МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ  
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСТИТЕТ)

**ЛАБОРАТОРНАЯ РАБОТА №2**

по курсу “Объектно-ориентированное программирование”

I семестр, 2021/22 учебный год

Студент: Чекменев Вячеслав Алексеевич

Группа:  *М8О-207Б-20*

Преподаватель: *Дорохов Евгений Павлович*

Москва, 2021

**Задание:** Разработать программу на языке C++ согласно варианту задания. Программа должна получать данные из стандартного ввода и выводить данные в стандартный вывод. Реализовать пользовательский литерал для работы с константами объектов созданного класса.

**Вариант №8:**

Создать класс Мoney для работы с денежными суммами. Сумма денег должна быть представлено двумя полями: типа unsigned long long для рублей и типа uint16\_t – для копеек. Дробная часть (копейки) при выводе на экран должна быть отделена от целой части запятой. Реализовать сложение сумм, вычитание, деление сумм, деление суммы на дробное число, yмножение на дробное число и операции сравнения.

**Описание программы:**

Исходный код разделён на 3 файла:

* money.h – описание класса деньги
* money.cpp – реализация класса деньги
* main.cpp – основная программа

**Дневник отладки:**

Ничего плохого не произошло.

**Вывод:**  
 В данной лабораторной работе я реализовал пользовательские литералы для перевода копеек в дробную часть рубля. Этот оператор, действительно, очень удобен.

**Листинг программ:**

**money.h:**

#ifndef MONEY\_H

#define MONEY\_H

#include <iostream>

class Money {

public:

Money();

Money(unsigned long long rub, uint16\_t kop);

Money(const Money &other);

Money &operator=(const Money &other);

Money(std::istream &is);

friend Money operator+(const Money& one, const Money &other);

friend Money operator-(const Money& one, const Money& other);

friend double operator/(const Money& one, const Money& other);

friend Money operator/(const Money& one, double arg);

friend Money operator\*(const Money& one, double arg);

bool operator==(const Money &other);

bool operator>(const Money &other);

bool operator<(const Money &other);

bool operator<=(const Money &other);

bool operator>=(const Money &other);

bool operator!=(const Money &other);

unsigned long long get\_rub() const;

uint16\_t get\_kop() const;

friend std::istream &operator>>(std::istream &is, Money &m);

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

virtual ~Money();

private:

unsigned long long rub\_;

uint16\_t kop\_;

};

long double operator""\_tokop(long double kop);

#endif // MONEY\_H

**money.cpp:**

#include "money.h"

Money::Money()

: rub\_(0), kop\_(0) {}

Money::Money(unsigned long long rub, uint16\_t kop)

: rub\_(rub), kop\_(kop) {}

Money::Money(const Money &other)

: Money(other.rub\_, other.kop\_) {}

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

{

is >> rub\_ >> kop\_;

}

unsigned long long Money::get\_rub() const {

return rub\_;

}

uint16\_t Money::get\_kop() const {

return kop\_;

}

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

{

if (this == &other) {

return \*this;

}

rub\_ = other.rub\_;

kop\_ = other.kop\_;

return \*this;

}

Money operator+(const Money& left, const Money& other) {

Money t;

t.rub\_ = left.rub\_ + other.rub\_;

t.kop\_ = left.kop\_ + other.kop\_;

if (t.kop\_ >= 100) {

t.rub\_;

t.kop\_ -= 100;

}

return t;

}

Money operator-(const Money& left, const Money& other) {

Money t;

Money r = left;

t.rub\_ = left.rub\_ - other.rub\_;

if (left.kop\_ < other.kop\_) {

t.rub\_--;

r.kop\_ = left.kop\_ + 100;

t.kop\_ = left.kop\_ - other.kop\_;

r.kop\_ -= 100;

}

else t.kop\_ = left.kop\_ - other.kop\_;

return t;

}

double operator/(const Money& left, const Money& other) {

double t, a, b;

a = (left.rub\_ \* 100 + left.kop\_);

b = (other.rub\_ \* 100 + other.kop\_);

t = a / b;

return t;

}

Money operator/(const Money& left, double arg) {

Money t;

double t2;

t2 = ((left.rub\_ \* 100 + left.kop\_) / arg) / 100;

t.rub\_ = (int)t2;

t.kop\_ = (t2 - (int)t2) \* 100;

return t;

}

Money operator\*(const Money& left, double arg) {

Money t;

double t2;

t2 = ((left.rub\_ \* 100 + left.kop\_) \* arg);

t.rub\_ = (int)(t2 / 100);

t.kop\_ = (t2 / 100 - (int)(t2 / 100)) \* 100;

return t;

}

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

return (this->rub\_ == other.rub\_ && this->kop\_ == other.kop\_);

}

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

return (this->rub\_ > other.rub\_ || (this->rub\_ == other.rub\_ && this->kop\_ > other.kop\_));

}

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

return (this->rub\_ < other.rub\_ || (this->rub\_ == other.rub\_ && this->kop\_ < other.kop\_));

}

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

return (this->rub\_ <= other.rub\_ && this->kop\_ <= other.kop\_);

}

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

return (this->rub\_ >= other.rub\_ && this->kop\_ >= other.kop\_);

}

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

return (this->rub\_ != other.rub\_ || this->kop\_ !=other.kop\_);

}

std::istream& operator>>(std::istream& is, Money& m)

{

is >> m.rub\_ >> m.kop\_;

return is;

}

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

{

os << m.rub\_ << ",";

if(m.kop\_ <= 0) {

os << "00" << std::endl;

}

else os << m.kop\_ << std::endl;

return os;

}

long double operator""\_tokop(long double kop){

return (kop / 100);

}

Money::~Money() {}

**main.cpp:**

#include "money.h"

int main(int argc, char const \*argv[])

{

Money number1;

Money number2;

std::cin >> number1 >> number2 >> std::endl;

std::cout << number1 << number1 << std::endl;

}