//动态多态满足条件

//1.有继承关系

//2.子类重写父类的虚函数

//重写，函数返回值，参数名，参数列表完全相同

//动态多态的使用

//父类的引用或指针，执行子类对象

#include<iostream>

using namespace std;

//动物类

class Animal {

public:

virtual void speak() {//虚函数

cout << "speaking" << endl;

}

};

//猫类

class Cat :public Animal {

public:

void speak() {//void 前面可加virtual

cout << "cat speak" << endl;

}

};

//执行说话

void dospeak(Animal& animal) {

animal.speak();

}

void test01() {

Cat cat;

dospeak(cat);

}

//void test01() {

//Cat cat;

//dospeak(cat);

//}这个代码的本意是让显示cat speak，但是在没有virtual时却显示spaeking

//父类和子类的变量是可以无需强制类型转换的，比如Animal sad=cat中，cat是子类的变量，但是在这里是之前变成父类的

//所以执行是speaking。这属于早绑定，在编译阶段确定函数地址

//如果想让猫说话，那么这个函数地址就不能提前绑定，要在运行阶段绑定，是地址晚绑定

int main(){

test01();

system("pause");

return 0;

}