МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ  
ФЕДЕРАЦИИ МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСТИТЕТ)

ЛАБОРАТОРНАЯ РАБОТА №2 по курсу объектно-ориентированное программирование I семестр, 2021/22 уч. год

Студент *Клитная Анастасия Викторовна, группа М80-208Б-20*

Преподаватель *Дорохов Евгений Павлович*

**Цель:**

● Изучение основ работы с классами в С++;

● Перегрузка операций и создание литералов

## Требования к программе

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 3 файлах:

1. main.cpp - исполняемый код.

2. Money.h - специальный файл .h, содержащий прототипы используемых мною функций.

3. Money.cpp - реализация функций для моего задания.

4. CMakeLists.txt - специальный дополнительный файл типа CMakeLists.

**Дневник отладки**

Во время выполнения лабораторной работы программа не нуждалась в отладке, все ошибки компиляции были исправлены, а также замечена одна опечатка. После их исправления программа работала так, как было задумано изначально.

**Недочёты**  
Недочётов не было обнаружено.

**Выводы**

Лабораторная работа №2 - это, по сути, та же самая лабораторная №1, только предусматривающая возможность перегрузки операторов. Лабораторная была выполнена успешно, в ее процессе были еще раз осознаны основные принципы ООП и перегрузки операторов.

**Исходный код**

Money.cpp

#include <iostream>

#include "Modulo.h"

#include <cmath>

Money::Money(){

rub = 1;

cop = 1;

}

Money::Money(std::istream &is){

is >> rub;

is >> cop;

}

Money::Money(unsigned long long first, unsigned long long second){

rub = first;

cop = second;

}

Money Money::operator +(Money& a){

this->rub = this->rub + a.rub;

this->cop += a.cop;

return \*this;//->rub%this->cop + a.rub%a.cop;

}

Money Money::operator -(Money& a){

this->rub = this->rub - a.rub;

this->cop -= a.cop;

return \*this;//->rub%this->cop - a.rub%a.cop;

}

Money Money::operator \*(Money& a){

this->rub = this->rub \* a.rub;

this->cop \*= a.cop;

return \*this;//(this->rub%this->cop) \* (a.rub%a.cop);

}

Money Money::operator /(Money& a){

this->rub = this->rub / a.rub;

this->cop /= a.cop;

return \*this;//(this->rub%this->cop) / (a.rub%a.cop);

}

Money Money::operator ++(){

this->cop++;

this->rub++;

return \*this;

}

Money Money::operator --(){

this->cop--;

this->rub--;

return \*this;

}

std::ostream& operator<<(std::ostream& os,const Money& a){

os << a.rub << "," << a.cop << std::endl;

return os;

}

bool Money::operator==(const Money& other){

return this->cop == other.cop && this->rub == other.rub;

}

Money::~Money(){

std::cout << "Money has deleted" << std::endl;

}

Money.h

#ifndef MONEY\_H

#define MONEY\_H

#include <iostream>

class Money {

public:

Money();

Money(std::istream &is);

Money(unsigned long long rub, unsigned long long cop);

Money operator +(Money& a);

Money operator -(Money& a);

Money operator \*(Money& a);

Money operator /(Money& a);

Money operator ++();

Money operator --();

bool operator ==(const Money& other);

friend std::ostream& operator<<(std::ostream& os,const Money& a);

~Money();

private:

unsigned long long rub;

unsigned long long cop;

};

#endif // MONEY\_H

Main.cpp

#include <iostream>

#include "Money.h"

int main(){

Money c(std::cin);

Money a(10, 6);

Money b(12, 5);

std::cout << "Money objects: "<< a << b << c << std::endl;

std::cout << "Sum: " << a+b << std::endl;

std::cout << "Division of residues " << a/b << std::endl;

std::cout << "Multiplication of residuals " << a\*b << std::endl;

std::cout << "Sum " << c+b << std::endl;

std::cout << "Operator -- : " << --a;

std::cout << "Operator ++ : " << ++a;

}