**2015研究生面向对象能力摸底测试**

1. 定义一个商品类CGoods，其中包含商品号(long no)、商品名(char \*p\_name)、商品价格(double price)三个数据成员，以及相应的构造函数、拷贝构造函数、析构函数、打印数据成员的成员函数。

2. 为CGoods类增加一个商品总数(int count)数据成员，并增加一个成员函数getCount()获取count的值，编写一个友元函数getName()获取商品名称p\_name。做如上修改后，重新实现CGoods类(与第1问相同的不用再重复)。

注释：根据功能要求，仔细考虑count应该定义成什么成员！

3. 为CGoods类定义小于运算符(‘<’)和不小于运算符(‘>=’)两个运算符重载函数。CGoods类对象大小的比较是根据其商品价格(price)的值的大小来实现的。(与第2问相同的不用再重复)

4. 以CGoods类为基类，派生出服装类CClothes和食品类CFood两个派生类，并在这两个类中分别增加一个表示品牌的指针数据成员(char \*p\_brand)和表示用途的成员函数(void usedFor()——可分别输出一条表示服装和食品用途的信息)。写出CClothes类和CFood类的完整定义(包括构造、析构和usedFor()成员函数的实现)。

5.为了能够采用动态联编的方式调用派生类的usedFor()成员函数，应该在CGoods类及其派生类CClothes和CFood类中作何改动？

**#include**<iostream>

**using** **namespace** std;

//商品类定义

**class** CGoods{

**protected**:

**long** no;

**char** \*p\_name;

**double** price;

**static** **int** *count*;

**public**:

//无参构造函数

**CGoods**();

//有参构造函数

**CGoods**(**long** no,**char** \*q\_name,**double** price);

//拷贝构造函数

**CGoods**(CGoods &goods);

//析构函数

**virtual** **~CGoods**();

**void** **printNo**();

**void** **printName**();

**void** **printPrice**();

**void** **print**();

**int** **getCount**();

**int** **getNo**();

**friend** **char**\* **getName**(CGoods& goods);

**bool** **operator <** (CGoods& goods);

**bool** **operator >=** (CGoods& goods);

**virtual** **void** **usedFor**();//虚函数，子类可重写此函数实现多态，注意虚函数与纯虚函数的区别

};

//商品类实现

**CGoods::CGoods**(){}

**CGoods::CGoods**(**long** no,**char** \*q\_name,**double** price){

**this**->no = no;

**this**->p\_name = q\_name;

**this**->price = price;

}

//拷贝构造函数

**CGoods::CGoods**(CGoods &goods){

**this**->no = goods.no;

**this**->p\_name = goods.p\_name;

**this**->price = goods.price;

**this**->*count* = goods.*count*;

}

**CGoods::~CGoods**(){}

**void** **CGoods::printNo**(){

cout << "商品号："<<**this**->no<<**endl**;

}

**void** **CGoods:: printName**(){

cout << "商品名："<<**this**->p\_name<<**endl**;

}

**void** **CGoods:: printPrice**(){

cout << "商品价格："<< **this**->price<<**endl**;

}

**void** **CGoods:: print**(){

cout << "商品号："<<**this**->no << "\t商品名："<< **this**->p\_name <<"\t商品价格:"<<showpoint<<**this**->price<<**endl**;

}

**int** **CGoods::getCount**(){

**return** **this**->*count*;

}

**int** **CGoods::getNo**(){

**return** **this**->no;

}

**char**\* **getName**(CGoods& goods){

**return** goods.p\_name;

}

**bool** **CGoods::operator <** (CGoods& goods){

**if**(**this**->price<goods.price){

**return** **true**;

}**else**{

**return** **false**;

}

}

**bool** **CGoods::operator >=** (CGoods& goods){

**if**(**this**->price<goods.price){

**return** **false**;

}**else**{

**return** **true**;

}

}

**void** **CGoods::usedFor**(){

cout << "商品号：" <<**this**->no<<"\t商品名："<<

**this**->p\_name<<"\t商品价格："<<**this**->price<<"\t\t商品总数："<<**this**->*count*<<**endl**;

}

**int** *CGoods::count*=10;

//衣服类定义

**class** CClothes:**public** CGoods{

**char** \*p\_brand;

**public**:

**CClothes**();

**CClothes**(CGoods& goods,**char** \*p\_brand);

**virtual** **~CClothes**();

**void** **usedFor**();

};

//衣服类实现

**CClothes::CClothes**(){}

**CClothes::CClothes**(CGoods& goods,**char** \*p\_brand):CGoods(goods),p\_brand(p\_brand){}

**CClothes::~CClothes**(){}

**void** **CClothes::usedFor**(){//重写自己的usedFor函数，实现多态

cout<<"品牌:"<<**this**->p\_brand<<"\t";

CGoods::usedFor();

}

//实物类定义

**class** CFood:**public** CGoods{

**char** \*p\_brand;

**public**:

**CFood**();

**CFood**(CGoods& goods,**char** \*p\_brand);

**~CFood**();

**void** **usedFor**();

};

//实物类实现

**CFood::CFood**(){}

**CFood::CFood**(CGoods& goods,**char** \*p\_brand):CGoods(goods),p\_brand(p\_brand){}

**CFood::~CFood**(){}

**void** **CFood::usedFor**(){

cout<<"品牌:"<<**this**->p\_brand<<"\t";

CGoods::usedFor();

}

//测试类

**int** **main**(){

CGoods goods1(1,"牛奶",5.20),goods2(2,"面包",6.00);

goods1.print();

//动态联编方式，父类接收子类对象，运行时才确定调用哪个对象的方法

CGoods \*clothes = **new** CClothes(goods1,"衣服");

clothes->usedFor();

**delete** clothes;

CGoods \*food = **new** CFood(goods2,"面包");

food->usedFor();

**delete** food;

//传统调用方式

CClothes clothes1(goods1,"衣服");

clothes1.usedFor();

CFood food1(goods2,"食物");

food1.usedFor();

cout<<"友元函数得到了商品号为："<<goods1.getNo()<<" 的商品名字是："<<getName(goods1)<<**endl**;

**if**(goods1<goods2){

cout<<"运算符<重载：goods1<goods2"<<**endl**;

}

**if**(goods2>=goods1){

cout << "运算符>=重载：goods2>=goods1"<<**endl**;

}

**return** 0;

}

测试结果：

