Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«**Пермский национальный исследовательский политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ**

Дисциплина: «Основы алгоритмизации и программирования»

Тема : "Классы и объекты. Использование конструкторов."

Выполнила работу

Студентка группы РИС-22-1Б

Верхоланцева Е. С.

Проверила

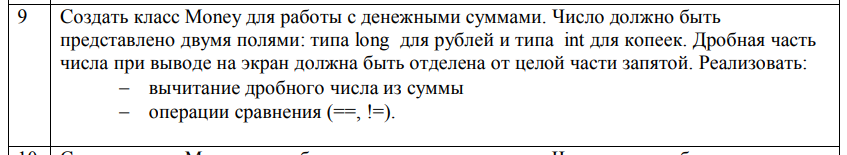
Доцент кафедры ИТАС

Полякова О.А.

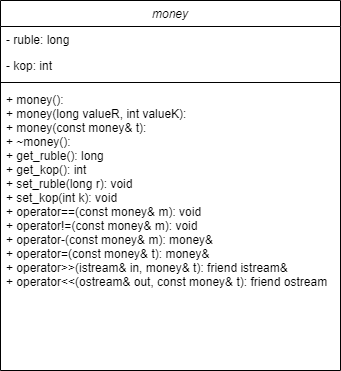
Г. Пермь-2

**Постановка задачи**

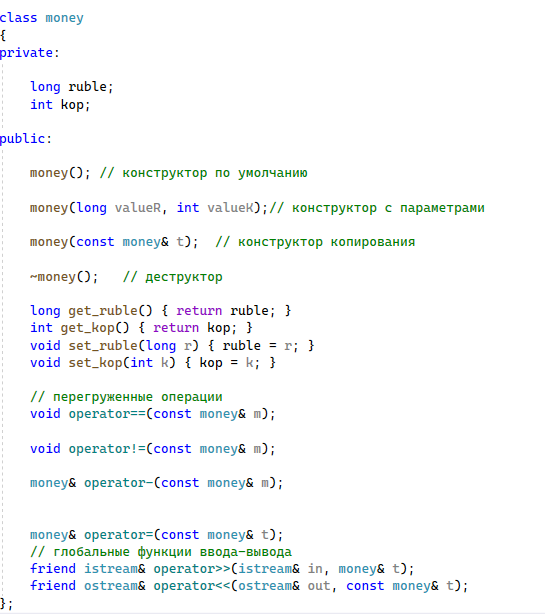
1. Определить пользовательский класс.  
2. Определить в классе следующие конструкторы: без параметров, с параметрами, копирования.  
3. Определить в классе деструктор.  
4. Определить в классе компоненты-функции для просмотра и установки полей данных (селекторы и модификаторы).  
5. Перегрузить операцию присваивания.  
6. Перегрузить операции ввода и вывода объектов с помощью потоков.  
7. Перегрузить операции указанные в варианте.  
8. Написать программу, в которой продемонстрировать создание объектов и работу всех перегруженных операций.



**Диаграмма классов**



**Описание класса**



**Определение компонентных функций**

money::money() // конструктор по умолчанию

{

cout << "Вызван конструктор без параметров для объекта " << this << "." << endl;

}

money::money(long valueR, int valueK) // конструктор с параметрами

{

ruble = valueR;

kop = valueK;

}

money::money(const money& t) // конструктор копирования

{

ruble = t.ruble;

kop = t.kop;

}

money::~money() { // деструктор

cout << "Вызван деструктор для объекта " << this << endl;

}

// перегрузка операции присваивания

money& money::operator=(const money& t)

{

// проверка на самоприсваивание

if (&t == this) return \*this;

ruble = t.ruble;

kop = t.kop;

return \*this;

}

// перегрузка функции - оператора ==

void money::operator==(const money& m)

{

if ((ruble == m.ruble) && (kop == m.kop))

{

cout << ruble << "," << kop << " и " << m.ruble << "," << m.kop << " равны." << endl;

}

else

{

cout << ruble << "," << kop << " и " << m.ruble << "," << m.kop << " не равны." << endl;

}

}

// перегрузка функции - оператора !=

void money::operator!=(const money& m)

{

if ((ruble != m.ruble) || (kop != m.ruble))

{

cout << ruble << "," << kop << " и " << m.ruble << "," << m.kop << " не равны." << endl;

}

else

{

cout << ruble << "," << kop << " и " << m.ruble << "," << m.kop << " равны." << endl;

}

}

// перегрузка бинарной операции вычитания

money& money::operator-(const money& m)

{

int t1 = ruble + m.ruble;

int t2 = kop + m.kop;

ruble = t1 + (t2 / 100);

kop = 0;

return \*this;

}

// перегрузка глобальной функции - операции ввода

istream& operator>>(istream& in, money& t)

{

cout << "Введите рубли: "; in >> t.ruble;

cout << "Введите копейки: "; in >> t.kop;

return in;

}

// перегрузка глобальной функции - операции вывода

ostream& operator<<(ostream& out, const money& t)

{

return (out << t.ruble << "," << t.kop);

}

**Функция main**

#include <iostream>

#include "money.h"

using namespace std;

int main()

{

setlocale(0, "Rus");

money m1(100, 10);

money m2(20,12);

cout << m1 - m2 << endl;

money m3(90, 80);

money m4(30, 40);

money m5(70, 60);

money m6(70, 60);

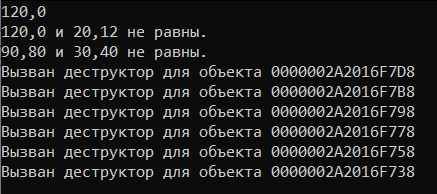
m1 != m2;

m3 == m4;

return 0;

}

**Результаты работы программы**

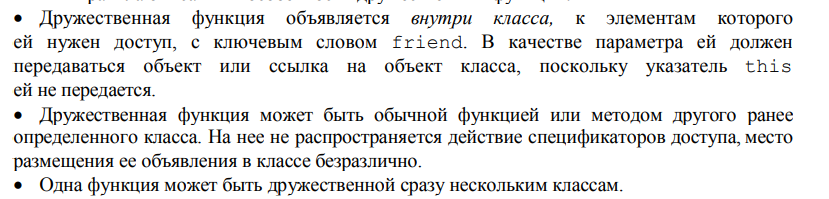


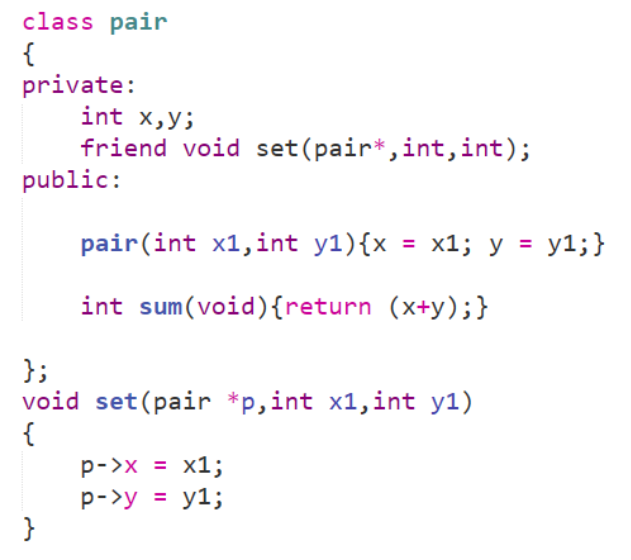
**Ответы на контрольные вопросы**

1. Для чего используются дружественные функции и классы?

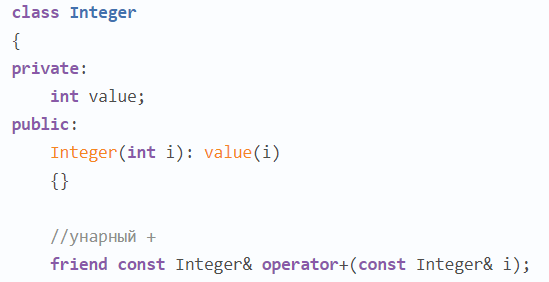
Дружественная функция - это функция, которая может получить доступ к закрытым членам класса, как если бы она была членом этого класса.

2. Сформулировать правила описания и особенности дружественных функций.



  
3. Каким образом можно перегрузить унарные операции?

Унарную операцию можно перегрузить:  
- Как компонентную функцию класса



- Как внешнюю (глобальную) функцию

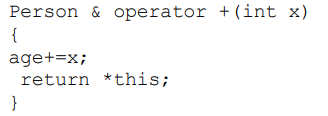
4. Сколько операндов должна иметь унарная функция-операция, определяемая внутри класса?

0   
5. Сколько операндов должна иметь унарная функция-операция, определяемая вне класса?

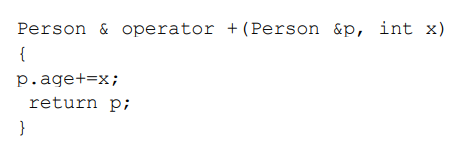
1

6. Сколько операндов должна иметь бинарная функция-операция, определяемая внутри класса?

1. Ее операнд – вызвавший ее же объект.

  
7. Сколько операндов должна иметь бинарная функция-операция, определяемая вне класса?

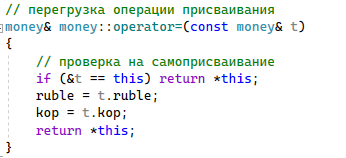
2 операнда типа класса.



8. Чем отличается перегрузка префиксных и постфиксных унарных операций?

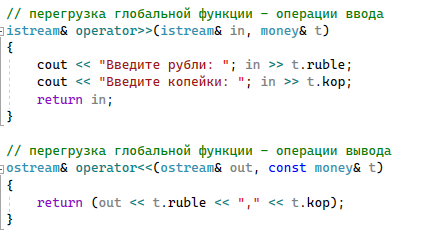
* если перегружается префиксная форма оператора ++, то в классе нужно реализовать операторную функцию operator++() без параметров;
* если перегружается префиксная форма оператора —, то в классе нужно реализовать операторную функцию operator—() без параметров;
* если перегружается постфиксная форма оператора ++, то в классе нужно реализовать операторную функцию operator++(int d) с одним целочисленным параметром. В этом случае параметр d не используется в функции. Он указывается только для того, чтобы указать что это именно постфиксная реализация оператора **++**. Имя d может быть заменено другим именем;
* если перегружается постфиксная форма оператора **—**, то в классе нужно реализовать операторную функцию operator—(int d) с одним параметром. Параметр d необходим для указания того, что перегружается именно постфиксная реализация оператора**—**

9. Каким образом можно перегрузить операцию присваивания?

  
10. Что должна возвращать операция присваивания?

Объект  
11. Каким образом можно перегрузить операции ввода-вывода?

При помощи дружественной глобальной ф-ии.



12. В программе описан класс class Student  
{  
Student& operator++();  
};  
и определен объект этого класса  
Student s;  
Выполняется операция  
++s;  
Каким образом, компилятор будет воспринимать вызов функции-операции?

Префиксный инкремент.

13. В программе описан класс  
class Student  
{  
friend Student& operator ++( Student&);  
};  
и определен объект этого класса  
Student s;  
Выполняется операция  
++s;  
Каким образом, компилятор будет воспринимать вызов функции-операции?

Постфиксный инкремент.

14. В программе описан класс class Student  
{  
bool operator<(Student &P);  
};  
и определены объекты этого класса  
Student a,b;  
Выполняется операция  
cout<<a<b;  
Каким образом, компилятор будет воспринимать вызов функции-операции?

Если а действительно меньше b вернется 1.

15. В программе описан класс  
class Student  
{  
friend bool operator >(const Person&, Person&)  
};  
и определены объекты этого класса  
Student a,b,  
Выполняется операция  
cout<<a>b;  
Каким образом, компилятор будет воспринимать вызов функции-операции?

Если а действительно больше b вернется 1.