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

ЛАБОРАТОРНАЯ РАБОТА №01

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.

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 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.

В программе не поддерживаются отрицательные суммы, и поэтому в программу добавлены соответствующие проверки.

Дневник отладки

Проблем не было.

Вывод

В данной лабораторной работе я узнала как работать с базовыми понятиями ООП на примере языка C++: классы, объекты классов, конструкторы класса, поля и методы класса, модификаторы доступа (private, public) к ним, дружественные функции класса.

До этого был опыт работы с ООП только на примере Java и C#.

Я определила пользовательский класс BritishMoney и реализовала математические операции над его объектами, применив на практике базовые принципы ООП.

Исходный код

BritishMoney.h:

```
#pragma once
#include <iostream>

using namespace std;

class BritishMoney
{
private:
   unsigned long long pounds;
   unsigned char shillings;
   unsigned char pennies;

   static const unsigned char penniesPerShilling = 12;
   static const unsigned char shillingsPerPound = 20;

public:
   BritishMoney();
   BritishMoney(unsigned long long pounds, unsigned char shillings, unsigned
char pennies);

   friend BritishMoney BritishMoneyFromPennies(unsigned long long pennies);
   friend unsigned long long BritishMoneyToPennies(const BritishMoney& money);

   friend BritishMoney Add(const BritishMoney& money1, const BritishMoney&
money2);
   friend BritishMoney Subtract(const BritishMoney& money1, const BritishMoney&
money2);
   friend BritishMoney Multiply(const BritishMoney& money, const double
factor);
   friend BritishMoney Multiply(const double factor, const BritishMoney&
money);
   friend BritishMoney Divide(const BritishMoney& money, const double factor);
   friend double Divide(const BritishMoney& money1, const BritishMoney&
money2);
```

```
   friend bool Equal(const BritishMoney& money1, const BritishMoney& money2);
   friend bool NotEqual(const BritishMoney& money1, const BritishMoney&
money2);
   friend bool Greater(const BritishMoney& money1, const BritishMoney& money2);
   friend bool GreaterOrEqual(const BritishMoney& money1, const BritishMoney&
money2);
   friend bool Less(const BritishMoney& money1, const BritishMoney& money2);
   friend bool LessOrEqual(const BritishMoney& money1, const BritishMoney&
money2);

   friend void WriteToStream(ostream& stream, const BritishMoney& money);
   friend void ReadFromStream(istream& stream, BritishMoney& money);
};
```

BritishMoney.cpp:

```
#include <stdexcept>
#include "BritishMoney.h"

BritishMoney::BritishMoney()
{
   this->pounds = 0;
   this->shillings = 0;
   this->pennies = 0;
}

BritishMoney::BritishMoney(unsigned long long pounds, unsigned char shillings,
unsigned char pennies)
{
   if (shillings >= shillingsPerPound || pennies >= penniesPerShilling)
      throw std::out_of_range("BritishMoney constructor: invalid values of
shillings or pennies");
   this->pounds = pounds;
   this->shillings = shillings;
   this->pennies = pennies;
}

BritishMoney BritishMoneyFromPennies(unsigned long long pennies)
{
   unsigned char pennies2 = pennies % BritishMoney::penniesPerShilling;
   unsigned long long shillings = pennies / BritishMoney::penniesPerShilling;
   unsigned char shillings2 = shillings % BritishMoney::shillingsPerPound;
   unsigned long long pounds = shillings / BritishMoney::shillingsPerPound;

   return BritishMoney(pounds, shillings2, pennies2);
}

unsigned long long BritishMoneyToPennies(const BritishMoney& money)
{
   unsigned long long pennies = (money.pounds * BritishMoney::shillingsPerPound
+ money.shillings) * BritishMoney::penniesPerShilling + money.pennies;
   return pennies;
}

BritishMoney Add(const BritishMoney& money1, const BritishMoney& money2)
{
```

```
   unsigned long long pennies1 = BritishMoneyToPennies(money1);
   unsigned long long pennies2 = BritishMoneyToPennies(money2);
   unsigned long long pennies3 = pennies1 + pennies2;
   return BritishMoneyFromPennies(pennies3);
}

BritishMoney Subtract(const BritishMoney& money1, const BritishMoney& money2)
{
   unsigned long long pennies1 = BritishMoneyToPennies(money1);
   unsigned long long pennies2 = BritishMoneyToPennies(money2);
   if (pennies1 < pennies2)
      throw std::out_of_range("BritishMoney Subtract: money1 less than
money2");
   unsigned long long pennies3 = pennies1 - pennies2;
   return BritishMoneyFromPennies(pennies3);
}

BritishMoney Multiply(const BritishMoney& money, const double factor)
{
   if (factor < 0.0)
      throw std::out_of_range("BritishMoney Multiply: second parameter less
than zero");
   unsigned long long pennies = BritishMoneyToPennies(money);
   unsigned long long pennies2 = (unsigned long long)(pennies * factor);
   return BritishMoneyFromPennies(pennies2);
}

BritishMoney Multiply(const double factor, const BritishMoney& money)
{
   if (factor < 0.0)
      throw std::out_of_range("BritishMoney Multiply: first parameter less than
zero");
   return Multiply(money, factor);
}

BritishMoney Divide(const BritishMoney& money, const double factor)
{
   if (factor == 0.0)
      throw std::out_of_range("BritishMoney Divide: second parameter is zero or
less");

   unsigned long long pennies = BritishMoneyToPennies(money);
   unsigned long long pennies2 = (unsigned long long)(pennies / factor);
   return BritishMoneyFromPennies(pennies2);
}

double Divide(const BritishMoney& money1, const BritishMoney& money2)
{
   unsigned long long pennies1 = BritishMoneyToPennies(money1);
   unsigned long long pennies2 = BritishMoneyToPennies(money2);
   if (pennies2 == 0)
      throw std::out_of_range("BritishMoney Divide: second parameter is zero");
   double factor = (double)pennies1 / (double)pennies2;
   return factor;
}

bool Equal(const BritishMoney& money1, const BritishMoney& money2)
```

```
{
   unsigned long long pennies1 = BritishMoneyToPennies(money1);
   unsigned long long pennies2 = BritishMoneyToPennies(money2);
   bool result = pennies1 == pennies2;
   return result;
}

bool NotEqual(const BritishMoney& money1, const BritishMoney& money2)
{
   unsigned long long pennies1 = BritishMoneyToPennies(money1);
   unsigned long long pennies2 = BritishMoneyToPennies(money2);
   bool result = pennies1 != pennies2;
   return result;
}

bool Greater(const BritishMoney& money1, const BritishMoney& money2)
{
   unsigned long long pennies1 = BritishMoneyToPennies(money1);
   unsigned long long pennies2 = BritishMoneyToPennies(money2);
   bool result = pennies1 > pennies2;
   return result;
}

bool GreaterOrEqual(const BritishMoney& money1, const BritishMoney& money2)
{
   unsigned long long pennies1 = BritishMoneyToPennies(money1);
   unsigned long long pennies2 = BritishMoneyToPennies(money2);
   bool result = pennies1 >= pennies2;
   return result;
}

bool Less(const BritishMoney& money1, const BritishMoney& money2)
{
   unsigned long long pennies1 = BritishMoneyToPennies(money1);
   unsigned long long pennies2 = BritishMoneyToPennies(money2);
   bool result = pennies1 < pennies2;
   return result;
}

bool LessOrEqual(const BritishMoney& money1, const BritishMoney& money2)
{
   unsigned long long pennies1 = BritishMoneyToPennies(money1);
   unsigned long long pennies2 = BritishMoneyToPennies(money2);
   bool result = pennies1 <= pennies2;
   return result;
}

void WriteToStream(ostream& stream, const BritishMoney& money)
{
   unsigned long long pounds = money.pounds;
   unsigned int shillings = money.shillings;
   unsigned int pennies = money.pennies;

   stream << "(" << pounds << "," << shillings << "," << pennies << ")";
}

void ReadFromStream(istream& stream, BritishMoney& money)
```

```
{
   unsigned long long pounds;
   unsigned int shillings;
   unsigned int pennies;
   char leftBracket, rightBracket, comma1, comma2;

   stream >> leftBracket >> pounds >> comma1 >> shillings >> comma2 >> pennies
>> rightBracket;
   money.pounds = pounds;
   money.shillings = shillings;
   money.pennies = pennies;
}
```

main.cpp:

```
// OOP, Lab 01, variant 9, Diana Kolpakova
// British Money

#include <iostream>
#include "BritishMoney.h"

using namespace std;

int main()
{
   cout << "oop_exercise_01 (c) Diana Kolpakova" << endl;
   cout << "British money format is (pounds,shillings,pennies)." << endl;

   BritishMoney money1;
   BritishMoney money2;
   double factor;

   cout << "Enter money1:";
   ReadFromStream(cin, money1);
   cout << "Enter money2:";
   ReadFromStream(cin, money2);
   cout << "Enter factor:";
   cin >> factor;

   cout << "Results:" << endl;
   cout << "money1 = "; WriteToStream(cout, money1); cout << endl;
   cout << "money2 = "; WriteToStream(cout, money2); cout << endl;
   cout << "factor = " << factor << endl;

   cout << "money1+money2 = "; WriteToStream(cout, Add(money1, money2)); cout
<< endl;
   cout << "money1-money2 = ";  WriteToStream(cout, Subtract(money1, money2));
cout << endl;
   cout << "money1*factor = ";  WriteToStream(cout, Multiply(money1, factor));
cout << endl;
   cout << "factor*money2 = ";  WriteToStream(cout, Multiply(factor, money2));
cout << endl;
   cout << "money1/money2 = " << Divide(money1, money2) << endl;
   cout << "money1/factor = ";  WriteToStream(cout, Divide(money1, factor));
cout << endl;
```

```
    cout << "money1==money2 = " << Equal(money1, money2) << endl;
    cout << "money1!=money2 = " << NotEqual(money1, money2) << endl;
    cout << "money1>money2 = " << Greater(money1, money2) << endl;
    cout << "money1>=money2 = " << GreaterOrEqual(money1, money2) << endl;
    cout << "money1<money2 = " << Less(money1, money2) << endl;
    cout << "money1<=money2 = " << LessOrEqual(money1, money2) << endl;
}
```

CMakeLists.txt:

```
cmake_minimum_required(VERSION 3.21)
project(oop_exercise_01)

set(CMAKE_CXX_STANDARD 14)

include_directories(.)

add_executable(oop_exercise_01
        BritishMoney.cpp
        BritishMoney.h
        main.cpp)
```