Московский Авиационный Институт
(Национальный исследовательский Университет)

Факультет: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»

Лабораторная работа
по курсу «ООП»

Тема:
Простые классы.

Студент:	Коротков Д.П.
Группа:	М80-208Б-18
Преподаватель:	Журавлев А.А.
Вариант:	9
Оценка:	
Дата:	

Москва
2019

1. Код программы на языке C++:

Money.hpp:

```
#ifndef __MONEY__
#define __MONEY__
#include <iostream>


struct money {
    unsigned long long pound;
    unsigned char shilling;
    unsigned char pension;
    money();
    money(unsigned long long po, unsigned char sh, unsigned char pe);
    void m_scan(std::istream& is);
    void m_print(std::ostream& os) const;
    money operator+ (const money& rhs) const;
    money operator/ (const double a) const;
    double operator/ (const money& rhs) const;
    money operator* (const double a) const;
    bool operator> (const money& rhs) const;
    bool operator>= (const money& rhs) const;
    bool operator< (const money& rhs) const;
    bool operator<= (const money& rhs) const;
    bool operator== (const money& rhs) const;
    void m_unif();
private:
    unsigned long long m_all() const;
    int m_cmp(const money& a) const;
};

std::istream& operator>> (std::istream& is, money& m);
std::ostream& operator<< (std::ostream& os, const money& m);
money operator"" _po (unsigned long long po);
money operator"" _sh (unsigned long long sh);
money operator"" _pe (unsigned long long pe);


#endif
```

Money.cpp:

```
#include "Money.hpp"
#include <iostream>

money::money() {
    pound = 0;
```

```
    shilling = 0;
    pension = 0;
}


money::money(unsigned long long po, unsigned char sh, unsigned char pe) {
    pound = po;
    shilling = sh;
    pension = pe;
}


unsigned long long money::m_all() const {
    return this->pension + this->shilling * 12 + this-> pound * 12 * 20;
}


void money::m_print(std::ostream& os) const {
    unsigned char zer = 0;
    os << this->pound << ' ' << this->shilling - zer << ' ' << this->pension - zer << '\n';
}


void money::m_scan(std::istream& is) {
    int sh, pe;
    is >> this->pound >> sh >> pe;
    this->shilling = sh;
    this->pension = pe;
}


money operator"" _po (unsigned long long po) {
    money tmp {po, 0, 0};
    return tmp;
}


money operator"" _sh (unsigned long long sh) {
    money tmp {0, sh, 0};
    return tmp;
}


money operator"" _pe (unsigned long long pe) {
    money tmp {0, 0, pe};
```

```
    return tmp;
}


std::istream& operator>> (std::istream& is, money& m) {
    m.m_scan(is);
    return is;
}


std::ostream& operator<< (std::ostream& os, const money& m) {
    m.m_print(os);
    return os;
}


int money::m_cmp(const money &a) const {
    if ((*this).m_all() > a.m_all()) return 1;
    else if ((*this).m_all() == a.m_all()) return 0;
    else return -1;

}


money money::operator+ (const money &rhs) const{
    money res;
    res.pound = this->pound + rhs.pound;
    res.shilling = this->shilling + rhs.shilling;
    res.pension = this->pension + rhs.pension;
    res.pension = this->pension + rhs.pension;
    res.m_unif();
    return res;
}


money money::operator/ (const double a) const{
    money res;
    unsigned long long all = (*this).m_all() / a;
    res.pound = all / 240;
    all %= 240;
    res.shilling = all / 12;
    all %= 12;
    res.pension = all;
    return res;
}
```

```
money money::operator* (const double a) const{
    money res;
    unsigned long long all = (*this).m_all() * a;
    res.pound = all / 240;
    all %= 240;
    res.shilling = all / 12;
    all %= 12;
    res.pension = all;
    return res;
}


double money::operator/ (const money& rhs) const{
    return (*this).m_all() / rhs.m_all();
}


bool money::operator> (const money& rhs) const{
    return (*this).m_cmp(rhs) > 0;
}


bool money::operator>= (const money& rhs) const{
    return (*this).m_cmp(rhs) >= 0;
}


bool money::operator< (const money& rhs) const{
    return (*this).m_cmp(rhs) < 0;
}


bool money::operator<= (const money& rhs) const{
    return (*this).m_cmp(rhs) <= 0;
}


bool money::operator== (const money& rhs) const{
    return (*this).m_cmp(rhs) == 0;
}


void money::m_unif() {
```

```
    this->pound += this->shilling / 20;
    this->shilling = this->shilling % 20;
    this->shilling += this->pension / 12;
    this->pension = this->pension % 12;
    this->pound += this->shilling / 20;
    this->shilling = this->shilling % 20;
}
```

main.cpp:

```
#include "Money.hpp"
#include <iostream>


signed main() {
    money l;
    l.m_scan(std::cin);
    unsigned long long rpo;
    int rsh, rpe;
    double div, prod;
    money r;
    std::cin >> r;
    std::cin >> div >> prod;
    money t;
    t = 1_po + 2_sh + 3_pe;
    money res {};
    int comp;
    comp = l > r;
    std::cout << ">: " << comp << '\n';
    comp = l >= r;
    std::cout << ">=: " << comp << '\n';
    comp = l < r;
    std::cout << "<: " << comp << '\n';
    comp = l <= r;
    std::cout << "<=: " << comp << '\n';
    comp = l == r;
    std::cout << "==: " << comp << '\n';
    std::cout << "sum: ";
    res = l + r;
    std::cout << res;
    std::cout << "digital division: ";
    res = l / div;
    res.m_print(std::cout);
    std::cout << "digital product: ";
    res = l * prod;
```

```
    res.m_print(std::cout);
    std::cout << "money division: " << l / r << '\n';
}
```

CmakeLists.txt:

```
cmake_minimum_required(VERSION 3.2)

project(lab2)

add_executable(lab2
        main.cpp
        Money.cpp
)

set_property(TARGET lab2 PROPERTY CXX_STANDARD 11)
```

test.sh:

```
#!/usr/bin/env bash

executable=$1

for file in test_??.test
do
  $executable < $file > tmp
  if cmp tmp ${file%%.test}.ans
  then
    echo Test "$file": SUCCESS
  else
    echo Test "$file": FAIL
  fi
  rm tmp
done
```

2. Ссылка на репозиторий на GitHub.

https://github.com/KorotkovDenis/oop_exercise_01

3. Набор testcases.

test_01.test:

```
1 1 1
1 1 1
1 1
```

test_02.test:

```
10 12 10
```

10 10 10
2 0
test_03.test:
0 0 0
100 100 100
10 13
test_04.test:
3 14 15
92 65 35
89 79
test_05.test:
27 19 1
0 0 1
28 1.5

4. Результаты выполнения тестов.

test_01.ans:
>: 0
>=: 1
<: 0
<=: 1
==: 1
sum: 2 2 2
digital division: 1 1 1
digital product: 1 1 1
money division: 1
test_02.ans:
>: 1
>=: 1
<: 0
<=: 0
==: 0
sum: 20 22 20
digital division: 5 6 5
digital product: 0 0 0
money division: 1
test_03.ans:
>: 0
>=: 0
<: 1
<=: 1
==: 0
sum: 100 100 100
digital division: 0 0 0

digital product: 0 0 0
money division: 0
test_04.ans:
>: 0
>=: 0
<: 1
<=: 1
==: 0
sum: 95 79 50
digital division: 0 0 10
digital product: 297 4 9
money division: 0
test_05.ans:
>: 1
>=: 1
<: 0
<=: 0
==: 0
sum: 27 19 2
digital division: 0 19 1
digital product: 41 18 7
money division: 6709

5. Объяснение результатов работы программы.

1) При запуске скрипта с аргументом ./test.sh ../builds/lab1 объекты l, r и два дробных числа div, prod в основной программе получают данные из файлов test_??.test.
2) Объекты l и r сравниваются операторами сравнения.
3) Объекты l и r складываются с помощью оператора + класса money, и результат выводится в стандартный поток вывода с помощью метода m_print().
4) Объект l делится на число div , и результат выводится в стандартный поток вывода с помощью метода m_print().
5) Объект l умножается на число prod , и результат выводится в стандартный поток вывода с помощью метода m_print().
6) Объект l делится на r с помощью оператора . класса money и результат выводится в стандартный поток вывода с помощью функции m_print().

6. Вывод.

Выполняя данную лабораторную я получил опыт работы с перегрузкой операторов в C++. Создал класс, соответствующий варианту моего задания, реализовал для него арифметические операции сложения, умножения, деления, а также операции сравнения.