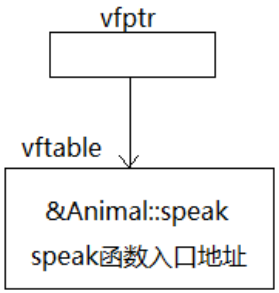
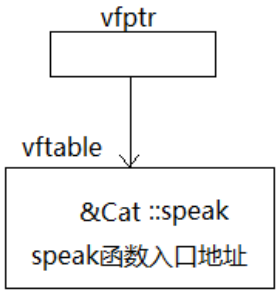
```cpp
// 动物类
class Animal {

public:

	// 虚函数 （采用的就是动态联编）
	virtual void speak() {
		cout << "动物在说话" << endl;
	}

};


// 猫类
class Cat :public Animal {

public:
	void speak() {
		cout << "猫在说话" << endl;
	}

};
```

当子类重写父类的虚函数后，那么子类的虚函数表入口地址就发生了覆盖（重写）。

Animal 类内部结构



Cat 类内部结构



vfptr 虚函数表指针
v virtual
f function
ptr pointer

vftable - 虚函数表
虚函数表记录了虚函数的函数入口地址

多态使用
父类指针或者引用指向子类对象

```cpp
Animal & animal = cat;
Animal * animal = new cat;

animal->speak();
```