本文重要介绍普通继承中如何写派生类的六个默认成员函数，主要是针对在派生类中，如何调用基类的六个默认成员函数

需要说明的一点就是，如果子类中没有调用父类的函数时，系统会自动生成一个。

**构造函数**

子类中有父类的成员，子类首先需要调用父类的构造函数，然后调用自己的构造函数。如果没有调用父类的构造函数，系统会自动生成一个。

在构造函数中，我们使用父类的匿名对象完成初始化。看下面的代码

**class** A

{

**public**:

A(**int** a = 0)

:\_a(a)

{

}

**private**:

**int** \_a;

};

**class** B : **public** A

{

**public**:

B(**int** a = 0,**int** b = 1)

:A(a)

,\_b(b)

{}

**private**:

**int** \_b;

};

上面的例子我们可以看出来，当我们写子类B的构造函数的时候，首先需要先调用 父类的匿名构造函数，完成对子类中包含的父类成员的初始化的工作。

**拷贝构造函数**

拷贝构造函数需要和构造函数区分开，因为我们的构造函数的参数是可变的，我们可以根据需要，为子类中包含的父类成员传递参数，但是在拷贝构造函数中，我们只能够传递的是子类的对象的引用，但是这个时候我们实际上还是可以调用父类的拷贝构造函数的，根据我们上一篇文章中学习过的赋值兼容规则，子类的对象是可以赋值给父类的对象的，所以这个时候我们可以拿子类的对象去实例化子类中包含的父类成员。此时实际上采用 的还是父类匿名对象。

拷贝构造函数应该写成是下面的形式

**B(const B& b)**

:A(**b)**

,\_b(**b.\_b)**

{}

这里语法规定子类中初始化父类的时候，必须在初始化列表中初始化父类，包括我们上面的构造函数也是这个样子，必须在初始化列表中初始化父类

**赋值运算符的重载**

拷贝构造函数应该如何使用呢，还是上面的思想，我们需要那子类的对象去为父类赋值，所以这个时候，我们可以在子类的赋值运算符重载的函数内部，调用父类的赋值运算符重载的函数，对父类的对象进行赋值。

**B&** operator=(const **B& b)**

{

**if** (this != &**b)**

{

operator=(**b);**

\_b = **b.\_b;**

}

return \*this*;* }

但是上面的问题，会出现一个问题就是，栈溢出，为什么会出现这样的问题呢，主要是在调用父类的赋值运算符重载的 时候，实际上是并没有调用父类的赋值运算符重载，这里调用的是子类的，这样子就形成了递归了，然后会一直调用下去。

解决办法是这样的，我们需要在调用父类的函数前面加上一个作用域限制符。  
A::operator(b);

这里为什么会调用子类的赋值运算符的重载呢，实际上是子类的函数名和父类的函数名构成了重定义，这个时候如果不加上作用域限制符就会出现问题了

**析构函数**

这里在写子类的析构函数的时候，是不允许子类调用父类的析构函数的，有两个原因

* 因为栈帧的关系，后开辟的空间先释放掉，意思就是说，父类是先开辟的空间，应该后释放，为了防止程序员写错代码，先释放了父类的，这个时候就会出现问题，所以要求自动的调用父类的析构函数
* 如果我们在实现析构函数的时候，忘记释放了，这个时候就会造成内存泄漏了，造成资源的浪费

基于上面的一些原因，要求就是只需要由系统自动的调用父类的析构函数即可。