1. mutex互斥量的原理，即在A线程上锁后，B线程运行同样的代码试图上锁会发生阻塞。并没什么🔒住数据之说，因此同一个互斥量，会所谓的锁住不同数据
2. 多线程要考虑极端情况，临时变量无法构造等等
3. 避免死锁

获取一个锁别再获取第二个

别在获取了锁时调用用户代码

1. 条件变量与锁的连用 先锁定 再等待信号，收到信号后解锁

Cond.wait(mutex, lambda)

当 lambda返回true 条件成立，继续锁定，并向下执行

当lambda返回false条件不成立，解锁，并且继续等待，其他线程无法回去锁

1. Future 期望可用于返回参数

Future<T> answer = async(&find\_answer, &class\_has\_it, args…)

调用answer.get() 会阻塞，直到async处于就绪状态，并获取结果