1. 类

3.1类和对象

Class human（类标签） C++中以类作为数据类型的变量就是对象

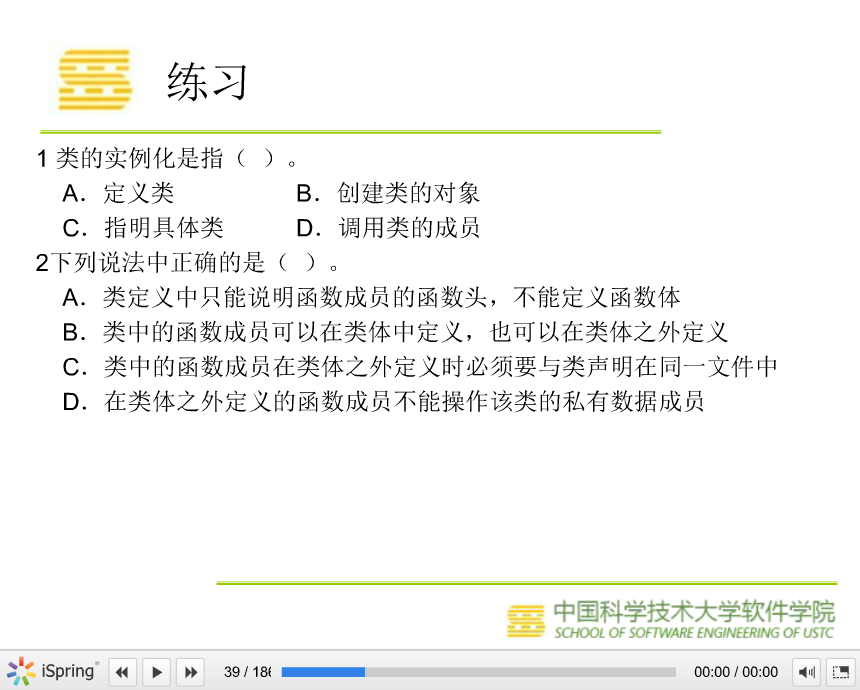
{ .............. }； human a；a就是对象

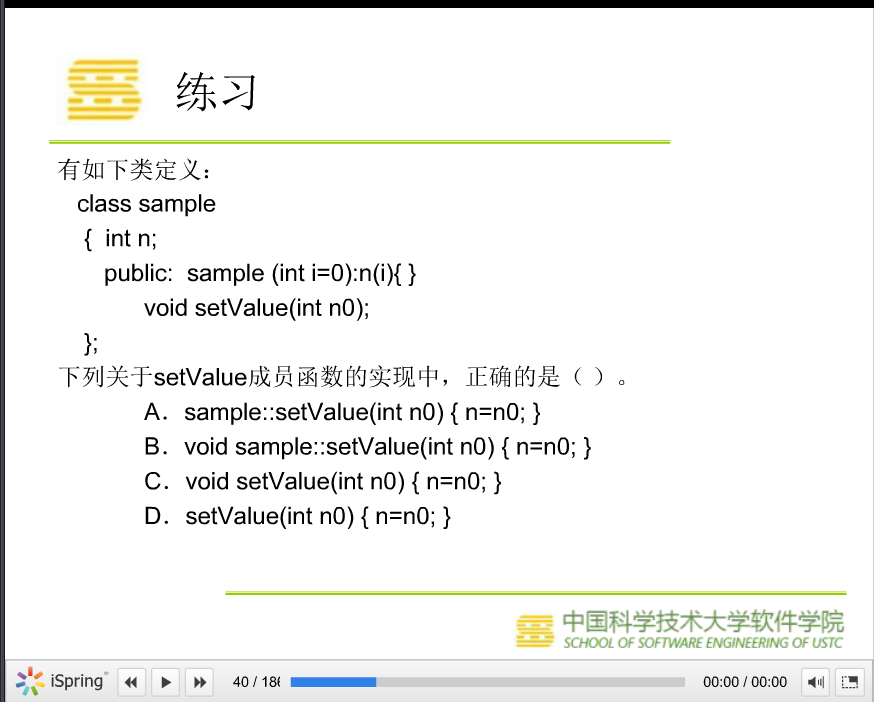
Private隐藏类实现，public暴露接口。Class创建类默认类成员是私有的，Struct相反。

类的私有成员只能由类的成员函数访问。

类的成员函数在类中声明，可在类内（inline内联方式）或者内外定义。

类外定义需要加在函数名前加：：，类内直接定义即可。

B,B

B

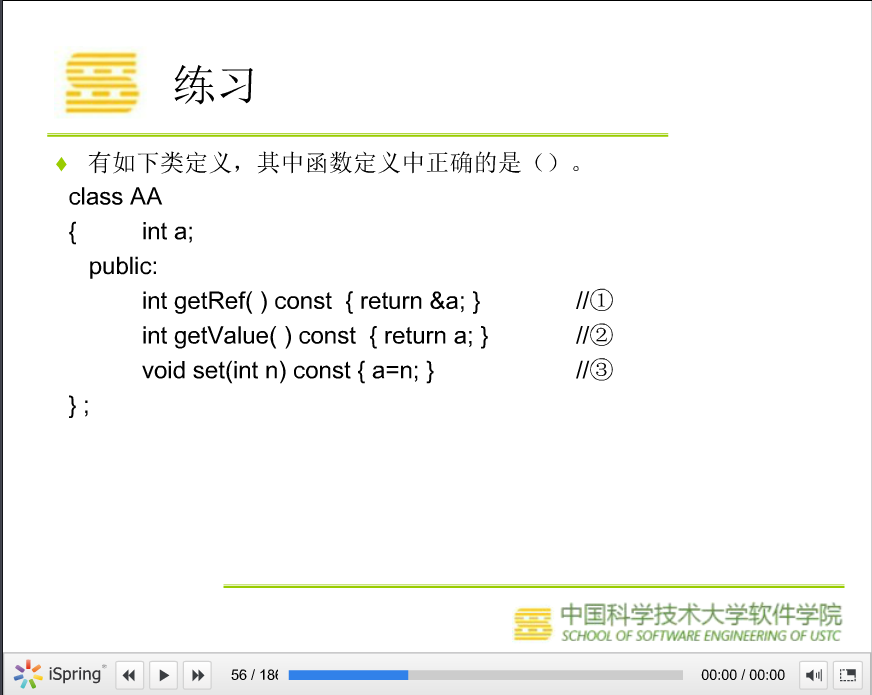
3.3效率和健壮性

通过引用来传递和返回对象，一般使用引用的方式，传值的话会降低效率，浪费内存。

用const标记函数的参数，表明该函数不会改变传来的值，直接对这个值进行操作。

一个函数不需改变所属对象的任何数据成员，用const标记一下。Const成员函数只能调用其他的const成员函数。（见p50）Const返回（p53）

常成员函数只能引用本类中的数据成员但不能修改它。

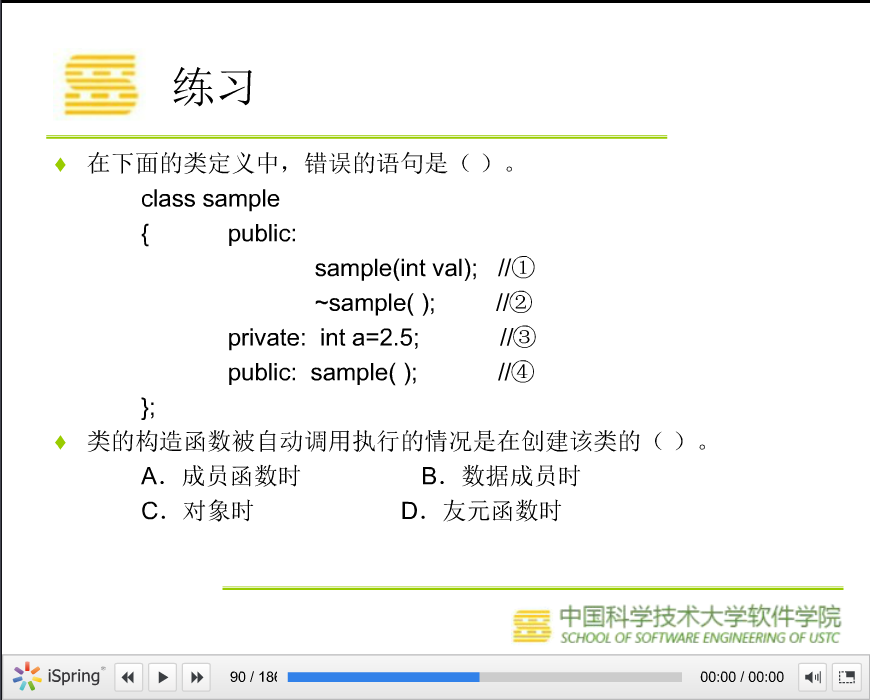


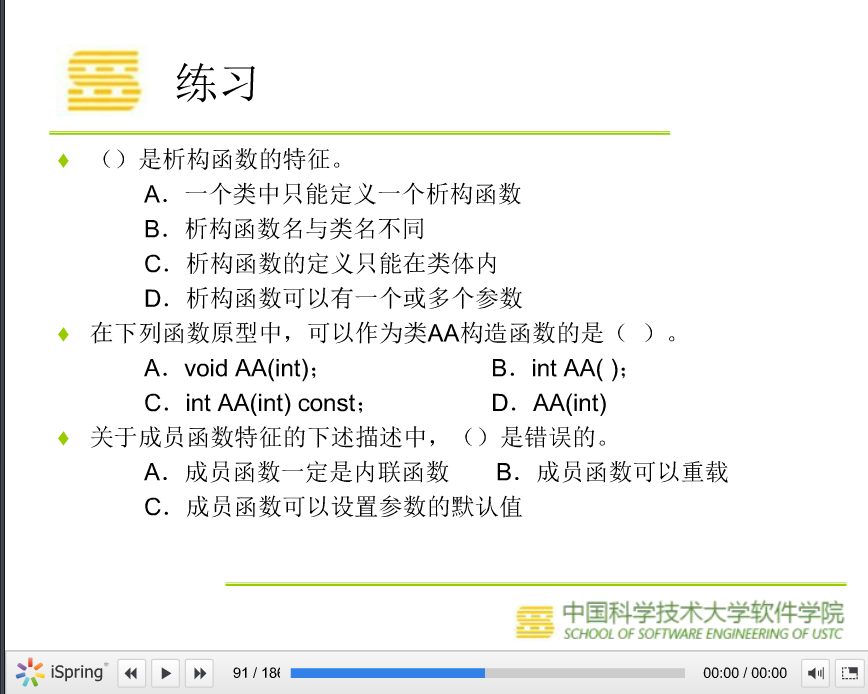
上题选2，一中返回类型与定义不同，3中不能对常成员函数的操作数赋值

3.5构造函数和析构函数

编译器会自动的调用这两个函数，不许我们做出调用动作。构造函数可有多个，析构函数最多一个。

构造函数是一种与类同名的成员函数，可以重载，不能有返回类型，当创建一个对象时，相应的构造函数（根据参数类型）会被自动调用。默认构造函数是没有参数的函数，若类中无构造函数，会自动生成一个默认构造函数，一旦声明了一个含参构造函数，系统就不会自动生成默认构造函数。构造函数主要用来对数据成员进行初始化和其他一些对象创建时需要处理的事务。



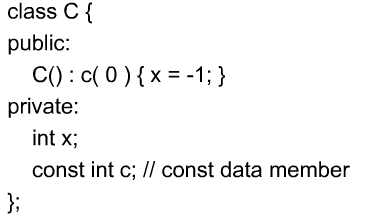


拷贝构造函数的参数类型是引用，第一个参数以后的参数都必须有默认值。若没有定义拷贝构造函数，系统会自动生成一个。主要作用是创建一个对象，此对象是另一个对象的拷贝品。

这两个对象指向同一个存储区域，即无论改变哪一个对象的内容，另一个都会随之改变。如果拷贝构造函数是私有的，其他的成员函数就不能通过传值的方法操作该类对象，必须使用类的引用。

转型构造函数用于类型转换，只有一个参数。

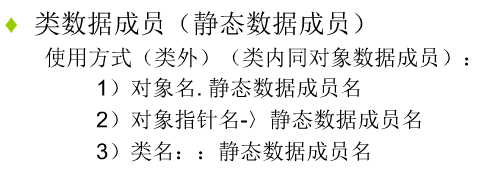
构造函数初始化：对const类型数据成员初始化时不能直接赋值，需要使用初始化列表赋值。其他的成员也可用列表赋值。



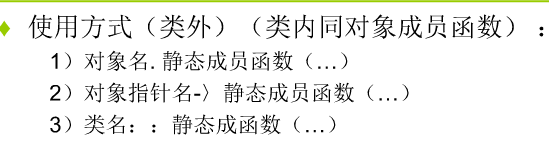
析构函数:对象被摧毁时，系统会自动调用析构函数 ~类名（）。不带参数，不能被重载，所以每个类只能有一个析构函数，并且没有返回类型。

3.7类数据成员和类成员函数

类数据成员：为一个类所有对象共享，声明前要加static，即为静态成员，声明时要在所有程序块外面，不会影响类与对象的sizeof。访问类成员需要使用：：。



类成员函数：静态成员函数只能访问其他的static成员，非静态成员函数可以访问static成员也可访问非static成员。



成员函数内的局部变量可定义为static，这样类中所有对象在调用该成员函数时都可共享该变量。

1. 继承

4.2概念和语法

可以一对多（单继承）或者多对一（多重继承）（父类-->子类）。继承方式默认为私有的。

派生类包括基类的成员和自己新加的成员。公有方式继承时，基类中公有的成员在派生类中也是公有的。

4.2.1继承中的私有成员

基类的私有成员在派生类中不可见，但派生类会为这些成员分配空间。派生类可以通过继承而来的公有成员函数间接访问基类的私有成员。

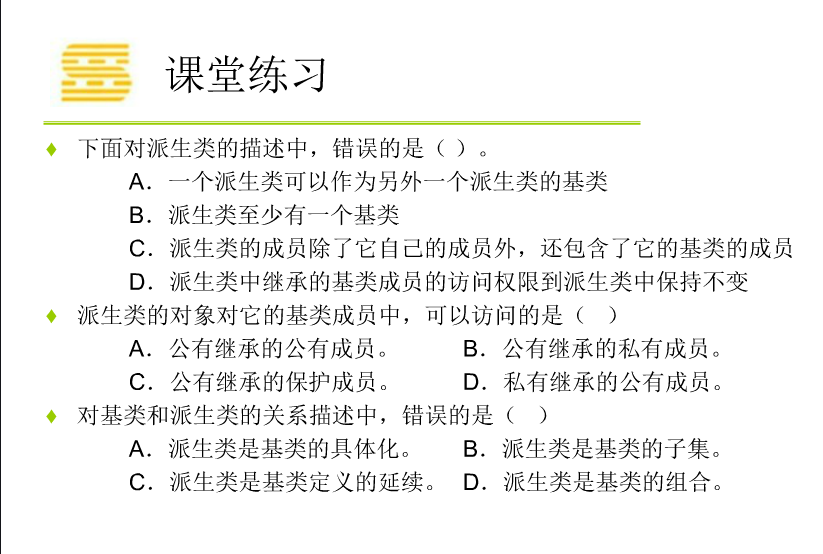
4.2.2改变访问限制

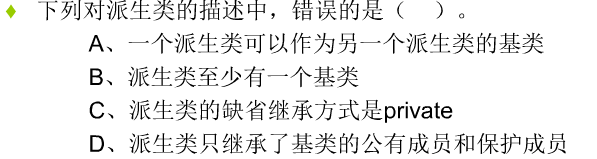
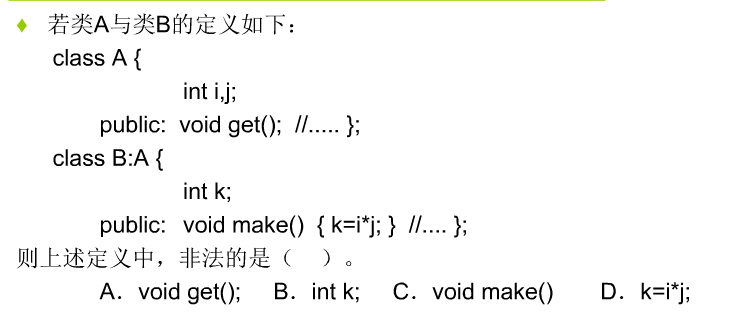
Using可以改变成员在派生类中的访问限制，using 基类名：：成员名。可以通过using把某些不适合派生类的基类成员变为私有或者保护。

4.2.3名字隐藏

派生类若新增了一个与基类某成员名字相同的成员，则会将隐藏基类中的同名成员。想要访问被隐藏的成员的话，需要使用“派生类对象.基类名：：被隐藏的成员”。

4.2.4间接继承





4.4保护成员

无继承时与私有成员一样只在本类中可见，但是继承后，保护成员在派生类中是可见的，但是派生类的对象不能访问继承来的保护成员，只能在类声明中访问。

派生类继承而来的保护成员依旧是保护成员。

派生类可以访问基类的保护成员，但是不能访问基类对象的保护成员，因为基类对象属于基类，不属于派生类。

一般不建议将成员定义为保护成员，可以先将其定义为私有成员，然后在设计一个可以进行访问的保护成员函数，通过这个保护成员函数访问。

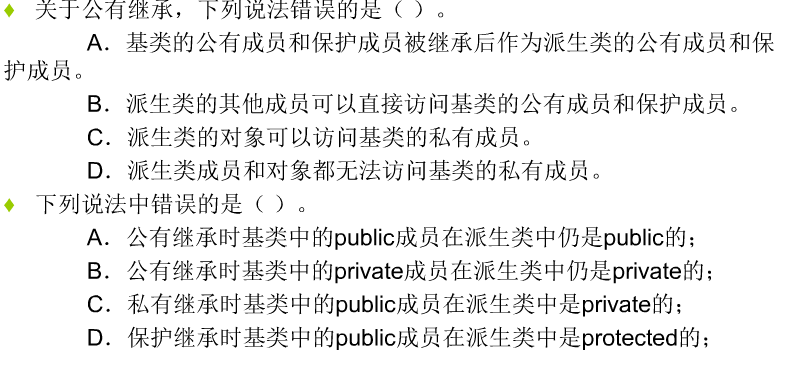
4.8补充

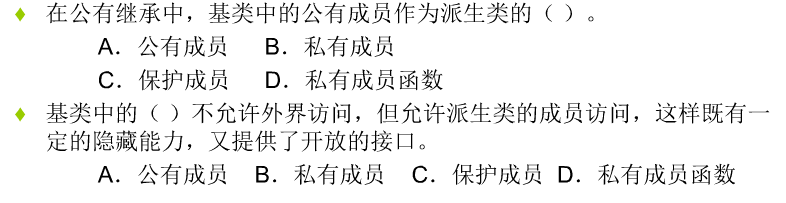
4.8.1保护继承

公有成员——》保护成员；保护成员不变；私有成员仅在基类中可见。

4.8.2私有继承

公有成员——》私有成员；保护成员——》私有成员；私有成员仅在基类中可见。

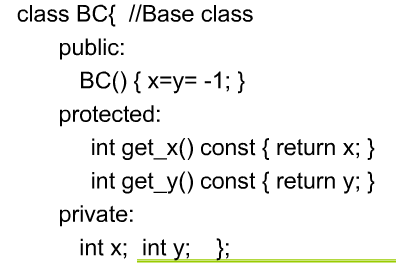
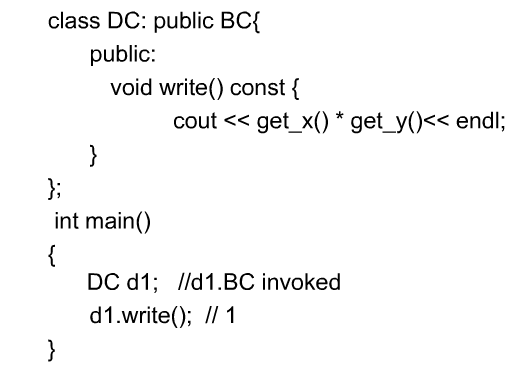




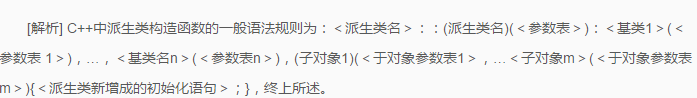
4.5继承机制下的构造函数和析构函数

4.5.1构造函数

创建派生类对象时，基类的缺省构造函数被自动调用，对派生类对象中的基类部分进行初始化。

派生类在定义对象时虽然可以自动调用基类的构造函数和析构函数，但是并不能继承他们。若在派生类中定义了构造函数，则由他来负责派生类对象中派生类部分的初始化。



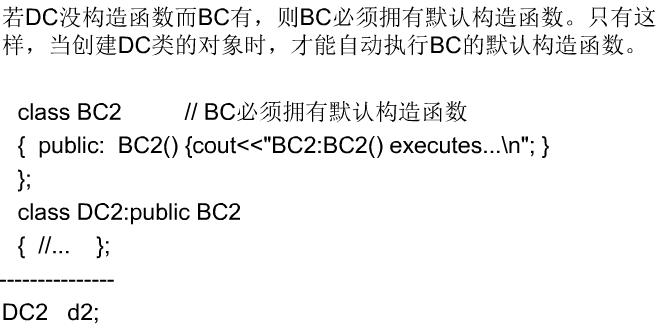
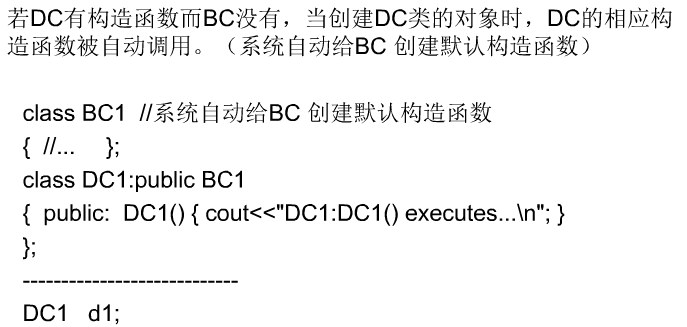
执行顺序：基类构造函数，数据成员是类对象的构造函数，派生类自己的构造函数。

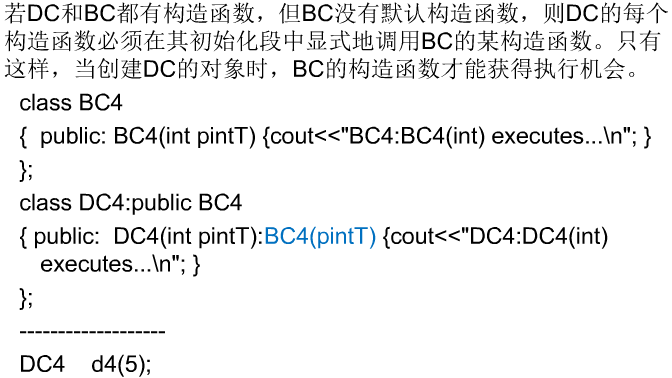
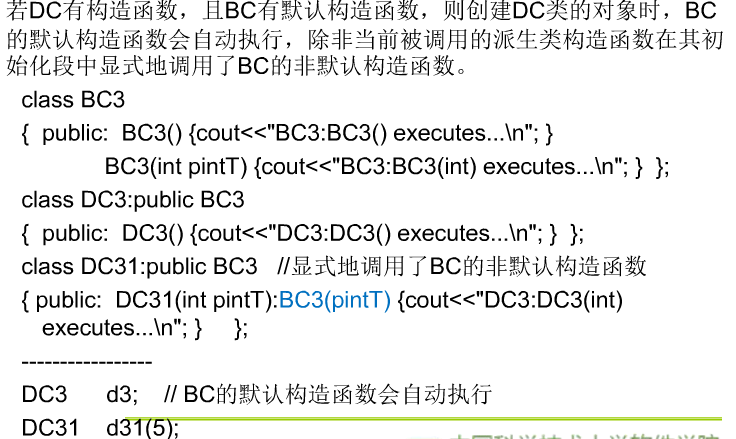
多米诺骨牌效应。

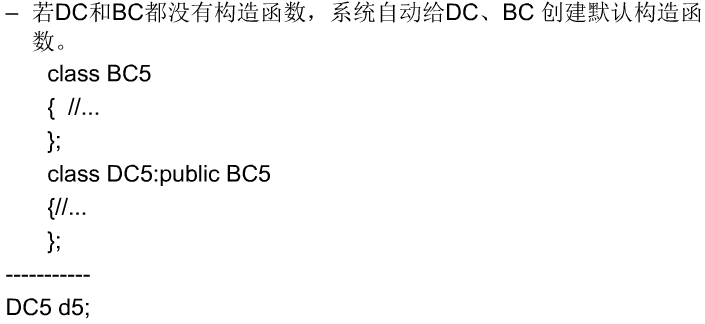
4.5.2派生类构造函数的规则

若基类有构造函数但是没有默认构造函数，派生类的构造函数必须显式调用基类的构造函数。

所以最好为基类定义一个默认构造函数，以防出错。若基类有构造函数，派生类也有，但是派生类没有显式调用基类构造函数，此时，创建一个派生类对象会自动调用基类的默认构造函数。



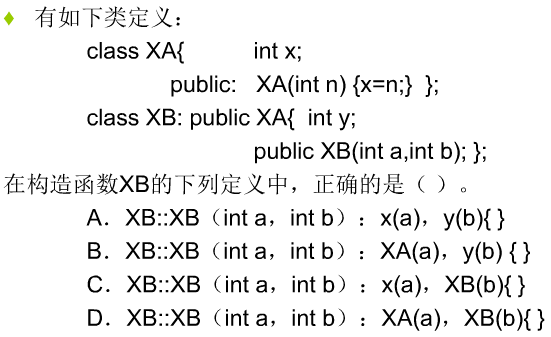
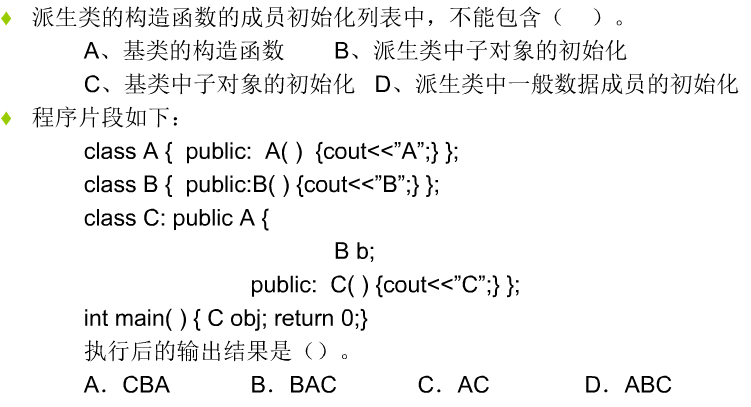




4.5.3继承机制下的析构函数

与构造函数相反，析构函数按照派生类到基类的顺序执行。保证最近分配的内存优先被释放。

因为类中只能有一个析构函数，所以不会出现二义性，不需要显式调用基类的析构函数。

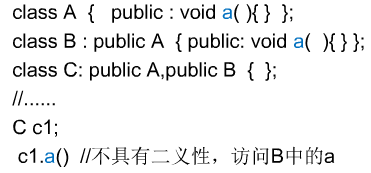
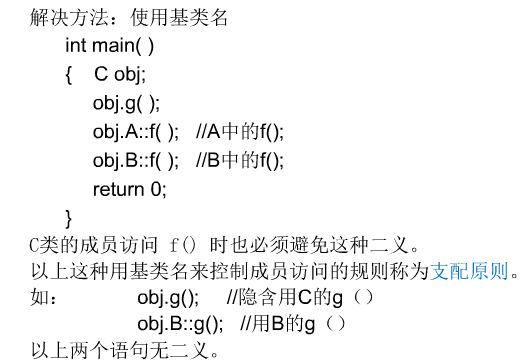
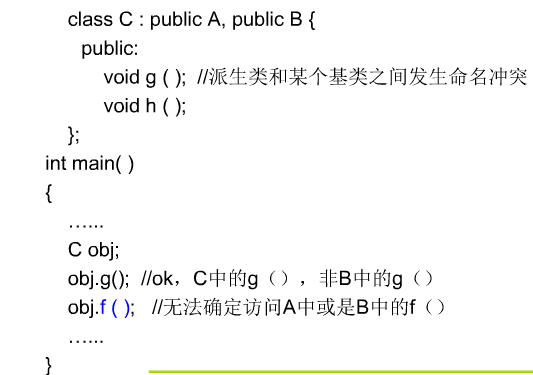


4.7多继承

4.7.1继承和访问规则

继承和访问的规则还是与单继承相同。多继承时，基类构造函数执行的顺序按照派生类声明的顺序依次调用。基类的构造函数按声明顺序调用，然后是数据成员所在类的构造函数，最后是派生类的构造函数。在派生类中出现调用的函数在多个基类中都存在，此时用

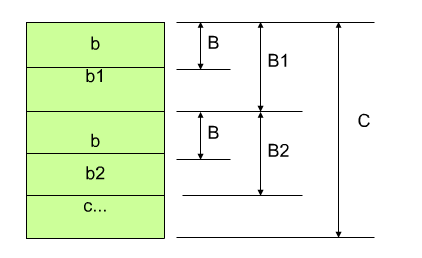
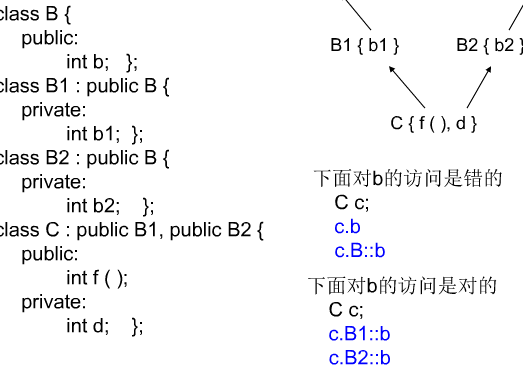
基类名：：数据成员的形式表示调用的是哪个基类里的成员。如果在基类和子类中分别定义的有名字相同的成员，派生类的占据支配地位



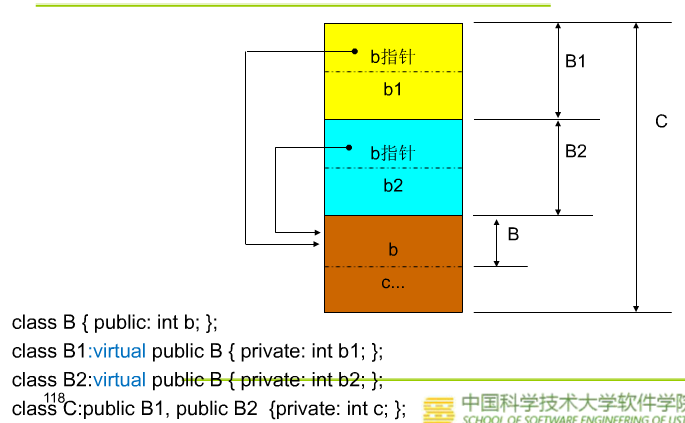
4.7.2虚基类

派生类间接继承了一个基类多次，当多条继承路径上有一个公共的基类时，如果希望只存储一个公共基类，可用虚基类机制，关键字virtual。

若不使用虚基类：

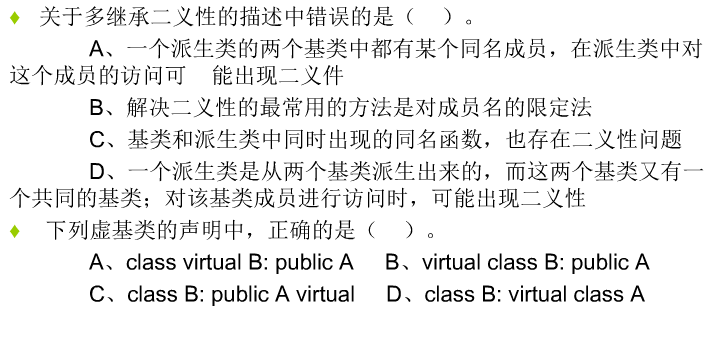
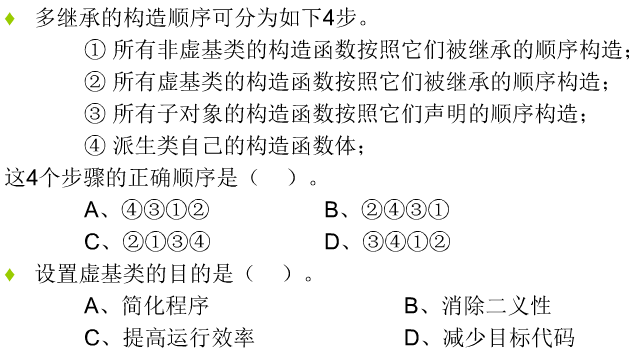


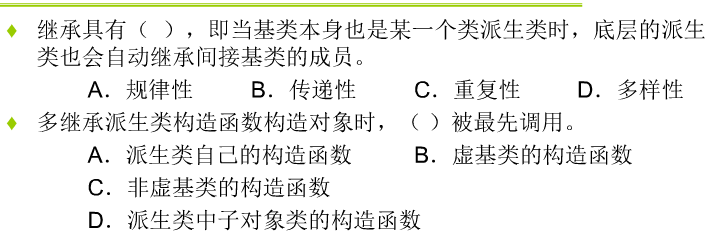
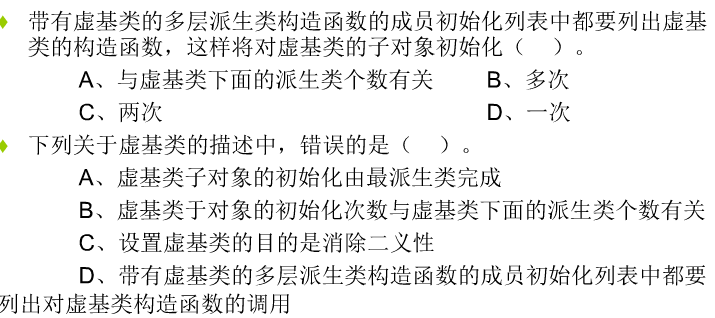
若使用虚基类：

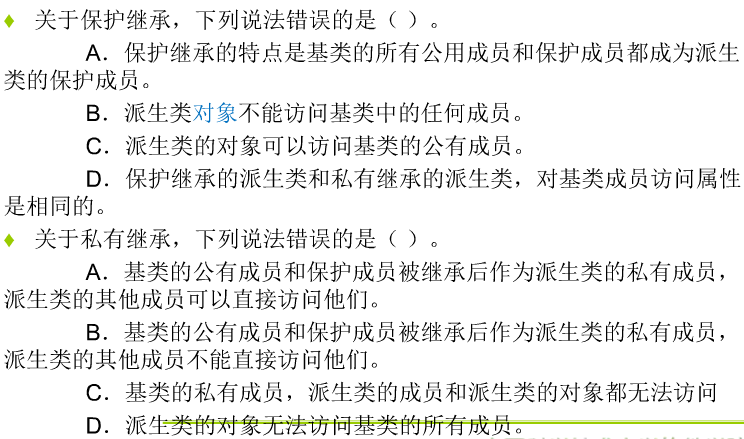
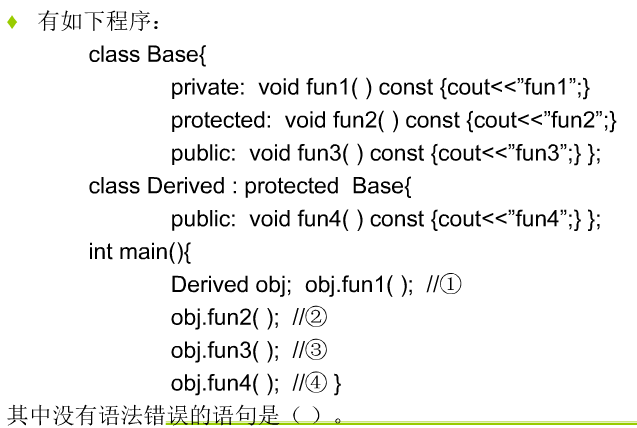
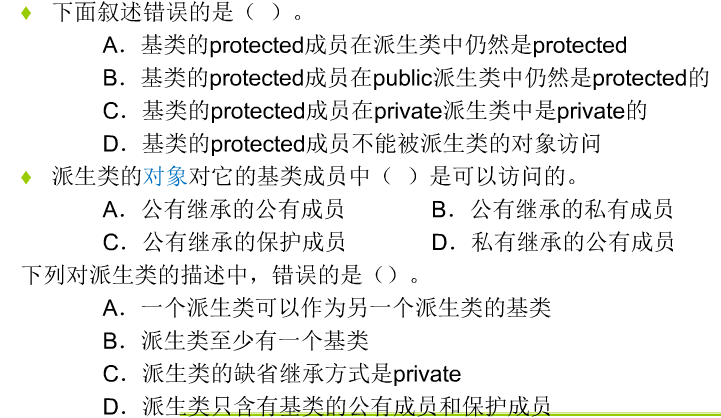
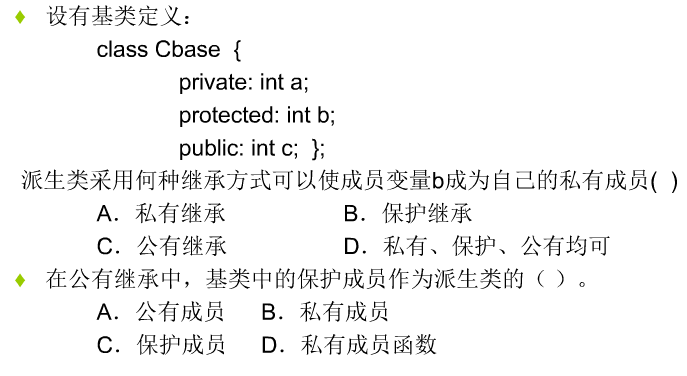


派生类中对于基类构造函数的调用，先调用虚基类的，再调非用虚基类的构造函数。

虚基类的构造函数只能由最派生类调用，同时虚基类的子对象由最派生类的构造函数通过调用虚基类的构造函数进行初始化。但虚基类的构造函数只能被调用一次。除了最派生类，其他的派生类也会调用虚基类构造函数，只是不会被执行，保证虚基类子对象只会通过最派生类构造函数初始化一次。







1. 多态

多态就是不同的对象调用名称相同的函数，而且操作的结果不同。同一接口，多种方法。

编译多态性：通过函数或者运算符重载实现。函数重载通过参数类型或个数确定调用哪个函数，运算符重载可以直接根据运算对象选择。

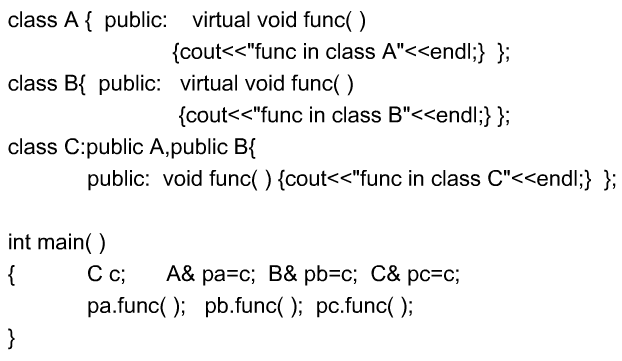
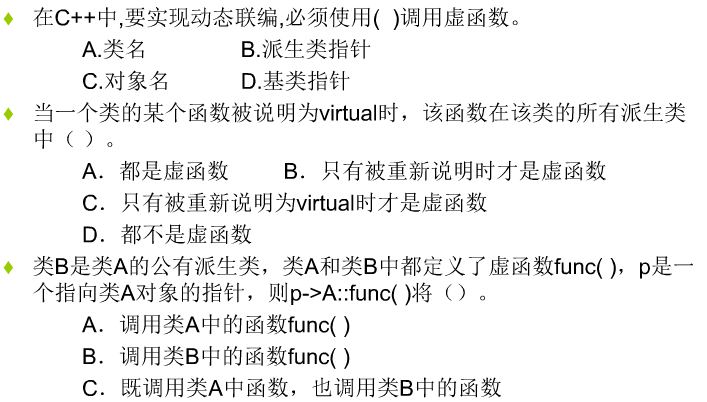
运行时多态性：在运行前无法确定该调用哪个，在程序中动态确定。

一个函数的绑定发生在运行时刻而非编译时刻，就称该函数是多态的。

多态的三个条件：继承，有同名函数，有指针或者引用。

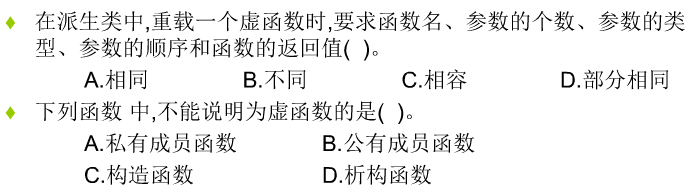
当基类的成员函数被声明为虚函数以后，即使派生类定义的同名函数前没有加virtual，也会被默认为虚函数。

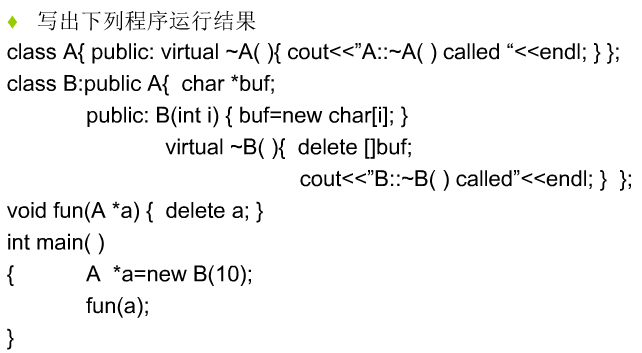
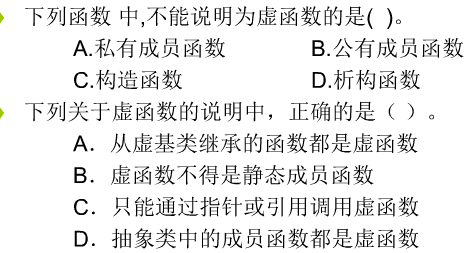
顶层函数不能为虚函数。



派生类中可以从基类继承虚函数。构造函数不能是虚函数，但析构函数可以是。只有非静态成员函数才可以是虚函数。

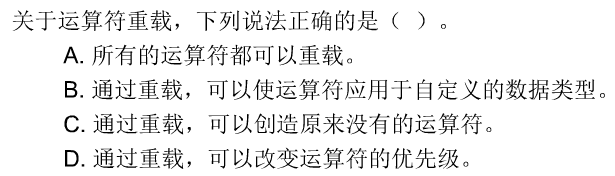
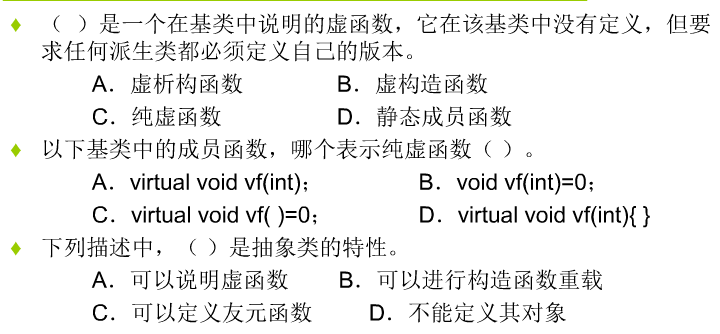
如果某几个类中的同名函数不是虚函数，测系统使用的是编译期绑定。若使用指针调用该函数，该指针属于哪个类类型就一直调用那个类的函数。





5.4抽象基类

有纯虚成员函数（只有虚函数才可以被声明为纯虚函数）的基类是抽象基类，抽象基类不能创建对象，但可以有派生类，如果其派生类能够覆盖其纯虚成员函数，则该派生类就不是抽象基类，可以创建对象。抽象基类的应用主要是在项目中想要每个类都有某些函数实现一定的功能，此时就可以把基类定义为抽象基类，这些函数为纯虚函数，所有的派生类为了能够创建对象就必须覆盖这些纯虚函数，从而可以保证这些类可以实现该功能。



第六章 操作符重载

6.1基本操作符重载

基类中重载的操作符都会被派生类继承。若要重载一个二元操作符，则其操作符重载函数只有一个参数，两个操作数的第一个操作数调用该函数，另一个作为参数同理，重载一个一元操作符，重载函数没有参数。



6.1.1操作符的优先级和语法

重载操作符函数不改变操作符的优先级和语法，

6.3用顶层函数进行操作符重载

顶层函数的形式进行操作符重载时，参数表中必须包含一个类的对象。

一般情况下若要重载一个二元操作符，则其操作符重载函数只有一个参数，但是用顶层函数进行二元运算符重载时，需要两个参数。同理，一元的就需要一个参数。

但是[],(),->,=四个操作符必须用类成员函数的形式进行重载。

6.4友元函数

用friend声明一个函数，该函数不是类的成员函数，可以在类外访问类的所有成员。

也可用friend将A类声明为另B类的友元，这样A类的所有函数都可以访问B类的所有成员。

Friend声明的友元是单向的，关系不能传递。

