

## 风云逸

≤	2018年9月							≥
日	一	二	三	四	五	六		
26	27	28	29	30	31	1		
2	3	4	5	6	7	8		
9	10	11	12	13	14	15		
16	17	18	19	20	21	22		
23	24	25	26	27	28	29		
30	1	2	3	4	5	6		

昵称: [风云逸](#)  
园龄: [4年10个月](#)  
粉丝: [18](#)  
关注: [2](#)  
[+加关注](#)

## 搜索

<input type="text"/>	找找看
<input type="text"/>	谷歌搜索

## 常用链接

[博客园](#) [首页](#) [新随笔](#) [联系](#) [订阅](#) [XML](#) [管理](#)

posts - 0, comments - 0, trackbacks - 0

## C++中struct和class的区别

有很多人在学习C++的时候只是了解了C++的语法，或者更高层次的人是理解了oo（面向对象），但在这样的学习过程中，往往一些最基本的问题反而被忽略了。当然，这些问题或许在你做工程中，不会有太大的影响，只要按照平时一贯的良好编码习惯，依旧可以写出好的代码；但某些时候，或许就是这些最基本的小问题，会让你的程序BUG难以发现，而注意到某些问题后，同时还将会提升你程序的性能。所以，还是让我们把这些最基本的问题——拾起，了解总比不了解好。

我原本打算用最俗的方式列出10个最基本的问题来排个TOP10的，后来发现每个问题要深入讨论的东西太多，于是就此打住，还是一个一个的来慢慢讨论，会更加清晰明了。

那么，最开始，就让我们来讨论一下一个最最基本，也最最容易被人忽视掉的问题——C++中的struct和class有什么区别？

如果谈到C中的struct和C++中的class的区别，你应该会告诉我很多。但我现在说的是C++中的struct，你还会依然那样告诉我吗？你会认为C中的struct和C++中的struct是一样的吗？

被我这样问道，或许你会吱吱呜呜的说：不一样吧。的确，是不一样，那么区别在哪里？

[我的随笔](#)[我的评论](#)[我的参与](#)[最新评论](#)[我的标签](#)

## 文章分类

[Android\(2\)](#)

其实，C和C++这两种语言，除了语法上相似，其理念是完全不同的。C++最初的想法就是对C进行扩充——“a better c”，但事实上，这样的“扩充”已经不能再称之为扩充了，我更愿意把C++当成是一种新的语言，而不仅仅是扩充。又或许，C++和C最大的关系，只是他们的名字，如果C++不叫C++，而叫D++，你可能就不会将它们俩的关系想得那么的紧密了。当然，这些话只是调侃，C++的确是在C的基础上发展起来的。

我之所以提到理念不同，关键就是指oo，这思想对整个软件编程的冲击太大，所以我会说C++更像是一种新的语言。

说了这么多废话，我们还是回到我们讨论的问题上来。

C++中的struct对C中的struct进行了扩充，它已经不再只是一个包含不同数据类型的数据结构了，它已经获取了太多的功能。

struct能包含成员函数吗？能！

struct能继承吗？能！！

struct能实现多态吗？能！！！

有很多人应该已经知道这样一个事实，但总有些不知道的人，看到这些会感到很惊讶。是的，当我第一次注意到这个事实的时候，我也同样很吃惊。

既然这些它都能实现，那它和class还能有什么区别？

最本质的一个区别就是默认的访问控制，体现在两个方面：

1) 默认的继承访问权限。struct是public的，class是private的。

如果不知道什么是public继承，什么是private继承的，可以去查书，这里暂不讨论。

你可以写如下的代码：

```
struct A
{
    char a;
};

struct B : A
{
    char b;
```

```
};
```

这个时候B是public继承A的。如果都将上面的struct改成class，那么B是private继承A的。这就是默认的继承访问权限。所以我们在平时写类继承的时候，通常会这样写：

```
struct B : public A
```

就是为了指明是public继承，而不是用默认的private继承。

当然，到底默认是public继承还是private继承，取决于子类而不是基类。我的意思是，struct可以继承class，同样class也可以继承struct，那么默认的继承访问权限是看子类到底是用的struct还是class。如下：

```
struct A{};
```

```
class B : A{}; //private继承
```

```
struct C : B{}; //public继承
```

2) struct作为数据结构的实现体，它默认的数据访问控制是public的，而class作为对象的实现体，它默认的成员变量访问控制是private的。

注意我上面的用词，我依旧强调struct是一种数据结构的实现体，虽然它是可以像class一样的用。我依旧将struct里的变量叫数据，class内的变量叫成员，虽然它们并无区别。其实，到底是用struct还是class，完全看个人的喜好，你可以将你程序里所有的class全部替换成struct，它依旧可以很正常的运行。但我给出的最好建议，还是：当你觉得你要做的更像是一种数据结构的话，那么用struct，如果你要做的更像是一种对象的话，那么用class。

当然，我在这里还要强调一点的就是，对于访问控制，应该在程序里明确的指出，而不是依靠默认，这是一个良好的习惯，也让你的代码更具可读性。

说到这里，很多了解的人或许都认为这个话题可以结束了，因为他们知道struct和class的“唯一”区别就是访问控制。很多文献上也确实只提到这一个区别。

但我上面却没有用“唯一”，而是说的“最本质”，那是因为，它们确实还有另一个区别，虽然那个区别我们平时可能很少涉及。那就是：“class”这个关键字还用于定义模板参数，就像“typename”。但关键字“struct”不用于定义模板参数。这一点在Stanley B.Lippman写的Inside the C++ Object Model有过说明。

问题讨论到这里，基本上应该可以结束了。但有人曾说过，他还发现过其他的“区别”，那么，让我们来看看，这到底是不是又一个区别。

还是上面所说的，C++中的struct是对C中的struct的扩充，既然是扩充，那么它就要兼容过去C中struct应有的所有特性。例如你可以这样写：

```
struct A //定义一个struct  
{  
    char c1;  
    int n2;  
    double db3;  
};  
A a={'p',7,3.1415926}; //定义时直接赋值
```

也就是说struct可以在定义的时候用{}赋初值。那么问题来了，class行不行呢？将上面的struct改成class，试试看。报错！噢~于是那人跳出来说，他又找到了一个区别。我们仔细看看，这真的又是一个区别吗？

你试着向上面的 struct中加入一个构造函数（或虚函数），你会发现什么？对，struct也不能用{}赋初值了。的确，以{}的方式来赋初值，只是用一个初始化列表 来对数据进行按顺序的初始化，如上面如果写成A a={'p',7};则c1,n2被初始化，而db3没有。这样简单的copy操作，只能发生在简单的数据结构上，而不应该放在对象上。加入一个构造函数 或是一个虚函数会使struct更体现出一种对象的特性，而使此{}操作不再有效。事实上，是因为加入这样的函数，使得类的内部结构发生了变化。而加入一个普通的成员函数呢？你会发现{}依旧可用。其实你可以将普通的函数理解成对数据结构的一种算法，这并不打破它数据结构的特性。至于虚函数和普通成员函数 有什么区别，我会具体写篇文章讨论。

那么，看到这里，我们发现即使是struct想用{}来赋初值，它也必须满足很多的约束条件，这些条件实际上就是让struct更体现出一种数据机构而不是类的特性。那为什么我们在上面仅仅将struct改成class，{}就不能用了呢？其实问题恰巧是我们之前所讲的——访问控制！你看看，我们忘记了什么？对，将struct 改成class的时候，访问控制由public变为private了，那当然就不能用{}来赋初值了。加上一个public，你会发现，class也是能用{}的，和struct毫无区别！！

做个总结，从上面的区别，我们可以看出，struct更适合看成是一个数据结构的实现体，class更适合看成是一个对象的实现体。所以我会提出什么时候用struct什么时候用class的建议。如果你有不同的看法，欢迎讨论。

好文要顶

关注我

收藏该文



友

风云逸关注 - 2粉丝 - 18

8

0