← C++ 数据封装　　C++ 文件和流 →

C++ 接口（抽象类）

接口描述了类的行为和功能，而不需要完成类的特定实现。

C++ 接口是使用**抽象类**来实现的，抽象类与数据抽象互不混淆，数据抽象是一个把实现细节与相关的数据分离开的概念。

如果类中至少有一个函数被声明为纯虚函数，则这个类就是抽象类。纯虚函数是通过在声明中使用 "= 0" 来指定的，如下所示：

```
class Box
{
   public:
      // 纯虚函数
      virtual double getVolume() = 0;
   private:
      double length;      // 长度
      double breadth;     // 宽度
      double height;      // 高度
};
```

设计**抽象类**（通常称为 ABC）的目的，是为了给其他类提供一个可以继承的适当的基类。抽象类不能被用于实例化对象，它只能作为**接口**使用。如果试图实例化一个抽象类的对象，会导致编译错误。

因此，如果一个 ABC 的子类需要被实例化，则必须实现每个虚函数，这也意味着 C++ 支持使用 ABC 声明接口。如果没有在派生类中重写纯虚函数，就尝试实例化该类的对象，会导致编译错误。

可用于实例化对象的类被称为**具体类**。

抽象类的实例

请看下面的实例，基类 Shape 提供了一个接口 **getArea()**，在两个派生类 Rectangle 和 Triangle 中分别实现了 **getArea()**：

实例

```
#include <iostream>

using namespace std;

// 基类
class Shape
{
public:
   // 提供接口框架的纯虚函数
   virtual int getArea() = 0;
   void setWidth(int w)
```

分类导航

HTML / CSS
JavaScript
服务端
数据库
移动端
XML 教程
ASP.NET
Web Service
开发工具
网站建设

```
   {
      width = w;
   }
   void setHeight(int h)
   {
      height = h;
   }
protected:
   int width;
   int height;
};

// 派生类
class Rectangle: public Shape
{
public:
   int getArea()
   {
      return (width * height);
   }
};
class Triangle: public Shape
{
public:
   int getArea()
   {
      return (width * height)/2;
   }
};

int main(void)
{
   Rectangle Rect;
   Triangle  Tri;

   Rect.setWidth(5);
   Rect.setHeight(7);
   // 输出对象的面积
   cout << "Total Rectangle area: " << Rect.getArea() << endl;

   Tri.setWidth(5);
   Tri.setHeight(7);
   // 输出对象的面积
   cout << "Total Triangle area: " << Tri.getArea() << endl;

   return 0;
}
```

当上面的代码被编译和执行时，它会产生下列结果：

```
Total Rectangle area: 35
Total Triangle area: 17
```

从上面的实例中，我们可以看到一个抽象类是如何定义一个接口 getArea()，两个派生类是如何通过不同的计算面积的算法来实现这个相同的函数。

设计策略

面向对象的系统可能会使用一个抽象基类为所有的外部应用程序提供一个适当的、通用的、标准化的接口。然后，派生类通过继承抽象基类，就把所有类似的操作都继承下来。

外部应用程序提供的功能（即公有函数）在抽象基类中是以纯虚函数的形式存在的。这些纯虚函数在相应的派生类中被实现。

这个架构也使得新的应用程序可以很容易地被添加到系统中，即使是在系统被定义之后依然可以如此。

← C++ 数据封装　　C++ 文件和流 →

点我分享笔记

反馈/建议