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

**ЛАБОРАТОРНАЯ РАБОТА №0.1**

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

Студент Карпов Даниил Антонович 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. Необходимо настроить сборку лабораторной работы с помощью CMake. Собранная программа должна называться oop\_exercise\_01 (в случае использования Windows oop\_exercise\_01.exe)

**Описание программы**

Исходный код лежит в 1 файлике:

1.main.cpp: основная программа.

**Дневник отладки**

Результат работы программы при тестовых данных: 1 5 1 2

First number:

Numerator is 1

Denomirator is 5

Second number:

Numerator is 1

Denomirator is 2

Add: Numerator is 7

Denomirator is 10

Subtract: Numerator is -3

Denomirator is 10

Multiply: Numerator is 1

Denomirator is 10

Devide: Numerator is 2

Denomirator is 5

**Недочёты**

**Выводы**

Эта лабораторная работа позволила мне научиться работать с основной в ООП вещью — классами. Как оказалось, создание класса — это очень просто и крайне удобно. До этого момента я не очень хорошо понимал, почему такой подход к программированию стал настолько широко распространённым, но теперь понял.

Ссылка на гитхаб**:**

**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());

}

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;

}

```