**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: Ильминский Никита

Группа: 80-207

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2019

1. Постановка задачи

В данной лабораторной работе мне необходимо реализовать класс *Money* для работы с денежными суммами. Сумма денег должна быть представлена 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. Для данного класса также необходимо реализовать функции сложения сумм, вычитания, деления сумм, деления суммы на дробное число, умножения на дробное число и операции сравнения.

1. Описание программы

В файле программы *main.cpp* реализован класс *Money*: для доступа *private* предоставлены два поля - *unsigned long long* переменная *rub* для хранения количества рублей данной суммы и *unsigned char* переменная *pen* для храненияколичества копеек; по модификатору доступа *public* располагаются все необходимые функции для работы с классом - конструктор (для создания объекта класса и одновременного заполнения его полей), функции *setRub(int)*, *setPen(int)*, *getRub()*, *setRub()* (для присваивания и вывода значений полей из main), функция *conv()*, переводящая значения денежных сумм в копейки (для дальнейшего применения в функциях деления, умножения и операциях сравнивания), и, наконец, предложенные в задании функции сложения *Money add(Money)*, вычитания *Money sub(Money)*, деления сумм *double div(Money)*, деления суммы на дробное число *double fraction\_div(double)*, умножения на дробное число *Money fraction\_mult(double mult)* и операции сравнения типа *bool*: *less(Money)*, *eq(Money)* и *greater(Money)*.

Наряду с этим имеется главная функция *int main()*, обеспечивающая пользовательский интерфейс для создания объектов класса *Money* и работы с ними: пользователю предлагается ввести значения двух денежных сумм, а также значения дробного делителя для первой суммы и дробного множителя для второй; программа работает с введенными значениями и выводит на экран значения данных денежных сумм через запятую (рубли, копейки), их сумму, разность, результат деления друг на друга, результат операций сравнения и результаты деления первой суммы и умножения второй суммы на данные дробные числа (в случае, когда частное равно 0, деление не производится и предусмотрен вывод сообщения об ошибке).

1. Набор testcases
2. 0 | 0 | 3 | 78 | .098 | 0.198
3. 0 | 36 | 0 | 40 | 0.6 | 1.5
4. 0 | 0 | 0 | 0 | 0 | 45.789
5. 500 | 35 | 200 | 75 | 0.1 | 0
6. Результаты выполнения тестов

**1)**

Enter rub1

0

Enter pen1

0

Enter rub2

3

Enter pen2

78

cash1 is 0 rub, 0 pen

cash2 is 3 rub, 78 pen

cash1 + cash2 is 3 rub, 78 pen

cash1 - cash2 is -3 rub, 78 pen

cash1 div cash2 is 0

cash1 < cash2?: 1

cash1 = cash2?: 0

cash1 > cash2?: 0

Enter cash1 fractional divider

.098

Enter cash2 fractional multiplier

0.198

cash1 div 0.098 is 0

cash2 mult 0.198 is 0 rub, 74 pen

**2)**

Enter rub1

0

Enter pen1

36

Enter rub2

0

Enter pen2

40

cash1 is 0 rub, 36 pen

cash2 is 0 rub, 40 pen

cash1 + cash2 is 0 rub, 76 pen

cash1 - cash2 is -0 rub, 4 pen

cash1 div cash2 is 0.9

cash1 < cash2?: 1

cash1 = cash2?: 0

cash1 > cash2?: 0

Enter cash1 fractional divider

0.6

Enter cash2 fractional multiplier

1.5

cash1 div 0.6 is 0.6

cash2 mult 1.5 is 0 rub, 60 pen

**3)**

Enter rub1

0

Enter pen1

0

Enter rub2

0

Enter pen2

0

cash1 is 0 rub, 0 pen

cash2 is 0 rub, 0 pen

cash1 + cash2 is 0 rub, 0 pen

cash1 - cash2 is 0 rub, 0 pen

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

cash1 < cash2?: 0

cash1 = cash2?: 1

cash1 > cash2?: 0

Enter cash1 fractional divider

0

Enter cash2 fractional multiplier

45.789

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

cash2 mult 45.789 is 0 rub, 0 pen

**4)**

Enter rub1

500

Enter pen1

35

Enter rub2

200

Enter pen2

75

cash1 is 500 rub, 35 pen

cash2 is 200 rub, 75 pen

cash1 + cash2 is 701 rub, 10 pen

cash1 - cash2 is 299 rub, 60 pen

cash1 div cash2 is 2.4924

cash1 < cash2?: 0

cash1 = cash2?: 0

cash1 > cash2?: 1

Enter cash1 fractional divider

0.1

Enter cash2 fractional multiplier

0

cash1 div 0.1 is 5003.5

cash2 mult 0 is 0 rub, 0 pen

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.\*/

#include <iostream>

#include <cmath>

using namespace std;

class Money {

unsigned long long rub;

unsigned char pen;

public:

Money(int r, int p) {

setRub(r);

setPen(p);

}

void setRub(int r) {

rub = r;

}

void setPen(int p) {

pen = p;

}

int getRub() {

return rub;

}

int getPen() {

return pen;

}

int conv() {

return rub \* 100 + pen;

}

Money add(Money m) {

Money result(0, 0);

int p = pen + m.getPen();

int r = rub + m.getRub();

result.setRub(r + p / 100);

result.setPen(p % 100);

return result;

}

Money sub(Money m) {

Money result(0, 0);

int p, r;

if (less(m)) {

p = m.getPen() - pen;

r = m.getRub() - rub;

} else {

p = pen - m.getPen();

r = rub - m.getRub();

}

if (p < 0) {

p = 100 + p;

r--;

}

result.setRub(r);

result.setPen(p);

return result;

}

double div(Money m) {

return (double)conv() / (double)m.conv();

}

double fraction\_div(double div) {

return conv() / div / 100;

}

Money fraction\_mult(double mult) {

Money result(0, 0);

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

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

return result;

}

bool less(Money m) {

return conv() < m.conv();

}

bool eq(Money m) {

return conv() == m.conv();

}

bool greater(Money m) {

return conv() > m.conv();

}

};

int main()

{

int rub, pen;

cout << "Enter rub1\n";

cin >> rub;

cout << "Enter pen1\n";

cin >> pen;

Money cash1(rub, pen);

cout << "Enter rub2\n";

cin >> rub;

cout << "Enter pen2\n";

cin >> pen;

Money cash2(rub, pen);

cout << "cash1 is " << cash1.getRub() << " rub, " << cash1.getPen() << " pen\n";

cout << "cash2 is " << cash2.getRub() << " rub, " << cash2.getPen() << " pen\n";

Money res = cash1.add(cash2);

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

res = cash1.sub(cash2);

if (cash1.less(cash2)) {

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

} else {

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

}

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

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

} else {

cout << "cash1 div cash2 is " << cash1.div(cash2) << endl << endl;

}

cout << "cash1 < cash2?: " << cash1.less(cash2) << endl;

cout << "cash1 = cash2?: " << cash1.eq(cash2) << endl;

cout << "cash1 > cash2?: " << cash1.greater(cash2) << endl << endl;

double div, mult;

cout << "Enter cash1 fractional divider\n";

cin >> div;

cout << "Enter cash2 fractional multiplier\n";

cin >> mult;

if (div == 0) {

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

} else {

cout << "cash1 div " << div << " is " << cash1.fraction\_div(div) << endl;

}

res = cash2.fraction\_mult(mult);

if (mult < 0) {

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

} else {

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

}

return 0;

}

1. Вывод

В результате выполнения данной лабораторной работы я изучил базовые понятия объектно-ориентированного программирования - классы и объекты, а также получил первый опыт работы с ними на языке C++. По моему мнению, классы имеют смысл с точки зрения обобщения свойств и действий для одного типа объекта, абстракции от деталей реализации и использования инкапсуляции для защиты информации от несанкционированного доступа. Однако у ооп есть и отрицательные черты - довольно непростая разработка, больший размер кода по сравнению с процедурами, сама парадигма ооп, которая многими специалистами признается как не имеющая смысла.

СПИСОК ЛИТЕРАТУРЫ

1. C++ Programming Tutorials Playlist - thenewboston [Электронный ресурс].  
   URL: <https://www.youtube.com/playlist?list=PLAE85DE8440AA6B83>  
   (дата обращения: 15.09.2019).