**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: Камеш Михаил

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 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.

1. Описание программы

В данной программе представлено решение проблемы выше следующим образом : данные, вводимые с клавиатуры, в связи с ограничением заданных нам типов данных (char) вводятся изначально в отдельную (int) переменную, а уже затем присваиваются соответствующим элементам класса.

Из-за ограничения (char) в 255 символов существуют соответствующие ограничения ввода в программу на количество шиллингов и пенни. Также существует ограничение деления на ноль.

Метод, реализуемой в этой задаче, состоит в том, что данные переводятся в единую валюту - пенни в другой переменной, а уже затем с ними производятся необходимые технические операции.

Всего в программе используются 9 переменных :

k : для визуального оформления программы через меню

a, b, c : переменные, для хранения данных при вводе с клавиатуры

funts, shillings, pennies - для хранения, как по условию, сумм внутри нашего класса BritishMoney

d, e, - для непосредственного перевода сумм в пенни, сравнения между собой, а также обратных операций.

1. Набор testcases

Тестовые данные данной программы представлены в виде примитивного рисованного меню, в котором можно выбирать осуществляемую команду :

0. Выход из программы

1. Добавить новую сумму
2. Вычесть новую сумму
3. Умножить на число
4. Разделить на число
5. Разделить на другую сумму
6. Сравнить с другой суммой
7. Обнулить сумму

В качестве примера, можно:

1. Поделить на ноль, что выдаст соответствующую ошибку
2. Умножить на ноль, сумма станет нулем
3. Умножить на единицу, сумма не изменится
4. Ввести число, больше 255, что выдаст соответствующую ошибку
5. Сравнить сумму с равной себе
6. Производить базовые операции, чтобы убедиться в правильной работе программы
7. Результаты выполнения тестов.

1 funt 2 shillings 3 pennies + 1 funt 2 shillings 3 pennies = 2 funts 4 shillings 4 pennies

2 funts 2 shillings 2 pennies / 2 funts 2 shillings 2 pennies = 1

2 funts 3 shillings 6 pennies - 2 funts 2 shillings 8 pennies = 0 funts 0 shillings 10 pennies

5 funts 4 shillings 2 pennies \* 0 = 0 funts 0 shillings 0 pennies

5 funts 4 shillings 2 pennies \* 1 = 5 funts 4 shillings 2 pennies

etc.

1. Листинг программы

/\*Камеш М.А. М80-207Б-18

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. \*/

#include <iostream>

#include <cmath>

using namespace std;

class BritishMoney

{

private:

double a,b,c;

double d,e;

unsigned long long funts;

unsigned char shillings;

unsigned char pennies;

public:

void readBritishMoney(void);

void printBritishMoney(void);

void calAddition(void);

void calDivision(void);

void Initialize(void);

void ConvertToPennies(void);

void ConvertToPennies2(void);

void ConvertToBasicSum(void);

void DivisionOperation(void);

void calcSum(void);

void calDivisionByNumber(void);

void MultOperation(void);

void calMultByNumber(void);

void calDifference(void);

void calcDiff(void);

void divideBySum(void);

void Compare(void);

};

void BritishMoney::Initialize(void)

{

funts = 0;

shillings = 0;

pennies = 0;

}

void BritishMoney::readBritishMoney(void)

{

cout << " Enter the sum" << endl;

cout<<"Enter amount of funts: ";

cin>>a;

cout<<"Enter amount of shillings: ";

cin>>b;

cout<<"Enter amount of pennies: ";

cin>>c;

if (b > 255 || c > 255)

{

cout << "Error, shillings and pennies can't be above 255, try again" << endl;

readBritishMoney();

}

}

void BritishMoney::calcSum(void)

{

ConvertToPennies();

ConvertToPennies2();

d += e;

ConvertToBasicSum();

}

void BritishMoney::calcDiff(void)

{

ConvertToPennies();

ConvertToPennies2();

if (d < e)

{

d = e - d;

ConvertToBasicSum();

cout << "Negative ";

}

if (d >= e)

{

d -= e;

ConvertToBasicSum();

}

}

void BritishMoney::printBritishMoney(void)

{

a = funts;

b = shillings;

c = pennies;

cout << "funts = " << a << ", shillings = " << b << ", pennies = " << c << endl;

}

void BritishMoney::calAddition(void)

{

readBritishMoney();

calcSum();

printBritishMoney();

}

void BritishMoney::calDifference(void)

{

readBritishMoney();

calcDiff();

printBritishMoney();

}

void BritishMoney::ConvertToPennies(void)

{

d = (int)pennies + (int)shillings \* 12 + funts \* 12 \* 20;

}

void BritishMoney::ConvertToPennies2(void)

{

e = c + b \* 12 + a \* 12 \* 20;

}

void BritishMoney::MultOperation(void)

{

d \*= a;

round(d);

ConvertToBasicSum();

}

void BritishMoney::DivisionOperation(void)

{

d /= a;

round(d);

ConvertToBasicSum();

}

void BritishMoney::calDivisionByNumber(void)

{

cout << "Enter number to divide sum by: ";

cin >> a;

while (a <= 0)

{

cout << "Can't divide by 0 or negative number" << endl;

cin >> a;

}

ConvertToPennies();

DivisionOperation();

printBritishMoney();

}

void BritishMoney::calMultByNumber(void)

{

cout << "Enter number to multiply sum by: ";

cin >> a;

ConvertToPennies();

MultOperation();

printBritishMoney();

}

void BritishMoney::ConvertToBasicSum(void)

{

Initialize();

while(d >= 12\*20)

{

d -= 12\*20;

funts++;

}

while (d >= 12)

{

d -= 12;

shillings ++;

}

pennies += d;

}

void BritishMoney::Compare(void)

{

readBritishMoney();

ConvertToPennies();

ConvertToPennies2();

if (d > e)

{

cout << "First sum is larger than second one" << endl;

}

else if (d == e)

{

cout << "First sum and second one are equal" << endl;

}

else if (d < e)

{

cout << "First sum is smaller than second one" << endl;

}

}

void BritishMoney::divideBySum(void)

{

readBritishMoney();

ConvertToPennies();

ConvertToPennies2();

cout << "First sum divided by second sum equals" << d/e << endl;

}

int main()

{

cout << " This is a program to work with British Money : "<< endl;

cout << " It can add, substract, divide, multiply and compare sums." << endl;

BritishMoney num;

num.Initialize();

num.readBritishMoney();

num.calcSum();

int k = 1;

while (k)

{

printf("===================================="

"\n|| MENU ||"

"\n||--------------------------------||"

"\n|| 0 - exit ||"

"\n|| 1 - Add to another sum ||"

"\n|| 2 - Find a difference ||"

"\n|| 3 - Multiply by number ||"

"\n|| 4 - Divide by number ||"

"\n|| 5 - Divide by another sum ||"

"\n|| 6 - Compare with another sum ||"

"\n|| 7 - Reset the sum ||"

"\n===================================="

"\n ==>");

scanf("%d", &k);

if (k == 1)

{

num.calAddition();

}

if (k == 2)

{

num.calDifference();

}

if (k == 3)

{

num.calMultByNumber();

}

if (k == 4)

{

num.calDivisionByNumber();

}

if (k == 5)

{

num.divideBySum();

}

if (k == 6)

{

num.Compare();

}

if (k == 7)

{

num.Initialize();

num.readBritishMoney();

num.calcSum();

}

}

return 0;

}

1. Вывод

В ходе выполнения работы были изучены базовые принципы работы с объектами, создана программа, изменяющая значения разной элементов класса с разными типами данных.

К сожалению, из-за лимитаций условия, некоторые вещи были реализованы некорректно, например работа с отрицательными числами. Во всех 3 классовых переменных (long long, unsigned char, unsigned char) отсутствуют отрицательные числа. Эту проблему можно было бы обойти, вводя дополнительные переменные для хранение знаков или чисел, но, по моему мнению, это выходит за рамки задания данной программы.

Умножение и деление для экономии памяти производится на дробные числа типа double, а пенни можно округлить командой “round” для наглядности и более удобного использования.

1. Используемая литература

1. Справочник по языку С++ [Электронный ресурс]. URL:

<https://stackoverflow.com/questions/55296835/console-application-menu-with-classes-c> (дата обращения: 19.09.2019).

2. Справочник по утилите CMake [Электронный ресурс]. URL:

<https://stackoverflow.com/questions/2481269/how-to-make-a-simple-c-makefile> (дата обращения: 12.09.2019).

3. Справочник по языку C++ [Электронный ресурс]. URL:

<https://www.geeksforgeeks.org/object-oriented-programming-in-cpp/>

(дата обращения: 12.09.2019).

4. Справочник по языку C++ [Электронный ресурс]. URL:

<https://www.geeksforgeeks.org/decision-making-c-c-else-nested-else/>

(дата обращения: 12.09.2019).

1. Ссылка на репозиторий GitHub

<https://github.com/kameshmihail/oop_exercise_01>