首页 HTML CSS JAVASCRIPT JQUERY BOOTSTRAP PYTHON3 PYTHON2 JAVA C C++

C++ 教程 (

C++ 教程

C++ 简介

C++ 环境设置

C++ 基本语法

C++ 注释

C++ 数据类型

C++ 变量类型

C++ 变量作用域

C++ 常量

C++ 修饰符类型

C++ 存储类

C++ 运算符

C++ 循环

C++ 判断

C++ 函数

C++ 数字

C++ 数组

C++ 字符串

C++ 指针

C++ 引用

C++ 日期 & 时间

C++ 基本的输入 输出

C++ 数据结构

C++ 面向对象

C++ 类 & 对象

C++ 继承

C++ 重载运算符 和重载函数

C++ 多态

C++ 数据抽象

C++ 数据封装

← C++ 多态

C++ 数据封装 →

C++ 数据抽象

数据抽象是指,只向外界提供关键信息,并隐藏其后台的实现细节,即只表现必要的信息而不呈现细节。

数据抽象是一种依赖于接口和实现分离的编程(设计)技术。

让我们举一个现实生活中的真实例子,比如一台电视机,您可以打开和关闭、切换频道、调整音量、添加外部组件(如喇叭、录像机、DVD播放器),但是您不知道它的内部实现细节,也就是说,您并不知道它是如何通过缆线接收信号,如何转换信号,并最终显示在屏幕上。

因此,我们可以说电视把它的内部实现和外部接口分离开了,您无需知道它的内部实现原理,直接通过它的外部接口(比如电源按钮、遥控器、声量控制器)就可以操控电视。

现在,让我们言归正传,就 C++ 编程而言,C++ 类为**数据抽象**提供了可能。它们向外界提供了大量用于操作对象数据的公共方法,也就是说,外界实际上并不清楚类的内部实现。

例如,您的程序可以调用 **sort()** 函数,而不需要知道函数中排序数据所用到的算法。 实际上,函数排序的底层实现会因库的版本不同而有所差异,只要接口不变,函数调 用就可以照常工作。

在 C++ 中,我们使用**类**来定义我们自己的抽象数据类型(ADT)。您可以使用类 **iost** ream 的 cout 对象来输出数据到标准输出,如下所示:

实例

```
#include <iostream>
using namespace std;

int main()
{
   cout << "Hello C++" <<endl;
   return 0;
}</pre>
```

在这里,您不需要理解 cout 是如何在用户的屏幕上显示文本。您只需要知道公共接口即可,cout 的底层实现可以自由改变。

访问标签强制抽象

在 C++ 中,我们使用访问标签来定义类的抽象接口。一个类可以包含零个或多个访问标签:

使用公共标签定义的成员都可以访问该程序的所有部分。一个类型的数据抽象视图是由它的公共成员来定义的。

Ⅲ 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

Advertisement



믦



反馈/建议

C++ 接口 (抽象 类)

C++ 高级教程

C++ 文件和流

C++ 异常处理

C++ 动态内存

C++ 命名空间

C++ 模板

C++ 预处理器

C++ 信号处理

C++ 多线程

C++ Web 编程

C++ 资源库

C++ STL 教程

C++ 标准库

C++ 有用的资源

C++ 实例

使用私有标签定义的成员无法访问到使用类的代码。私有部分对使用类型的代码 隐藏了实现细节。

访问标签出现的频率没有限制。每个访问标签指定了紧随其后的成员定义的访问级别。指定的访问级别会一直有效,直到遇到下一个访问标签或者遇到类主体的关闭右括号为止。

数据抽象的好处

数据抽象有两个重要的优势:

类的内部受到保护,不会因无意的用户级错误导致对象状态受损。

类实现可能随着时间的推移而发生变化,以便应对不断变化的需求,或者应对那 些要求不改变用户级代码的错误报告。

如果只在类的私有部分定义数据成员,编写该类的作者就可以随意更改数据。如果实现发生改变,则只需要检查类的代码,看看这个改变会导致哪些影响。如果数据是公有的,则任何直接访问旧表示形式的数据成员的函数都可能受到影响。

数据抽象的实例

C++ 程序中,任何带有公有和私有成员的类都可以作为数据抽象的实例。请看下面的实例:

实例

```
#include <iostream>
using namespace std;
class Adder{
   public:
      // 构造函数
     Adder(int i = 0)
        total = i;
      // 对外的接口
      void addNum(int number)
          total += number;
      // 对外的接口
      int getTotal()
          return total;
      };
   private:
     // 对外隐藏的数据
      int total;
};
int main( )
{
   Adder a;
   a.addNum(10);
   a.addNum(20);
  a.addNum(30);
   cout << "Total " << a.getTotal() <<endl;</pre>
```

python教 python入 到精通

授课模式· 在结 +课后录 内容包含 人工智能 栈+pyth +』





return 0;

当上面的代码被编译和执行时,它会产生下列结果:

Total 60

}

上面的类把数字相加,并返回总和。公有成员 addNum 和 getTotal 是对外的接口, 用户需要知道它们以便使用类。私有成员 total 是用户不需要了解的,但又是类能正常 工作所必需的。

设计策略

抽象把代码分离为接口和实现。所以在设计组件时,必须保持接口独立于实现,这 样,如果改变底层实现,接口也将保持不变。

在这种情况下,不管任何程序使用接口,接口都不会受到影响,只需要将最新的实现 重新编译即可。

◆ C++ 多态

在线实例

C++ 数据封装 →

② 点我分享笔记

具 · HTML 实例 · CSS 实例 集设置 JavaScript · HTML 实例 ASCII 字符集 · Ajax 实例 8859-1 · jQuery 实例 · XML 实例 符号 · Java 实例 · JSON 格式 化工具

· HTML 字符 · HTML ISO-· HTML 实体 · HTML 拾色

字符集&工

最新更新 · Python redis 使... Windows10 MYSQ... · Docke 镜 像加速 · Debian Docker 安装 · C 库函数

· Linux

groupadd ...

· CSS var() 函数

站点信息

- · 意见反馈
- · 合作联系
- · 免责声明
- · 关于我们
- · 文章归档

关注微信



Copyright © 2013-2019 菜鸟教程 runoob.com All Rights Reserved. 备案号:闽ICP备15012807号-1



