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

ЛАБОРАТОРНАЯ РАБОТА №1

по курсу “Объектно-ориентированное программирование”

I семестр, 2021/22 учебный год

Студентка: *Варламова Анна Борисовна*

Группа: *М8О-207Б-20*

Преподаватель: *Дорохов Евгений Павлович, каф. 806*

Москва, 2021

Задание:

Разработать программу на языке C++ согласно варианту задания. Программа должна получать данные из стандартного ввода и выводить данные в стандартный вывод.

Вариант №8:

Создать класс Money 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, умножение 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:

Проблем не возникло.

Тестирование:

```
First summ:
56 37
Second summ:
12 12
Number to div and multiply
5.8

56,37
12,12
68,49
44,25
4.65099
9,71
326,94

<: false
>: true
```

```
<=: false
>=: true
==: false
!=: true



First summ:
11 0
Second summ:
11 0
Number to div and multiply
67

11,00
11,00
22,00
0,00
1
0,16
737,00

<: false
>: false
<=: true
>=: true
==: true
!=: false
```

Вывод:

При выполнении работы я на практике познакомилась с базовыми принципами ООП, реализовала свой класс и функции для работы с ним. Также я перегрузила операторы ввода/вывода для удобной работы с реализованным классом. Всё то же самое мне придётся делать в следующих лабораторных.

Исходный код:

money.h:

```
#ifndef MONEY_H
#define MONEY_H
```

```
#include <iostream>

class Money
{
public:
    Money();
    Money(unsigned long long rubles, uint16_t penny);
    Money(const Money &other);
    Money(std::istream &is);
    Money &operator=(const Money &other);

    unsigned long long getRub() const;
    uint16_t getKop() const;

    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);

    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;
};

#endif // MONEY_H
```

money.cpp:

```
#include "money.h"

Money::Money()
    : rub(0), kop(0) {}

Money::Money(unsigned long long rubles, uint16_t penny)
    : rub(rubles), kop(penny) {}

Money::Money(const Money &other)
    : Money(other.rub, other.kop) {}

Money::Money(std::istream &is)
{
    is >> rub >> kop;
```

```
}

unsigned long long Money::getRub() const {
    return rub;
}

uint16_t Money::getKop() 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& one, const Money& other) {
    Money temp;
    temp.rub = one.rub + other.rub;
    temp.kop = one.kop + other.kop;
    if(temp.kop >= 100) {
        temp.rub++;
        temp.kop -= 100;
    }
    return temp;
}

Money operator-(const Money& one, const Money& other) {
    Money temp;
    Money r = one;
    temp.rub = one.rub - other.rub;
    if(one.kop < other.kop) {
        temp.rub--;
        r.kop = one.kop + 100;
        temp.kop = one.kop - other.kop;
        r.kop -= 100;
    }
    else temp.kop = one.kop - other.kop;
    return temp;
}

double operator/(const Money& one, const Money& other) {
    double temp, a, b;
    a = ((one.rub * 100) + one.kop);
    b = (( other. rub * 100) + other. kop);
    temp = a / b;
    return temp;
}

Money operator/(const Money& one, double arg) {
    Money temp;
```

```
    double tempN;
    tempN = (((one.rub * 100) + one.kop) / arg) / 100;
    temp. rub = (int)tempN;
    temp. kop = (tempN - (int)tempN) * 100;
    return temp;
}

Money operator*(const Money& one, double arg) {
    Money temp;
    double tempN;
    tempN = (((one.rub * 100) + one.kop) * arg);
    temp. rub = (int)(tempN / 100);
    temp. kop = (tempN / 100 - (int)(tempN / 100)) * 100;
    return temp;
}

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;
}
```

```
Money::~Money() {}
```

main.cpp:

```
#include <iostream>
#include "money.h"

int main(void) {

   Money a1;
   Money a2;
   float arg;

   std::cout << "First summ:" << std::endl;
   std::cin >> a1;
   std::cout << "Second summ:" << std::endl;
   std::cin >> a2;
   std::cout << "Number to div and multiply" << std::endl;
   std::cin >> arg;
   if(arg == 0){
      std::cout << "cannot be divided by zero" << std::endl;
      return 0;
   }
   std::cout << std::endl;

   std::cout << a1;

   std::cout << a2;

   std::cout << a1+a2;

   std::cout << a1 - a2;

   std::cout << a1 / a2 << std::endl;

   std::cout << a1 / arg;

   std::cout << a1 * arg << std::endl;

   if(a1 < a2) std::cout << "<:" << std::endl;
   else std::cout << "<: false" << std::endl;

   if(a1 > a2) std::cout << ">: true" << std::endl;
   else std::cout << ">: false" << std::endl;

   if(a1 <= a2) std::cout << "<=: true" << std::endl;
   else std::cout << "<=: false" << std::endl;

   if(a1 >= a2) std::cout << ">=: true" << std::endl;
   else std::cout << ">=: false" << std::endl;

   if(a1 == a2) std::cout << "==: true" << std::endl;
   else std::cout << "==: false" << std::endl;
```

```
    if(a1 != a2) std::cout << "!=: true" << std::endl;
    else std::cout << "!=: false" << std::endl;

    return 0;
}
```