# C++\_多态\_01\_多态简介

## 引言

在C++中，多态（Polymorphism）是面向对象编程的重要特性之一，多态性主要通过这几个机制来实现：函数重载、运算符重载、虚函数和模板。下面我们将详细讲解多态及其相关概念。

## 多态的定义

简单来说，多态就是“一个接口，多种实现”，同一操作，传入不同对象，产生不同的行为和结果。

多态性是面向对象程序设计的一大支柱，在程序中指的是在基类中定义的属性或服务被派生类继承之后，可以表现出不同的行为，也就是指一段程序能够处理不同类型对象的能力。类的多态体现在面向对象程序设计的许多方面，函数重载，运算符重载，虚函数的覆盖继承以及模板均是多态性的体现。

举个例子：

比如买票这件事情，普通人买票时，做出的动作是全价买票；学生买票时，做出的动作是半价买票；军人买票时，做出的动作是优先买票。

比如去北京这件事情，甲去北京，做出的动作是坐飞机；乙去北京，做出的动作是坐火车；丙去北京，做出的动作是坐汽车。

## 多态相关概念

1. 多态性：源程序中相同的程序元素在不同的条件下具有不同的语法解释，从而产生不同的动作，C++语言称这些程序元素具有多态性。
2. 多态的实现：对具有多态性的程序元素作出最终明确的语法解释，这称为多态的实现。

## 多态的分类

### 实现角度分类

多态从实现的角度可以分为两类，编译时多态和运行时多态。

前者是在编译的过程中确定了同名操作的具体操作对象，而后者则是在程序运行过程中才动态地确定操作所针对的具体对象。

* 编译时多态（静态多态）：

主要通过函数重载、运算符重载和模板来实现。

在编译时决定调用哪个函数。

* 运行时多态（动态多态）：

主要通过虚函数和继承来实现。

在运行时根据对象的实际类型决定调用哪个函数。

动态多态采用一种延迟绑定技术，普通的函数调用，在编译期间就已经确定了调用的函数的地址，所以无论怎样调用，总是那个函数。

但是拥有虚函数的类，声明时则自动在类中增加一个虚函数指针，该指针指向的是一个虚函数表，虚函数表中存着每个虚函数真正对应的函数地址。在调用虚函数时，首先去查虚函数表，然后在确定调用的是哪一个函数，所以，调用的函数是在运行时才会确定的。

在声明派生类对象时，虚函数指针指向的是派生类对应的虚函数表。在对象被创建之后(以指针为例)，无论是基类指针还是派生类指针指向这个对象，虚函数表是不会改变的，虚表指针的指向也是不会变的。

### 应用角度分类

多态从应用的角度可以分为两类：通用性多态和特定性多态。

两者的区别是前者用来系统地描述语义上相关的一组类型，对工作的类型不加限制，允许对不同类型的值执行相同的代码；后者用用来描述语义上无关联的类型间的关系，只对有限数量的类型有效，而且对不同类型的值可能要执行不同的代码。

通用性多态分为：参数多态(parametric)和包含多态(inclusion)；

特定性多态分为：重载多态(overloading)和强制多态(coercion)。

## 总结

从上面可看出，通用性多态性是真正的多态性,特定性多态性只是表面的多态性。因为重载只允许某一个符号有多种类型,而它所表示的值分别具有不同的类型。类似地,隐式类型转换也不是真正的多态，因为在运算开始前，各值必须转换为所要求的类型,而输出类型也与输入类型无关。相比之下,派生类与继承却是真正的多态，类型参数化也是一种纯正的多态，同一对象或函数在不同的类型上下文中统一使用而不需采用隐式类型转换、运行时检测或其它限制。

以上就是今天要讲的内容，后续会有更多内容。

## 参考资料

版权声明：本文参考了其他资料和CSDN博主的文章，遵循CC 4.0 BY-SA版权协议，现附上参考出处链接及本声明。

1. <https://blog.csdn.net/2301_78916684/article/details/139210176>
2. <https://blog.csdn.net/martian665/article/details/140962799>
3. <https://blog.csdn.net/qcyfred/article/details/53446940>
4. <https://blog.csdn.net/molangmolang/article/details/140754165>
5. <https://blog.csdn.net/2301_78029441/article/details/142362100>
6. https://blog.csdn.net/hackergin/article/details/73431401