多态是面向对象程序设计语言中数据抽象和继承之外的第三个基本特征。

多态性改善了代码的可读性和组织性，同时也使创建的程序具有可扩展性，项目不仅在最初创建时期可以扩展，而且当项目在需要有新的功能时也能扩展。

C++支持静态多态（编译时多态）和动态多态（运行时多态）。

静态多态： 运算符重载和函数重载

动态多态： 继承和虚函数

理解：

静态多态即函数地址是在编译阶段就已经绑定好了，如函数的重载，传入的参数在编译阶段就已经绑定了。

动态多态即函数地址在运行阶段才能决定，这就是晚绑定。如虚函数，在函数运行时才知道传入的是什么类型的参数。

用一句话总结多态：父类的引用或指针指向子类对象。

虚函数原理解析：

在父类Animal中speak函数声明为虚函数后，该虚函数结构发生了改变，生成了vfptr(virtual function pointer)虚函数表指针，指向了vftable(virtual function table)虚函数表，该表中存的是 &Animal::speak 。



Class Animal

{

Public:

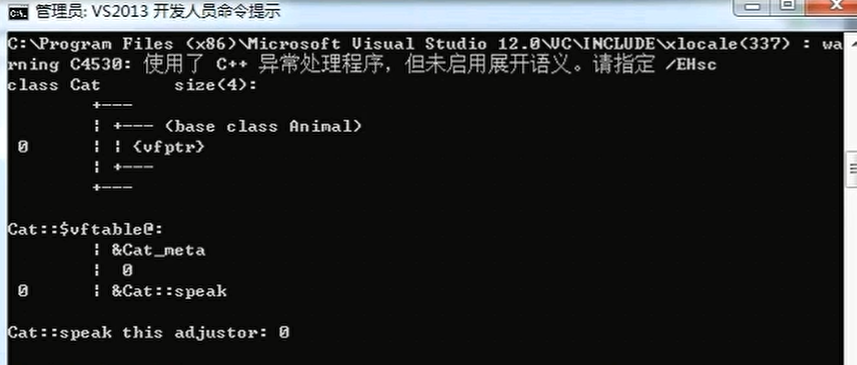
Virtual void speak()

{

}

}

子类Cat继承父类Animal，当子类Cat没有定义自己的speak函数时，Cat继承了父类的vfptr和vbtable，但是Cat的vfptr并不是指向父类的vbtable，而是指向自己的vbtable。这是在Cat对象创建的时候，调用构造函数时，Cat的vfptr就指向了自己的vftable。



Class Cat : public Animal

{  
public:

}

如果子类Cat定义了自己的speak函数，那么子类Cat就会生成 &Cat::speak覆盖掉自己的vftable中的 &Animal::speak， 但是父类中的vftable 还是 &Animal::speak。

重写：子类Cat写父类Animal的虚函数speark，这种写法叫重写。重写规定返回值，参数个数，类型，顺序都要相同。



Class Cat : public Animal

{  
public:

Void speak()

{  
cout << “ 小猫在说话” << endl;

}

}