Задача #2. Рациональные числа

Введение

Разработайте класс рационального числа, описывающего дробь с возможностью использовать числитель и знаменатель неограниченной длины.

Рациональные числа могут оказаться крайне полезными, когда использование вещественных чисел в виде floating point дает слишком большую накапливающуюся погрешность, и становится, например, уже невозможно дать ответ на вопрос x > y, т.к. погрешность d(x) вычисления x больше | y – x |.

Класс обладает схожим набором функциональности, что и встроенные вещественные типы в C++.

Требования

1. Конструкторы не должны задавать неявное преобразование типа.
2. Операторы преобразования не должны задавать неявное преобразование типа.
3. Вам нет необходимости поддерживать отрицательные рациональные числа.
4. В академических целях при решении задачи следует избегать использования стандартных классов STL для управления памятью (vector, unique_ptr, ...). Можно использовать std::string, но не для внутреннего представления рационального числа.
5. Класс должен называться rational и находится в пространстве apa (arbitrary-precision arithmetic).
6. Ваше решение должно содержать ровно два файла: rational.h с объявлениями и rational.cpp с определениями. Никаких makefile'ов не требуется.
7. Деление длинного целого числа на ноль приводит к тому же эффекту, что и деление обычного int'a.
8. В этом задании не должна использоваться библиотека boost::operators.
9. Если при конструировании или зачитывании вашего числа из std::istream входная строка не удовлетворяет формату, должно быть брошено исключение apa::wrong_format.
10. Дробь в Вашем рациональном числе дожна быть приведена к сокращенному виду после операций над ней. Т.е. рациональное число, созденное по "2 / 6", будет выводится как "1 / 3".

Минимальный набор операций, поддерживаемых классом:

Конструкторы

1. Без параметров.
2. Копирования.
3. Задание целого числителя и знаменателя в виде типа int. Знаменатель имеет дефолтное значение, равное 1.
4. Такой же конструктор, но числитель и знаменатель заданы в виде строки std::string. Также дефолтное значение знаменателя – 1.

Операторы

1. Копирующее присваивание.
2. Возможность использования в условных выражениях (true, если не ноль).

3. Ввод/вывод в стандартные потоки (std::ostream/std::istream).
    a. Ввод/вывод числителя и знаменателя аналогичен обычным целым числам.
    b. При выводе между числителем и знаменателем через пробел ставится знак ‘/’, даже, если знаменатель равен 1.
    c. Аналогично, символ деления ‘/’ ожидается при вводе. Числитель и знаменатель отделены пробелом.
4. Унарные +=, *=, /=
5. Бинарные +, *, /
6. Сравнения (==, !=, <, <=, >=, >)

Функции

1. string *str*() const;
   возвращает строковое представление числа, аналогичное тому, что описано в формате вывода.

Рекомендации

1. Лучше делать базу системы счисления как можно больше. Например, uint32_t.
2. Если какая-то функция может быть реализована через public интерфейс класса, сделайте ее внешней.

Дополнительные усложнения (по желанию)

1. Small object optimization. **+0,5 балла при успешной сдаче.**
2. Copy on write (ленивое копирование). **+0,5 балла при успешной сдаче.**
3. Move semantics (constructor & operator=). **+0,25 балла при успешной сдаче.**