第十三章

派生类和基类的特殊关系

基类指针可以在不进行显式类型转换的情况下指向派生类对象；

基类引用可以在不进行显示类型转化的情况下引用派生类对象；

引用兼容性属性也能够将基类对象初始化为派生类对象

C++有三种继承关系：公有继承、保护继承、私有继承

公有继承

最常用，它建立一种is-a关系，即派生类对象也是一个基类对象，可以对基类对象执行的任何操作， 也可以对派生类对象执行。

公有继承不建立has-a关系。不能通过从Fruit类派生出Lunch类在午餐中添加水果.

公有继承不能建立is-like-a 关系，继承可以在基类的基础上添加属性，但不能删除基类的属性。不能通过Shark派生出Lawyer类

公有继承不建立is-implemented-as-a(作为…来实现)关系可以通过数组来实现栈，但从Array类派生出Stack类是不合适的，因为栈不是数组

公有继承不建立uses-a关系，例如计算机可以使用激光打印机，但从Compuler类派生出Printer类是没有意义的

多态公有继承

希望同一个方法在派生类和基类中的行为是不同的。方法的行为应取决于调用该方法的对象。这种较复杂的行为成为多态——具有多种形态。

两种机制:

1、在派生类中重新定义基类的方法

2、使用虚方法

如果要在派生类重新定义基类的方法，通常应将基类方法声明为虚的。这样，程序将根据对象类型而不是引用或指针的类型来选择方法版本。为基类声明一个虚析构函数也是一种惯例

派生类并不能直接访问基类的私有数据，而必须使用基类的共有方法才能访问这些数据。访问的方法取决于方法。

有关虚函数：

在基类方法的声明中使用关键字virtual可使该方法在基类以及所有的派生类中是虚的

如果使用指向对象的引用或指针来调用虚方法，程序将使用为对象类型定义的方法，而不使用为引用或指针来调用虚方法。

如果定义的类将被用作基类，则应将那些要在派生类中重新定义的方法声明为虚的

抽象基类

用椭圆派生出圆将导致信息冗余。

解决方法：从Ellipsse和Circle类中抽象出他们的共性，将这些特性放到一个ABC中，然后从该ABC派生出Circle类。

Area()方法是两个类共有的，但是定义不同，不能在ABC中实现Area()方法，因为它没有包含必要的数据成员。C++通过使用纯虚函数提供未实现的函数。纯虚函数声明的结尾是=0；

当类声明中包含纯虚函数时，则不能创建该类对象。包含纯虚函数的类只用作基类。要成为真正的ABC，必须至少包含一个纯虚函数。

protected:

类中保护方法，派生类方法可以调用这些方法，但他们并不是派生类对象的公有接口的组成部分。

继承和动态内存分配

第一种情况：派生类不使用new

不需要为派生类重新定义显式析构函数、复制构造函数和赋值运算符

默认用到基类的时候，都会直接调用基类中定义的

第二种情况：派生类使用new

在这种情况下，必须为派生类定义显式析构函数、复制构造函数、赋值运算符。

派生类析构函数自动调用基类的析构函数，故其自身的职责是对派生类构造函数执行工作的进行清理。