**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 1**

Тема: Простые классы на языке С++

Студент: Эссаулов Андрей

Группа: 80-207

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2019

1. Постановка задачи

Создать класс BritishMoney для работы с денежными суммами в старой британской система. Сумма

денег должна быть представлено 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.

1. Описание программы

Программа содержит класс BritishMoney, который имеет поля unsigned long long для фунтов и два unsigned char для шиллингов и пенни.

Также в классе определен конструктор для начальной инициализации переменных и метод WriteMoney, который выводит в консоль текущее значение всех полей класса.

Все действия над классом BritishMoney реализованы в виде внешних функций, которые принимают 2 экземпляра класса(либо 1 экземпляр и 1 число) и возвращают результирующий экземпляр. В операциях суммы и вычитания реализована проверка на выход за границы размерности переменной. В операции деления реализована проверка деления на ноль. Операции сравнения переводят поля фунтов и шиллингов в пенни и сравнивают основываясь на сумме трех полей.

Работа с программой осуществляется следующим образом:

Программа дважды запрашивает ввод 3 значений (фунты, шиллинги, пенни), после чего запрашивает ввод той операции, которую надо провести над введенными данными (коды операций выводятся в консоль). В итоге программа выводит результат операции и завершается.

1. Набор testcases

Тестам подаются значения 2 раза по 3 значения, которые интерпретируются как фунты, шиллинги и пенни. И в конце тесту подается код команды, которую он должен выполнить.

Коды операций:

1) + 2) - 3) / 4) / (float) 5) \* (float) 6) > 7) < 8) ==

**1-ый тест:**

2 6 17

4 8 20

1

**2-ой тест:**

0 2 10

0 1 12

2

**3-ий тест:**

8 12 7

0 1 2

3

1. Результаты выполнения тестов.

1) Funts - 6, shilings - 14, pennys - 37

2) Funts - 0, shilings - 0, pennys - 10

3) Funts - 8, shilings - 12, pennys - 3

1. Листинг программы

// Лабораторная №1 Эссаулов Андрей M80-207Б-18

// Создать класс BritishMoney для работы с денежными суммами в старой британской система. Сумма

// денег должна быть представлено 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.

#include <iostream>

class BritishMoney

{

public:

BritishMoney(unsigned long long f, unsigned char s, unsigned char p )

{

funts = f;

shilings = s;

pennys = p;

}

~BritishMoney()

{

}

void WriteMoney()

{

std::cout << "Funts - " << funts << ", shilings - " << (unsigned int)shilings << ", pennys - " << (unsigned int)pennys << std::endl;

}

unsigned long long funts;

unsigned char shilings;

unsigned char pennys;

};

BritishMoney summ (const BritishMoney a, const BritishMoney b)

{

BritishMoney result(0, 0, 0);

if (a.pennys + b.pennys > UINT8\_MAX)

{

unsigned int buf = a.pennys + b.pennys;

result.pennys += buf % 12;

result.shilings += buf / 12;

}

else {

result.pennys += a.pennys + b.pennys;

}

if (a.shilings + b.shilings > UINT8\_MAX)

{

unsigned int buf = a.shilings + b.shilings;

result.shilings += buf % 20;

result.funts += buf / 20;

}

else{

result.shilings += a.shilings + b.shilings;

}

result.funts += a.funts + b.funts;

return result;

}

BritishMoney subtraction (const BritishMoney a, const BritishMoney b)

{

BritishMoney result(0, 0 ,0);

int p, s, f = 0;

p = a.pennys - b.pennys;

s = a.shilings - b.shilings;

f = a.funts - b.funts;

if (p < 0){

while (p < 0){

if (s > 0){

s--;

p += 12;

}else if (f > 0){

f--;

s += 19;

p += 12;

}else{

break;

}

}

}

if (s < 0)

{

while (s < 0)

{

if (f > 0)

{

f--;

s += 12;

}

else if (p > 12)

{

p -= 12;

s++;

}

else

{

break;

}

}

}

if(f < 0)

{

while (f < 0)

{

if (s > 20)

{

s -= 20;

f++;

}

else if (p > (12 \* 20))

{

p -= (12\*20);

f++;

}

else

{

break;

}

}

}

result.pennys = p<0?0:p;

result.shilings = s<0?0:s;

result.funts = f<0?0:f;

return result;

}

BritishMoney divide (const BritishMoney a, const BritishMoney b)

{

BritishMoney result(0, 0, 0);

if (b.pennys != 0) result.pennys = a.pennys / b.pennys; else result.pennys = a.pennys;

if (b.shilings != 0) result.shilings = a.shilings / b.shilings; else result.shilings = a.shilings;

if (b.funts != 0) result.funts = a.funts / b.funts; else result.funts = a.funts;

return result;

}

BritishMoney divide (const BritishMoney a, const float b)

{

BritishMoney result(0,0,0);

if (b != 0)

{

result.pennys = a.pennys / b;

result.shilings = a.shilings / b;

result.funts = a.funts / b;

}

else

{

result.pennys = a.pennys;

result.shilings = a.shilings;

result.funts = a.funts;

}

return result;

}

BritishMoney multiplication (const BritishMoney a, const float b)

{

BritishMoney result(0,0,0);

result.pennys = a.pennys \* b;

result.shilings = a.shilings \* b;

result.funts = a.funts \* b;

return result;

}

bool equal (const BritishMoney a, const BritishMoney b)

{

BritishMoney result(0,0,0);

unsigned long long first, second;

first = a.pennys + a.shilings \* 12 + a.funts \* 12 \* 20;

second = b.pennys + b.shilings \* 12 + b.funts \* 12 \* 20;

return (first==second?true:false);

}

bool morethan (const BritishMoney a, const BritishMoney b)

{

BritishMoney result(0,0,0);

unsigned long long first, second;

first = a.pennys + a.shilings \* 12 + a.funts \* 12 \* 20;

second = b.pennys + b.shilings \* 12 + b.funts \* 12 \* 20;

return (first>second?true:false);

}

bool lessthan (const BritishMoney a, const BritishMoney b)

{

BritishMoney result(0,0,0);

unsigned long long first, second;

first = a.pennys + a.shilings \* 12 + a.funts \* 12 \* 20;

second = b.pennys + b.shilings \* 12 + b.funts \* 12 \* 20;

return (first<second?true:false);

}

bool moreequalthan (const BritishMoney a, const BritishMoney b)

{

BritishMoney result(0,0,0);

unsigned long long first, second;

first = a.pennys + a.shilings \* 12 + a.funts \* 12 \* 20;

second = b.pennys + b.shilings \* 12 + b.funts \* 12 \* 20;

return (first>=second?true:false);

}

bool lessequalthan (const BritishMoney a, const BritishMoney b)

{

BritishMoney result(0,0,0);

unsigned long long first, second;

first = a.pennys + a.shilings \* 12 + a.funts \* 12 \* 20;

second = b.pennys + b.shilings \* 12 + b.funts \* 12 \* 20;

return (first<=second?true:false);

}

int main()

{

BritishMoney first(0, 0 , 0);

BritishMoney second(0, 0, 0);

int bufs, bufp;

std::cout << "Enter first bill\nfunts\t shilings\t pennys" << std::endl;

std::cin >> first.funts >> bufs >> bufp;

first.shilings = bufs;

first.pennys = bufp;

std::cout << "--------------------------------------------" << std::endl << std::endl;

std::cout << "Enter second bill\nfunts\t shilings\t pennys" << std::endl;

std::cin >> second.funts >> bufs >> bufp;

second.shilings = bufs;

second.pennys = bufp;

std::cout << "--------------------------------------------" << std::endl << std::endl;

std::cout << "Enter operation code: " << std::endl << "1) +" << std::endl << "2) -" << std::endl << "3) /"

<< std::endl << "4) / (float)" << std::endl << "5) \* (float)" << std::endl << "6) >"<< std::endl << "7) <"

<< std::endl << "8) ==" << std::endl;

int switch\_var;

std::cin >> switch\_var;

switch (switch\_var)

{

case 1:

summ(first, second).WriteMoney();

break;

case 2:

subtraction(first, second ).WriteMoney();

break;

case 3:

divide(first, second).WriteMoney();

break;

case 4:

std::cout << "Enter divider" << std::endl;

int div;

std::cin >> div;

divide(first, div).WriteMoney();

break;

case 5:

std::cout << "Enter multiplier" << std::endl;

int mul;

std::cin >> mul;

multiplication(first, mul).WriteMoney();

break;

case 6:

if (morethan(first, second)) std::cout << "first > second - true" << std::endl;

else std::cout << "first > second - false" << std::endl;

break;

case 7:

if (lessthan(first, second)) std::cout << "first < second - true" << std::endl;

else std::cout << "first < second - false" << std::endl;

break;

case 8:

if (equal(first, second)) std::cout << "first == second - true" << std::endl;

else std::cout << "first == second - false" << std::endl;

break;

default:

std::cout << "Uncorrect enter" << std::endl;

break;

}

return 1;

}

[Код на github](https://github.com/PromZona/oop_exercise_01)

1. Выводы

Объектно-ориентированное программирование позволяет предоставлять пользовательские типы данных в виде объектов, что может упростить понимание программы.

Данная программа может быть улучшена с точки зрения функционала, если мы вместо функций будем использовать перегрузку операторов. Кроме того в качестве удобства использования можно зациклить пользовательский мод для возможности множественного обращения.

СПИСОК ЛИТЕРАТУРЫ

1. Руководство по написанию кода на C++ [Электронный ресурс]. URL:<https://metanit.com/cpp/tutorial/>

Дата обращения: 10.09.2019

1. Документация по C++ [Электронный ресурс]. URL: <https://docs.microsoft.com/ru-ru/cpp>

Дата обращения 12.09.2019