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

ЛАБОРАТОРНАЯ РАБОТА №2

по курсу “Объектно-ориентированное программирование» 1 семестр,

2021/22 уч. год

Студент: *Колпакова Диана Саргаевна, группа М8О-208Б-20*
Преподаватель: *Дорохов Евгений Павлович*

Задание

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. Операции над объектами реализовать в виде перегрузки операторов. Реализовать пользовательский литерал для работы с константами объектов созданного класса.

Вариант 9:

Создать класс BritishMoney для работы с денежными суммами в старой британской системе. Сумма денег должна быть представлена тремя полями: типа unsigned long long для фунтов стерлингов, типа unsigned char – для шиллингов, unsigned char – для пенсов (пенни). Реализовать сложение сумм, вычитание, деление сумм, деление суммы на дробное число, умножение на дробное число и операции сравнения. 1 фунт = 20 шиллингов, 1 шиллинг = 12 пенни.

Описание программы

Исходный код лежит в 3 файлах:

1. main.cpp: часть программы, отвечающая за взаимодействие с пользователем через консоль. В ней происходит инициализация объектов и вызов функций работы с ними;
2. britishmoney.h: описание класса сумм британских денег BritishMoney;
3. britishmoney.cpp: реализация класса BritishMoney.

Также используется файл CMakeLists.txt с конфигурацией CMake для автоматизации сборки программы.

Дневник отладки

Проблем не было.

Вывод

В данной лабораторной работе продолжилось знакомство с ООП в языке

C++.

Была изучена и реализована перегрузка операторов, а также заданы пользовательские литералы – константы класса BritishMoney.

Я переопределила пользовательский класс BritishMoney, реализовав функции-операции над экземплярами класса в виде перегрузки операторов

Исходный код

britishmoney.h:

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



class BMoney {
    friend BMoney operator+(const BMoney &m1, const BMoney &m2); // друж ф-ям есть доступ к приват полям и ф-ям класса
    friend BMoney operator-(const BMoney &m1, const BMoney &m2);
    friend BMoney operator/(BMoney &m1, BMoney &m2);
    friend bool operator==(const BMoney &m1, const BMoney &m2);
    friend bool operator!=(const BMoney &m1, const BMoney &m2);
    friend bool operator>(const BMoney &m1, const BMoney &m2);
    friend bool operator<(const BMoney &m1, const BMoney &m2);
    friend bool operator>=(const BMoney &m1, const BMoney &m2);
    friend bool operator<=(const BMoney &m1, const BMoney &m2);
    friend BMoney PtoSum(unsigned long long tmp_p);

    public:
        BMoney(); // явный конструктор по умолчанию (все равно 0)
        BMoney(unsigned long long a, uint16_t b, uint16_t c); // конструктор с
```

```
параметрами, инициализация напрямую в коде
        BMoney(std::istream &is); // конструктор из istream
        BMoney operator/(double C);
        BMoney operator*(double C);
        void operator>>(std::ostream &os);
        ~BMoney(); // деструктор

        unsigned long long ToPenny() const;
        BMoney operator=(const BMoney &other);
        void Translate();
        void Print(std::ostream &os);
        bool Empty() const;

    private:
        unsigned long long ps;
        uint16_t sh;
        uint16_t p;

};

BMoney operator "" _ps(unsigned long long ps);
BMoney operator "" _s(unsigned long long sh);
BMoney operator "" _p(unsigned long long p);

#endif
```

britishmoney.cpp:

```
#include "britishmoney.h"
#include <cmath>
```

```
uint16_t ps_sh = 20;
uint16_t sh_p = 12;

BMoney::BMoney() {
    ps = 0;
    sh = 0;
    p = 0;
    std:: cout << "\t\t\t\t~virtual wallet created by default~" << std:: endl;
}

BMoney::BMoney(unsigned long long a, uint16_t b, uint16_t c) {
    if (ps < 0 || sh < 0 || p < 0) {
        std:: cout << "Parameters must be positive or zero integer numbers" <<
std:: endl;
    }
    else {
        ps = a;
        sh = b;
        p = c;
    }
    std:: cout << "\t\t\t\t~virtual wallet created according to parameters~" <<
std:: endl;
}

BMoney::BMoney(std::istream &is) {
    std:: cout << "Please enter your wallet data in order [pounds] [shillings]
[pennies]: " << std:: endl;
    is >> ps >> sh >> p;
    while (ps < 0 || sh < 0 || p < 0) {
        std:: cout << "Invalind input. Try again." << std:: endl;
        is >> ps >> sh >> p;
    }
    std:: cout << "\t\t\t\t~virtual wallet created via istream~" << std:: endl;
}


bool operator==(const BMoney &m1, const BMoney &m2) {
    if (m1.ps == m2.ps && m1.sh == m2.sh && m1.p == m2.p)
```

```
        return 1;
    return 0;
}

bool operator!=(const BMoney &m1, const BMoney &m2) {
    if (m1.ps != m2.ps || m1.sh != m2.sh || m1.p != m2.p)
        return 1;
    return 0;
    //return !Equal(m1, m2);
}

bool operator>(const BMoney &m1, const BMoney &m2) {
    unsigned long long tmp1 = m1.ToPenny();
    unsigned long long tmp2 = m2.ToPenny();
    if (tmp1 > tmp2)
        return 1;
    return 0;
}

bool operator<=(const BMoney &m1, const BMoney &m2) {
    return !(m1 > m2);
}

bool operator<(const BMoney &m1, const BMoney &m2) {
    return m2 > m1;
}

bool operator>=(const BMoney &m1, const BMoney &m2) {
    return !(m2 > m1);
}

bool BMoney::Empty() const {
    if (ps == 0 && sh == 0 && p == 0)
        return 1;
    return 0;
}
```

```
BMoney operator+(const BMoney& m1, const BMoney &m2) {
    BMoney res;
    res.p = (m1.p + m2.p) % sh_p;
    res.sh = (m1.sh + m2.sh + (m1.p + m2.p) / sh_p) % ps_sh;
    res.ps = m1.ps + m2.ps + (m1.sh + m2.sh + (m1.p + m2.p) / sh_p) / ps_sh;
    return res;
}

unsigned long long BMoney::ToPenny() const {
    unsigned long long res = ps * ps_sh * sh_p + sh * sh_p + p;
    //std:: cout << res << std:: endl;
    return res;
};

BMoney PtoSum(unsigned long long tmp_p) {
    BMoney res;
    res.ps = tmp_p / (ps_sh * sh_p);
    tmp_p %= (ps_sh * sh_p);
    res.sh = tmp_p / sh_p;
    res.p = tmp_p % sh_p;
    return res;
}

BMoney operator-(const BMoney &m1, const BMoney &m2) {
    if (m1 < m2) {
        std:: cout << "The operation could not be performed. The first sum is
less than the second." << std:: endl;
        return BMoney(); // возвращение нулевого кошелька
    }
    unsigned long long tmp = m1.ToPenny() - m2.ToPenny();
    return PtoSum(tmp);
}

BMoney operator/(BMoney &m1, BMoney &m2) {
    if (!m2.Empty()) {
        unsigned long long tmp = m1.ToPenny() / m2.ToPenny();
        return PtoSum(tmp);
    }
```

```
    std:: cout << "The operation could not be performed. The second sum equals
null." << std:: endl;
    return BMoney();
}

BMoney BMoney::operator/(double C) { // все функции класса (не friend)
обязательно должны иметь [назв-е класса]::
    if (C == 0) {
        std:: cout << "The operation could not be performed. The number equals
null." << std:: endl;
        return BMoney();
    }
    unsigned long long tmp = this->ToPenny() / C;
    return PtoSum(tmp);
}

BMoney BMoney::operator*(double C) {
    unsigned long long tmp = ToPenny() * C;
    return PtoSum(tmp);
}

void BMoney::Print(std::ostream &os) { // totally works
    os << ps << " pounds " << sh << " shillings " << p << " pennies " <<
std::endl;
}

BMoney BMoney::operator=(const BMoney &other)
{
    ps = other.ps;
    sh = other.sh;
    p = other.p;
    return *this;
}

BMoney operator "" _ps(unsigned long long ps)
{
    return BMoney(ps, 0, 0);
}
```

```
BMoney operator "" _s(unsigned long long sh)
{
    return BMoney(0, sh, 0);
}

BMoney operator "" _p(unsigned long long p)
{
    return BMoney(0, 0, p);
}

void BMoney::Translate() {
    if (p > sh_p || sh > ps_sh) {
        sh += p / sh_p;
        p %= sh_p;
        ps += sh / ps_sh;
        sh = (sh % ps_sh) + p / sh_p;
    }
}

BMoney::~BMoney() {
    std:: cout << "\t\t\t\t ~wallet has been deleted~" << std:: endl;
}
```

main.cpp:

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

int main(void) {

    double arg;

    BMoney a1(1, 2, 3);
    BMoney a2(std::cin);
    a2.Translate();
    std::cin >> arg;
    a1.Print(std::cout);
    a2.Print(std::cout);
    std::cout << "a1 < a2 = " << (a1 < a2) << std::endl;
```

```
    std::cout << "a1 > a2 = " << (a1 > a2) << std::endl;
    std::cout << "a1 <= a2 = " << (a1 <= a2) << std::endl;
    std::cout << "a1 >= a2 = " << (a1 >= a2) << std::endl;
    std::cout << "a1 == a2 = " << (a1 == a2) << std::endl;
    std::cout << "a1 != a2 = " << (a1 != a2) << std::endl;
    BMoney a3 = a1 + a2;
    std::cout << "a1 + a2 = " << std::endl;
    a3.Print(std::cout);
    BMoney a4 = a2 - 15_p;
    std::cout << "a2 - 15 pennies = " << std::endl;
    a4.Print(std::cout);
    BMoney a5 = a2 / a1;
    std::cout << "a2 / a1 = " << std::endl;
    a5.Print(std::cout);
    BMoney a6 = a1 * arg;
    std::cout << "a1 * C = " << std::endl;
    a6.Print(std::cout);
    BMoney a7 = a1 / arg;
    std::cout << "a1 / C = " << std::endl;
    a7.Print(std::cout);

    return 0;
}
```

CMakeLists.txt:

```
cmake_minimum_required(VERSION 3.10)
project(lab0.1)

set(CMAKE_CXX_STANDARD 17)

add_executable(lab0.1 britishmoney.h britishmoney.cpp main.cpp)
```