<https://www.2cto.com/kf/201108/98926.html>

持久化的对象，是已经存储到[数据库](https://www.2cto.com/database/)或保存到本地硬盘中的对象，我们称之为持久化对象。下面介绍C++中的持久化对象。

　　持久对象（persistent objects）广泛应用于游戏、分布式数据库[系统](https://www.2cto.com/os/)、多媒体以及图形应用程序中。目前C++并不直接支持持久性（persistence）（但有一些在C++未来版本中添加持久性和反射（reflection）的建议）。

　　持久对象可以在创建它的程序的作用域之外保持自身状态。把对象写入一个文件并在以后重建之，或者把对象传送到一台远程机器，就是这样的例子。对持久性的支持并不象第一眼看上去那样简单，同一对象的大小和内存布局在不同的平台上可能并不相同，而不同的字节次序（byte ordering），或称为endian-ness，使事情更加复杂化。

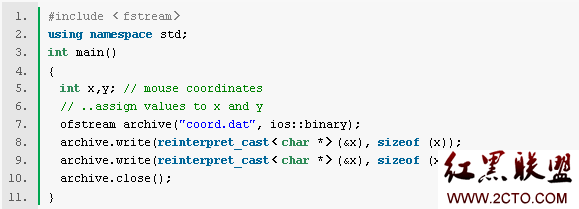
　　在下文中我将讨论如何实现持久性，而无须求助于DCOM和CORBA之类的第三方框架。对于小型和可移植的应用程序而言，这是一种有效并令人满意的方案。

　　序列化（serialization）基础

　　为了使一个对象持久存在，必须把它的状态保存在非易失的存储设备中。考虑一个录制和播放MP3文件的应用程序，每首单曲都表示为一个包含标题、唱片、歌手、时间、速率、录制日期以及相应的MP3文件的对象，该应用程序在跟踪列表中显示最近播放的曲目。你的目标是通过序列化，也就是把对象写入一个文件，使MP3对象成为持久对象，同时通过反序列化（deserialization）在下一个session中重建这些对象。

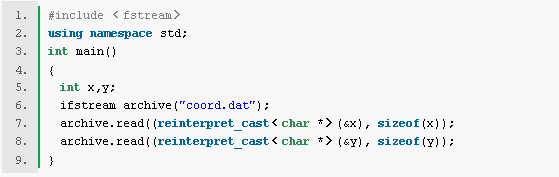
　　序列化内置数据类型

　　每个对象最终都由内置数据成员组成，如int, bool, char[]等等。你的第一个任务是把这样的类型写入一个输出文件流（ofstream）中。应用程序必须这些值存储为相应的二进制形式，基于这个目的，应使用write() 和read() 成员函数。write() 以某个变量的地址和大小为参数，把该变量的位模式写入一个文件流中。read() 的两个参数为char\*和long类型，分别指示内存缓冲区的地址和字节大小。下面的例子演示如何在ofstream中保存两个整数：



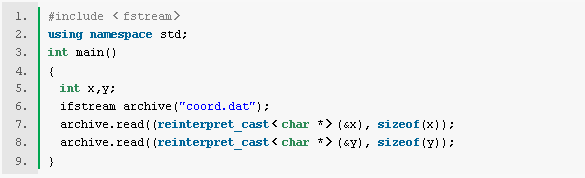
　　使用reinterpret\_cast＜＞是必要的，因为write()的第一个参数类型为const char\*，但&x和&y是int\*类型。

　　以下代码读取刚才存储的值：



　　序列化对象

　　要序列化一个完整的对象，应把每个数据成员写入文件中：

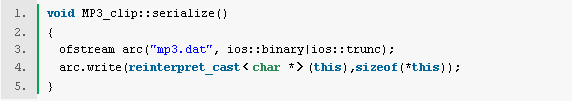


实现deserialize() 需要一些技巧，因为你需要为字符串分配一个临时缓冲区。做法如下：



性能优化

　　你可能会感到迷惑，为什么不把整个对象一次性转储到文件中，而必须对每个数据成员进行序列化呢？换句话说，难道不能用下面的方式实现serialize() 吗？



　　不行，不能这样做。这种方式至少存在两个问题。通常，当被序列化的对象还包含其它一些对象时，你不能简单地把该对象转储到一个文件中并指望以后从中重建一个有效的对象。在我们的例子中，外层对象包含一个std::string成员，一个浅拷贝（shallow copy）操作会把std::string成员归档，但其值是时变的，意思是说每次运行程序时都可能改变。

　　更糟的是，由于std::string事实上并不包含一个字符数组，而是一个指针，使用浅拷贝试图重建原始字符串是不可能的。为克服这个问题，程序没有序列化string对象，而是归档其含有的字符和长度。一般来说，指针，数组和句柄应以相同的方式进行处理。

　　另一个问题设计到多态对象。每个多态对象都含有一个vtpr，即一个指向虚拟函数地址分配表的隐藏指针。vtpr的值是时变的，如果你把整个多态对象转储到一个文件中，然后强行把归档后的数据添加到一个新的对象上，则其vptr可能无效并导致未定义的行为。再次提醒，解决方案是只对非时变的数据成员进行序列化和反序列化。另一种方法是计算vptr的确切偏移量，在从文件重建对象时不要动它。记住，vptr的位置是与实现相关的，因此这样的代码是不可移植的。

　　小结

　　虽然C++不直接支持对象持久性，但手工实现它并不难，只要你遵从一些基本的准则：首先把每个复合对象分解为原始数据类型，然后对这些原始数据类型进行序列化。当序列化数据时，记住要跳过时变的值。在反序列化过程中，读取刚才存储的值。处理string对象、数组和句柄需要一些技巧：总是要对它们解引用，存储它们所指向的值。记住在一个单独的字段中存储string或数组的大小。

　　希望通过以上内容的介绍，能够给你带来帮助。