## МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

**(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСТИТЕТ)**

ЛАБОРАТОРНАЯ РАБОТА №1

## по курсу объектно-ориентированное программирование I семестр, 2021/22 уч. год

Студент: *Соколов Даниил Витальевич, группа М8О-207Б-20*

Преподаватель: *Дорохов Евгений Павлович*

# Условие

Задание: 3: Создать класс Rational для работы с рациональными дробями.Реализовать все арифметические операции. Реализовать операции сравнения Разработать программу на языке C++ согласно варианту задания. Программа должна получать данные из стандартного ввода и выводить данные в стандартный вывод.

# Описание программы

Исходный код лежит в файле main.cpp

Я создал класс Racional, для такого объекта как рациональная дробь. Определил в классе все операции над рациональными дробями вместе с функциями. Простая работа.

# Дневник отладки

Ошибок не наблюдалось.

# Недочёты

Недочётов не заметил.

# Вывод

В данной лабораторной работе была написан обычный класс без особого применения принципов ООП. Я приобрел навыки написания не только классов, но и операции над объектами класса. А также научился писать конструктуры к классам. Работа очень простая, особенно для тех, кто уже был знаком с данной парадигмой.

# Исходный код

#include <iostream> #include <string.h> class Racional

{ public:

Racional() : numerator(0), denominator(0){};

Racional(int a, int b) : numerator(a), denominator(b){};

int get\_num() const

{

return numerator;

}

int get\_den() const

{

return denominator;

}

friend Racional operator+(const Racional &rac1, const Racional &rac2);

friend Racional operator-(const Racional &rac1, const Racional &rac2);

friend Racional operator\*(const Racional &rac1, const Racional &rac2);

friend Racional operator/(const Racional &rac1, const Racional &rac2);

friend std::ostream &operator<<(std::ostream &out, const Racional &rac);

friend std::istream &operator>>(std::istream &in, Racional &rac);

private:

int numerator;

int denominator;

Racional reduce()

{

bool \_end = false;

for (int i = 2; i <= abs(this->numerator); i++)

{

while (this->numerator % i == 0 && this->denominator % i == 0)

{

this->numerator /= i;

this->denominator /= i;

}

}

}

};

std::ostream &operator<<(std::ostream &out, const Racional &rac)

{

out << "Numerator is " << rac.get\_num() << std::endl;

out << "Denomirator is " << rac.get\_den() << std::endl;

return out;

}

std::istream &operator>>(std::istream &in, Racional &rac)

{

in >> rac.numerator;

in >>rac.denominator;

return in;

}

Racional operator+(const Racional &rac1, const Racional &rac2)

{

Racional res(rac1.get\_num() \* rac2.get\_den() + rac1.get\_den() \* rac2.get\_num(), rac1.get\_den()

\* rac2.get\_den());

res.reduce();

return res;

};

Racional operator-(const Racional &rac1, const Racional &rac2)

{

Racional res(rac1.get\_num() \* rac2.get\_den() - rac1.get\_den() \* rac2.get\_num(), rac1.get\_den()

\* rac2.get\_den());

res.reduce();

return res;

};

Racional operator\*(const Racional &rac1, const Racional &rac2)

{

Racional res(rac1.get\_num() \* rac2.get\_num(), rac1.get\_den() \* rac2.get\_den());

res.reduce();

return res;

};

Racional operator/(const Racional &rac1, const Racional &rac2)

{

Racional res(rac1.get\_num() \* rac2.get\_den(), rac1.get\_den() \* rac2.get\_num());

res.reduce();

return res;

};

bool operator>(const Racional &rac1, const Racional &rac2)

{

return (rac1.get\_num() \* rac2.get\_den() - rac2.get\_num() - rac1.get\_den() > 0);

}

bool operator<(const Racional &rac1, const Racional &rac2)

{

return (rac1.get\_num() \* rac2.get\_den() - rac2.get\_num() - rac1.get\_den() < 0);

}

bool operator==(const Racional &rac1, const Racional &rac2)

{

return (rac1.get\_den() == rac2.get\_den() && rac2.get\_num() == rac2.get\_den());

}

int main()

{

Racional a;

Racional b;

//std::cout<<"Write first racional number: ";

std::cin>>a;

//std::cout<<"Write second racional number: ";

std::cin>>b;

std::cout<<"First number: \n"<<a<<std::endl;

std::cout<<"Second number: \n"<<b<<std::endl;

Racional c;

c = a + b;

std::cout << "Add: "<< c;

c = a - b;

std::cout << "Subtract: "<<c;

c = a \* b;

std::cout << "Multiply: "<<c;

c = a / b;

std::cout << "Devide: "<< c;

return 0;

}