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_print(std::ostream& os) const;
    void m_scan(std::istream& is);
    bool m_equal(const money& a) const;
    void m_sum(const money& lhs, const money& rhs);
    void m_dig_div(const money& m, const double a);
    double m_div(const money& m) const;
    void m_dig_prod(const money& m, const double a);
    void m_unif();
private:
    unsigned long long all;
};

#endif
```

Money.cpp:

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

money::money() {
    pound = 0;
    shilling = 0;
    pension = 0;
    all = 0;
}


money::money(unsigned long long po, unsigned char sh, unsigned char pe) {
    pound = po;
    shilling = sh;
    pension = pe;
    all = pe + sh * 12 + po * 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;
    this->all = this->pension + this->shilling * 12 + this->pound * 12 * 20;
}


bool money::m_equal(const money &a) const {
     return this->pound == a.pound && this->shilling == a.shilling && this->pension
== a.pension;
}


void money::m_sum(const money &lhs, const money &rhs) {
    this->pound = lhs.pound + rhs.pound;
    this->shilling = lhs.shilling + rhs.shilling;
    this->pension = lhs.pension + rhs.pension;
    this->all = lhs.all + rhs.all;
}


void money::m_dig_div(const money &m, const double a) {
    this->pound = m.pound / a;
    this->shilling = m.shilling / a;
    this->pension = m.pension / a;
    this->all = this->pension + this->shilling * 12 + this->pound * 12 * 20;
}


void money::m_dig_prod(const money &m, const double a) {
    this->pound = m.pound * a;
    this->shilling = m.shilling * a;
    this->pension = m.pension * a;
    this->all = this->pension + this->shilling * 12 + this->pound * 12 * 20;
}
```

```
double money::m_div(const money &m) const {
    return this->all / m.all;
}


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;
    std::cin >> rpo >> rsh >> rpe;
    std::cin >> div >> prod;
    money r {rpo, int(rsh), int(rpe)};
    money res {};
    if (l.m_equal(r)) {
        std::cout << "equal: YES\n";
    } else {
        std::cout << "equal: NO\n";
    }
    std::cout << "sum: ";
    res.m_sum(l, r);
    res.m_print(std::cout);
    std::cout << "digital division: ";
    res.m_dig_div(l, div);
    res.m_print(std::cout);
    std::cout << "digital product: ";
    res.m_dig_prod(l, prod);
    res.m_print(std::cout);
    std::cout << "money division: " << l.m_div(r) << '\n';
```

```
}
```
CmakeLists.txt:
```
cmake_minimum_required(VERSION 3.2)

project(lab1)

add_executable(lab1
        main.cpp
        Money.cpp
)

set_property(TARGET lab1 PROPERTY CXX_STANDART 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:
equal: YES
sum: 2 2 2
digital division: 1 1 1
digital product: 1 1 1
money division: 1
test_02.ans:
equal: NO
sum: 20 22 20
digital division: 5 6 5
digital product: 0 0 0
money division: 1
test_03.ans:
equal: NO
sum: 100 100 100
digital division: 0 0 0
digital product: 0 0 0
money division: 0
test_04.ans:
equal: NO
sum: 95 79 50
digital division: 0 0 0
digital product: 237 82 161
money division: 0
test_05.ans:
equal: NO
sum: 27 19 2
digital division: 0 0 0
digital product: 40 28 1
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 методом m_equal().
3) Объекты l и r складываются с помощью метода m_sun() 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 зкщв , и результат выводится в стандартный поток вывода с помощью метода m_print().
6) Объект l делится на r с помощью метода m_div() класса money и результат выводится в стандартный поток вывода с помощью функции m_print().

6. Вывод.

Выполняя данную лабораторную я получил опыт работы с простыми классами, с системой сборки Cmake, с системой контроля версий GitHub, а также изучил основы работы с классами в C++. Создал класс, соответствующий варианту моего задания, реализовал для него арифметические операции сложения, умножения, деления, а также операцию сравнения.