) sender();

if(i==1 && ui->lineEdit\_1->text()!="e")

{

QString lable=ui->lineEdit\_1->text()+button->text();

ui->lineEdit\_1->setText(lable);

}

else if(i==2 && ui->lineEdit\_2->text()!="e")

{

QString lable=ui->lineEdit\_2->text()+button->text();

ui->lineEdit\_2->setText(lable);

}

else if (i==3 && ui->lineEdit\_3->text()!="e")

{

QString lable=ui->lineEdit\_3->text()+button->text();

ui->lineEdit\_3->setText(lable);

}

else if(i==4 && ui->lineEdit\_4->text()!="e")

{

QString lable=ui->lineEdit\_4->text()+button->text();

ui->lineEdit\_4->setText(lable);

}

else if(i==5 && ui->lineEdit\_5->text()!="e")

{

QString lable=ui->lineEdit\_5->text()+button->text();

ui->lineEdit\_5->setText(lable);

}

else if(i==6)

{

QString lable=ui->lineEdit\_from->text()+button->text();

ui->lineEdit\_from->setText(lable);

}

else if(i==7)

{

QString lable=ui->lineEdit\_befor->text()+button->text();

ui->lineEdit\_befor->setText(lable);

}

else if(i==8 && ui->lineEdit\_5->text()!="e")

{

QString lable=ui->lineEdit\_accuracy->text()+button->text();

ui->lineEdit\_accuracy->setText(lable);

}

}

void Calculator::**on\_pushButton\_next\_clicked**()

{

if(i<8)

i=i+1;

}

void Calculator::**on\_pushButton\_point\_clicked**()

{

if (i==1 && !(ui->lineEdit\_1->text().contains('.')) && ui->lineEdit\_1->text()!=0 && ui->lineEdit\_1->text()!="e")

ui->lineEdit\_1->setText(ui->lineEdit\_1->text()+'.');

else if (i==2 && !(ui->lineEdit\_2->text().contains('.')) && ui->lineEdit\_2->text()!=0 && ui->lineEdit\_2->text()!="e")

ui->lineEdit\_2->setText(ui->lineEdit\_2->text()+'.');

else if (i==3 && !(ui->lineEdit\_3->text().contains('.')) && ui->lineEdit\_3->text()!=0 && ui->lineEdit\_3->text()!="e")

ui->lineEdit\_3->setText(ui->lineEdit\_3->text()+'.');

else if (i==4 && !(ui->lineEdit\_4->text().contains('.')) && ui->lineEdit\_4->text()!=0 && ui->lineEdit\_4->text()!="e")

ui->lineEdit\_4->setText(ui->lineEdit\_4->text()+'.');

else if (i==5 && !(ui->lineEdit\_5->text().contains('.')) && ui->lineEdit\_5->text()!=0 && ui->lineEdit\_5->text()!="e")

ui->lineEdit\_5->setText(ui->lineEdit\_5->text()+'.');

else if (i==6 && !(ui->lineEdit\_from->text().contains('.')) && ui->lineEdit\_from->text()!=0 && ui->lineEdit\_from->text()!="e")

ui->lineEdit\_from->setText(ui->lineEdit\_from->text()+'.');

else if (i==7 && !(ui->lineEdit\_befor->text().contains('.')) && ui->lineEdit\_befor->text()!=0 && ui->lineEdit\_befor->text()!="e")

ui->lineEdit\_befor->setText(ui->lineEdit\_befor->text()+'.');

else if (i==8 && !(ui->lineEdit\_accuracy->text().contains('.')) && ui->lineEdit\_accuracy->text()!=0 && ui->lineEdit\_accuracy->text()!="e")

ui->lineEdit\_accuracy->setText(ui->lineEdit\_accuracy->text()+'.');

}

void Calculator::**on\_pushButton\_replace\_clicked**()

{

double numbers;

if(i==1)

{

numbers=(ui->lineEdit\_1->text()).toDouble();

numbers=numbers\*-1;

ui->lineEdit\_1->setText(QString::number(numbers));

}

else if(i==2)

{

numbers=(ui->lineEdit\_2->text()).toDouble();

numbers=numbers\*-1;

ui->lineEdit\_2->setText(QString::number(numbers));

}

else if(i==3)

{

numbers=(ui->lineEdit\_3->text()).toDouble();

numbers=numbers\*-1;

ui->lineEdit\_3->setText(QString::number(numbers));

}

else if(i==4)

{

numbers=(ui->lineEdit\_4->text()).toDouble();

numbers=numbers\*-1;

ui->lineEdit\_4->setText(QString::number(numbers));

}

else if(i==5)

{

numbers=(ui->lineEdit\_5->text()).toDouble();

numbers=numbers\*-1;

ui->lineEdit\_5->setText(QString::number(numbers));

}

else if(i==6)

{

numbers=(ui->lineEdit\_from->text()).toDouble();

numbers=numbers\*-1;

ui->lineEdit\_from->setText(QString::number(numbers));

}

else if(i==7)

{

numbers=(ui->lineEdit\_befor->text()).toDouble();

numbers=numbers\*-1;

ui->lineEdit\_befor->setText(QString::number(numbers));

}

}

void Calculator::**on\_pushButton\_e\_clicked**()

{

if (i==1 && ui->lineEdit\_1->text()==0)

ui->lineEdit\_1->setText("e");

else if (i==2 && ui->lineEdit\_2->text()==0)

ui->lineEdit\_2->setText("e");

else if (i==3 && ui->lineEdit\_3->text()==0)

ui->lineEdit\_3->setText("e");

else if (i==4 && ui->lineEdit\_4->text()==0)

ui->lineEdit\_4->setText("e");

else if (i==5 && ui->lineEdit\_5->text()==0)

ui->lineEdit\_5->setText("e");

else if (i==8 && ui->lineEdit\_accuracy->text()==0)

ui->lineEdit\_accuracy->setText("e");

}

void Calculator::**on\_pushButton\_clear\_clicked**()

{

if(i==1)

ui->lineEdit\_1->setText("");

if(i==2)

ui->lineEdit\_2->setText("");

if(i==3)

ui->lineEdit\_3->setText("");

if(i==4)

ui->lineEdit\_4->setText("");

if(i==5)

ui->lineEdit\_5->setText("");

if(i==6)

ui->lineEdit\_from->setText("");

if(i==7)

ui->lineEdit\_befor->setText("");

if(i==8)

ui->lineEdit\_accuracy->setText("");

}

void Calculator::**on\_pushButton\_clear\_2\_clicked**()

{

ui->lineEdit\_1->setText("");

ui->lineEdit\_2->setText("");

ui->lineEdit\_3->setText("");

ui->lineEdit\_4->setText("");

ui->lineEdit\_5->setText("");

ui->lineEdit\_from->setText("");

ui->lineEdit\_befor->setText("");

ui->lineEdit\_accuracy->setText("");

ui->lineEdit\_show->setText("");

i=1;

}

void Calculator::**on\_pushButton\_root\_clicked**()

{

double t;

if(i==1)

{

t=ui->lineEdit\_1->text().toDouble();

t=sqrt(t);

ui->lineEdit\_1->setText(QString::number(t));

}

if(i==2)

{

t=ui->lineEdit\_2->text().toDouble();

t=sqrt(t);

ui->lineEdit\_2->setText(QString::number(t));

}

if(i==3)

{

t=ui->lineEdit\_3->text().toDouble();

t=sqrt(t);

ui->lineEdit\_3->setText(QString::number(t));

}

if(i==4)

{

t=ui->lineEdit\_4->text().toDouble();

t=sqrt(t);

ui->lineEdit\_4->setText(QString::number(t));

}

if(i==5)

{

t=ui->lineEdit\_5->text().toDouble();

t=sqrt(t);

ui->lineEdit\_5->setText(QString::number(t));

}

if(i==6)

{

t=ui->lineEdit\_from->text().toDouble();

t=sqrt(t);

ui->lineEdit\_from->setText(QString::number(t));

}

if(i==7)

{

t=ui->lineEdit\_befor->text().toDouble();

t=sqrt(t);

ui->lineEdit\_befor->setText(QString::number(t));

}

if(i==8)

{

t=ui->lineEdit\_accuracy->text().toDouble();

t=sqrt(t);

ui->lineEdit\_accuracy->setText(QString::number(t));

}

}

double **F**(double y1, double y2,double y3,double y4,double y5, double x)

{

return y1\*pow(x,3)+y2\*pow(x,2)+y3\*x+y4+y5;

}

void Calculator::**on\_pushButton\_equally\_clicked**()

{

double eps,a,b,x,x0,x1,y1,y2,y3,y4,y5;

if(ui->lineEdit\_1->text()=="e")

y1=M\_E;

else

y1=ui->lineEdit\_1->text().toDouble();

if(ui->lineEdit\_2->text()=="e")

y2=M\_E;

else

y2=ui->lineEdit\_2->text().toDouble();

if(ui->lineEdit\_3->text()=="e")

y3=M\_E;

else

y3=ui->lineEdit\_3->text().toDouble();

if(ui->lineEdit\_4->text()=="e")

y4=M\_E;

else

y4=ui->lineEdit\_4->text().toDouble();

if(ui->lineEdit\_5->text()=="e")

y5=M\_E;

else

y5=ui->lineEdit\_5->text().toDouble();

if(ui->lineEdit\_accuracy->text()=="e")

eps=M\_E;

else

eps=ui->lineEdit\_accuracy->text().toDouble();

a=ui->lineEdit\_from->text().toDouble();

b=ui->lineEdit\_befor->text().toDouble();

x0=(a+b)/2;

do

{

if(F(y1,y2,y3,y4,y5,a)\*F(y1,y2,y3,y4,y5,x0)<0)

b=x0;

else

a=x0;

x1=x0;

x0=(a+b)/2;

if((x1-x0)<0)

x=(x1-x0)\*-1;

else

x=x1-x0;

}while(x>eps);

ui->lineEdit\_show->setText(QString::number(x0));

}

**3.2 Задача Коммивояжёра**

Main.cpp

#include "kommivoyazher.h"

#include <QApplication>

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

{

QApplication a(*argc*, argv);

Kommivoyazher w;

w.show();

return a.exec();

}

Kommivoyazher.h

#ifndef KOMMIVOYAZHER\_H

#define KOMMIVOYAZHER\_H

#include <QMainWindow>

#include <QPainter>

#include <QtMath>

QT\_BEGIN\_NAMESPACE

namespace **Ui** { class **Kommivoyazher**; }

QT\_END\_NAMESPACE

class **Kommivoyazher** : public QMainWindow

{

Q\_OBJECT

public:

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

~***Kommivoyazher***();

private:

Ui::Kommivoyazher \*ui;

protected:

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

private slots:

void **on\_pushButton\_clicked**();

};

#endif // KOMMIVOYAZHER\_H

Kommivoyazher.cpp

#include "kommivoyazher.h"

#include "ui\_kommivoyazher.h"

Kommivoyazher::**Kommivoyazher**(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::Kommivoyazher)

{

ui->setupUi(this);

}

Kommivoyazher::~***Kommivoyazher***()

{

delete ui;

}

int mas[6][6]{

{-1,-1,-1,-1,-1,-1},

{-1,-1,-1,-1,-1,-1},

{-1,-1,-1,-1,-1,-1},

{-1,-1,-1,-1,-1,-1},

{-1,-1,-1,-1,-1,-1},

{-1,-1,-1,-1,-1,-1}

};

int k=0;

int mas\_result[12];

QString strok="";

int n=0;

void Kommivoyazher::***paintEvent***(QPaintEvent \*)

{

QPainter painter;

painter.begin(this);

int l = 180;

double th=2\*M\_PI/6;

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

{

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

{

if(i!=j)

{

int x1 = 260 + cos(th \* i) \* l;

int y1 = 270 - sin(th \* i) \* l;

int x2 = 260 + cos(th \* j) \* l;

int y2 = 270 - sin(th \* j) \* l;

painter.setPen(QPen(Qt::darkCyan, 1, Qt::SolidLine));

painter.drawLine(x1, y1, x2, y2);

}

}

}

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

{

painter.setPen(QPen(Qt::darkCyan, 3, Qt::SolidLine));

painter.setBrush(QBrush(Qt::cyan, Qt::SolidPattern));

painter.drawEllipse(240 + cos(th \* i) \* l, 250 - sin(th \* i) \* l, 50, 50);

painter.setPen(QPen(Qt::black, 5, Qt::SolidLine));

painter.drawText(260 + cos(th \* i) \* l, 280 - sin(th \* i) \* l, QString::number(i+1));

}

int x1,x2,y1,y2,i=0;

while(i<12)

{

if (ui->lineEdit\_number->text().toInt()!=0)

{

x1 = 260 + cos(th \* (mas\_result[i]-1)) \* l;

x2 = 260 + cos(th \* (mas\_result[i+1]-1)) \* l;

y1 = 270 - sin(th \* (mas\_result[i]-1)) \* l;

y2 = 270 - sin(th \* (mas\_result[i+1]-1)) \* l;

painter.setPen(QPen(Qt::darkCyan, 5, Qt::SolidLine));

painter.drawLine(x1, y1, x2, y2);

}

i+=2;

}

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

{

if(ui->lineEdit\_number->text().toInt()==i+1)

{

painter.setPen(QPen(Qt::darkCyan, 3, Qt::SolidLine));

painter.setBrush(QBrush(Qt::darkCyan, Qt::SolidPattern));

painter.drawEllipse(240 + cos(th \* i) \* l, 250 - sin(th \* i) \* l, 50, 50);

}

painter.setPen(QPen(Qt::black, 5, Qt::SolidLine));

painter.drawText(260 + cos(th \* i) \* l, 280 - sin(th \* i) \* l, QString::number(i+1));

}

painter.end();

}

double **Rating\_zero**(int k, int l)

{

double min = 1000, sum;

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

{

if (l != i && mas[k][i] != -1 && mas[k][i] < min)

min = mas[k][i];

}

sum = min;

min = 1000;

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

{

if (k != i && mas[i][l] != -1 && mas[i][l] < min)

min = mas[i][l];

}

sum = sum + min;

return sum;

}

void **Reduction**(int k, int l)

{

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

{

mas[k][i] = -1;

}

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

{

mas[i][l] = -1;

}

}

void **Paths**()

{

double mas\_di[6];

double mas\_dj[6];

double min = 10000;

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

{

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

{

if (mas[i][j] < min && mas[i][j] != -1)

min = mas[i][j];

}

mas\_di[i] = min;

if (min == 10000)

mas\_di[i] = 0;

min = 10000;

}

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

{

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

{

if (mas[i][j] != -1)

mas[i][j] = mas[i][j] - mas\_di[i];

}

}

/////////////////////////////////////////////////////////////////////

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

{

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

{

if (mas[j][i] < min && mas[j][i] != -1)

min = mas[j][i];

}

mas\_dj[i] = min;

if (min == 10000)

mas\_dj[i] = 0;

min = 10000;

}

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

{

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

{

if (mas[j][i] != -1)

mas[j][i] = mas[j][i] - mas\_dj[i];

}

}

/////////////////////////////////////////////////////////////////////

int str = 0, st = 0;

double max = 0;

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

{

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

{

if (mas[i][j] == 0)

{

if (Rating\_zero(i, j) > max)

{

str = i;

st = j;

max = Rating\_zero(i, j);

}

}

}

}

if (str == 0 && st == 0)

{

n=0;

return;

}

///////////////////////////////////////////////////////////////////

Reduction(str, st);

////////////////////////////////////////////////////////////////////

str++;

st++;

mas\_result[k\*2]=str;

mas\_result[k\*2+1]=st;

k++;

if (k != 6)

Paths();

k=0;

}

void **Answer**(int n)

{

int i=0;

int current = 0;

while (i<12)

{

if(n==mas\_result[i])

{

strok=strok+" → "+QString::number(mas\_result[i+1]);

current=i+1;

}

i += 2;

}

n=mas\_result[current];

k++;

if (k != 6)

Answer(n);

}

void Kommivoyazher::**on\_pushButton\_clicked**()

{

mas[0][1]=ui->lineEdit\_i0j1->text().toInt();

mas[0][2]=ui->lineEdit\_i0j2->text().toInt();

mas[0][3]=ui->lineEdit\_i0j3->text().toInt();

mas[0][4]=ui->lineEdit\_i0j4->text().toInt();

mas[0][5]=ui->lineEdit\_i0j5->text().toInt();

mas[1][0]=ui->lineEdit\_i1j0->text().toInt();

mas[1][2]=ui->lineEdit\_i1j2->text().toInt();

mas[1][3]=ui->lineEdit\_i1j3->text().toInt();

mas[1][4]=ui->lineEdit\_i1j4->text().toInt();

mas[1][5]=ui->lineEdit\_i1j5->text().toInt();

mas[2][0]=ui->lineEdit\_i2j0->text().toInt();

mas[2][1]=ui->lineEdit\_i2j1->text().toInt();

mas[2][3]=ui->lineEdit\_i2j3->text().toInt();

mas[2][4]=ui->lineEdit\_i2j4->text().toInt();

mas[2][5]=ui->lineEdit\_i2j5->text().toInt();

mas[3][0]=ui->lineEdit\_i3j0->text().toInt();

mas[3][1]=ui->lineEdit\_i3j1->text().toInt();

mas[3][2]=ui->lineEdit\_i3j2->text().toInt();

mas[3][4]=ui->lineEdit\_i3j4->text().toInt();

mas[3][5]=ui->lineEdit\_i3j5->text().toInt();

mas[4][0]=ui->lineEdit\_i4j0->text().toInt();

mas[4][1]=ui->lineEdit\_i4j1->text().toInt();

mas[4][2]=ui->lineEdit\_i4j2->text().toInt();

mas[4][3]=ui->lineEdit\_i4j3->text().toInt();

mas[4][5]=ui->lineEdit\_i4j5->text().toInt();

mas[5][0]=ui->lineEdit\_i5j0->text().toInt();

mas[5][1]=ui->lineEdit\_i5j1->text().toInt();

mas[5][2]=ui->lineEdit\_i5j2->text().toInt();

mas[5][3]=ui->lineEdit\_i5j3->text().toInt();

mas[5][4]=ui->lineEdit\_i5j4->text().toInt();

int mas\_copy[6][6];

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

{

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

{

mas\_copy[i][j]=mas[i][j];

}

}

n= ui->lineEdit\_number->text().toInt();

Paths();

if(n>0 && n<=6)

{

repaint();

strok = QString::number(n);

Answer(n);

int sum=0;

int i=0;

while(i<12)

{

sum = sum + mas\_copy[mas\_result[i]-1][mas\_result[i+1]-1];

i+=2;

}

ui->lineEdit\_show->setText(strok);

ui->lineEdit\_sum->setText(QString::number(sum));

}

else

{

ui->lineEdit\_show->setText("Нет решения");

ui->lineEdit\_sum->setText("Нет");

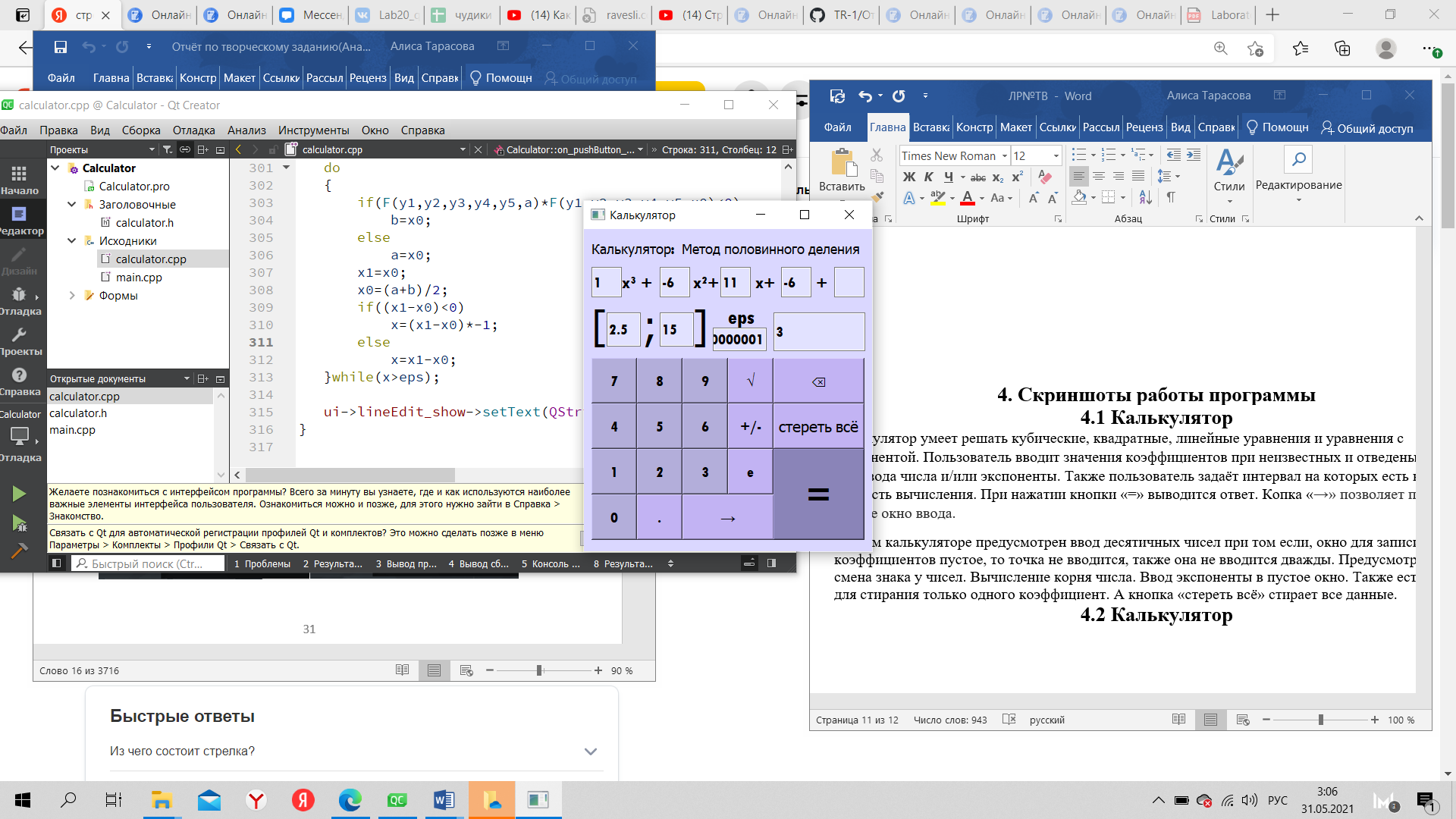
}

}

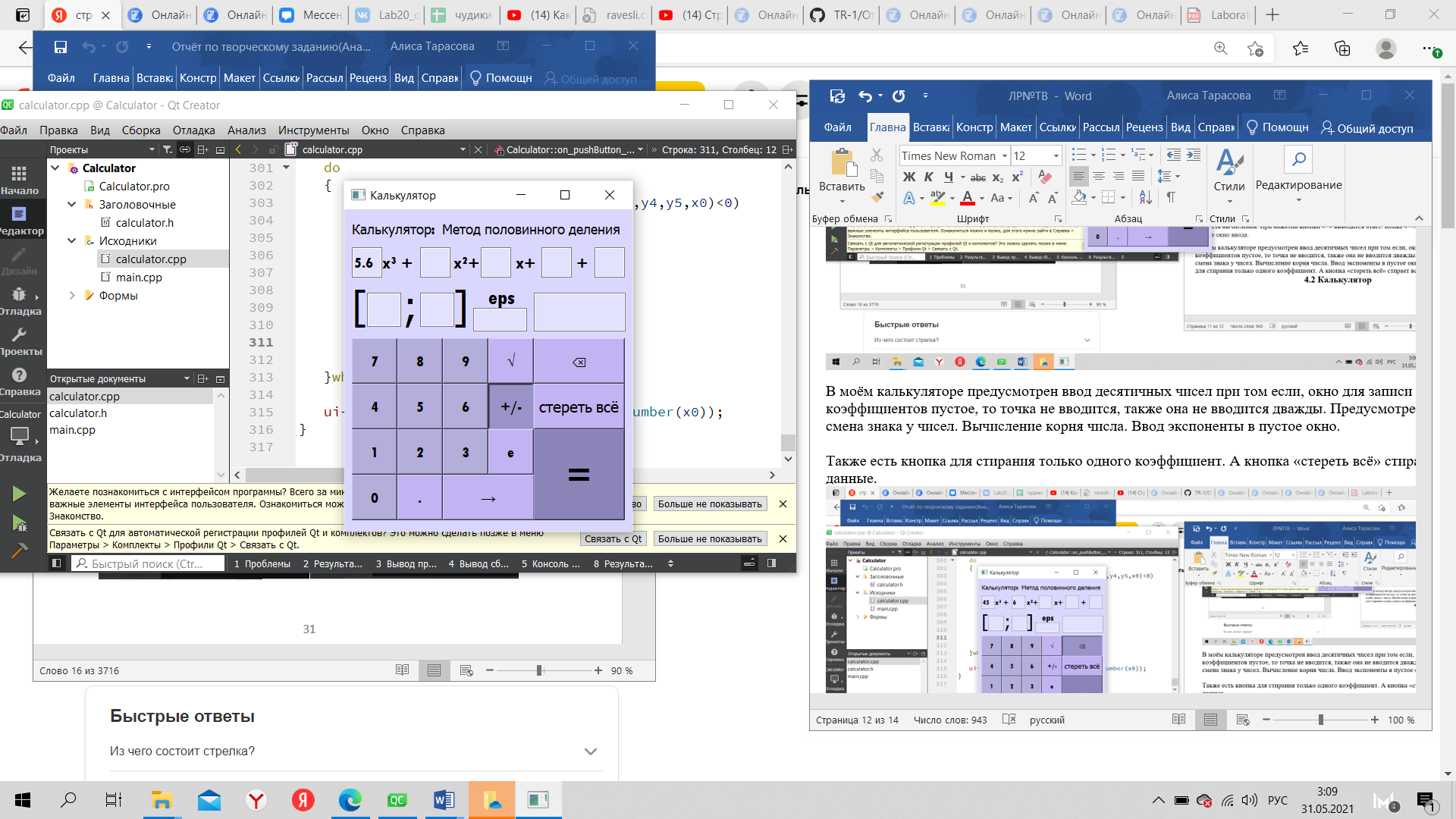
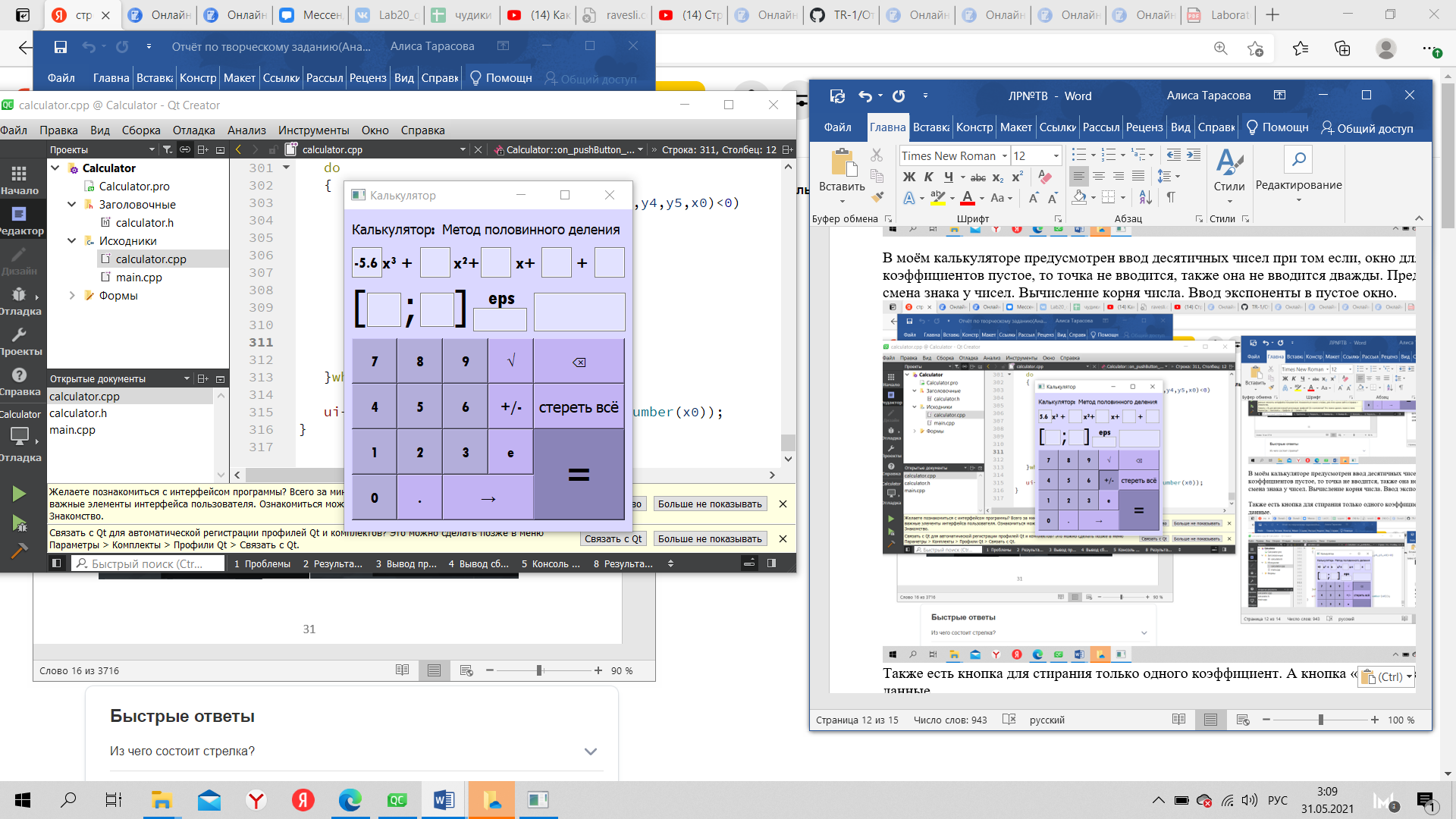
**4. Скриншоты работы программы**

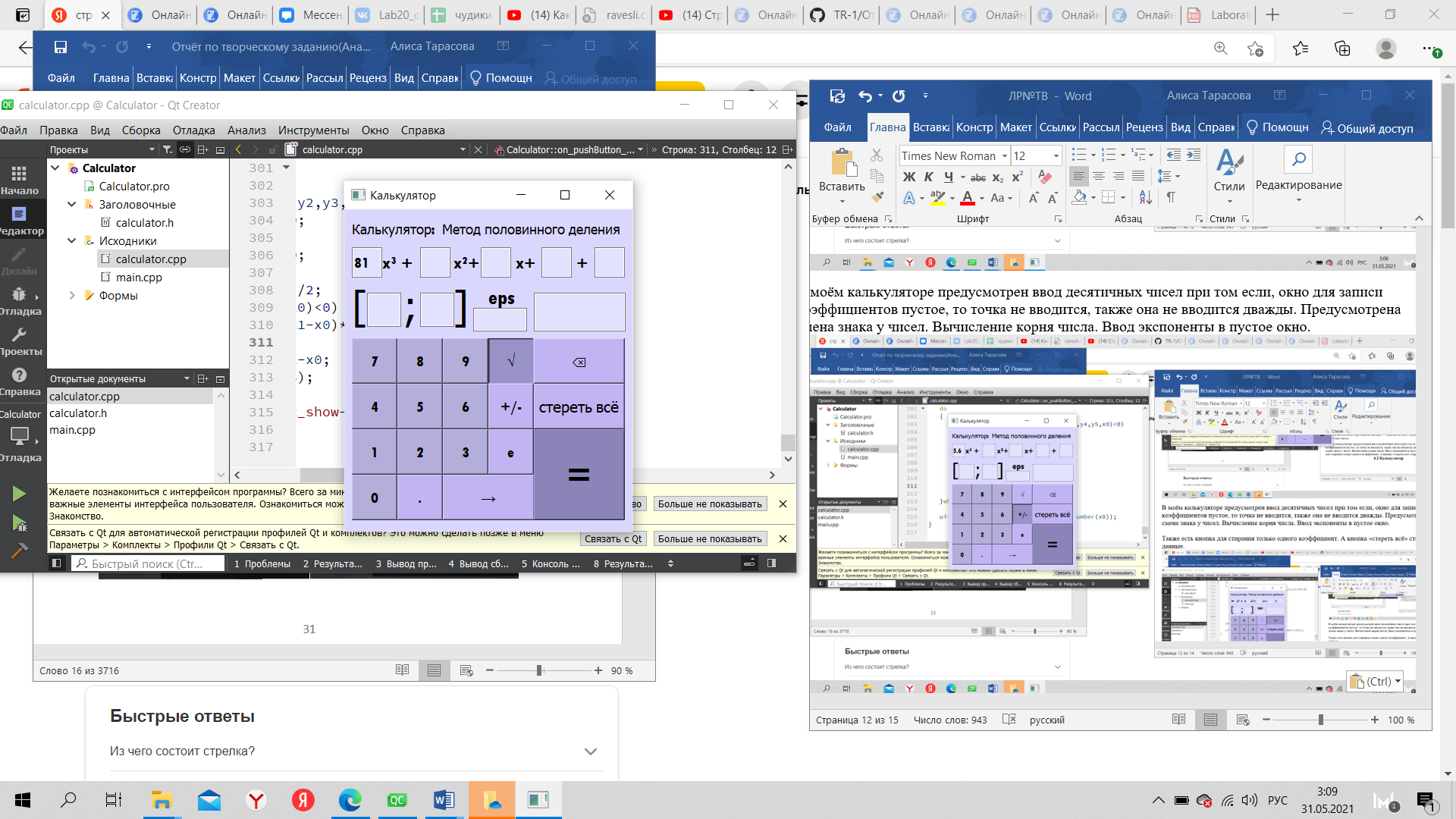
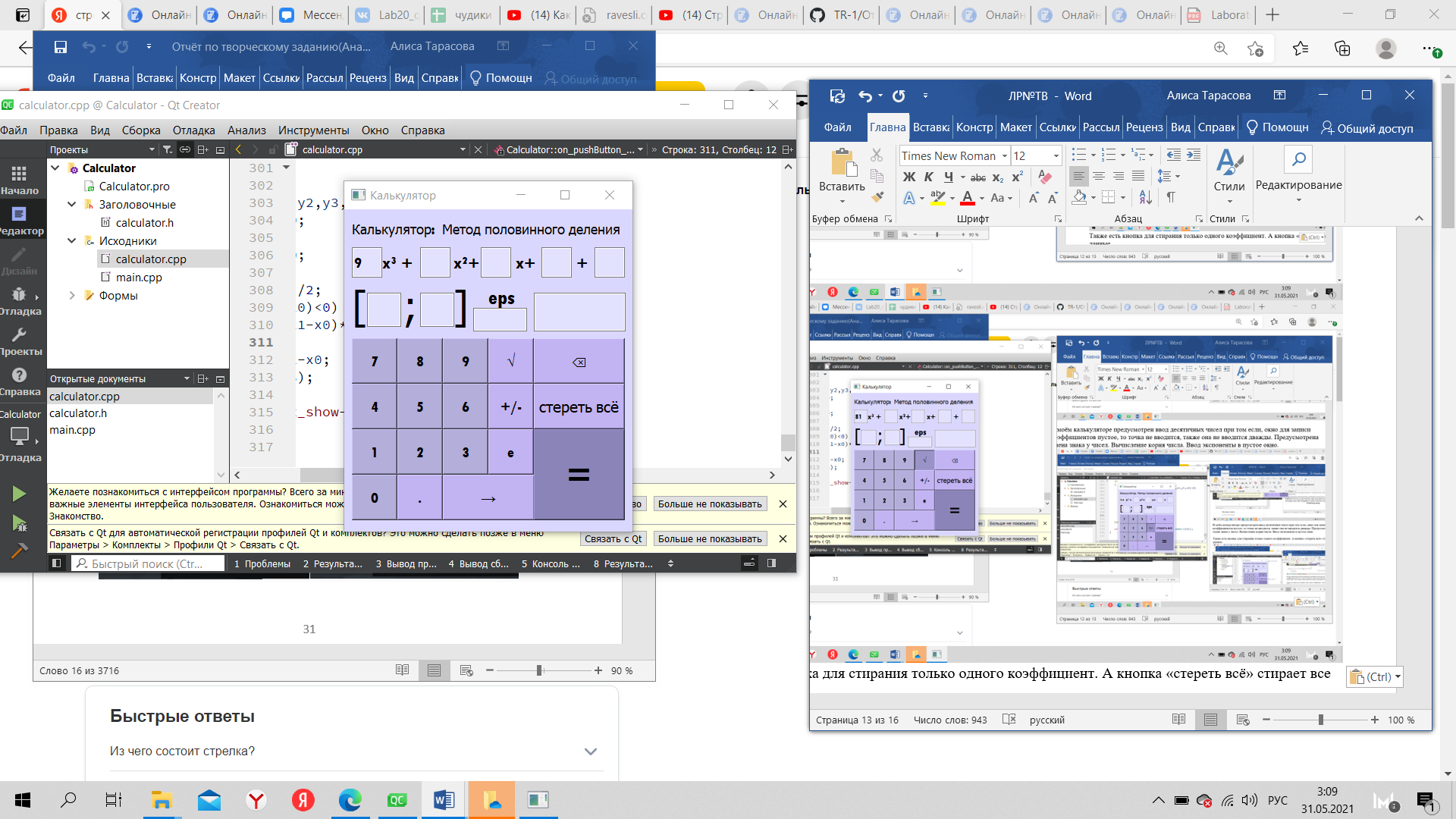
**4.1 Калькулятор**

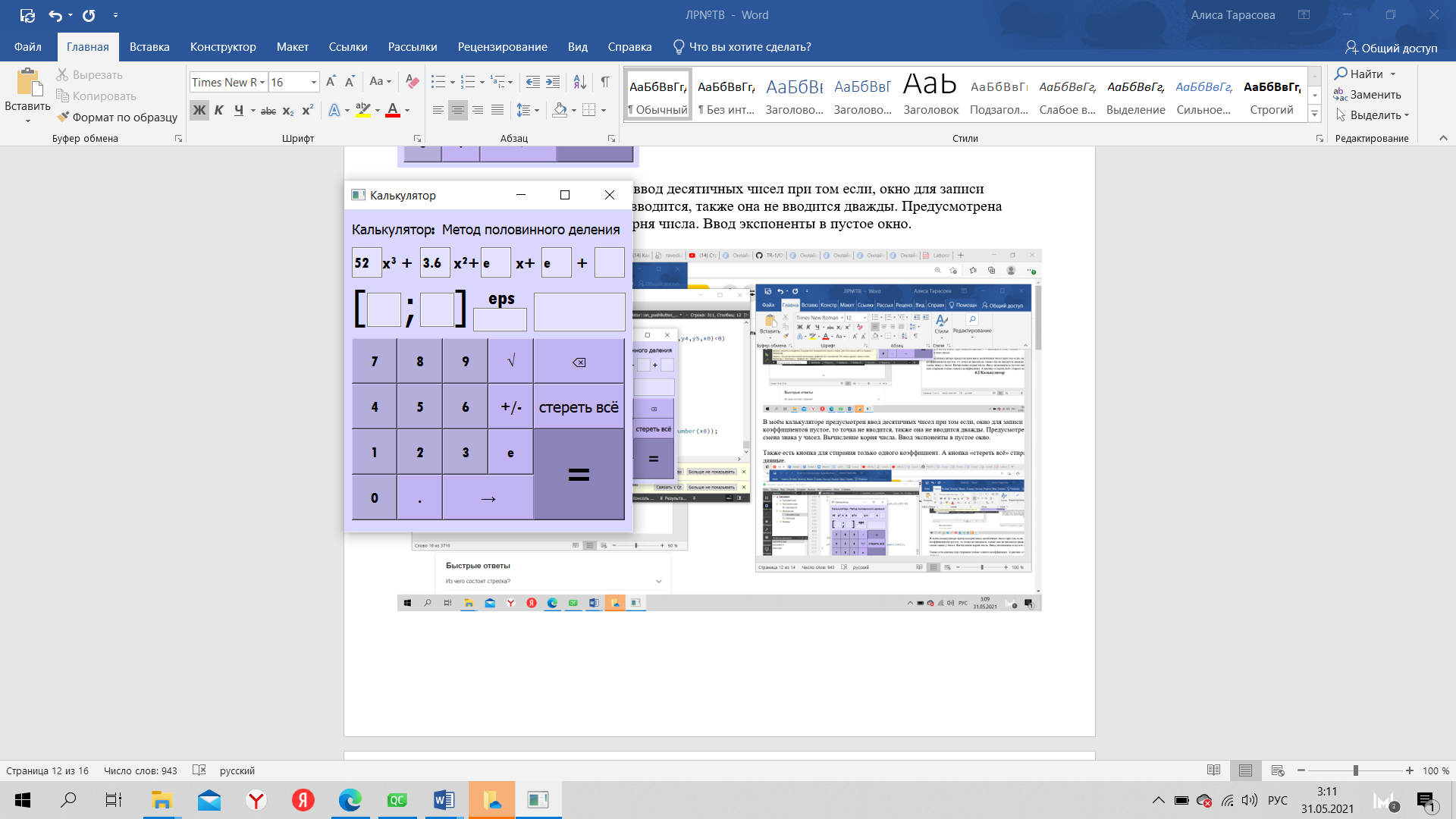
Калькулятор умеет решать кубические, квадратные, линейные уравнения и уравнения с экспонентой. Пользователь вводит значения коэффициентов при неизвестных и отведены два поля для ввода числа и/или экспоненты. Также пользователь задаёт интервал на которых есть корень и точность вычисления. При нажатии кнопки «=» выводится ответ. Копка «→» позволяет перейти в другое окно ввода.



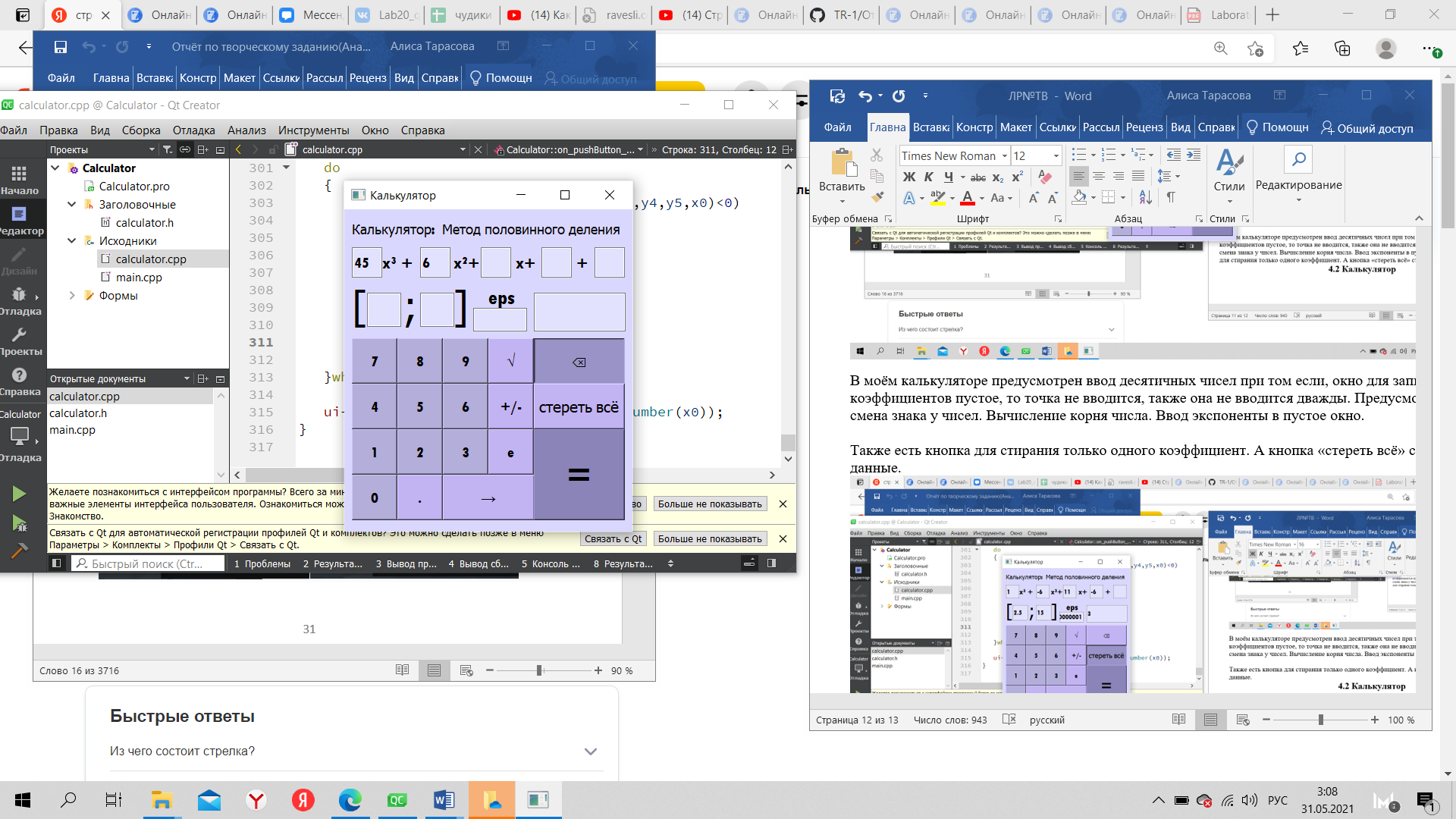
В моём калькуляторе предусмотрен ввод десятичных чисел при том если, окно для записи коэффициентов пустое, то точка не вводится, также она не вводится дважды. Предусмотрена смена знака у чисел. Вычисление корня числа. Ввод экспоненты в пустое окно.

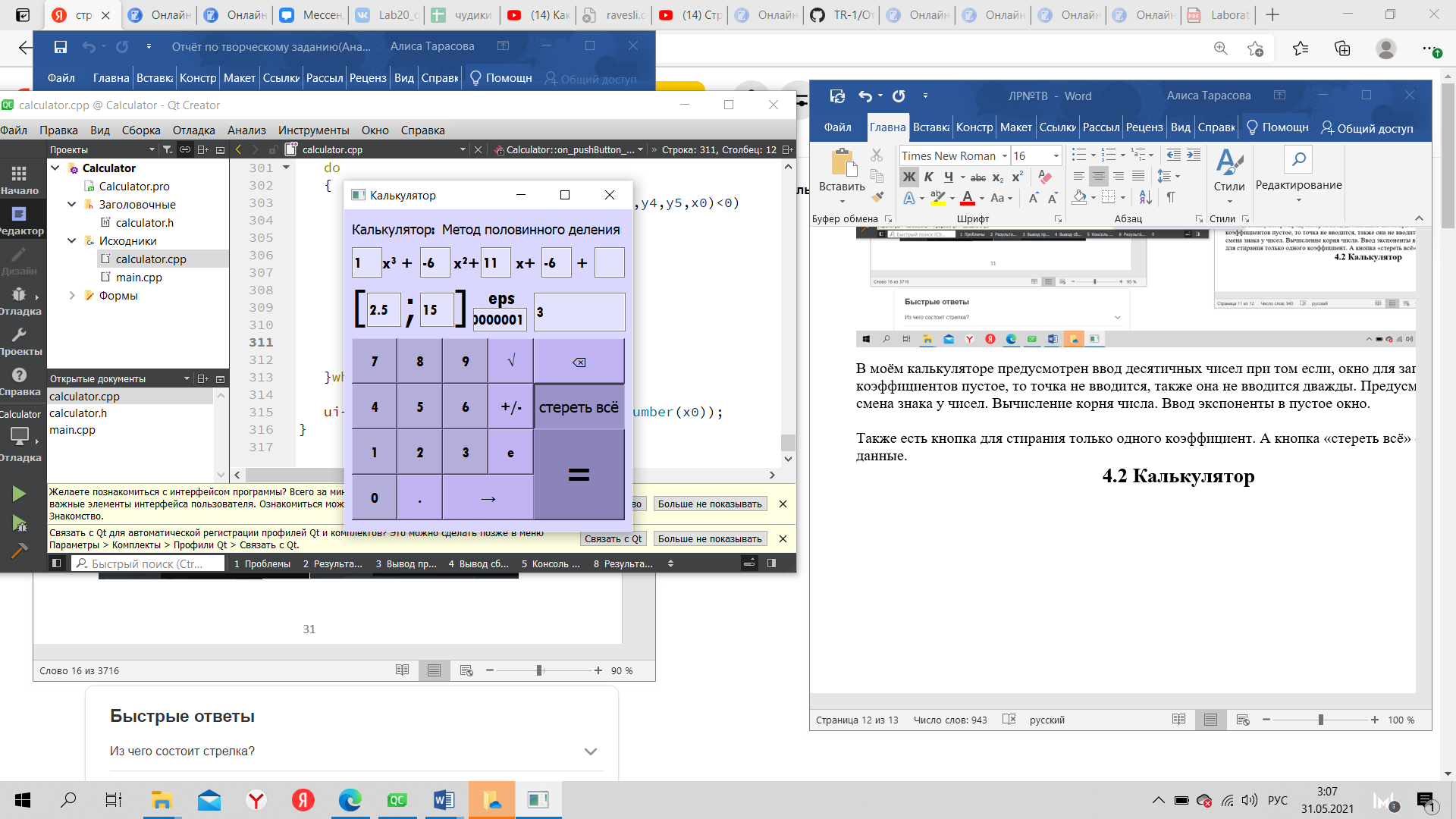
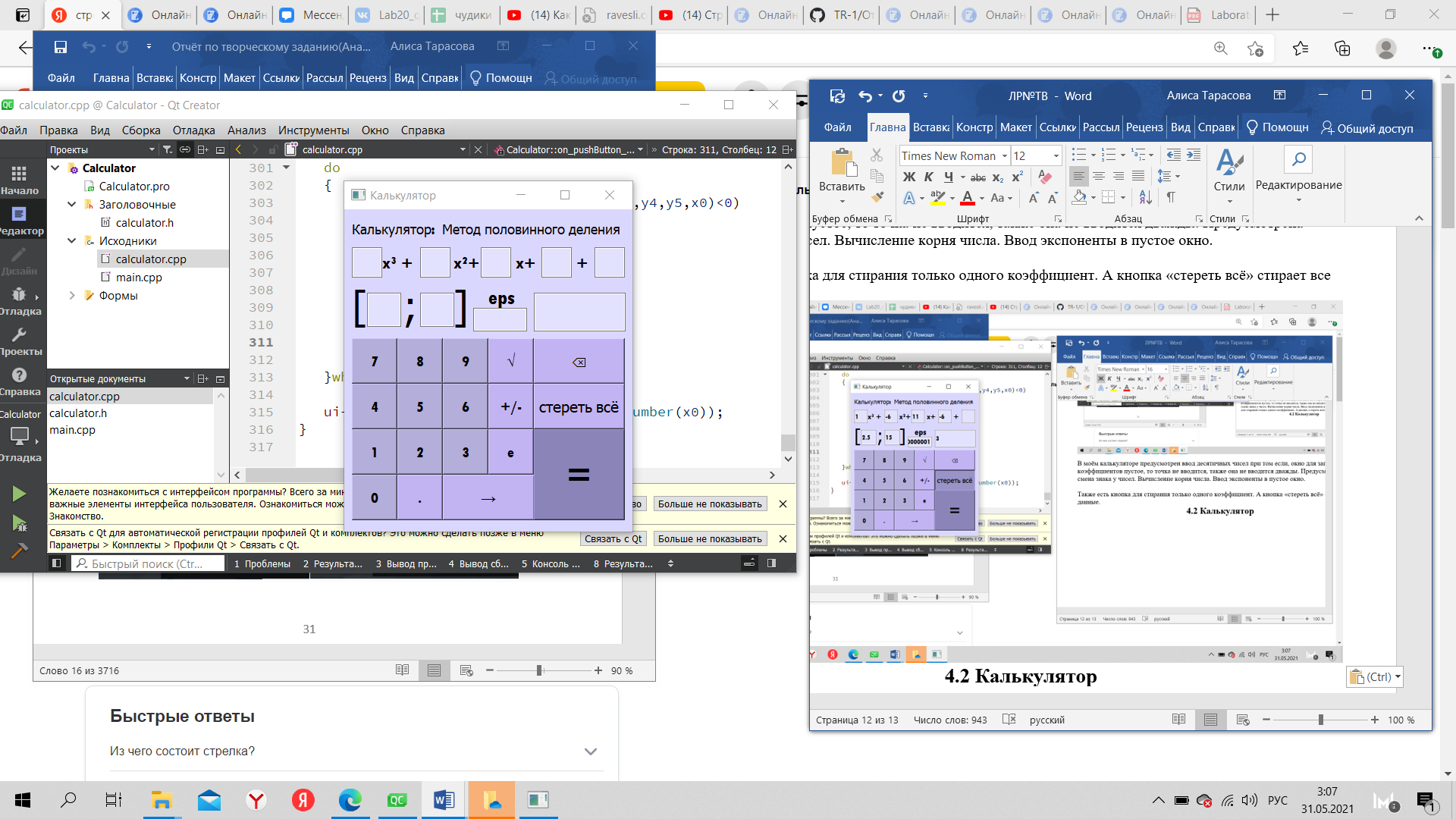
 



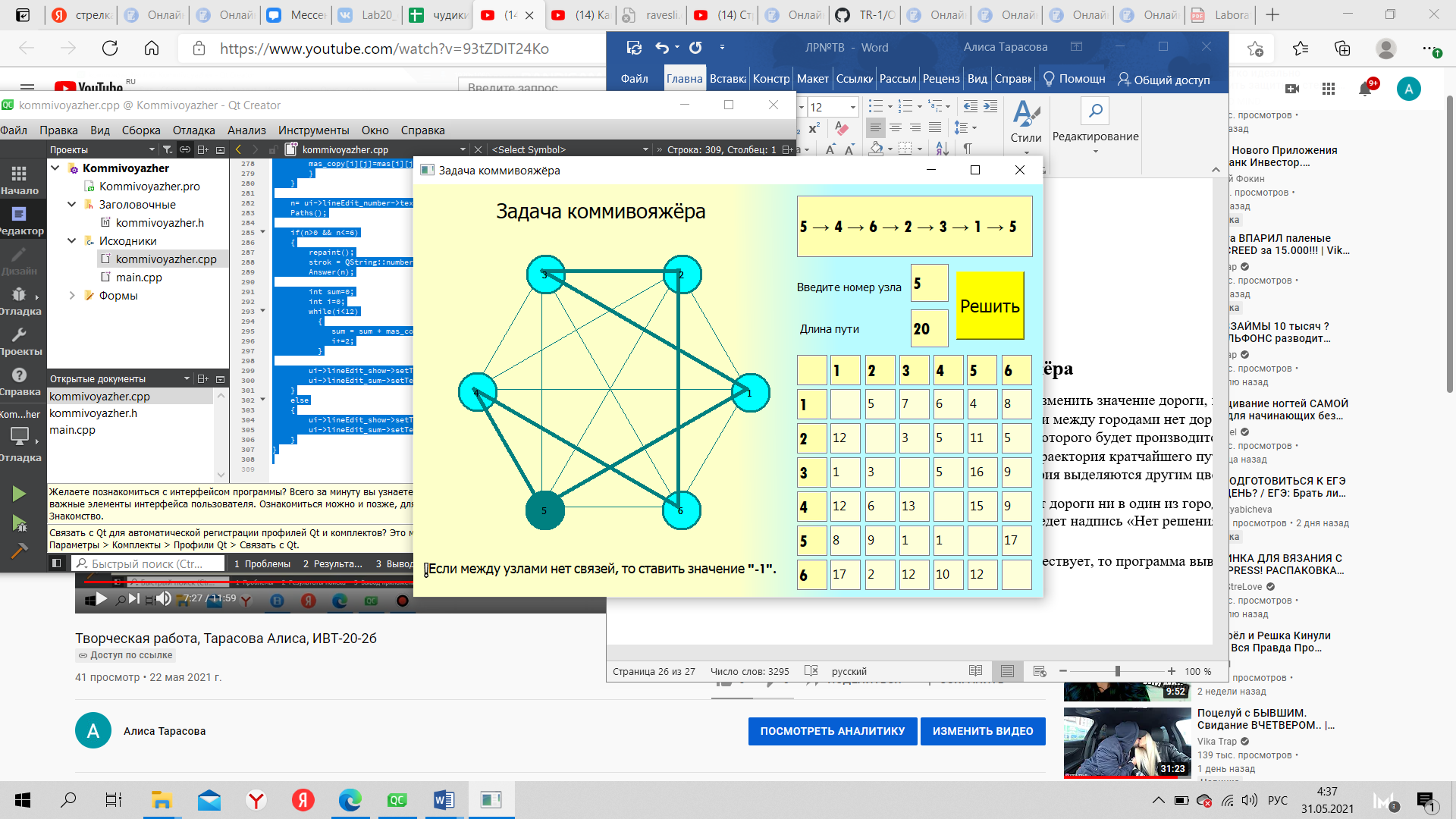
Также есть кнопка для стирания только одного коэффициент. А кнопка «стереть всё» стирает все данные.

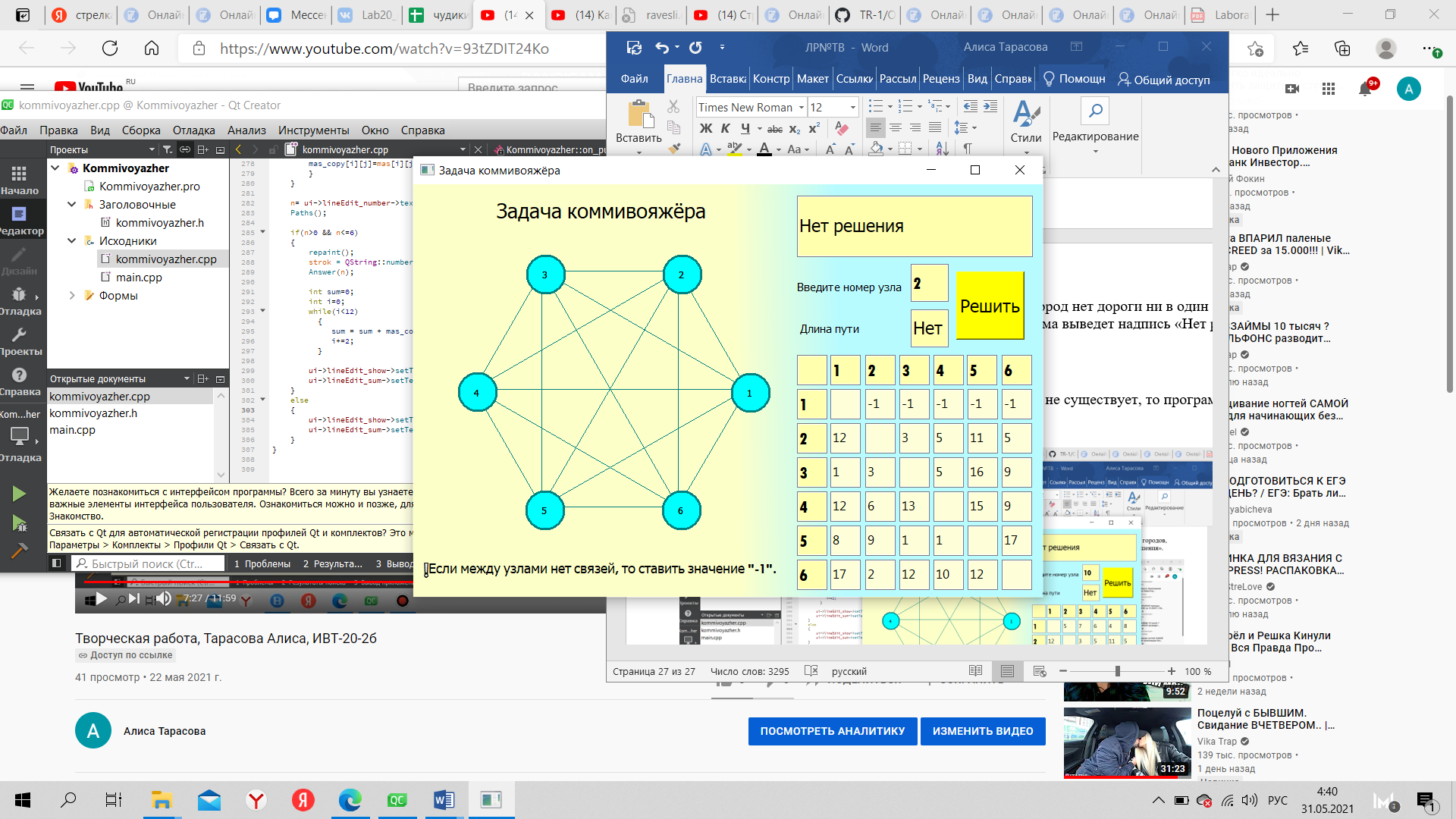
 

**4.2 Задача Коммивояжёра**

По умолчанию введены длины дорог, но пользователь может изменить значение дороги, введя его с клавиатуры в определённую ячейку. Но есть условие, что если между городами нет дороги, то значение должно стоять -1. Также есть окно для ввода узла, с которого будет производится поиск кратчайшего пути. После нажатие кнопки «Решить» выводит траектория кратчайшего пути и его длина. Визуально город с которого начинается путь, и траектория выделяются другим цветом.



Также предусмотрена ситуация, когда из какого-либо город нет дороги ни в один из городов, следовательно, траекторию построить нельзя и программа выведет надпись «Нет решения».



Также если пользователь введёт номер города, который не существует, то программа выведет надпись «Нет решения».

