

Практическая работа №3 – Вещественные числа

Цель работы

Знакомство с методами валидации вещественных чисел при выполнении математических операций.

Теоретическое введение

Работа с вещественными числами в Qt аналогична работе с целыми числами – их можно конвертировать из строки в число, так и обратно – из числа в строку:

```
double value_1 = text_1.toDouble(&correct_1);  
double value_2 = text_2.toDouble(&correct_2);  
...  
double sum = value_1 + value_2;  
QString result = QString::number(sum);
```

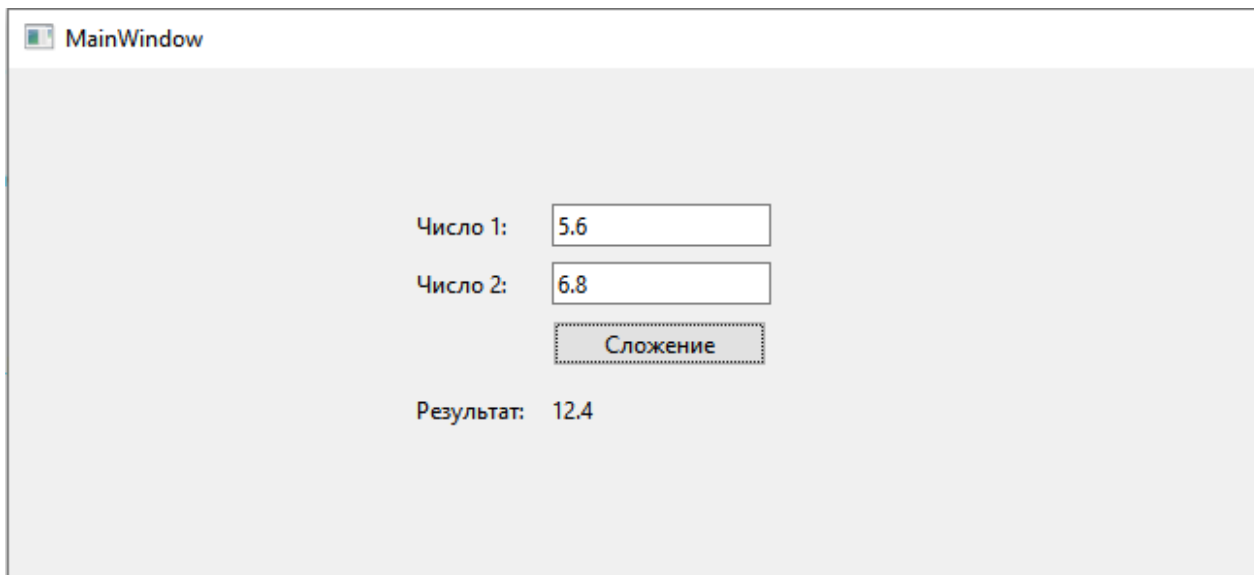
Однако, при работе с вещественными числами поддерживается работа только с десятичной системой счисления.

Так, код из предыдущей работы будет выглядеть следующим образом:

```
void MainWindow::on_pushButton_sum_clicked()    // метод обработки нажатия на кнопку суммы  
{  
    QString text_1 = ui->lineEdit_value_1->text();    // получение текстового значения первого числа  
    QString text_2 = ui->lineEdit_value_2->text();    // получение текстового значения второго числа  
  
    bool correct_1 = false;    // значение по умолчанию false  
    bool correct_2 = false;    // значение по умолчанию false  
  
    double value_1 = text_1.toDouble(&correct_1);    // конвертируем строку в вещественное число  
    double value_2 = text_2.toDouble(&correct_2);    // конвертируем строку в вещественное число  
  
    if(correct_1 && correct_2) // если оба значения корректны  
    {  
        double sum = value_1 + value_2;    // считаем сумму двух чисел  
        QString result = QString::number(sum); // конвертируем число в строку, в 10ю ССЧ  
        ui->label_result_value->setText(result);    // выводим полученную сумму на экран  
    }  
    else    // если значения некорректные  
    {  
        ui->label_result_value->setText("Ошибка!");    // выводим полученную сумму на экран  
    }  
}
```

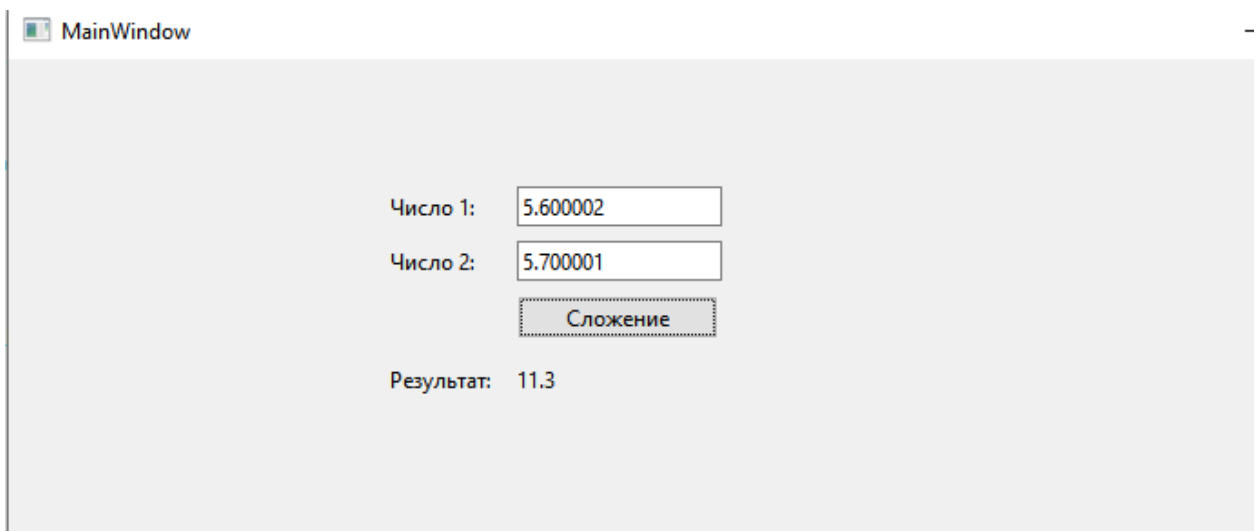
Код суммы вещественных чисел

В случае, если числа небольшие, ответ, полученный в результате выполнения математической операции, будет достаточно точным:



Результат суммы вещественных чисел

Однако, если значение после запятой небольшое, оно может быть округлено автоматически, по умолчанию, до 6 символов:



Результат сложения больших чисел

Для отображения точного значения, необходимо указать тип форматирования строки при конвертации числа в строку:

```
QString result = QString::number(sum, 'f', 10);
```

В данном случае указан формат “f” и 10 знаков после запятой:

Число 1: 6.70000001

Число 2: 5.40000001

Сложение

Результат: 12.1000000200

Вывод 10 цифр после запятой

Всего существует 5 форматов:

g – автоматический выбор формата в нижнем регистре

G – автоматический выбор формата в верхнем регистре

e – экспоненциальный вид в нижнем регистре [-]9.9e[+|-]999

E – экспоненциальный вид в верхнем регистре [-]9.9E[+|-]999

f – указание точного количества символов после запятой

Код в данном случае будет выглядеть следующим образом:

```

void MainWindow::on_pushButton_sum_clicked()    // метод обработки нажатия на кнопку суммы
{
    QString text_1 = ui->lineEdit_value_1->text();    // получение текстового значения первого числа
    QString text_2 = ui->lineEdit_value_2->text();    // получение текстового значения второго числа

    bool correct_1 = false;    // значение по умолчанию false
    bool correct_2 = false;    // значение по умолчанию false

    double value_1 = text_1.toDouble(&correct_1);    // конвертируем строку в вещественное число
    double value_2 = text_2.toDouble(&correct_2);    // конвертируем строку в вещественное число

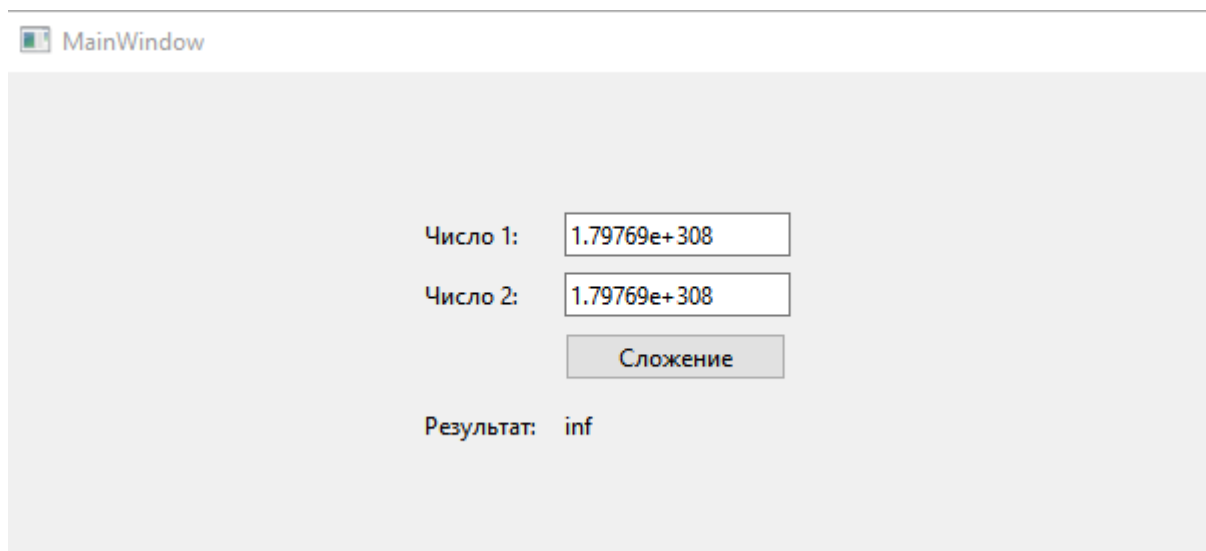
    if(correct_1 && correct_2)    // если оба значения корректны
    {
        double sum = value_1 + value_2;    // считаем сумму двух чисел
        QString result = QString::number(sum, 'f', 10);    // форматируем строку на 10 символов после запятой
        ui->label_result_value->setText(result);    // выводим полученную сумму на экран
    }
    else    // если значения некорректные
    {
        ui->label_result_value->setText("Ошибка!");    // выводим полученную сумму на экран
    }
}

```

Вывод 10 знаков после запятой

Другой проблемой, при работе с вещественными числами, может быть выход за пределы допустимых значений типа данных (машинный эпсилон) – для типа данных `double` этот предел равен от $-1.79769\text{e}+308$ до $+1.79769\text{e}+308$.

Таким образом, при сложении очень больших чисел, будет получено значение «бесконечность» (infinity):



Значение бесконечность при выполнении математической операции

Проверить значение на бесконечность можно проверить в Qt с помощью метода `qIsInf()`:

```

void MainWindow::on_pushButton_sum_clicked()    // метод обработки нажатия на кнопку суммы
{
    QString text_1 = ui->lineEdit_value_1->text();    // получение текстового значения первого числа
    QString text_2 = ui->lineEdit_value_2->text();    // получение текстового значения второго числа

    bool correct_1 = false;    // значение по умолчанию false
    bool correct_2 = false;    // значение по умолчанию false

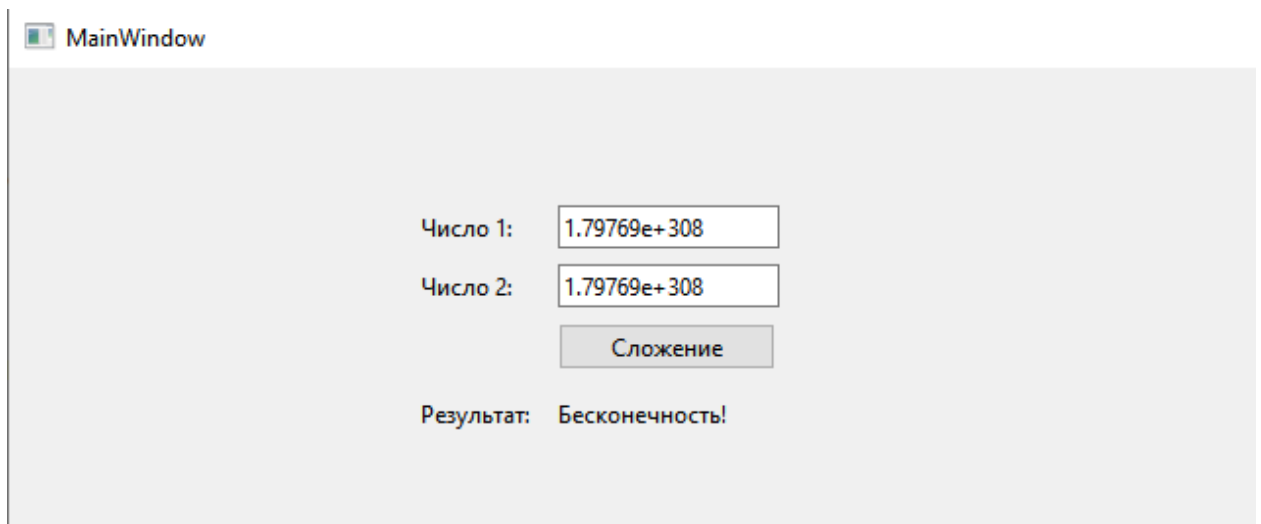
    double value_1 = text_1.toDouble(&correct_1);    // конвертируем строку в вещественное число
    double value_2 = text_2.toDouble(&correct_2);    // конвертируем строку в вещественное число

    if(correct_1 && correct_2) // если оба значения корректны
    {
        double sum = value_1 + value_2;    // считаем сумму двух чисел

        if(qIsInf(sum))    // если значение является бесконечностью
        {
            ui->label_result_value->setText("Бесконечность!");    // выводим ошибку
        }
        else
        {
            QString result = QString::number(sum, 'f', 10); // форматируем строку на 10 символов после запятой
            ui->label_result_value->setText(result);    // выводим полученную сумму на экран
        }
    }
    else // если значения некорректные
    {
        ui->label_result_value->setText("Ошибка!");    // выводим полученную сумму на экран
    }
}

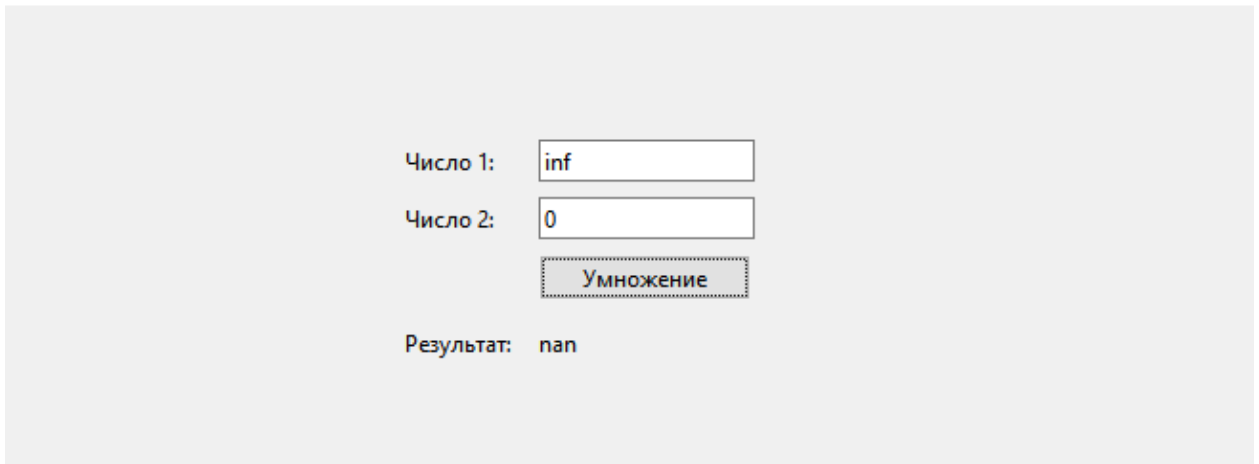
```

Проверка числа на бесконечность



Результат проверки на бесконечность

Также проблемой может быть некорректное значение, полученное в результате математической операции – деление на ноль, умножение или деление на бесконечность и т.п.:



Умножение бесконечность на ноль

Результатом умножения бесконечность на ноль, закономерно является NaN (not a number). Проверка таких значений в Qt выполняется методом `qIsNaN()`:

```
void MainWindow::on_pushButton_sum_clicked() // метод обработки нажатия на кнопку суммы
{
    QString text_1 = ui->lineEdit_value_1->text(); // получение текстового значения первого числа
    QString text_2 = ui->lineEdit_value_2->text(); // получение текстового значения второго числа

    bool correct_1 = false; // значение по умолчанию false
    bool correct_2 = false; // значение по умолчанию false

    double value_1 = text_1.toDouble(&correct_1); // конвертируем строку в вещественное число
    double value_2 = text_2.toDouble(&correct_2); // конвертируем строку в вещественное число

    if(correct_1 && correct_2) // если оба значения корректны
    {
        double mul = value_1 * value_2; // считаем произведение двух чисел

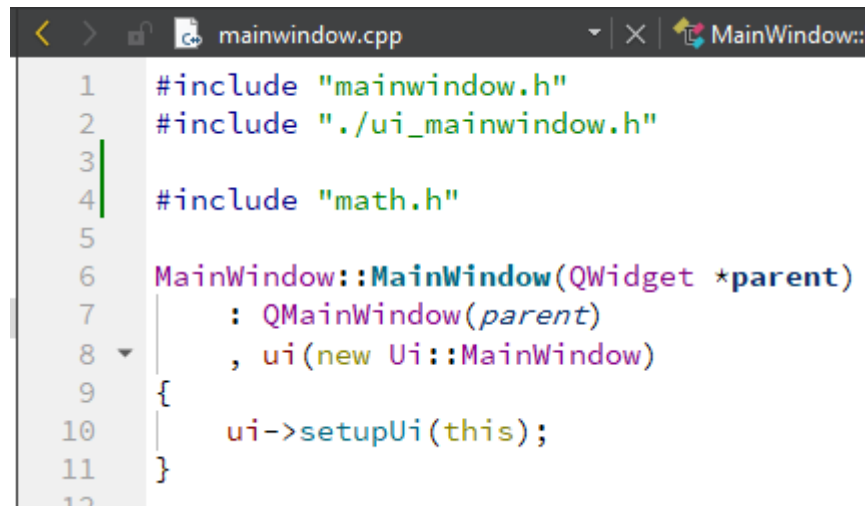
        if(qIsInf(mul)) // если значение является бесконечностью
        {
            ui->label_result_value->setText("Бесконечность!"); // выводим ошибку
        }
        else if(qIsNaN(mul)) // если значение не является числом
        {
            ui->label_result_value->setText("Не число!"); // выводим ошибку
        }
        else
        {
            QString result = QString::number(mul, 'f', 10); // форматируем строку на 10 символов после запятой
            ui->label_result_value->setText(result); // выводим результат
        }
    }
    else // если значения некорректные
    {
        ui->label_result_value->setText("Ошибка!"); // выводим ошибку
    }
}
```

Проверка на не число

Выполнять данные проверки необходимо при выполнении любых математических операций, т.к. пользователь может ввести любое число, а

также нельзя исключать вероятность наличия ошибок в коде, которые могут привести к ошибке.

Для выполнения сложных математических операций, таких, как синус, косинус и т.д. необходимо подключить библиотеку математических функций. Подключается она путём добавления директивы `#include "math.h"` в начало файла кода программы:



```
mainwindow.cpp
1  #include "mainwindow.h"
2  #include "./ui_mainwindow.h"
3
4  #include "math.h"
5
6  MainWindow::MainWindow(QWidget *parent)
7      : QMainWindow(parent)
8      , ui(new Ui::MainWindow)
9  {
10     ui->setupUi(this);
11 }
12
```

Подключение библиотеки математических функций

Для расчёта синуса в радианах используется функция `sin()`:

```
double s = sin(0.523599);    // 30 градусов
```

Для расчёта синуса в градусах, необходимо выполнить перевод градусов в радианы:

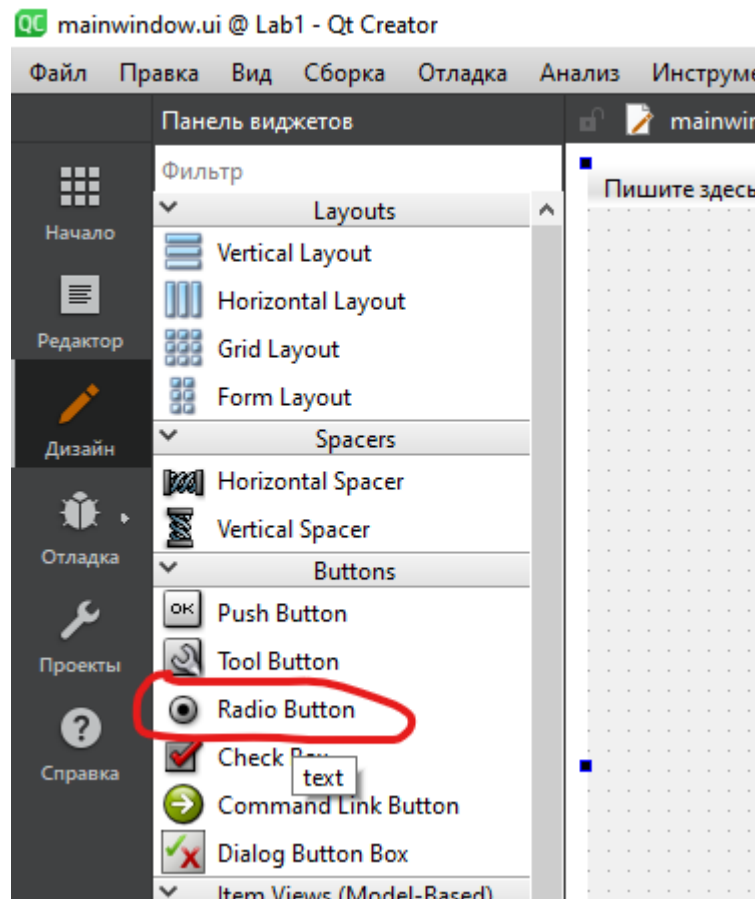
```
double grad = 30.0;
double rad = grad * M_PI / 180.0;
double s = sin(rad);
```

Или в одну строку:

```
double s = sin(30.0 * M_PI / 180.0);
```

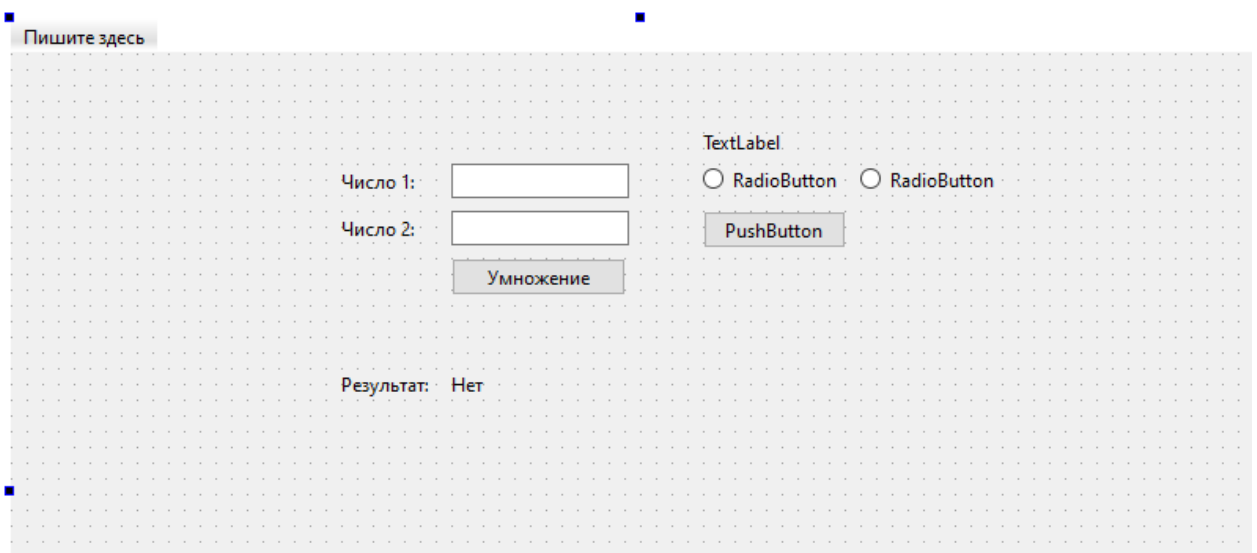
Для выбора, как считать значение – в градусах, или радианах, на интерфейсе можно использовать переключатель – `radioButton`.

В редакторе интерфейса он находится в блоке `Buttons`:



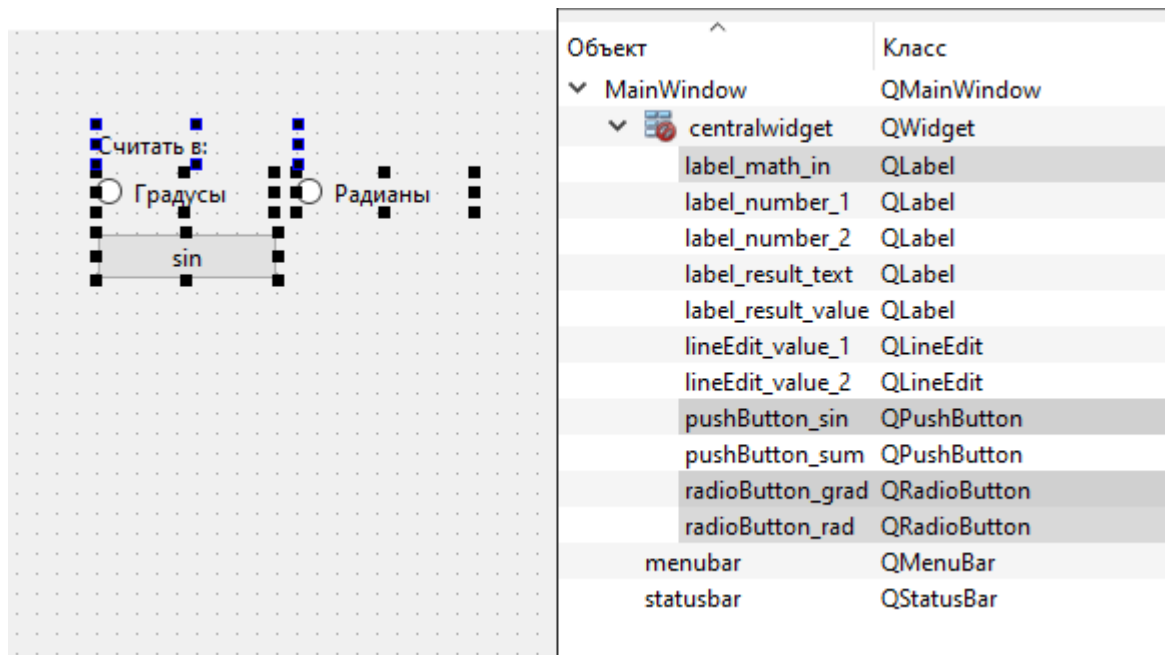
Выбор переключателя

Добавим 2 виджета Radio Button, Label и Push Button:



Переключатели на интерфейсе

Переключатели необходимо назвать “radioButton_grad” и “radioButton_rad” соответственно, а также переименовать название и указать текст для всех остальных добавленных виджетов:



Указание всех свойств виджетов

Реализуем действие на нажатие кнопки синуса:

```
// обработка нажатия кнопки синуса
void MainWindow::on_pushButton_sin_clicked()
{
}

```

Обработка нажатия кнопки синуса

Для проверки, нажат ли Radio Button, необходимо использовать метод `isChecked()`:

```

// обработка нажатия кнопки синуса
void MainWindow::on_pushButton_sin_clicked()
{
    // добавить все необходимые проверки!
    double value = ui->lineEdit_value_1->text().toDouble();

    if(ui->radioButton_grad->isChecked())    // если выбран расчёт в градусах
    {
        double s = sin(value * M_PI / 180.0);    // переводим градусы в радианы и считаем синус
        ui->label_result_value->setNum(s);
    }
    else if(ui->radioButton_grad->isChecked())    // если выбран расчёт в радианах
    {
        double s = sin(value);    // считаем синус сразу в радианах
        ui->label_result_value->setNum(s);
    }
    else    // если не выбран никакой расчёт
    {
        ui->label_result_value->setText("Выберите тип!");
    }
}
}

```

Обработка значений Radio Button

Таким образом возможна организация множественного выбора, на примере выбора, в каких единицах производить расчёт.

Задание

Используя программу, созданную в практической работе №2, расширить её следующим образом:

1. Заменить все действия с целыми числами на вещественные.
2. Указание системы счисления в данной работе не требуется (все числа вводятся в десятичной системе счисления).
3. Выполнить валидацию входных и выходных значений математических операций.
4. Реализовать тригонометрические функции – синус, косинус, тангенс, котангенс, арксинус, арккосинус.
5. Расчёт тригонометрических функций должен производиться в градусах и радианах – по выбору пользователя, путём переключателя.

Контрольные вопросы для защиты работы

1. Реализация дополнительной математической операции.