CMemoryState类没有基类。  
CMemoryState提供了便利的方式以检测程序中的“内存空洞”。“内存空洞”发生于已经为一个对象分配了内存，而在长期不再需要时也没有释放。这样的内存空洞最终可能导致内存溢出错误。有几种方式来为程序分配或重新分配内存：

|  |  |
| --- | --- |
| · | 使用运行库中的函数的malloc/free家族。 |
| · | 使用Windows API内存管理函数LocalAlloc/LocalFree和GlobalAlloc/Global Free。 |
| · | 使用C++ new和delete操作。 |

CMemoryState 诊断只能用于帮助检测内存空洞，该空洞的产生是由于使用new操作分配了内存而没有使用delete进行删除。另外两组内存管理函数是针对非C++程序的，不推荐在同一程序中使用new和delete的混合。当需要文件及内存分配的行跟踪，那么提供一个附加的宏DEBUG\_NEW来替换new操作。通常，不管是否使用了new操作，都需要使用DEBUG\_NEW宏。  
与其它诊断相同，CMemoryState诊断也仅仅在程序的调试版中有效。  
调试版必须有\_DEBUG常量定义。  
如果怀疑程序中有内存空洞，那么可以使用Checkpoint, Difference及DumpStatistics 函数来找出程序执行完成后，两个不同点之间内存状态（分配内存的对象）的不同。该信息对于决定函数是否清除了所有分配内存的对象非常有用。  
如果仅仅知道当内存分配或内存释放时不均衡发生于何处，它并没有提供足够的信息，因此可以使用DumpAllObjectSince函数来转储自从前一次调用Checkpoint函数以来分配的所有对象。这种转储显示了内存分配次序，源文件及被分配对象所在的行（条件是使用了DEBUG\_NEW）以及派生对象、其地址、大小。DumpAllObjectSince对象也调用了每一个对象的Dump函数以提供有关的信息。  
如果要了解有关如何使用CMemoryState和其它诊断的信息，请参阅联机文档“Visual C++程序员指南”中的“MFC调试支持”。  
注意：  
CMemoryState类型的对象声明以及成员函数的调用必须用#ifdefined (\_DEBUG)/#endif括起来，以便仅在程序的调试版本中包含内存诊断。