# 单例模式

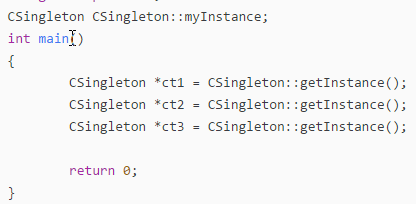
参考资料：https://blog.csdn.net/cjbct/article/details/79266057

基本思想：一个类只有一个实例对象，且将构造函数、拷贝构造函数以及赋值操作符函数声明为private或protect（这样方便子类继承）以达到阻止用户实例化一个类；同时提供一个public&static的方法来获取该类对象，这个对象即该类的唯一实例化对象。

饿汉式单例实现：即类产生的时候就创建好实例对象，这是一种空间换时间的方式

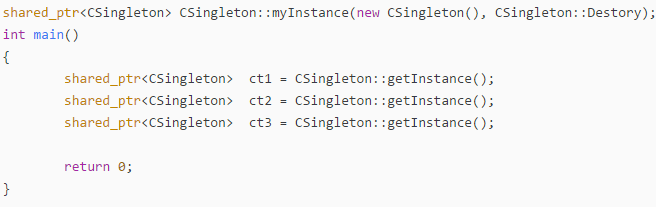
懒汉式单例实现：即在需要的时候，才创建对象，这是一种时间换空间的方式

## 饿汉模式

.h .c

上面的方式是直接实例化一个对象，如果改为指针的方式，即存储的是静态指针变量，然后源文件中使用new申请一个实例的指针。这种方式存在一个问题：**如果单例模式的类中申请了其他资源，就无法释放而导致内存泄漏**。解决这个问题的一种方式是使用c++11的share\_ptr智能指针：

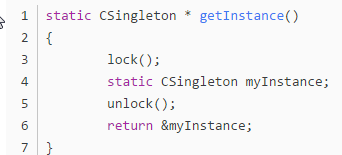
 .h

 .c

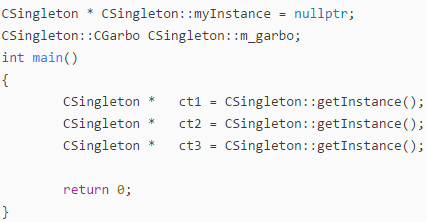
上面的方法中，使用了share\_ptr删除器的功能，因为share\_ptr在引用计数为0的时候会自动释放内存。而通过指定删除器是为了解决智能指针不能调用私有的析构函数的问题（虽然将析构函数改为public也可以，但是违背了单例模式的原则）。所以这里定义了一个静态的destory函数，即可通过类名直接调用的函数以用于替代析构函数。其实我们发现实际上并没有在这个函数中进行指针内存的释放而是需要一个析构函数的替代，否则编译不过。同样的为了满足编译格式上的需求或者说是一种门禁，destory函数的形参也是必须的，且是该对象的指针。

## 懒汉模式

在第一次调用获取实例API的时候才创建实例对象。基本实现如下：一种是返回实例对象的模式，一种是返回堆区指针的模式。

 .h 

为了实现线程安全加入了锁机制进行保护

 .h  .c

static CGarbo m\_garbo —— 它的作用就是帮助单例模式的类实现析构时对内存区域的正确释放。另外为了保证多线程下的访问，申请实例对象的时候加入锁：

