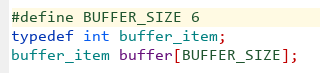
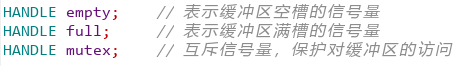
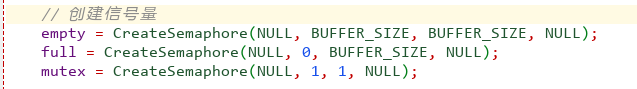
1. 缓冲区



缓冲区由数组实现，长度为6，每个元素的数据类型为int（重命名为buffer\_item）。

1. 信号量





1. mutex为互斥信号量，用于互斥的访问缓冲区，作为锁。
2. full信号量用来判断缓冲区是否有数据，初始值为0。当生产者生产了1个数据，就会向等待接收full信号的消费者传递。
3. empty信号量用来判断缓冲区是否有空缓冲区，初始值为BUFFER\_SIZE。当消费者消费了1个数据，就会向等待接受empty信号的生产者传递。
4. HANDLE 是Windows API中广泛使用的一种数据类型，它代表一个句柄（handle）。句柄是一种抽象的指针，用来标识和访问操作系统所管理的各种对象，如窗口、进程、线程、文件、事件、互斥体、信号量等
5. 信号量（Semaphore）的，它是Windows API中的一个同步对象，常用于控制多个线程或进程对共享资源的访问。信号量维护一个计数器，线程可以对其进行增加（ReleaseSemaphore函数）或减少（通过WaitForSingleObject函数等待信号量时）。当计数器为非零时，等待的线程可以继续执行；当计数器为零时，线程将被挂起，直到计数器由其他线程增加为止。

HANDLE CreateSemaphore(

LPSECURITY\_ATTRIBUTES lpSemaphoreAttributes, // 信号量的安全属性

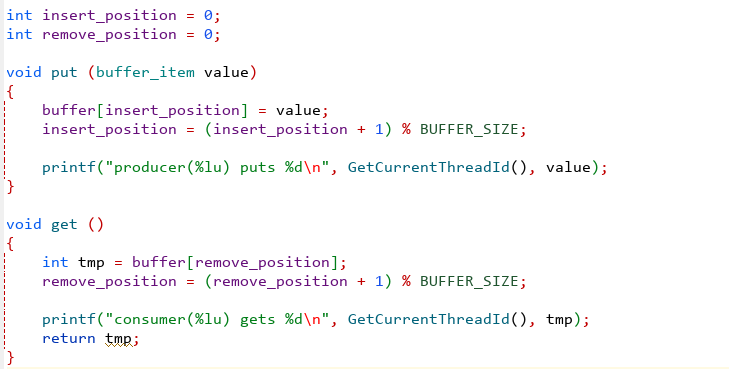
LONG lInitialCount, // 初始计数

LONG lMaximumCount, // 最大计数

LPCSTR lpName // 信号量的名称

);

1. 缓冲区的写入写出



insert\_position用来表示下一个该put的位置，remove\_position用来表示表示下一个该get的位置，每次使用完会加一取余，维持边界，整体缓冲区数据结构是一个队列，在put和get的时候还会输出当前线程的标识号。

1. 生产者



1. srand(GetCurrentThreadId())使用当前线程ID初始化随机数发生器，确保每个线程生成不同的随机序列；
2. 用随机数生成一个item作为写入；
3. WaitForSingleObject(empty, INFINITE); emoty 减一，等待直到有空的缓冲区位置可用。这保证了不会在缓冲区满时尝试写入数据，避免数据丢失或覆盖
4. WaitForSingleObject(mutex, INFINITE); 获取互斥锁，确保在修改缓冲区时不会有其他线程干扰（即保护临界区）
5. ReleaseSemaphore(mutex, 1, NULL)之后释放互斥锁；
6. ReleaseSemaphore(full, 1, NULL); // 信号量full增加1，表示缓冲区中已有一个新的满槽，消费者线程可以检查并消费
7. 消费者



1. WaitForSingleObject(full, INFINITE); 等待直到缓冲区中有数据可消费（即full信号量的计数大于0）
2. WaitForSingleObject(mutex, INFINITE); 获取互斥锁，确保在访问和更新共享缓冲区时不与其他线程冲突
3. get();消费数据，调用get()函数从缓冲区中取出数据并打印信息
4. ReleaseSemaphore(mutex, 1, NULL); 释放互斥锁，允许其他等待的线程（无论是生产者还是消费者）继续执行
5. ReleaseSemaphore(empty, 1, NULL); 增加empty信号量的计数，表示缓冲区中有一个空位可供生产者填充数据
6. main

本进程在return前会sleep10秒，便于查看各个线程的情况。

通过命令行能确定生产者和消费者的数量。第一个参数为生产者数量，第二个为消费者数量。



