**Stack And Queue**

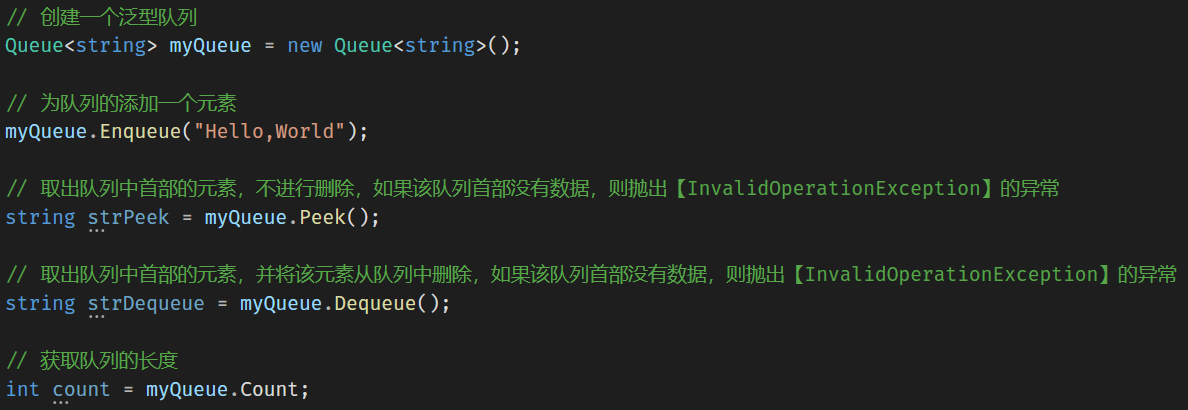
**开篇**

-> Stack和Queue是属于两种特殊的集合，区别于平常所使用的诸如List的集合， 它们是不存在索引器的，也就是说我们不能通过索引来获取指定下标的元素，并且其添加元素的方式和平常的集合也是存在区别的，但是不可否认的是，Stack和Queue其本身的实现就是一个集合

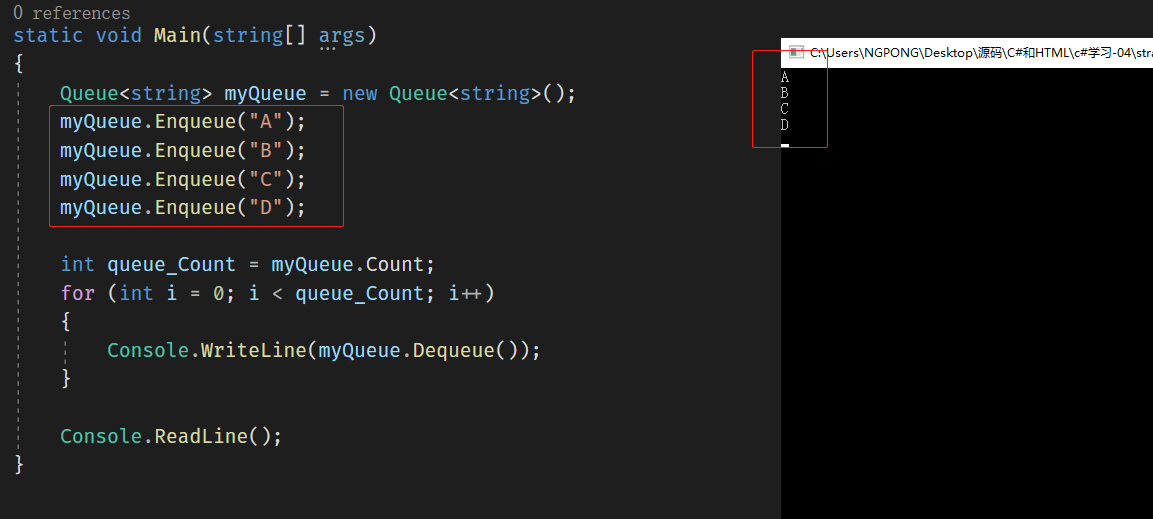
**Queue**

-> Queue在获取元素的时候需要牢记这一特性：先进先出

-> 关于Queue的一些常用函数



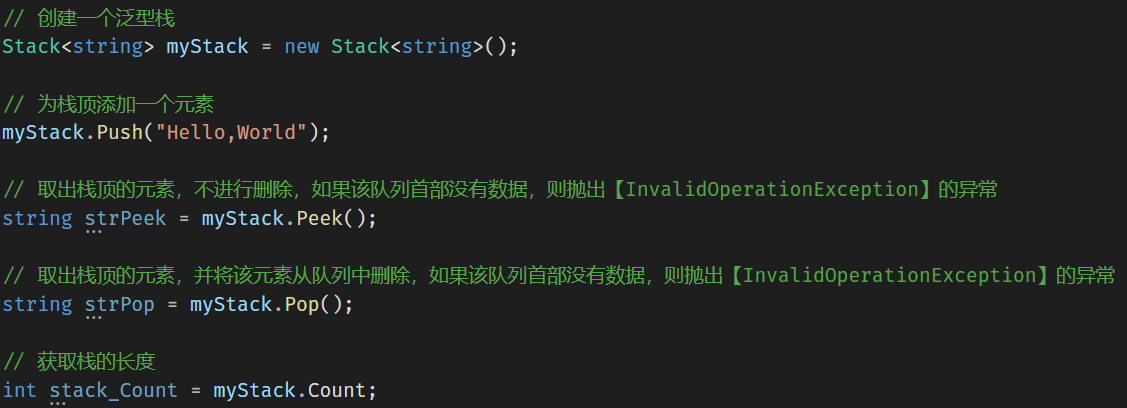
-> Queue的先进先出特性



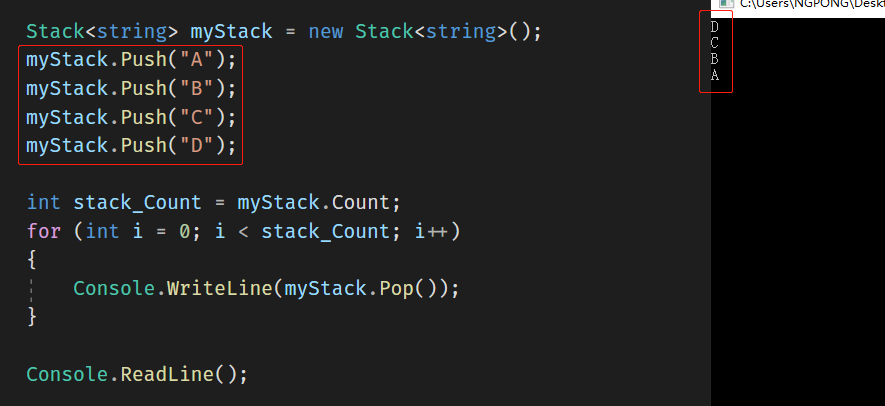
**Stack**

-> Stack在获取元素的时候需要牢记这一特性：后进先出

-> 关于Stack的一些常用函数

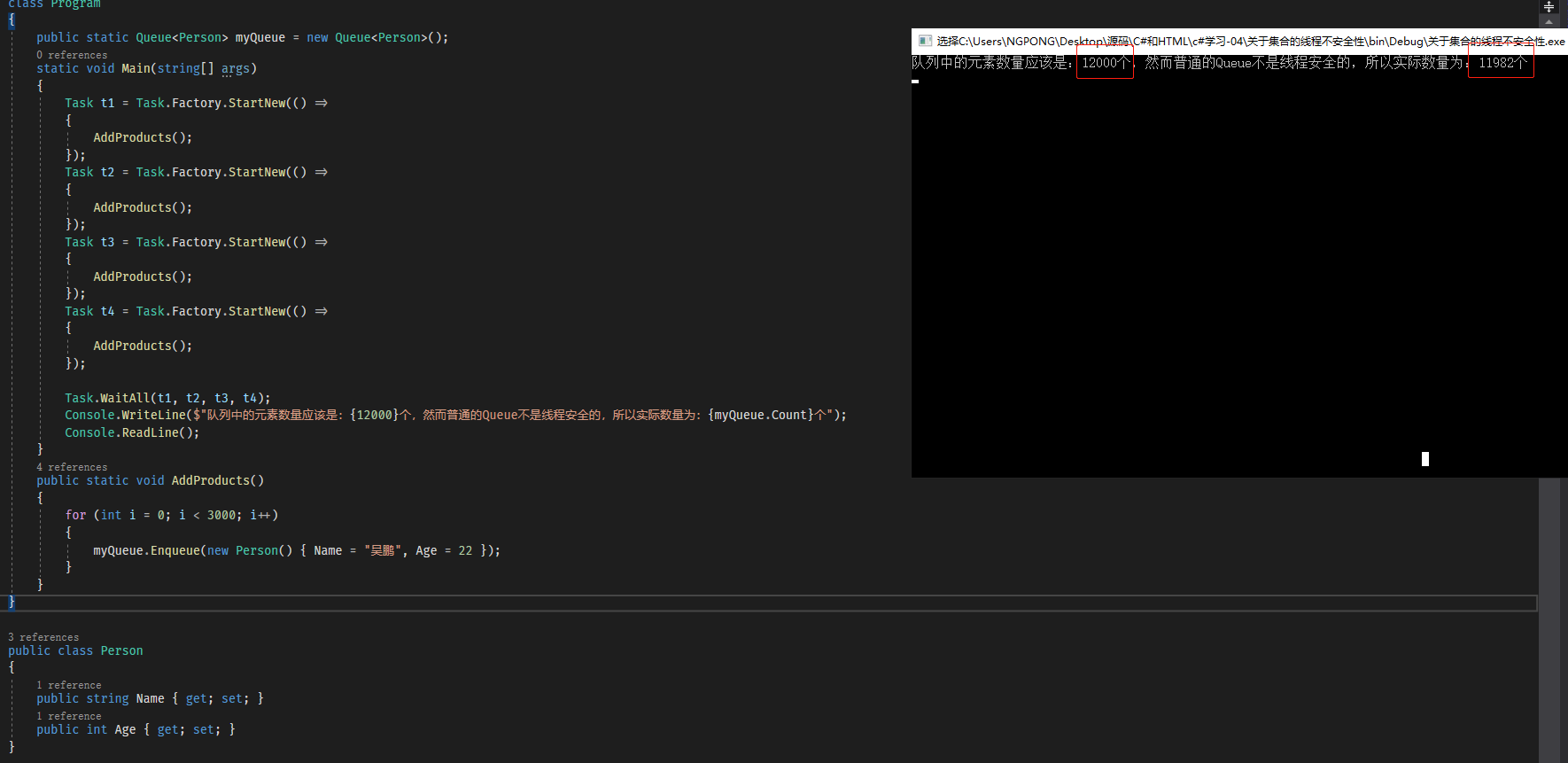


-> Stack的后进先出特性



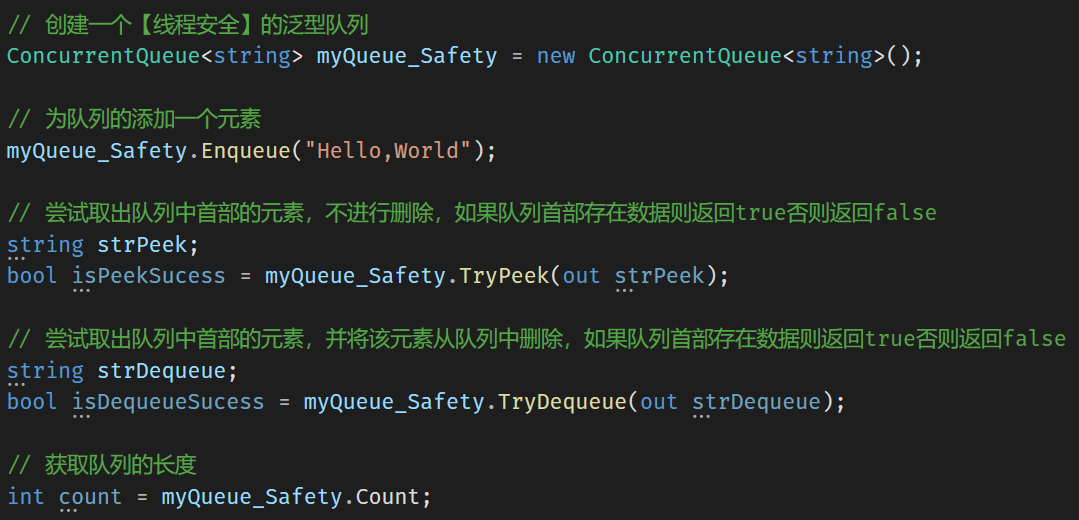
**关于Stack和Queue中的线程安全问题**

-> Stack和Queue不是线程安全的，不止它们两个，包括所有的集合其实都不是线程安全的，比方说我们通过自己所创建的几个线程同时进行往集合里面添加元素的任务，这时候我们会发现所需添加的元素数量和最终的集合的元素数量是不相同的，因为在某一时刻存在着一个并发的问题，如下图所示

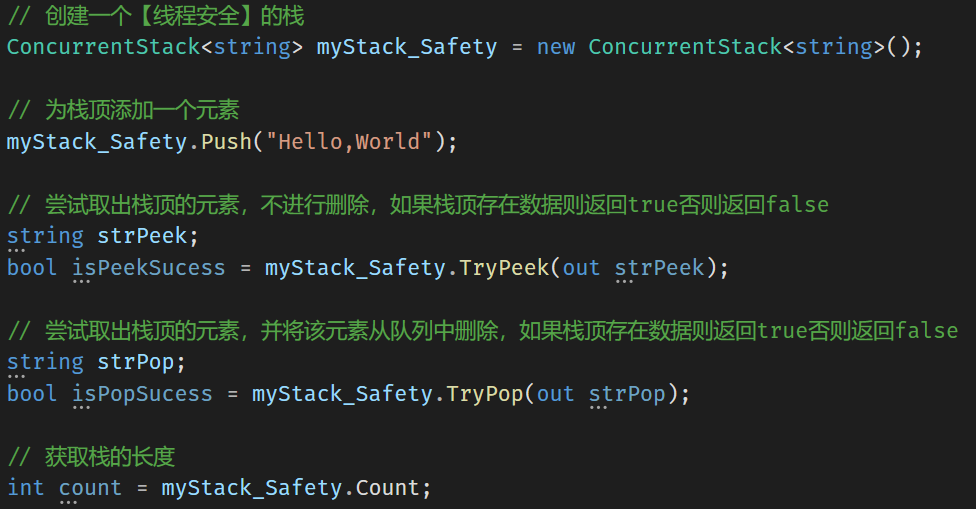


-> 如何解决这一问题？我们可以使用Lock关键字产生一个悲观锁来实现这一功能，但是Lock会实现不同线程进行相同资源之间的等待情况，所以这是一个比较浪费性能的事情，而微软在 .NET Framework 4.0 之后推出了几个关于线程安全的集合，其底层的构造已经帮我们实现了线程安全的问题，我们可以通过使用它们来很好的规避这一问题

1. ConcurrentQueue



1. ConcurrentStack



-> 关于更多的信息可以参考如下文章

<https://www.cnblogs.com/chenwolong/p/LoveFuTing.html>

<https://www.cnblogs.com/chenwolong/p/Queue.html>

<https://docs.microsoft.com/zh-cn/dotnet/api/system.collections.queue?view=netframework-4.8>

<https://docs.microsoft.com/zh-cn/dotnet/api/system.collections.stack?view=netframework-4.8>