Lesson 7

1.类的函数指针：

a) 普通成员函数指针包含类名信息以及 const，指向具体函数时必须加上&符号。

b) static 函数的指针不包含 class 名，&符号也不是必须。

2.改写生产者消费者，封装一个车间类（指定缓冲区大小，以及生产者、消费者的数目）。

3.分析程序框架：

a) Condition 内部含有 MutexLock 的引用。因为多个 Condition 要与同一个 MutexLock 进行交互。

b) Buffer 是在原生的 Queue 上面包装了一层互斥与同步的机制，从而实现了一个线程安全的缓冲区。

c) Buffer 包含了 MutexLock 和 Condition。

d) 两个 Thread 持有 Buffer 的引用，是因为多个线程要与同一个 Buffer 交互。

e) 一个类持有另一个类的引用，这叫做类的关联。

f) 一个类的数据成员中含有另一个类的对象，这叫做类的组合。

g) Buffer 内置了同步与互斥的机制，使得生产者进程和消费者进程不必关心竞态问题，这使得模块之间的独

立性增强，这符号软件工程中“高内聚，低耦合”的原则。

4.关于头文件和前向声明：

a) 使用了类的指针、引用，采用前向声明即可。

b) 如果使用了类的对象，或者使用对象、指针、引用调用了数据成员或者函数，此时必须使用头文件。

5.单例模式的编写：

a) 构造函数设为私有，此时无法生成对象

b) 编写一个成员函数来生成对象，但是无法调用。

c) 将该函数设为 static，此时可以生成对象，但是对象不唯一。

d) 添加一个 static 指针成员，仅当该指针为 NULL（也就是第一次访问时）才去生成对象。

e) 但是此时的代码在多线程环境下存在竞态问题。

f) 于是我们每次检查指针前都要进行加锁。

g) 此时每次获取对象都要加锁，锁争用过多，影响效率，于是我们引入“双重锁”模式（Double Check Lock Pattern， DCLP）。这种模式采用了两重判断，其中内部的判断采用了锁，保证结果的正确性，外面的检查保证大部分线程不会进入争用锁。

h) 后面我们采用 pthread_once 编写单例模式。