C 中的虚函数(virtualfunction) PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/255/2021_2022_C___E4_B8_AD_E7_9A_84_E8_c67_255027.htm 虚函数是C 中用于实现多态(polymorphism)的机制。核心理念就是通过基类访问派生类定义的函数。假设我们有下面的类层次：class A{public:virtual void foo() { cout }.class B: public A{public:virtual void foo() { cout }. 那么，在使用的时候，我们可以：A * a = new B().a->foo(). // 在这里，a虽然是指向A的指针，但是被调用的函数(foo)却是B的! 这个例子是虚函数的一个典型应用，通过这个例子，也许你就对虚函数有了一些概念。它虚就虚在所谓“ 推迟联编 ”或者“ 动态联编 ”上，一个类函数的调用并不是在编译时刻被确定的，而是在运行时刻被确定的。由于编写代码的时候并不能确定被调用的是基类的函数还是哪个派生类的函数，所以被成为“ 虚 ”函数。 虚函数只能借助于指针或者引用来达到多态的效果，如果是下面这样的代码，则虽然是虚函数，但它不是多态的：class A{public:virtual void foo().}.class B: public A{virtual void foo().}.void bar(){A a.a.foo(). // A::foo()被调用}1.1 多态 在了解了虚函数的意思之后，再考虑什么是多态就很容易了。仍然针对上面的类层次，但是使用的方法变的复杂了一些：void bar(A * a){a->foo(). // 被调用的是A::foo()还是B::foo()？} 因为foo()是个虚函数，所以在bar这个函数中，只根据这段代码，无从确定这里被调用的是A::foo()还是B::foo()，但是可以肯定的说：如果a指向的是A类的实例，则A::foo()被调用，如果a指向的是B类的实例，则B::foo()被调用。 这种同一代码可以产生不同效果的特点，被称为“ 多态

”。1.2 多态有什么用？ 多态这么神奇，但是能用来做什么呢？这个命题我难以用一两句话概括，一般的C 教程（或者其它面向对象语言的教程）都用一个画图的例子来展示多态的用途，我就不再重复这个例子了，如果你不知道这个例子，随便找本书应该都有介绍。我试图从一个抽象的角度描述一下，回头再结合那个画图的例子，也许你就更容易理解。 在面向对象的编程中，首先会针对数据进行抽象（确定基类）和继承（确定派生类），构成类层次。这个类层次的使用者在使用它们的时候，如果仍然在需要基类的时候写针对基类的代码，在需要派生类的时候写针对派生类的代码，就等于类层次完全暴露在使用者面前。如果这个类层次有任何的改变（增加了新类），都需要使用者“ 知道 ”（针对新类写代码）。这样就增加了类层次与其使用者之间的耦合，有人把这种情况列为程序中的“ bad smell ”之一。 多态可以使程序员脱离这种窘境。再回头看看1.1中的例子，bar()作为A-B这个类层次的使用者，它并不知道这个类层次中有多少个类，每个类都叫什么，但是一样可以很好的工作，当有一个C类从A类派生出来后，bar()也不需要“ 知道 ”（修改）。这完全归功于多态--编译器针对虚函数产生了可以在运行时刻确定被调用函数的代码。 