# Задание по программированию: Класс Rational

Условие

В этой задаче вам надо разработать класс для представления рациональных чисел и внедрить его в систему типов языка С++ так, чтобы им можно было пользоваться естественным образом. Задание состоит из нескольких частей. К каждой части приложен файл с заготовкой программы, который имеет следующую структуру

#include <iostream>

using namespace std;

 // Комментарии, которые говорят, что именно нужно реализовать в этой программе

 int main() {

    // Набор юнит-тестов для вашей реализации

    cout << "OK" << endl;

    return 0;

}

Вам нужно, **не меняя тело функции main,** реализовать то, что предложено в комментариях, так, чтобы ваша программа выводила на экран сообщение OK. Функция main содержит ряд юнит-тестов, которые проверяют правильность вашей реализации. Если эти тесты находят ошибку в вашей реализации, то вместо OK программа выведет подсказку о том, что именно работает неправильно.

Когда ваша программа начнёт выводить OK, можете отправлять свой файл на проверку. Ваша реализация будет тестироваться на другом, более полном наборе тестов. **То есть, если ваша программа выводит OK, то это не даёт гарантии, что ваша реализация будет принята тестирующей системой.** В случае неудачной посылки тестирующая система также выдаст вам подсказку о том, какая именно часть реализации работает неправильно.

Внимательно изучите набор юнит-тестов для **каждой** заготовки решения. Тесты описывают требования, предъявляемые к вашей реализации, которые могут быть не указаны в условии задач

На проверку можно отправлять весь файл с вашей реализацией: функцию main удалять из него не нужно.

## Часть 1

В первой части Вам надо реализовать класс Rational, который представляет собой рациональное число вида p/q, где p — целое, а q — натуральное и диапазоны возможных значений p, q таковы, что могут быть представлены типом int. При этом, класс Rational должен иметь следующий интерфейс:

class Rational {

public:

  Rational();

  Rational(int numerator, int denominator);

  int Numerator() const;

  int Denominator() const;

};

Класс Rational должен гарантировать, что p/q — это несократимая дробь. Например, код

Rational r(4, 6);

cout << r.Numerator() << '/' << r.Denominator();

должен выводить «2/3» — именно поэтому мы делаем Rational классом, а не структурой. Структура позволила бы нарушить этот инвариант:

struct Rational {

    int numerator, denominator;

};

Rational r{2, 3};

r.numerator = 6; // Нарушен инвариант несократимости

Список требований, предъявляемых к реализации интерфейса класса Rational:

1. Конструктор по умолчанию должен создавать дробь с числителем 0 и знаменателем 1.
2. При конструировании объека класса Rational с параметрами p и q должно выполняться сокращение дроби p/q (здесь вам может пригодиться решение задачи «Наибольший общий делитель»).
3. Если дробь p/q отрицательная, то объект Rational(p, q) должен иметь отрицательный числитель и положительный знаменатель.
4. Если дробь p/q положительная, то объект Rational(p, q) должен иметь положительные числитель и знаменатель (обратите внимание на случай Rational(-2, -3)).
5. Если числитель дроби равен нулю, то знаменатель должен быть равен 1.

**Гарантируется, что ни в одном из тестов, на которых будет тестироваться ваша реализация, не будет знаменателя, равного нулю.**

### Замечание:

В стандарте языка C++17 для вычисления НОД имеется стандартная функция gcd. Допустимо использовать данную функцию для сокращения дробей. Подробнее с данной функцией и её ограничениями можно ознакомиться по ссылке: <https://en.cppreference.com/w/cpp/numeric/gcd>

## Часть 2

Реализуйте для класса Rational операторы ==, + и - так, чтобы операции с дробями можно было записывать естественным образом. Например, следующий код должен быть валидным:

const Rational r = Rational(1, 2) + Rational(1, 3) - Rational(1, 4);

if (r == Rational(7, 12)) {

  cout << "equal";

}

**Гарантируется, что во всех тестах, на которых будет тестироваться ваша реализация, числители и знаменатели ВСЕХ дробей (как исходных, так и получившихся в результате соответствующих арифметических операций) будут укладываться в диапазон типа int.**

## Часть 3

Аналогично предыдущей части, реализуйте операторы \* и /. Например, следующий код должен быть валидным:

const Rational r = Rational(1, 2) \* Rational(1, 3) / Rational(1, 4);

if (r == Rational(2, 3)) {

  cout << "equal";

}

**Как и в предыдущем случае, гарантируется, что во всех тестах, на которых будет тестироваться ваша реализация, числители и знаменатели ВСЕХ дробей (как исходных, так и получившихся в результате соответствующих арифметических операций) будут укладываться в диапазон типа int.**

**Кроме того, гарантируется, что ни в одном из тестов, на которых будет тестироваться ваша реализация, не будет выполняться деление на ноль.**

## Часть 4

В этой части вам нужно реализовать операторы ввода и вывода для класса Rational. В результате у вас должен работать, например, такой код

Rational r;

cin >> r;

cout << r;

## Часть 5

Наконец, вам нужно реализовать возможность использовать объекты класса Rational в качестве элементов контейнера set и ключей в контейнере map. Пример:

set<Rational> rationals;

rationals.insert(Rational(1, 2));

rationals.insert(Rational(1, 3));

map<Rational, string> name;

name[Rational(1, 2)] = "одна вторая";