**C++中可以在构造函数中调用另一个构造函数吗**

题目如下：问下列代码的打印结果为0吗？

**[cpp]** [view plain](http://blog.csdn.net/race604/article/details/6921678) [copy](http://blog.csdn.net/race604/article/details/6921678)

1. #include <stdlib.h>
2. #include <iostream>
3. **using** **namespace** std;
5. **struct** CLS
6. {
7. **int** m\_i;
8. CLS( **int** i ) : m\_i(i){}
9. CLS()
10. {
11. CLS(0);
12. }
13. };
14. **int** main()
15. {
16. CLS obj;
17. cout << obj.m\_i << endl;
19. system("PAUSE");
20. **return** 0;
21. }

打印结果是不定的，不一定为0  
  
代码奇怪的地方在于构造函数中调用了自己的另一个构造函数  
  
我们知道，当定义一个对象时，会按顺序做2件事情：  
1）分配好内存（非静态数据成员是未初始化的）  
2）调用构造函数（构造函数的本意就是初始化非静态数据成员）  
  
显然上面代码中，CLS obj；这里已经为obj分配了内存，然后调用默认构造函数，但是默认构造函数还未执行完，却调用了另一个构造函数，这样相当于产生了一个匿名的临时CLS对象，它调用CLS(int)构造函数，将这个匿名临时对象自己的数据成员m\_i初始化为0；但是obj的数据成员并没有得到初始化。于是obj的m\_i是未初始化的，因此其值也是不确定的  
  
从这里，我们归纳如下：  
1）在c++里，由于构造函数允许有默认参数，使得这种构造函数调用构造函数来重用代码的需求大为减少  
2）如果仅仅为了一个构造函数重用另一个构造函数的代码，那么完全可以把构造函数中的公共部分抽取出来定义一个成员函数(推荐为private),然后在每个需要这个代码的构造函数中调用该函数即可  
3）偶尔我们还是希望在类的构造函数里调用另一个构造函数，可以按下面方式做：  
**在构造函数里调用另一个构造函数的关键是让第二个构造函数在第一次分配好的内存上执行，而不是分配新的内存，这个可以用标准库的placement new做到**：  
先看看标准库中placement new的定义

**[cpp]** [view plain](http://blog.csdn.net/race604/article/details/6921678) [copy](http://blog.csdn.net/race604/article/details/6921678)

1. **inline** **void** \*\_\_cdecl operator **new**(**size\_t**, **void** \*\_P)
2. {
3. **return** (\_P);
4. }

可见没有分配新的内存。  
  
正确的方式：

**[cpp]** [view plain](http://blog.csdn.net/race604/article/details/6921678) [copy](http://blog.csdn.net/race604/article/details/6921678)

1. **struct** CLS
2. {
3. **int** m\_i;
4. CLS( **int** i ) : m\_i(i){}
5. CLS()
6. {
7. **new** (**this**)CLS(0);
8. }
9. };

另： **若构造函数调用自身，则会出现无限递归调用，是不允许的。**

--------结束转载------------

所以，在实际使用的时候，**单纯的在构造函数中调用其它的构造函数，只是会产生一个临时的匿名变量**。如果仅仅是为了重用代码，可以把重用的代码封装成一个新的函数。