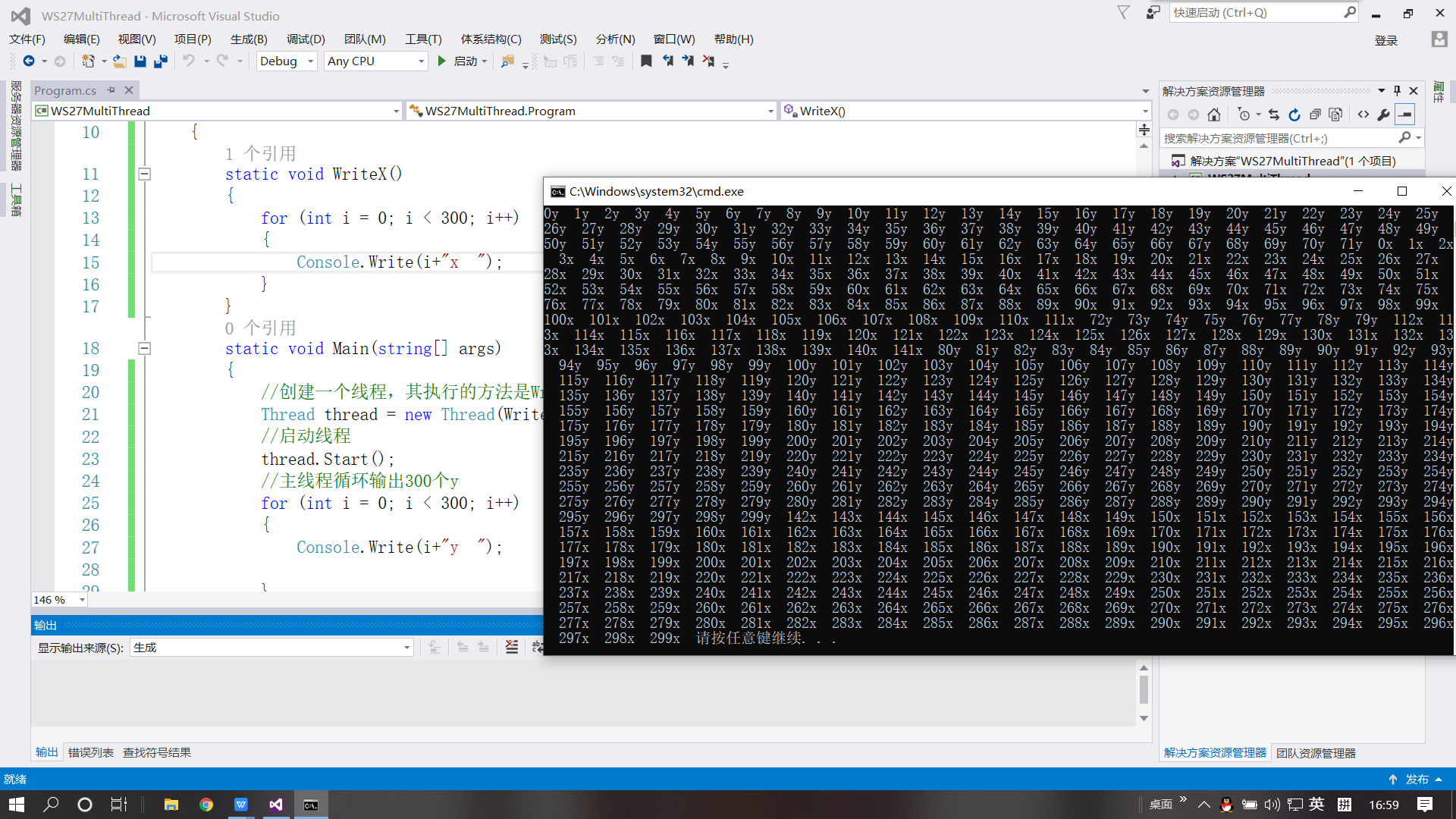
# 实验五 Windows多线程编程和线程同步

### （一）线程创建

1. 回答：程序运行结果是什么？解释运行结果



解：程序首先运行主函数的循环输出y，经过多次调式，大概70-80次循环后，开始执行线程thread里面循环输出x，但之后发现存在先执行线程的情况，查找资料，线程执行前需要时间处理调度的问题，但是提前抢占系统资源的函数先执行

1. 若注释掉thread1.Join()和thread2.Join()，信息“售票完毕，主线程退出”会在thread1和thread2终止后再显示吗？为什么？

解：不会，Join()方法是让主线程阻塞，直到线程终止为止，再唤醒主线程继续执行，所以注释掉thread1.Join()和thread2.Join()后，程序再启动两个线程的同时，代码还会继续向下执行，可能会先执行Console.WriteLine("售票完毕，主线程退出");，也存在执行线程时主线程继续向下执行；

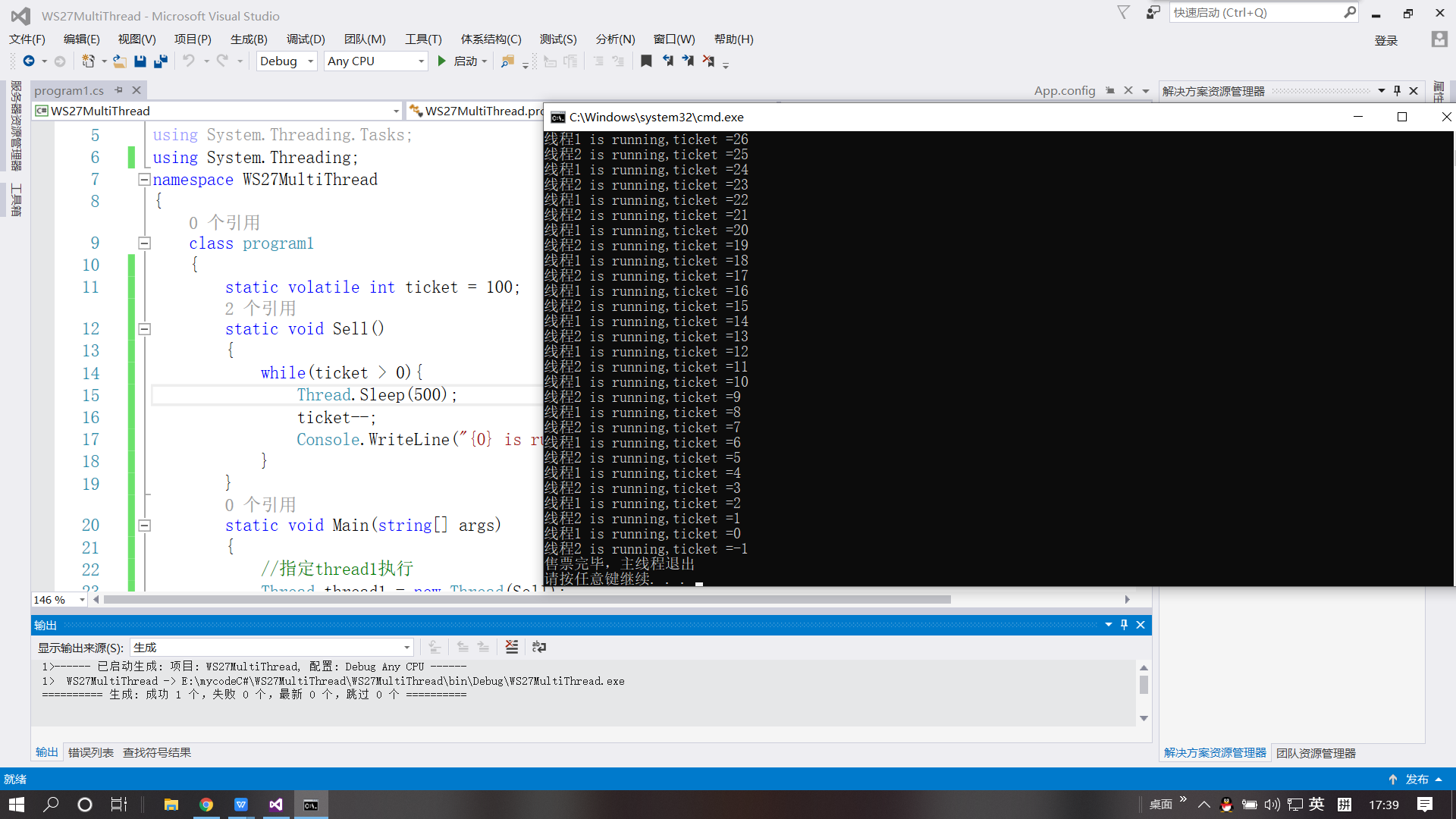
### （二） 线程同步

1. 续上述程序，取消thread1.join()和thread2.join()的注释，运行程序。程序运行结果符合预期吗？为什么？运行几次程序。程序每次执行结果一样吗？为什么？

解：程序运行结果不符合预期；因为最后一个线程显示的票数总是大于0，

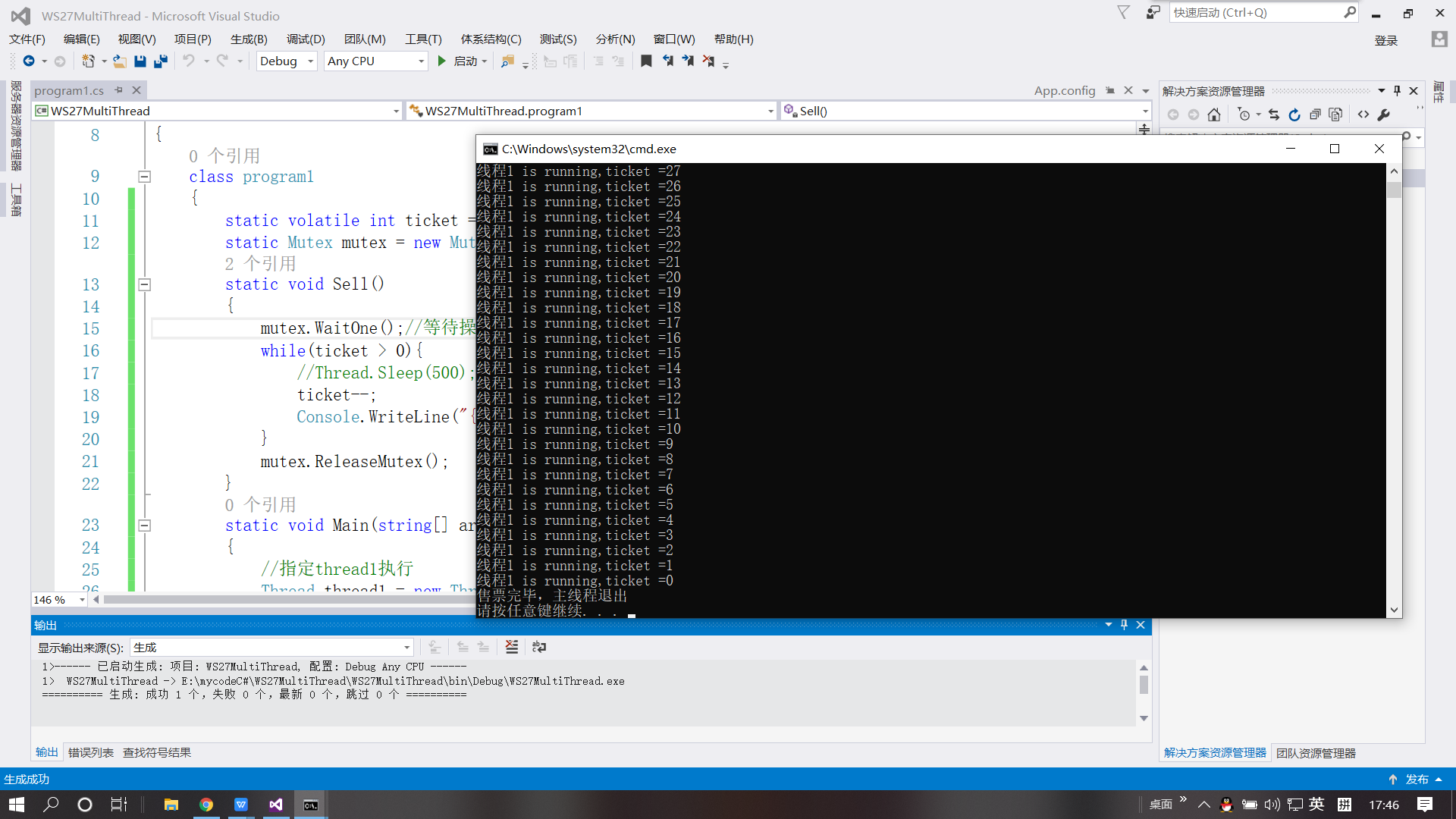
程序每次执行结果不一样，最后一个票数都大于0且不同，线程2并没有获得现时票数

1. 在函数Fun()内，ticket—前增加一句“Thread.Sleep(500);”，其作用是让当前线程阻塞500毫秒，然后再唤醒线程继续工作。再次运行程序（或多运行几次程序），会不会出现ticket=-1的情况？为什么？提示：分析当ticket = 1时，两线程的可能调度情况。



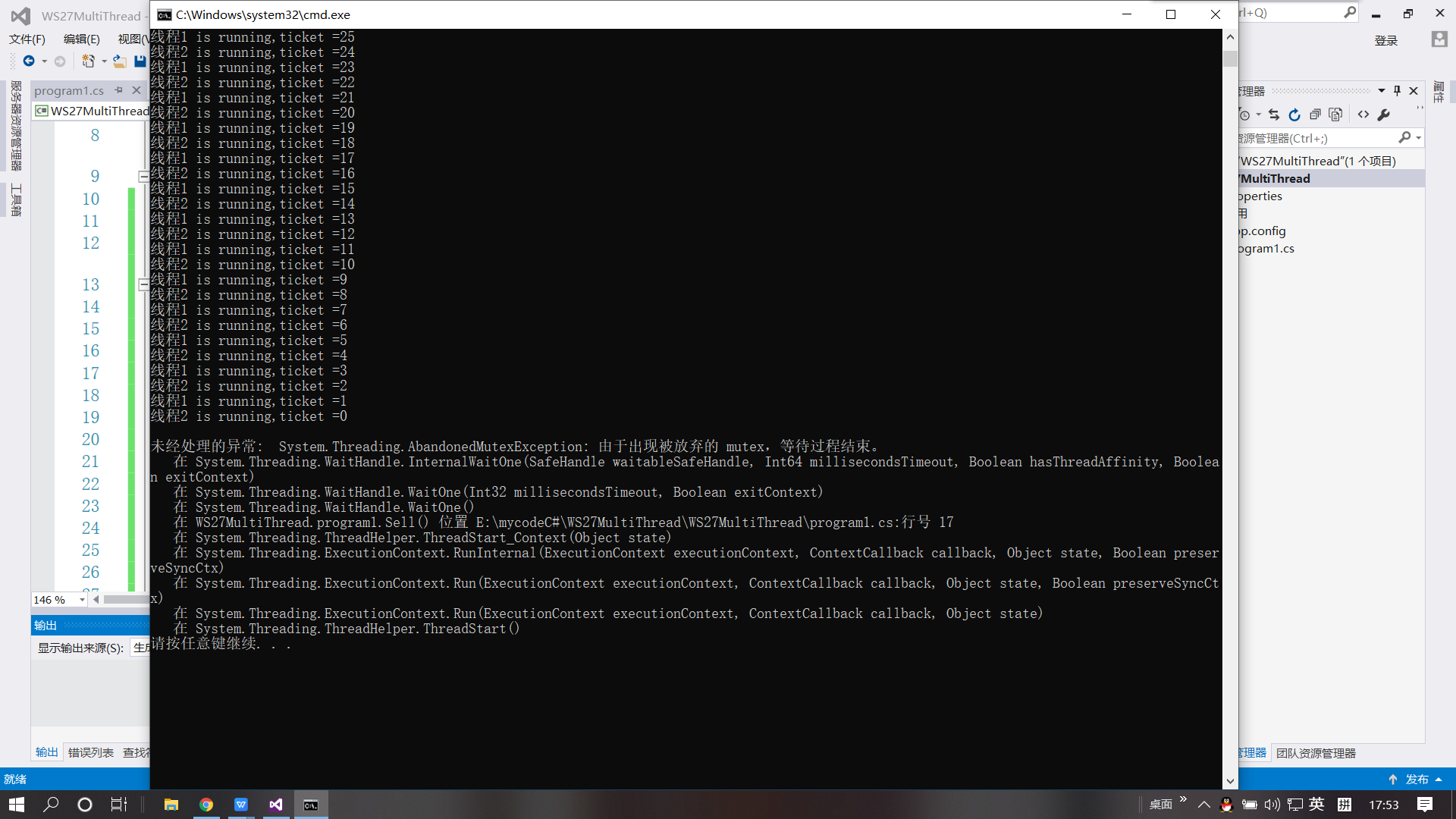
解：会出现ticket=-1的情况，因为剩余一张票时候，两个线程都判断ticket>0为真，然后进入Sleep()阻塞，在各获得一张票的情况下减一，导致最终票数为-1；

1. 产生上述问题，是因为两线程共享全局变量ticket，而没有任何互斥措施。下面我们实现对tickets的互斥访问。修改代码后，程序运行结果是什么？分析运行结果

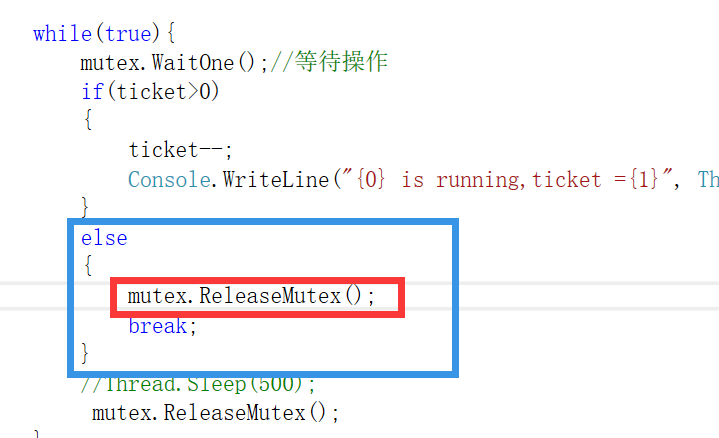


解：始终只有线程1在执行；因为mutex放在了循环外部，所以线程2一直在等待线程1释放，但是ticket>0的时候线程1始终没有开锁，导致线程2不能够进入waitone之后的代码

1. 更改上述代码，程序运行结果是什么？为什么会出错？该如何修改代码？



解：输出结果是正确的，但是程序报错了，当票数为0的时候，循环跳出来了，但是没有解锁，可以在else中添加解锁:



### （三） 用信号量实现生产者-消费者问题

1. 将consumer函数中的full.WaitOne()和mutex.WaitOne()交换位置。

反复运行程序几次。会出现死锁情况吗？若有，是什么原因导致死锁？

解：会产生死锁现象；消费者获取信号，但是生产者还在mutex.WaitOne()这个锁卡住，所以没有执行full.Rlease()这个解锁操作，导致消费者一直在等待full.WaitOne()这个锁，导致mutex.Release()这个锁没有解锁，所以导致死锁。

1. （选做题）将consumer函数中的full.WaitOne()和mutex.WaitOne()位置交换回来。交换mutex.ReleaseMutex()和empty.Release(); 重新运行程序。运行结果正常吗？可以交换mutex和empty的signal操作的位置吗？若不可以，为什么？若可以，先signal(mutex)再signal(empty)，与先signal(empty)再signal(mutex)，哪个操作更好？

解：重新运行程序。运行结果正常；

先signal(empty)再signal(mutex) 操作更好，因为生产者一直等待empty的解锁，而先signal(empty)可以先让生产线程先解第一个锁，增加程序效率。