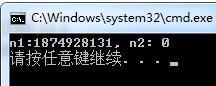
问题来源：

由于面试题中，考官出了一道简单的程序输出结果值的题：如下，

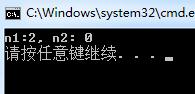
1. **class A**
2. {
3. **private:**
4. int n1;
5. int n2;
7. **public:**
8. A():n2(0),n1(n2+2){}
10. **void Print(){**
11. cout << "n1:" << n1 << ", n2: " << n2 <<endl;
12. }
13. };
15. int main()
16. {
18. A a;
19. a.Print();
21. **return 1;**
22. }

这时，那个考生这样回答：n1是2，n2是0。  
在我电脑输出结果为：

如果你也这样回答，那么你肯定不懂初始化成员列表的顺序。

如果我把A类中构造函数改成：

1. A()
2. {
3. n2 = 0;
4. n1 = n2 +2;
5. }

那么此时输出结果为：

分析：

1、成员变量在使用**初始化列表初始化时**，与构造函数中初始化成员列表的顺序无关，只**与定义成员变量的顺序有关**。因为成员变量的初始化次序是根据变量在内存中次序有关，而内存中的排列顺序早在编译期就根据变量的定义次序决定了。这点在EffectiveC++中有详细介绍。

2、如果不使用初始化列表初始化，在**构造函数内初始化时**，此时与成员变量在**构造函数中的位置有关**。

3、注意：类成员在定义时，是不能初始化的

4、注意：类中const成员常量必须在构造函数初始化列表中初始化。

5、注意：类中static成员变量，必须在类外初始化。

6、静态变量进行初始化顺序是基类的静态变量先初始化，然后是它的派生类。直到所有的静态变量都被初始化。这里需要注意全局变量和静态变量的初始化是不分次序的。这也不难理解，其实静态变量和全局变量都被放在公共内存区。可以把静态变量理解为带有“作用域”的全局变量。在一切初始化工作结束后，main函数会被调用，如果某个类的构造函数被执行，那么首先基类的成员变量会被初始化。 

* bbb的成员变量定义：
* private:
  + int n1;
  + int n2;
* bbb的构造函数：
* bbb::bbb()
* :n2(1),
* n1(2)
* {
* }
* 汇编代码：
* 00401535 mov eax,dword ptr [ebp-4]
* 00401538 mov dword ptr [eax+4],2
* 0040153F mov ecx,dword ptr [ebp-4]
* 00401542 mov dword ptr [ecx+8],1
* 然后依照派生链初始化派生类的成员函数。

.总结：

   变量的初始化顺序就应该是：

* 1 基类的静态变量或全局变量
* 2 派生类的静态变量或全局变量
* 3 基类的成员变量
* 4 派生类的成员变量