|  |  |
| --- | --- |
| ДИСЦИПЛИНА | Технологии индустриального программирования |
| ИНСТИТУТ | ИПТИП |
| КАФЕДРА | Индустриального программирования |
| ВИД УЧЕБНОГО МАТЕРИАЛА | Методические указания к практическим занятиям по дисциплине |
| ПРЕПОДАВАТЕЛЬ | Александров Алексей Сергеевич |
| СЕМЕСТР | 3 семестр, 2024/2025 уч. год |

# Практическое занятие №4

## Вещественные числа в Qt. Бесконечность и не число.

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);

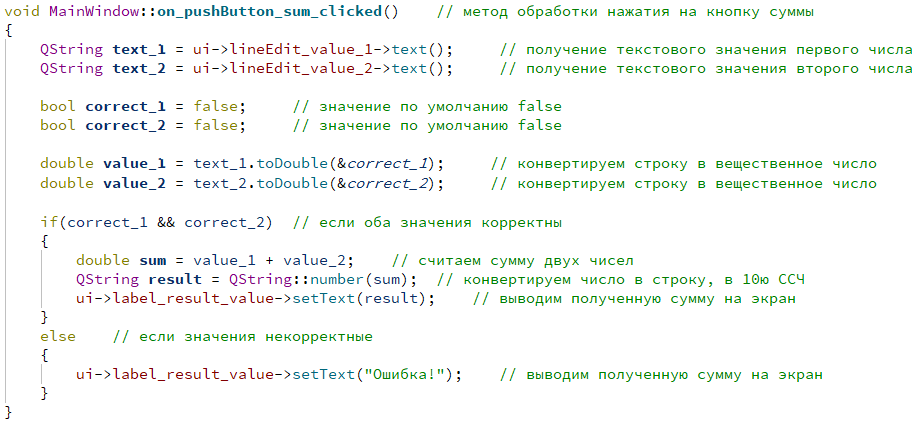
…

double sum = value\_1 + value\_2;

QString result = QString::number(sum);

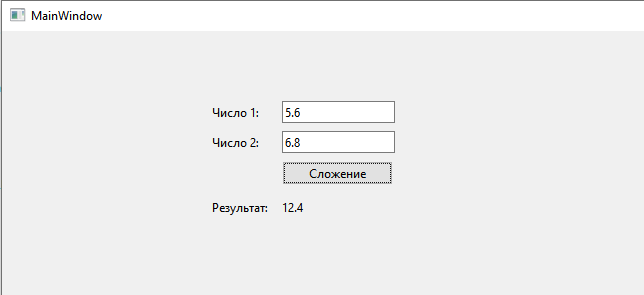
Однако, при работе с вещественными числами поддерживается работа только с десятичной системой счисления.

Так, код из предыдущей работы будет выглядеть следующим образом:



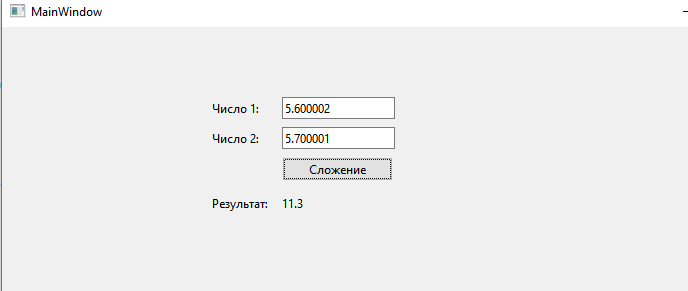
Код суммы вещественных чисел

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:

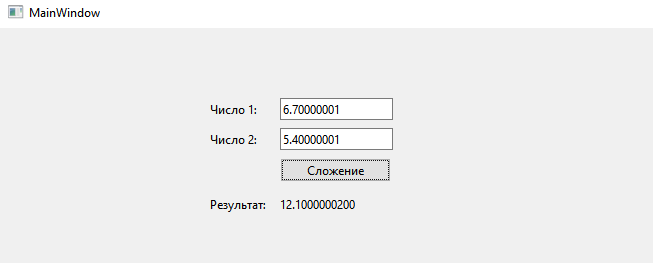


Результат сложения небольших чисел

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:



Вывод 10 цифр после запятой

Всего существует 5 форматов:

g – автоматический выбор формата в нижнем регистре

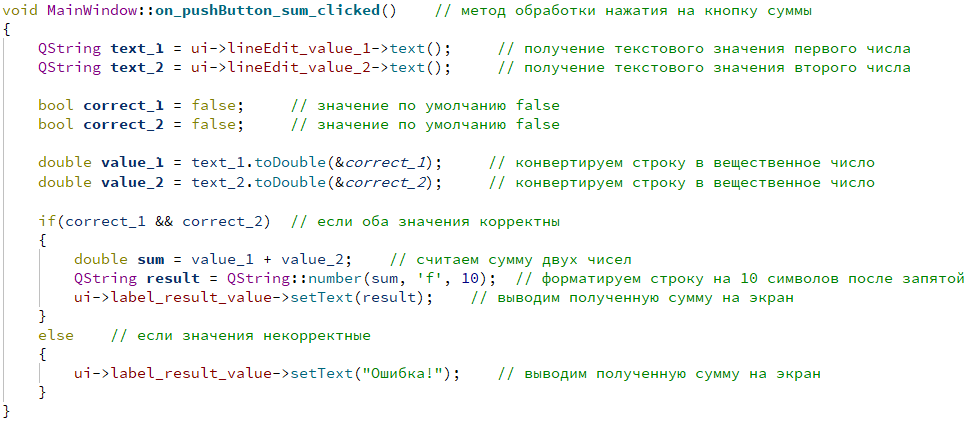
G – автоматический выбор формата в верхнем регистре

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

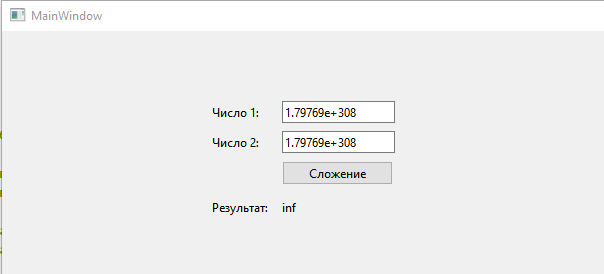
Код в данном случае будет выглядеть следующим образом:



Вывод 10 знаков после запятой

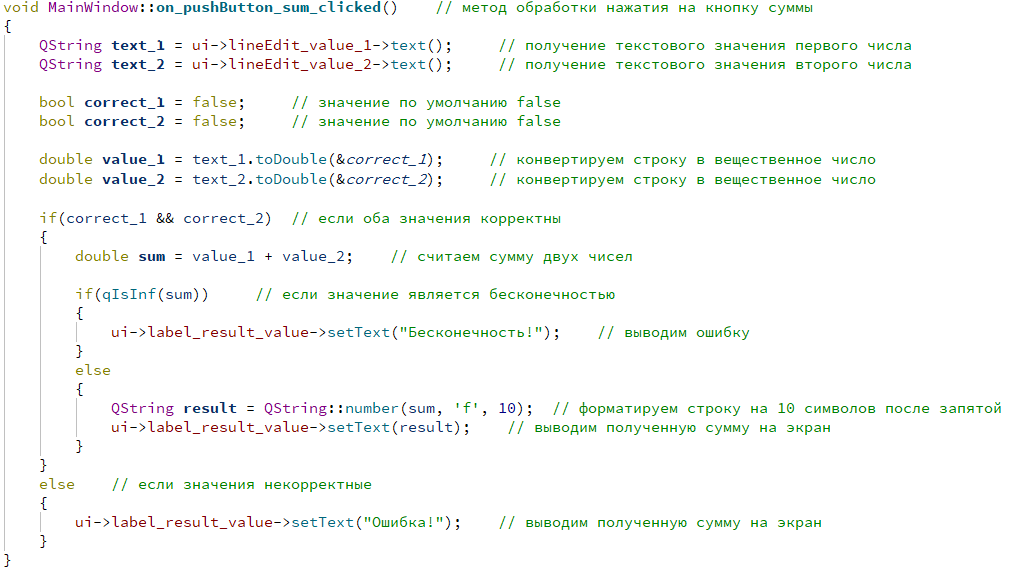
Другой проблемой, при работе с вещественными числами, может быть выход за пределы допустимых значений типа данных (машинный эпсилон) – для типа данных double этот предел равен от -1.79769e+308 до +1.79769e+308.

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

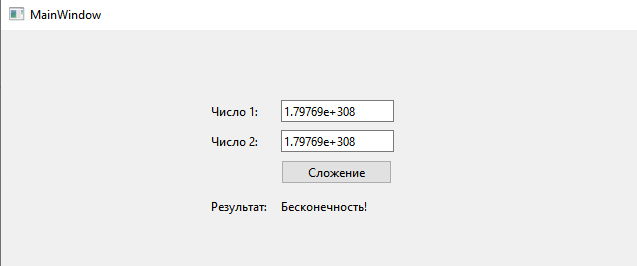


Значение бесконечноть при выполнении математической операции

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

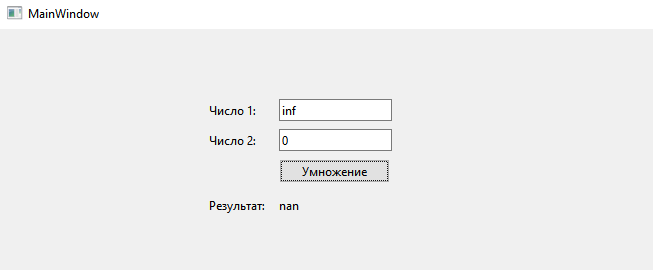


Проверка числа на бесконечность



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

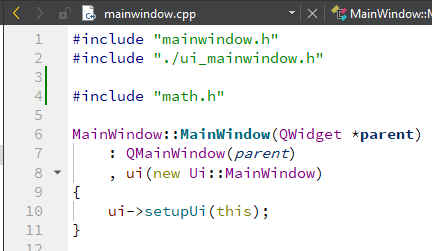
Результатом умножения бесконечность на ноль, закономерно является NaN (not a number). Проверка таких значений в Qt выполняется методом qIsNan():



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:



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);

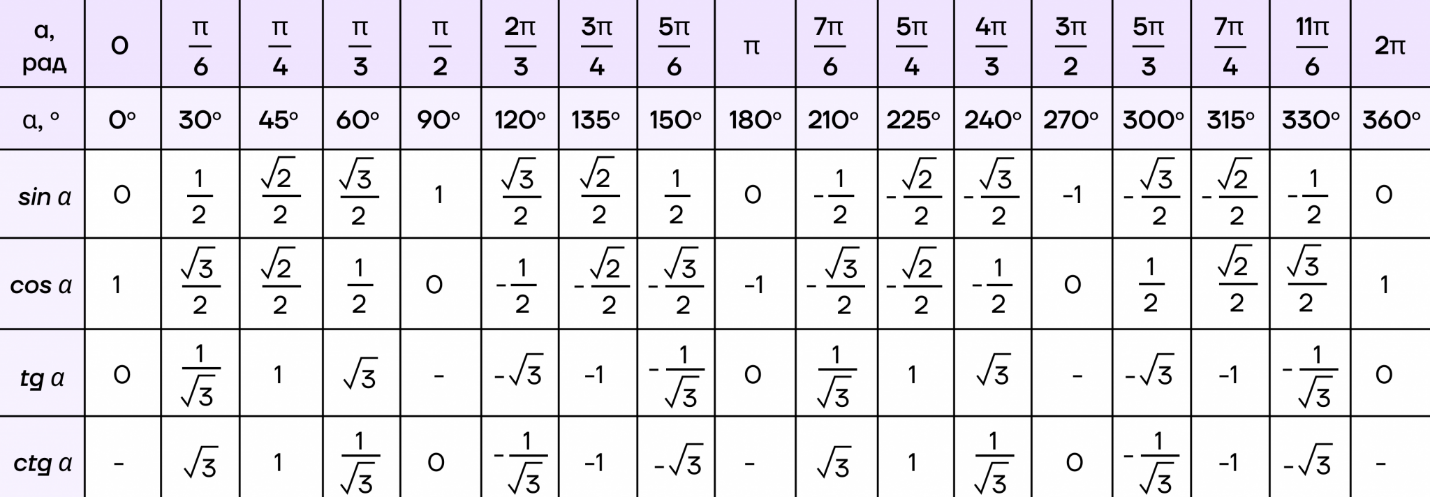
Аналогично можно посчитать обратное значение – arcsin():

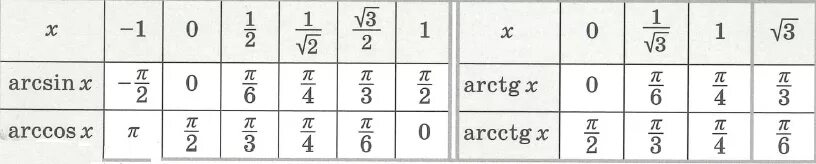
double s = asin(0.5); // 0.523599 или в радианах

Полученное значение можно перевести в градусы:

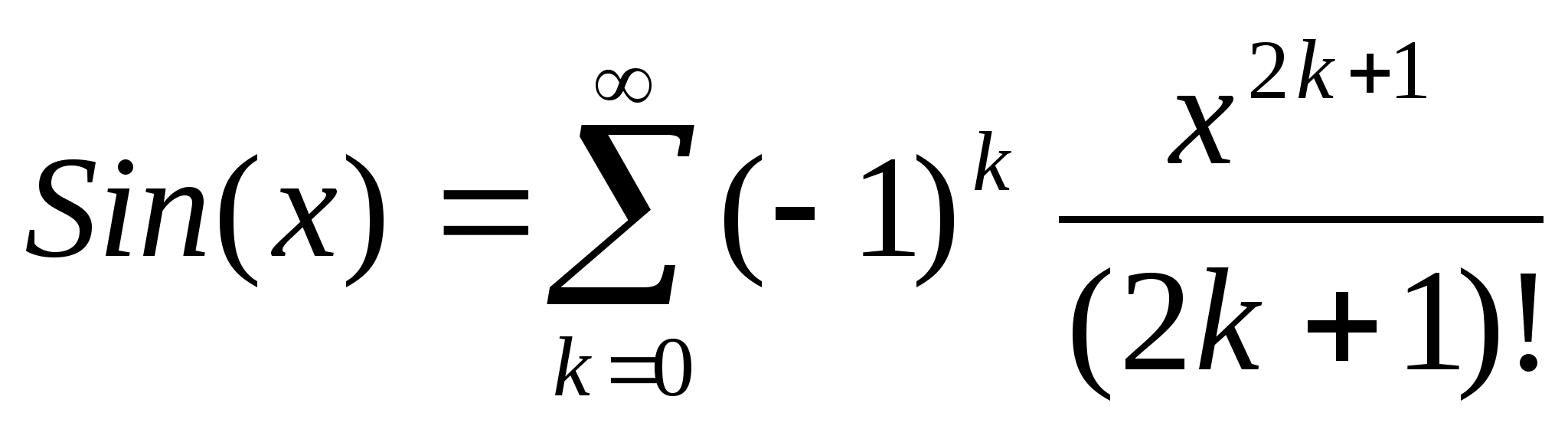
double s = asin(0.5) \* 180.0 / M\_PI; // 30 градусов

Проверить полученные значения можно по таблице тригонометрических функций:





Однако, стоит учитывать, что все тригонометрические функции в Qt считаются при помощи разложения их в ряд Тейлора до определённого N:



Из-за этого значения, получаемые такими функциями, являются приближёнными, поэтому полученные значения необходимо округлять:

double s = sin(180.0 \* M\_PI / 180.0); // 1.22465e-16

При округлении:

double s = sin(180.0 \* M\_PI / 180.0);

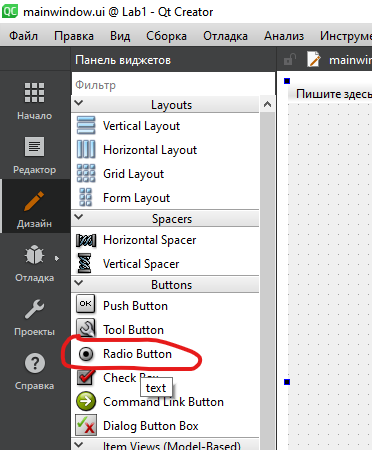
s = round(s \* 10) / 10; // округление до 1 знака после запятой

// s = 0

Таким образом, при значении 180 градусов, функция sin() вернула значение близкое к нулю – 1.22465e-16, или 1.22 \* 10 в степени -16. Округление же, позволило получить правильное значение.

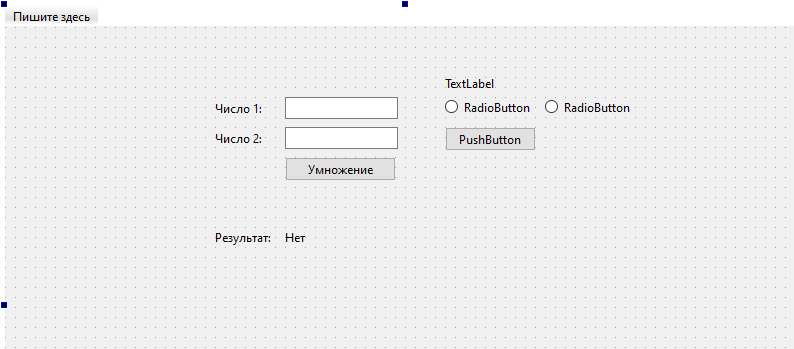
Для выбора, как считать значение – в градусах, или радианах, на интерфейсе можно использовать переключатель – radioButton.

В редакторе интерфейса он находится в блоке Buttons:



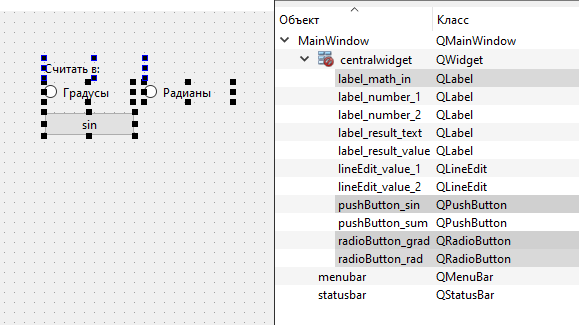
Выбор переключателя

Добавим 2 виджета Radio Button, Label и Push Button:



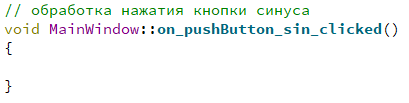
Переключатели на интерфейсе

Переключатели необходимо назвать “radioButton\_grad” и “radioButton\_rad” соответственно, а также переименовать название и указать текст для всех остальных добавленных виджетов:



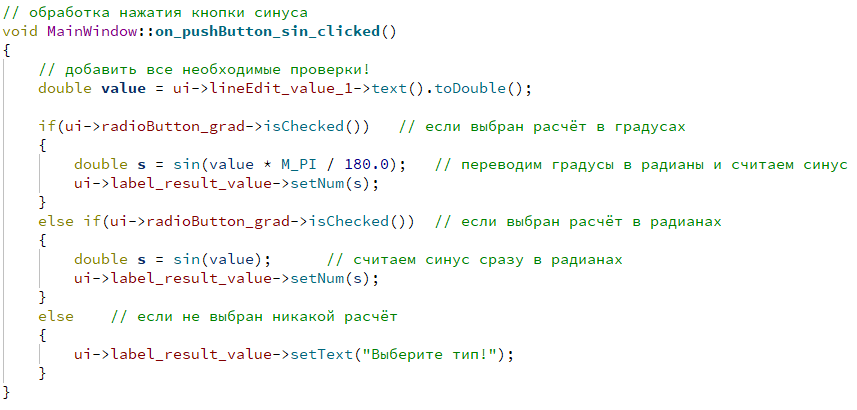
Указание всех свойств виджетов

Реализуем действие на нажатие кнопки синуса:



Обработка нажатия кнопки синуса

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



Обработка значений Radio Button

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