МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСТИТЕТ)

ЛАБОРАТОРНАЯ РАБОТА №0.2

по курсу объектно-ориентированное программирование 3 семестр, 2021/22 уч. Год

Студент Абросимов Алексей Дмитриевич, группа М8О-207Б-20
Преподаватель Дорохов Евгений Павлович

Условие
Задание: Вариант 3: Рациональная (несократимая) дробь. Разработать программу на языке C++ согласно варианту задания. Программа на C++ должна собираться с помощью системы сборки CMake. Программа должна получать данные из стандартного ввода и выводить данные в стандартный вывод. Реализовать над объектами реализовать в виде перегрузки операторов. Реализовать пользовательский литерал для работы с константами объектов созданного класса.

Описание программы
Исходный код лежит в 1 файлике:
1.main.cpp: основная программа.

Дневник отладки
Результат программы при тестовых данных: 1 5

```
First number:
Numerator is 1
Denomirator is 5

Second number:
Numerator is 21
Denomirator is 5

Add:
Numerator is 22
Denomirator is 5

Subtract:
Numerator is -4
Denomirator is 1

Multiply:
Numerator is 21
Denomirator is 25

Devide:
Numerator is 1
Denomirator is 21
```

Недочёты

Выводы
В этой лабораторной работе я подробнее ознакомился с перегрузкой операторов и узнал, что такое пользовательский литерал. Пользовательский литерал оказался удобной вещью, потому что с его помощью можно уменьшить «количество» кода и не теряя при этом читаемость.

Ссылка на гитхаб: https://github.com/yungalexxxey/oop_labs/tree/main/lab0.2

Исходный код
main.cpp:

```
#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());
}

Racional operator""_rac(const char *str)
{
int num = 0;
int den = 0;
int i = 0;
for (; i < strlen(str); i++)
{
if (str[i] != '.')
{
num = num * 10 + str[i] - 48;
continue;
}
break;
}
++i;
for (; i < strlen(str); i++)
{
den = den * 10 + str[i] - 48;
}
return Racional(num, den);
}

int main()
{
Racional a;
Racional b=21.5_rac;
std::cin>>a;
std::cout<<"First number: \n"<<a<<std::endl;
std::cout<<"Second number: \n"<<b<<std::endl;
Racional c;
c = a + b;
std::cout << "Add: \n"<< c<<std::endl;
```

```
c = a - b;
std::cout << "Subtract: \n"<<c<<std::endl;
c = a * b;
std::cout << "Multiply: \n"<<c<<std::endl;
c = a / b;
std::cout << "Devide: \n"<< c<<std::endl;
return 0;
}
```