**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 2**

Тема: Перегрузка операторов в 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 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.

Операции сложения, вычитания, умножения, деления, сравнения (на равенство, больше и меньше) должны быть выполнены в виде перегрузки операторов.

Необходимо реализовать пользовательский литерал для работы с константами типа **BritishMoney**.

1. Описание программы

Проблемы напоминает предыдущую задачу со следующими дополнительными ограничениями : все операции должны быть перегружены, должен быть введен пользовательский литерал для работы с суммами.

Для решения поставленной нам проблемы были имплементированы следующие изменения структуры программы по сравнению с предыдущей:

1. Введены 3 пользовательских литерала для работы с суммами : \_fun, \_sh, \_pen для соответственно фунтов, шиллингов и пенни.

2. Все операции над суммами были перегружены под разные типы данных для более удобной работы с суммами в “теле” программы.

3. В предыдущей программе операции были реализованы как операции между 6 переменными внутри одной классовой переменной, тут же используются 3 классовые переменные с 3 переменными внутри для хранения денежных сумм.

В остальном сам принцип программы остался неизменным, валюта переводится в пенни для следующих операций и дальнейшего перевода обратно, однако перевод в пенни осуществляется уже автоматически через литералы.

1. Набор testcases
2. 1\_fun 4\_sh 6\_pen + 0\_fun 0.5\_sh 4\_pen
3. 180\_fun 412.7\_sh 761.3\_pen + 832\_fun 123\_sh -125.4\_pen
4. 5\_fun 4\_sh 7\_pen - 3\_fun 6\_sh 9\_pen
5. 245.2\_fun 186.2\_sh 367.3\_pen - 393.2\_fun 123.4\_sh 252.8\_pen
6. 6\_fun 7\_sh 11\_pen \* 0
7. 5.4\_fun 3.6\_sh 1.3\_pen \* -23.7
8. 5\_fun 5\_sh 5\_pen / 1\_fun 1\_sh 1\_pen
9. 120\_fun 5.4\_sh 7\_pen / 73\_fun 13\_sh 4\_pen
10. 4\_fun 4\_sh 8\_pen / 2
11. 542\_fun 340\_sh 23\_pen / 41.6
12. 5\_fun 6\_sh 7\_pen > 3\_fun 30\_sh 15\_pen
13. 126\_fun 7\_sh 5\_pen > 124\_fun 73.4\_sh 7\_pen
14. Результаты выполнения тестов.
15. 1 funts 5 shillings 4 pennies
16. 1041 funts 8 shillings 8 pennies
17. 1 funts 17 shillings 10 pennies
18. -144 funts 7 shillings 7 pennies
19. 0 funts 0 shillings 0 pennies
20. -132 funts 7 shillings 5 pennies
21. 5
22. 1.63302
23. 2 funts 2 shillings 4 pennies
24. 13 funts 8 shillings 9 pennies
25. Sum1 > Sum2
26. Sum1 < Sum2
27. Листинг программы

|  |  |
| --- | --- |
|  |  |
|  | \*/ Камеш М.А. группа М80-207Б-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.  Операции сложения, вычитания, умножения, деления, сравнения (на равенство, больше и меньше) должны быть выполнены в виде перегрузки операторов.  Необходимо реализовать пользовательский литерал для работы с константами типа BritishMoney. \*/ |
|  |  |
|  |  |
|  |  |
|  | #include<iostream> |
|  | using namespace std; |
|  |  |
|  | class BritishMoney |
|  | { |
|  | private: |
|  |  |
|  | public: |
|  | long double funts = 0, shillings = 0, pennies = 0; |
|  | BritishMoney(long long r = 0, unsigned char i = 0, unsigned char k = 0) |
|  | { |
|  | funts = r; |
|  | shillings = i; |
|  | pennies = k; |
|  | } |
|  | void print() |
|  | { |
|  | cout << funts << " funts " << shillings << " shillings " << pennies << " pennies " << endl; |
|  | } |
|  |  |
|  | friend BritishMoney operator + (BritishMoney &, BritishMoney &); |
|  | friend BritishMoney operator - (BritishMoney &, BritishMoney &); |
|  | friend BritishMoney operator \* (BritishMoney &, long double &); |
|  | friend long double operator / (BritishMoney &, BritishMoney &); |
|  | friend BritishMoney operator / (BritishMoney &, long double &); |
|  | friend char operator > (BritishMoney &, BritishMoney &); |
|  | }; |
|  |  |
|  | long double operator"" \_fun(long double x ) |
|  | { |
|  | return x \* 240; |
|  | } |
|  |  |
|  | long double operator"" \_sh( long double x ) |
|  | { |
|  | return x \* 12; |
|  | } |
|  |  |
|  | long double operator"" \_pen( long double x ) |
|  | { |
|  | return x; |
|  | } |
|  |  |
|  | BritishMoney operator + (BritishMoney &c1, BritishMoney &c2) |
|  | { |
|  | c1.pennies = (c1.pennies + c2.pennies + c1.funts + c2.funts + c1.shillings + c2.shillings); |
|  | c1.funts = 0; |
|  | c1.shillings = 0; |
|  | if (c1.pennies > 0) |
|  | { |
|  | while(c1.pennies >= 12\*20) |
|  | { |
|  | c1.pennies -= 12\*20; |
|  | c1.funts++; |
|  | } |
|  | while (c1.pennies >= 12) |
|  | { |
|  | c1.pennies -= 12; |
|  | c1.shillings ++; |
|  | } |
|  | } |
|  | else if (c1.pennies < 0) |
|  | { |
|  | c1.pennies\*=(-1); |
|  | while(c1.pennies >= 12\*20) |
|  | { |
|  | c1.pennies -= 12\*20; |
|  | c1.funts++; |
|  | } |
|  | while (c1.pennies >= 12) |
|  | { |
|  | c1.pennies -= 12; |
|  | c1.shillings ++; |
|  | } |
|  | c1.funts\*=(-1); |
|  | } |
|  | return BritishMoney(c1.funts, c1.shillings, c1.pennies); |
|  | } |
|  |  |
|  |  |
|  | BritishMoney operator - (BritishMoney &c1, BritishMoney &c2) |
|  | { |
|  | c1.pennies = (c1.funts + c1.shillings + c1.pennies) - (c2.funts + c2.shillings + c2.pennies); |
|  | c1.funts = 0; |
|  | c1.shillings = 0; |
|  | if (c1.pennies > 0) |
|  | { |
|  | while(c1.pennies >= 12\*20) |
|  | { |
|  | c1.pennies -= 12\*20; |
|  | c1.funts++; |
|  | } |
|  | while (c1.pennies >= 12) |
|  | { |
|  | c1.pennies -= 12; |
|  | c1.shillings ++; |
|  | } |
|  | } |
|  | else if (c1.pennies < 0) |
|  | { |
|  | c1.pennies\*=(-1); |
|  | while(c1.pennies >= 12\*20) |
|  | { |
|  | c1.pennies -= 12\*20; |
|  | c1.funts++; |
|  | } |
|  | while (c1.pennies >= 12) |
|  | { |
|  | c1.pennies -= 12; |
|  | c1.shillings ++; |
|  | } |
|  | c1.funts\*=(-1); |
|  | } |
|  | return BritishMoney(c1.funts, c1.shillings, c1.pennies); |
|  | } |
|  |  |
|  | BritishMoney operator \* (BritishMoney &c1, long double &temp) |
|  | { |
|  | c1.pennies = (c1.funts+c1.shillings+c1.pennies) \* temp; |
|  | c1.funts = 0; |
|  | c1.shillings = 0; |
|  | if (c1.pennies > 0) |
|  | { |
|  | while(c1.pennies >= 12\*20) |
|  | { |
|  | c1.pennies -= 12\*20; |
|  | c1.funts++; |
|  | } |
|  | while (c1.pennies >= 12) |
|  | { |
|  | c1.pennies -= 12; |
|  | c1.shillings ++; |
|  | } |
|  | } |
|  | else if (c1.pennies < 0) |
|  | { |
|  | c1.pennies\*=(-1); |
|  | while(c1.pennies >= 12\*20) |
|  | { |
|  | c1.pennies -= 12\*20; |
|  | c1.funts++; |
|  | } |
|  | while (c1.pennies >= 12) |
|  | { |
|  | c1.pennies -= 12; |
|  | c1.shillings ++; |
|  | } |
|  | c1.funts\*=(-1); |
|  | } |
|  | return BritishMoney(c1.funts, c1.shillings, c1.pennies); |
|  | } |
|  |  |
|  |  |
|  | BritishMoney operator / (BritishMoney &c1, long double &temp) |
|  | { |
|  | c1.pennies = (c1.funts+c1.shillings+c1.pennies)/ temp; |
|  | c1.funts = 0; |
|  | c1.shillings = 0; |
|  | if (c1.pennies > 0) |
|  | { |
|  | while(c1.pennies >= 12\*20) |
|  | { |
|  | c1.pennies -= 12\*20; |
|  | c1.funts++; |
|  | } |
|  | while (c1.pennies >= 12) |
|  | { |
|  | c1.pennies -= 12; |
|  | c1.shillings ++; |
|  | } |
|  | } |
|  | else if (c1.pennies < 0) |
|  | { |
|  | c1.pennies\*=(-1); |
|  | while(c1.pennies >= 12\*20) |
|  | { |
|  | c1.pennies -= 12\*20; |
|  | c1.funts++; |
|  | } |
|  | while (c1.pennies >= 12) |
|  | { |
|  | c1.pennies -= 12; |
|  | c1.shillings ++; |
|  | } |
|  | c1.funts\*=(-1); |
|  | } |
|  | return BritishMoney(c1.funts, c1.shillings, c1.pennies); |
|  | } |
|  |  |
|  | long double operator / (BritishMoney &c1, BritishMoney &c2) |
|  | { |
|  | return (c1.funts+c1.shillings+c1.pennies)/(c2.funts+c2.shillings+c2.pennies); |
|  | } |
|  |  |
|  | char operator > (BritishMoney &c1, BritishMoney &c2) |
|  | { |
|  | if ((c1.funts+c1.shillings+c1.pennies) - (c2.funts+c2.shillings+c2.pennies) > 0) |
|  | { |
|  | return '>'; |
|  | } |
|  | else if ((c1.funts+c1.shillings+c1.pennies) - (c2.funts+c2.shillings+c2.pennies) < 0) |
|  | { |
|  | return '<'; |
|  | } |
|  | else if ((c1.funts+c1.shillings+c1.pennies) - (c2.funts+c2.shillings+c2.pennies) == 0) |
|  | { |
|  | return '='; |
|  | } |
|  | } |
|  |  |
|  |  |
|  | int main() |
|  | { |
|  | long double temp = 3; |
|  | char compare; |
|  | BritishMoney c1, c2, c3; |
|  | cout << "This program does arithmetic operation with British currencies : funts, shillings and pounds. It adds, substracts, multiplies, devides and compares sums and sums with numbers" << endl; |
|  | cout << "Addition" << endl; |
|  | c1.funts = 1.0\_fun; |
|  | c1.shillings = 4.0\_sh; |
|  | c1.pennies = 6.0\_pen; |
|  | c2.funts = 0.0\_fun; |
|  | c2.shillings = 0.5\_sh; |
|  | c2.pennies = 4.0\_pen; |
|  | c3 = c1 + c2; |
|  | c3.print(); |
|  | c1.funts = 180.0\_fun; |
|  | c1.shillings = 412.7\_sh; |
|  | c1.pennies = 761.3\_pen; |
|  | c2.funts = 832.0\_fun; |
|  | c2.shillings = 123.0\_sh; |
|  | c2.pennies = -125.4\_pen; |
|  | c3 = c1 + c2; |
|  | c3.print(); |
|  | cout << "Substraction" << endl; |
|  | c1.funts = 5.0\_fun; |
|  | c1.shillings = 4.0\_sh; |
|  | c1.pennies = 7.0\_pen; |
|  | c2.funts = 3.0\_fun; |
|  | c2.shillings = 6.0\_sh; |
|  | c2.pennies = 9.0\_pen; |
|  | c3 = c1 - c2; |
|  | c3.print(); |
|  | c1.funts = 245.2\_fun; |
|  | c1.shillings = 186.2\_sh; |
|  | c1.pennies = 367.3\_pen; |
|  | c2.funts = 393.2\_fun; |
|  | c2.shillings = 123.4\_sh; |
|  | c2.pennies = 252.8\_pen; |
|  | c3 = c1 - c2; |
|  | c3.print(); |
|  | cout << "Multiplication by number" << endl; |
|  | c1.funts = 6.0\_fun; |
|  | c1.shillings = 7.0\_sh; |
|  | c1.pennies = 11.0\_pen; |
|  | temp = 0; |
|  | c3 = c1 \* temp; |
|  | c3.print(); |
|  | c1.funts = 5.4\_fun; |
|  | c1.shillings = 3.6\_sh; |
|  | c1.pennies = 1.3\_pen; |
|  | temp = -23.7; |
|  | c3 = c1 \* temp; |
|  | c3.print(); |
|  | cout << "Division by sum" << endl; |
|  | c1.funts = 5.0\_fun; |
|  | c1.shillings = 5.0\_sh; |
|  | c1.pennies = 5.0\_pen; |
|  | c2.funts = 1.0\_fun; |
|  | c2.shillings = 1.0\_sh; |
|  | c2.pennies = 1.0\_pen; |
|  | temp = c1 / c2; |
|  | cout << temp << endl; |
|  | c1.funts = 120.0\_fun; |
|  | c1.shillings = 5.4\_sh; |
|  | c1.pennies = 7.0\_pen; |
|  | c2.funts = 73.0\_fun; |
|  | c2.shillings = 13.0\_sh; |
|  | c2.pennies = 4.0\_pen; |
|  | temp = c1 / c2; |
|  | cout << temp << endl; |
|  | cout << "Division by number" << endl; |
|  | c1.funts = 4.0\_fun; |
|  | c1.shillings = 4.0\_sh; |
|  | c1.pennies = 8.0\_pen; |
|  | temp = 2; |
|  | c3 = c1 / temp; |
|  | c3.print(); |
|  | c1.funts = 542.0\_fun; |
|  | c1.shillings = 340.0\_sh; |
|  | c1.pennies = 23.0\_pen; |
|  | temp = 41.6; |
|  | c3 = c1 / temp; |
|  | c3.print(); |
|  | cout << "Comparison" << endl; |
|  | c1.funts = 5.0\_fun; |
|  | c1.shillings = 6.0\_sh; |
|  | c1.pennies = 7.0\_pen; |
|  | c2.funts = 3.0\_fun; |
|  | c2.shillings = 30.0\_sh; |
|  | c2.pennies = 15.0\_pen; |
|  | compare = c1 > c2; |
|  | cout << "Sum1 " << compare << " Sum2" << endl; |
|  | c1.funts = 126.0\_fun; |
|  | c1.shillings = 7.0\_sh; |
|  | c1.pennies = 5.0\_pen; |
|  | c2.funts = 124.0\_fun; |
|  | c2.shillings = 73.4\_sh; |
|  | c2.pennies = 7.0\_pen; |
|  | compare = c1 > c2; |
|  | cout << "Sum1 " << compare << " Sum2" << endl; |
|  | return 0; |
|  | } |

1. Вывод

В результате провередия работы было лучше изучена классовая структура языка с++. Была переделана работа с указанными требованиями по заданию. Были изучены такие функции, как литералы - user defined literals (udl) и перегрузка операторов.

В данной программе также появились некоторые неудобства, тут их было 2. Первая - это проблема ввода переменных пользовательского типа с клавиатуры. Для этого после ввода приходится приводить типи данных переменных и искусственно умножать на требуемую константу, от чего теряется смысл литералов. Вторая - трудность создания дополнительных функций для вычислений внутри оператора.

1. Литература

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

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

(дата обращения: 28.09.2019).

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

<https://www.geeksforgeeks.org/operator-overloading-c/>

(дата обращения: 28.09.2019).

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

<https://en.cppreference.com/w/cpp/language/user_literal>

(дата обращения: 30.09.2019).

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

<https://www.geeksforgeeks.org/user-defined-literals-cpp/>

(дата обращения: 28.09.2019).

1. Ссылка на github

<https://github.com/kameshmihail/oop_exercise_02>