**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: Ильминский Никита

Группа: 80-207

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2019

1. Постановка задачи

В данной лабораторной работе мне необходимо выполнить перегрузку операторов сложения, вычитания, умножения, деления, сравнения (на равенство, больше и меньше) для раннее реализованного класса *Money* для работы с денежными суммами. Также необходимо реализовать пользовательский литерал для работы с константами типа *Money*.

1. Описание программы

В файле программы *main.cpp* для ранее описанного класса *Money* реализованы функции перегрузки операторов:

friend Money operator+ (Money m1, Money m2);

friend Money operator- (Money m1, Money m2);

friend Money operator\* (Money m1, double mult);

friend double operator/ (Money m1, Money m2);

friend bool operator< (Money m1, Money m2);

friend bool operator== (Money m1, Money m2);

friend bool operator!= (Money m1, Money m2);

friend bool operator> (Money m1, Money m2);

а также три пользовательских литерала:

Money operator"" \_r(unsigned long long rub);

Money operator"" \_k(unsigned long long kop);

Money operator"" \_rk(long double rub\_kop).

В основной функции main при помощи реализованных пользовательских литералов производится ввод 4 наборов тестов (по два объекта класса Money в каждом), для каждого из которых при помощи перегруженных операторов по очереди производится сложение, вычитание и деление. Далее пользователю предлагается ввести значение десятичного множителя для выполнения умножения для каждой денежной суммы и сразу выводится результат для каждого произведения. И, в завершение, для каждого набора тестов программа выводит результаты сравнения на больше, меньше, равенство и неравенство.

1. Набор testcases
2. 2988770\_r, 23.45\_rk;
3. 99\_k, 687\_k;
4. 2\_r + 93\_k, 0\_r + 10\_k;
5. 0\_r + 45879\_k, 0\_k;
6. Результаты выполнения тестов

cash1 is 2988770 rub, 0 kop

cash2 is 23 rub, 45 kop

cash3 is 0 rub, 99 kop

cash4 is 6 rub, 87 kop

cash5 is 2 rub, 93 kop

cash6 is 0 rub, 10 kop

cash7 is 458 rub, 79 kop

cash8 is 0 rub, 0 kop

The results of operations:

Addition:

cash1 + cash2 is 2988793 rub, 45 kop

cash3 + cash4 is 7 rub, 86 kop

cash5 + cash6 is 3 rub, 3 kop

cash7 + cash8 is 458 rub, 79 kop

Substraction:

cash1 - cash2 is 2988746 rub, 55 kop

cash3 - cash4 is -5 rub, 88 kop

cash5 - cash6 is 2 rub, 83 kop

cash7 - cash8 is 458 rub, 79 kop

Division:

cash1 div cash2 is 127453

cash3 div cash4 is 0.144105

cash5 div cash6 is 29.3

Can't divide: cash2 is 0! Aborting...

Multiplication:

Enter the desired multiplicator value:

.43

cash1 mult 0.43 is 1285171 rub, 10 kop

cash2 mult 0.43 is 10 rub, 8 kop

cash3 mult 0.43 is 0 rub, 42 kop

cash4 mult 0.43 is 2 rub, 95 kop

cash5 mult 0.43 is 1 rub, 25 kop

cash6 mult 0.43 is 0 rub, 4 kop

cash7 mult 0.43 is 197 rub, 27 kop

cash8 mult 0.43 is 0 rub, 0 kop

Comparision:

cash1 < cash2?: no

cash3 = cash4?: no

cash5 != cash6?: yes

cash7 > cash8?: yes

1. Листинг программы

//Ильминский Никита М80-207Б-18

/\*Создать класс Money для работы с денежными суммами. Сумма денег должна быть представлено двумя

полями: типа unsigned long long для рублей и типа unsigned char – для копеек. Дробная часть (копейки) при

выводе на экран должна быть отделена от целой части запятой. Реализовать сложение сумм, вычитание,

деление сумм, деление суммы на дробное число, умножение на дробное число и операции сравнения.\*/

/\*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 Money.\*/

#include <iostream>

#include <cmath>

using namespace std;

class Money {

unsigned long long rub;

unsigned char kop;

public:

Money(int r, int p) {

setRub(r);

setKop(p);

}

void setRub(int r) {

rub = r;

}

void setKop(int p) {

kop = p;

}

int getRub() {

return rub;

}

int getKop() {

return kop;

}

int conv() {

return rub \* 100 + kop;

}

friend Money operator+ (Money m1, Money m2);

friend Money operator- (Money m1, Money m2);

friend Money operator\* (Money m1, double mult);

friend double operator/ (Money m1, Money m2);

friend bool operator< (Money m1, Money m2);

friend bool operator== (Money m1, Money m2);

friend bool operator!= (Money m1, Money m2);

friend bool operator> (Money m1, Money m2);

double fraction\_div(double div) {

return conv() / div / 100;

}

};

//OPERATOR OVERLOAD

Money operator+ (Money m1, Money m2) {

Money result(0, 0);

int p = m1.getKop() + m2.getKop();

int r = m1.getRub() + m2.getRub();

result.setRub(r + p / 100);

result.setKop(p % 100);

return result;

}

Money operator- (Money m1, Money m2) {

Money result(0, 0);

int p, r;

if (m1 < m2) {

p = m2.getKop() - m1.getKop();

r = m2.getRub() - m1.getRub();

} else {

p = m1.getKop() - m2.getKop();

r = m1.getRub() - m2.getRub();

}

if (p < 0) {

p = 100 + p;

r--;

}

result.setRub(r);

result.setKop(p);

return result;

}

Money operator\* (Money m, double mult) {

Money result(0, 0);

result.setKop(fmod(abs(m.conv() \* mult), 100));

result.setRub(abs(m.conv() \* mult) / 100);

return result;

}

double operator/ (Money m1, Money m2) {

return (double)m1.conv() / (double)m2.conv();

}

bool operator< (Money m1, Money m2) {

return m1.conv() < m2.conv();

}

bool operator> (Money m1, Money m2) {

return m1.conv() > m2.conv();

}

bool operator== (Money m1, Money m2) {

return m1.conv() == m2.conv();

}

bool operator!= (Money m1, Money m2) {

return !(m1 == m2);

}

//UDL

Money operator"" \_r(unsigned long long rub) {

return Money(rub, 0);

}

Money operator"" \_k(unsigned long long kop) {

return Money(kop / 100, kop % 100);

}

Money operator"" \_rk(long double rub\_kop) {

int sum = rub\_kop \* 100;

return Money(sum / 100, sum % 100);

}

int main()

{

Money cash1 = 2988770\_r;

Money cash2 = 23.45\_rk;

Money cash3 = 99\_k;

Money cash4 = 687\_k;

Money cash5 = 2\_r + 93\_k;

Money cash6 = 0\_r + 10\_k;

Money cash7 = 0\_r + 45879\_k;

Money cash8 = 0\_k;

cout << "cash1 is " << cash1.getRub() << " rub, " << cash1.getKop() << " kop\n";

cout << "cash2 is " << cash2.getRub() << " rub, " << cash2.getKop() << " kop\n";

cout << "cash3 is " << cash3.getRub() << " rub, " << cash3.getKop() << " kop\n";

cout << "cash4 is " << cash4.getRub() << " rub, " << cash4.getKop() << " kop\n";

cout << "cash5 is " << cash5.getRub() << " rub, " << cash5.getKop() << " kop\n";

cout << "cash6 is " << cash6.getRub() << " rub, " << cash6.getKop() << " kop\n";

cout << "cash7 is " << cash7.getRub() << " rub, " << cash7.getKop() << " kop\n";

cout << "cash8 is " << cash8.getRub() << " rub, " << cash8.getKop() << " kop\n\n";

cout << "The results of operations:\n";

cout << "Addition:\n";

Money res = cash1 + cash2;

cout << "cash1 + cash2 is " << res.getRub() << " rub, " << res.getKop() << " kop\n";

res = cash3 + cash4;

cout << "cash3 + cash4 is " << res.getRub() << " rub, " << res.getKop() << " kop\n";

res = cash5 + cash6;

cout << "cash5 + cash6 is " << res.getRub() << " rub, " << res.getKop() << " kop\n";

res = cash7 + cash8;

cout << "cash7 + cash8 is " << res.getRub() << " rub, " << res.getKop() << " kop\n\n";

cout << "Substraction:\n";

res = cash1 - cash2;

if (cash1 < cash2) {

cout << "cash1 - cash2 is -" << res.getRub() << " rub, " << res.getKop() << " kop\n";

} else {

cout << "cash1 - cash2 is " << res.getRub() << " rub, " << res.getKop() << " kop\n";

}

res = cash3 - cash4;

if (cash3 < cash4) {

cout << "cash3 - cash4 is -" << res.getRub() << " rub, " << res.getKop() << " kop\n";

} else {

cout << "cash3 - cash4 is " << res.getRub() << " rub, " << res.getKop() << " kop\n";

}

res = cash5 - cash6;

if (cash5 < cash6) {

cout << "cash5 - cash6 is -" << res.getRub() << " rub, " << res.getKop() << " kop\n";

} else {

cout << "cash5 - cash6 is " << res.getRub() << " rub, " << res.getKop() << " kop\n";

}

res = cash7 - cash8;

if (cash7 < cash8) {

cout << "cash7 - cash8 is -" << res.getRub() << " rub, " << res.getKop() << " kop\n";

} else {

cout << "cash7 - cash8 is " << res.getRub() << " rub, " << res.getKop() << " kop\n\n";

}

cout << "Division:\n";

if (cash2.conv() == 0) {

cout << "Can't divide: cash2 is 0! Aborting...\n\n";

} else {

cout << "cash1 div cash2 is " << cash1 / cash2 << endl << endl;

}

if (cash4.conv() == 0) {

cout << "Can't divide: cash2 is 0! Aborting...\n\n";

} else {

cout << "cash3 div cash4 is " << cash3 / cash4 << endl << endl;

}

if (cash6.conv() == 0) {

cout << "Can't divide: cash2 is 0! Aborting...\n\n";

} else {

cout << "cash5 div cash6 is " << cash5 / cash6 << endl << endl;

}

if (cash8.conv() == 0) {

cout << "Can't divide: cash2 is 0! Aborting...\n\n";

} else {

cout << "cash7 div cash8 is " << cash7 / cash8 << endl << endl;

}

cout << "Multiplication:\n";

double mult;

cout << "Enter the desired multiplicator value:\n";

cin >> mult;

res = cash1 \* mult;

if (mult < 0) {

cout << "cash1 mult " << mult << " is -" << res.getRub() << " rub, " << res.getKop() << " kop\n";

} else {

cout << "cash1 mult " << mult << " is " << res.getRub() << " rub, " << res.getKop() << " kop\n";

}

res = cash2 \* mult;

if (mult < 0) {

cout << "cash2 mult " << mult << " is -" << res.getRub() << " rub, " << res.getKop() << " kop\n";

} else {

cout << "cash2 mult " << mult << " is " << res.getRub() << " rub, " << res.getKop() << " kop\n";

}

res = cash3 \* mult;

if (mult < 0) {

cout << "cash3 mult " << mult << " is -" << res.getRub() << " rub, " << res.getKop() << " kop\n";

} else {

cout << "cash3 mult " << mult << " is " << res.getRub() << " rub, " << res.getKop() << " kop\n";

}

res = cash4 \* mult;

if (mult < 0) {

cout << "cash4 mult " << mult << " is -" << res.getRub() << " rub, " << res.getKop() << " kop\n";

} else {

cout << "cash4 mult " << mult << " is " << res.getRub() << " rub, " << res.getKop() << " kop\n";

}

res = cash5 \* mult;

if (mult < 0) {

cout << "cash5 mult " << mult << " is -" << res.getRub() << " rub, " << res.getKop() << " kop\n";

} else {

cout << "cash5 mult " << mult << " is " << res.getRub() << " rub, " << res.getKop() << " kop\n";

}

res = cash6 \* mult;

if (mult < 0) {

cout << "cash6 mult " << mult << " is -" << res.getRub() << " rub, " << res.getKop() << " kop\n";

} else {

cout << "cash6 mult " << mult << " is " << res.getRub() << " rub, " << res.getKop() << " kop\n";

}

res = cash7 \* mult;

if (mult < 0) {

cout << "cash7 mult " << mult << " is -" << res.getRub() << " rub, " << res.getKop() << " kop\n";

} else {

cout << "cash7 mult " << mult << " is " << res.getRub() << " rub, " << res.getKop() << " kop\n";

}

res = cash8 \* mult;

if (mult < 0) {

cout << "cash8 mult " << mult << " is -" << res.getRub() << " rub, " << res.getKop() << " kop\n\n";

} else {

cout << "cash8 mult " << mult << " is " << res.getRub() << " rub, " << res.getKop() << " kop\n\n";

}

cout << "Comparision:\n";

cout << "cash1 < cash2?: " << (cash1 < cash2 ? "yes" : "no") << endl;

cout << "cash3 = cash4?: " << (cash1 == cash2 ? "yes" : "no") << endl;

cout << "cash5 != cash6?: " << (cash1 != cash2 ? "yes" : "no") << endl;

cout << "cash7 > cash8?: " << (cash1 > cash2 ? "yes" : "no") << endl << endl;

return 0;

}

1. Вывод

В результате выполнения данной лабораторной работы я получил навыки работы с перегрузкой операторов и пользовательскими литералами. Процесс работы оказался довольно прост и понятен, так как стандарт языка C++11 представляет удобные инструменты для реализации.

Пользовательские литералы повышают читаемость кода, помогают более понятно определять классы, а позволяют обнаруживать ошибки во время компиляции, а не во время выполнения (как при вызове конструктора).

Перегруженные операторы также упрощают код и процесс его написания, делая легче работу с объектами класса. Однако при этом нужно помнить правила, так как существуют определенные ограничения:

1. почти любой существующий оператор в C++ может быть перегружен; **исключениями являются**: **?:** (тернарный оператор), **sizeof**, **::** (оператор видимости), **.** (оператор выбора члена), **.\*** (указатель как оператор выбора члена).

2. **возможно перегрузить только существующие операторы.**

3.  **по крайней мере один из операндов перегруженного оператора должен быть пользовательского типа данных.**

4.  **изначальное количество операндов, поддерживаемых оператором, изменить невозможно.**

5. **по умолчанию все операторы сохраняют свой**[**приоритет и ассоциативность**](https://ravesli.com/urok-38-prioritet-operatsij-assotsiativnost/).

СПИСОК ЛИТЕРАТУРЫ

1. Уроки C++ [Электронный ресурс].

URL: [https://ravesli.com](https://ravesli.com/) (дата обращения: 30.09.2019).

1. User Defined Literals [Электронный ресурс].

URL: <https://www.modernescpp.com/index.php/user-defined-literals> (дата обращения: 30.09.2019).