

☰ C++ 教程 🌙

C++ 教程

C++ 简介

C++ 环境设置

C++ AI 编程助手

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++ 数据结构

C++ 继承 →

C++ 类 & 对象

C++ 在 C 语言的基础上增加了面向对象编程，C++ 支持面向对象程序设计。类是 C++ 的核心特性，通常被称为用户定义的类型。

类用于指定对象的形式，是一种用户自定义的数据类型，它是一种封装了数据和函数的组合。类中的数据称为成员变量，函数称为成员函数。类可以被看作是一种模板，可以用来创建具有相同属性和行为的多个对象。

C++ 类定义

定义一个类需要使用关键字 `class`，然后指定类的名称，并类的主体是包含在一对花括号中，主体包含类的成员变量和成员函数。

定义一个类，本质上是定义一个数据类型的蓝图，它定义了类的对象包括了什么，以及可以在这个对象上执行哪些操作。

☰ 分类导航

HTML / CSS

JavaScript

服务端

数据库

数据分析

移动端

XML 教程

ASP.NET

Web Service

开发工具

网站建设



反馈/建议

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++ 结构体(struct)

C++ vector 容器

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++ 接口（抽象类）

C++ 高级教程



The diagram illustrates the syntax of a C++ class definition with the following components and annotations:

- 关键字** (Keyword): Points to the `class` keyword.
- 类名** (Class Name): Points to the `classname` identifier.
- Access specifiers:** Points to the `private/public/protected` access modifiers.
- // 访问修饰符: private/public/protected**: Annotation for the access specifiers.
- Date members/variables;**: Points to the member variables.
- // 变量** (Variable): Annotation for the member variables.
- Member functions() {}**: Points to the member functions.
- // 方法** (Method): Annotation for the member functions.
- };**: Points to the closing brace and semicolon.
- // 分号结束一个类** (Semicolon ends a class): Annotation for the closing brace and semicolon.

以下实例我们使用关键字 **class** 定义 Box 数据类型，包含了三个成员变量 `length`、`breadth` 和 `height`：

```
class Box
{
    public:
        double length;    // 盒子的长度
        double breadth;   // 盒子的宽度
        double height;    // 盒子的高度
};
```

关键字 **public** 确定了类成员的访问属性。在类对象作用域内，公共成员在类的外部是可访问的。您也可以指定类的成员为 **private** 或 **protected**，这个我们稍后会进行讲解。



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
C++ 测验
C++ <iostream>
C++ <fstream>
C++ <sstream>
C++ <iomanip>
C++ <array>
C++ <vector>

定义 C++ 对象

类提供了对象的蓝图，所以基本上，对象是根据类来创建的。声明类的对象，就像声明基本类型的变量一样。

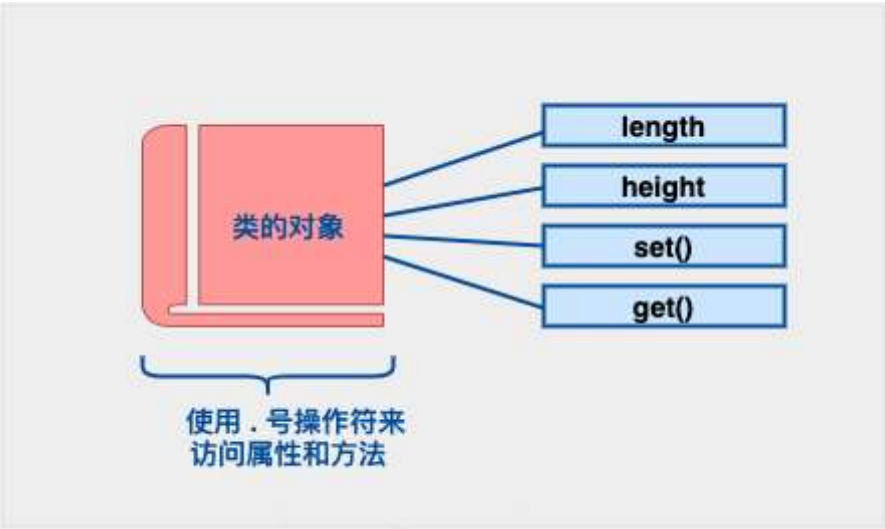
下面的语句声明了类 Box 的两个对象：

```
Box Box1;           // 声明 Box1，类型为 Box
Box Box2;           // 声明 Box2，类型为 Box
```

对象 Box1 和 Box2 都有它们各自的数据成员。

访问数据成员

类的对象的公共数据成员可以使用直接成员访问运算符 `.` 来访问。



为了更好地理解这些概念，让我们尝试一下下面的实例：

实例

```
#include <iostream>

using namespace std;
```

C++ <list>

C++ <forward_list>

C++ <deque>

C++ <stack>

C++ <queue>

C++ <priority_queue>

C++ <set>

C++ <unordered_set>

C++ <map>

C++ <unordered_map>

C++ <bitset>

C++ <algorithm>

C++ <iterator>

C++ <functional>

C++ <numeric>

C++ <complex>

C++ <valarray>

C++ <cmath>

C++ <string>

C++ <regex>

C++ <ctime>

```
class Box
{
    public:
        double length;    // 长度
        double breadth;    // 宽度
        double height;    // 高度
        // 成员函数声明
        double get(void);
        void set( double len, double bre, double hei );
};
// 成员函数定义
double Box::get(void)
{
    return length * breadth * height;
}

void Box::set( double len, double bre, double hei)
{
    length = len;
    breadth = bre;
    height = hei;
}

int main( )
{
    Box Box1;           // 声明 Box1, 类型为 Box
    Box Box2;           // 声明 Box2, 类型为 Box
    Box Box3;           // 声明 Box3, 类型为 Box
    double volume = 0.0;    // 用于存储体积

    // box 1 详述
    Box1.height = 5.0;
    Box1.length = 6.0;
    Box1.breadth = 7.0;

    // box 2 详述
    Box2.height = 10.0;
    Box2.length = 12.0;
    Box2.breadth = 13.0;
```



C++ <chrono>
C++ <thread>
C++ <mutex>
C++ <condition_variable>
C++ <future>
C++ <atomic>
C++ <type_traits>
C++ <typeinfo>
C++ <exception>
C++ <stdexcept>
C++ <cstdio>
C++ <cstdlib>
C++ <memory>
C++ <new>
C++ <utility>
C++ <random>
C++ <locale>
C++ <codecvt>
C++ <cassert>
C++ <wchar>
C++ <limits>

```
// box 1 的体积
volume = Box1.height * Box1.length * Box1.breadth;
cout << "Box1 的体积: " << volume <<endl;

// box 2 的体积
volume = Box2.height * Box2.length * Box2.breadth;
cout << "Box2 的体积: " << volume <<endl;

// box 3 详述
Box3.set(16.0, 8.0, 12.0);
volume = Box3.get();
cout << "Box3 的体积: " << volume <<endl;
return 0;
}
```

当上面的代码被编译和执行时，它会产生下列结果：

Box1 的体积：210
Box2 的体积：1560
Box3 的体积：1536

需要注意的是，私有的成员和受保护的成员不能使用直接成员访问运算符 (.) 来直接访问。我们将在后续的教程中学习如何访问私有成员和受保护的成员。

类 & 对象详解

到目前为止，我们已经对 C++ 的类和对象有了基本的了解。下面的列表中还列出了其他一些 C++ 类和对象相关的概念，可以点击相应的链接进行学习。

概念	描述
类成员函数	类的成员函数是指那些把定义和原型写在类定义内部的函数，就像类定义中的其他变量一样。



类访问修饰符	类成员可以被定义为 public、private 或 protected。默认情况下是定义为 private。
构造函数 & 析构函数	类的构造函数是一种特殊的函数，在创建一个新的对象时调用。 类的析构函数也是一种特殊的函数，在删除所创建的对象时调用。
C++ 拷贝构造函数	拷贝构造函数，是一种特殊的构造函数，它在创建对象时，是使用同一类中之前创建的对象来初始化新创建的对象。
C++ 友元函数	友元函数 可以访问类的 private 和 protected 成员。
C++ 内联函数	通过内联函数，编译器试图在调用函数的地方扩展函数体中的代码。
C++ 中的 this 指针	每个对象都有一个特殊的指针 this ，它指向对象本身。
C++ 中指向类的指针	指向类的指针方式如同指向结构的指针。实际上，类可以看成是一个带有函数的结构。
C++ 类的静态成员	类的数据成员和函数成员都可以被声明为静态的。

[← C++ 数据结构](#)[C++ 继承 →](#)

6 篇笔记



写笔记



在线实例

- [HTML 实例](#)
- [CSS 实例](#)
- [JavaScript 实例](#)
- [Ajax 实例](#)
- [jQuery 实例](#)
- [XML 实例](#)
- [Java 实例](#)

字符集&工具

- [HTML 字符集设置](#)
- [HTML ASCII 字符集](#)
- [JS 混淆/加密](#)
- [PNG/JPEG 图片压缩](#)
- [HTML 拾色器](#)
- [JSON 格式化工具](#)
- [随机数生成器](#)

最新更新

- [PyTorch 数据处...](#)
- [PyTorch 神经网络...](#)
- [PyTorch 基础](#)
- [PyTorch 安装](#)
- [PyTorch 简介](#)
- [PyTorch 教程](#)
- [Tailwind CSS 布...](#)

站点信息

- [意见反馈](#)
- [免责声明](#)
- [关于我们](#)
- [文章归档](#)

关注微信

