请你来说一下C++中类成员的访问权限

参考回答：C++通过 public、protected、private 三个关键字来控制成员变量和成员函数的访问权限，它们分别表示公有的、受保护的、私有的，被称为成员访问限定符。在类的内部（定义类的代码内部），无论成员被声明为 public、protected 还是 private，都是可以互相访问的，没有访问权限的限制。在类的外部（定义类的代码之外），只能通过对象访问成员，并且通过对象只能访问 public 属性的成员，不能访问 private、protected 属性的成员

访问控制

	public	**protected**	**private**
类成员是否可以访问	Yes	Yes	Yes
友元函数是否可以访问	Yes	Yes	Yes
子类是否可以访问	Yes	Yes	No
类的实例化对象是否可以访问	Yes	No	No

继承

	public	**protected**	**private**
public继承	public	protected	private
protected继承	protected	protected	private
private继承	private	private	private

● 请你来说一下C++中struct和class的区别

在C++中，可以用struct和class定义类，都可以继承。区别在于：struct的默认继承权限和默认访问权限是public，而class的默认继承权限和默认访问权限是private。

另外，class还可以定义模板类形参，比如template <class T, int i>。

参考网址

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

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。

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。

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。

● 请你回答一下C++类内可以定义引用数据成员吗?

可以，必须通过成员函数初始化列表初始化。