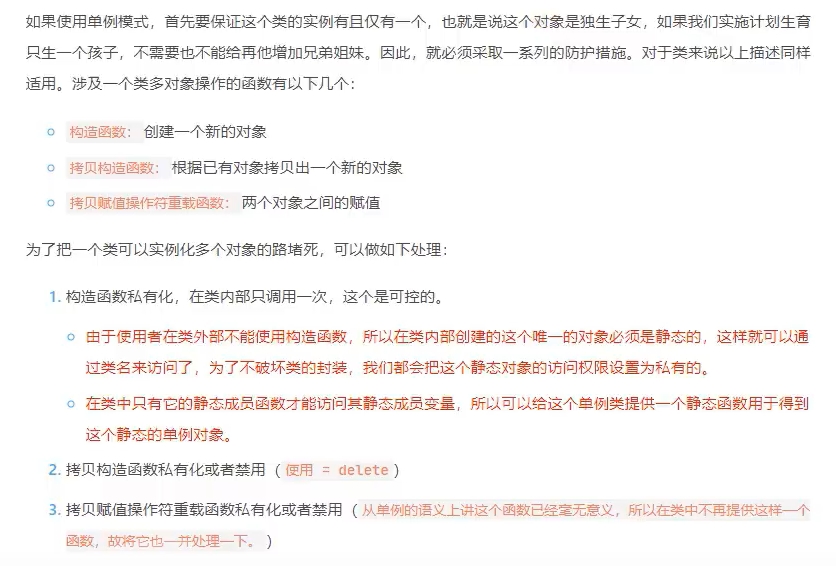
先给出单例模式的定义：

在一个项目中，全局范围内，某个类的示例有且只有一个，通过这个唯一示例向其他数据提供数据的全局访问，这种模式就是单例模式，一个经典的应用就是任务队列。

需要注意的是，单例模式作用的对象是一个类，这种模式下的类保证某些类在系统中全局唯一，并保证这些类在整个程序中提供一个具有规范的访问接口。这个模式解决了资源管理，状态一致和模块解耦的问题。举例，这个其实可以应用于一个全局变量的访问权限封装，避免一个资源方面的使用竞争等。

可以预见到，单例模式的一个基本要求是一个**单例**的特性。而如果要实现这个特性，仅仅在设计时去确保它只创建一次将是骗人骗己的。我们需要做的会是一个绝育，确保这个类会且只会被创建一次。



要让一个类只能创建一次，我们必须堵死所有可以自由创建类以及可能导致类的复制的方法。显然，我们一个去禁止单例模式的类对象的new创建使用，要如此的话我们就需要从它的构造函数入手。

我们可以直接删除(delete)掉默认构造函数，同时拒绝提供任何带参构造。同时，需要我们得知道c++内部是会自动提供一个拷贝构造函数的，所以对于这个函数我们也得进行删除，对应的，与拷贝构造函数绑定的=重载函数也应该删除。至此，我们删除了一些与类的创建有关的一些函数。当然方法不止于此。

我们还可以不删除函数而是将对应的函数放在private中实现等效的作用。毕竟private是无法通过类对象访问的，这样就避免了在外部程序通过调用new方法去创建多个类对象了。

总总这些，只是解决了绝育这一步，但是我们需要考虑的还有我们对于这个单例模式下的类还是需要一个具体的示例的。而这个实例在设计中一般是通过静态成员变量来进行访问的，而且这个成员变量应该是本类的一个指针，也就是说，这个指针是提供给外界的唯一类接口，我们所有对这个类的访问和操作都必须经过这个指针。

接下来，我们需要对这个指针进行初始化。毕竟在类中，你是无法对一个静态成员变量进行初始化的。

类名\* 类名作用域::静态成员指针=new 类名;

这个格式就是一般的单例模式的类的静态成员类指针初始化语句。

在然后我们需要将这个接口提供给程序，毕竟现在我们只是实例化了这个静态成员变量，由于是类的成员变量，且一般来说我们是将其放在一个private作用域的，所以我们是没有办法通过一个类对象去进行访问的。事实上，由于我们已经对类进行了绝育，所以我们其实是无法额外new一个对象出来的。这时候，我们一般需要额外定义一个静态公有方法，这个方法用于放回一个指针，而这个指针就是我们前面创建的静态类成员指针。也就是说，我们需要一个可通过类名直接调用的静态成员函数来充当接口的作用。

至此，我们已经有了一个单例类的指针，而且我们可以在项目中的任何地方去取得这个指针，只要我们包含对应的文件。当然，需要注意单例这个属性，不要在多文件中去错误的创建多个静态类成员指针。保证单例的性质。

接下来看一下这种设计的可行性基础。或者说，我们需要重新认识下静态成员等的特性。

首先，静态成员是属于类的，所有类都共享一份。并且能通过对象和类名来进行访问，当然，一般来说，对于静态成员只通过类名来进行访问好进行区分。这是单例模式使用这种静态成员来实现单例的基础。只有这种唯一性能够保证我们在整个程序中都保证唯一。

其次，我们需要注意下我们的静态类指针一般和构造函数都是房子private中的。这里就需要注意下静态成员的访问权限了。静态成员在初始化和定义时是无视其的作用域，也就是说，即使我们声明这个静态成员的属性为private。我们仍可在外部对这个静态成员进行定义和初始化。或者说，我们必须在外部进行对应的初始化和定义。一般来说，我们的初始化和定义都会是绑定在一条语句的。一条只含有定义的语句将会导致一个未定义行为，毕竟你不知道对应的二进制文件位置处是一个怎么样的文本串。话又说回来了，对于一个初始化了的静态成员，我们在二进制文件中的位置是.date段，对于未初始化的成员变量，其的位置是在.bss段。扯远了，回来。

在进行初始化和定义后，这个类静态变量的访问权限将会回复，一个静态公有变量将还是能够通过类名进行直接访问，但是如果是私有或者是保护的，那么类名就没有权限去访问到这个成员了。这时就需要用到一个类接口。一般来说，还是将这个接口设计为静态公有函数以进行区分，当然理论上你还是可以通过一个普通的公有成员函数。但是其实这样做并不是很安全。注意设计模式中的**依赖倒转原则**，我们完全可以通过一个静态函数来充当一个中间层来屏蔽影响，这样同时能够提供更好的逻辑上的隔离。

就比如我们可以通过这种设计来达到一种懒加载或者说延迟初始化的效果，我们完全可以在外层初始化这个静态指针时初始化为一个nullptr，然后在之后进行利用这个指针去找到对应的示例时再去进行对象的创建，而这个创建可以在对应的静态公有函数中去。