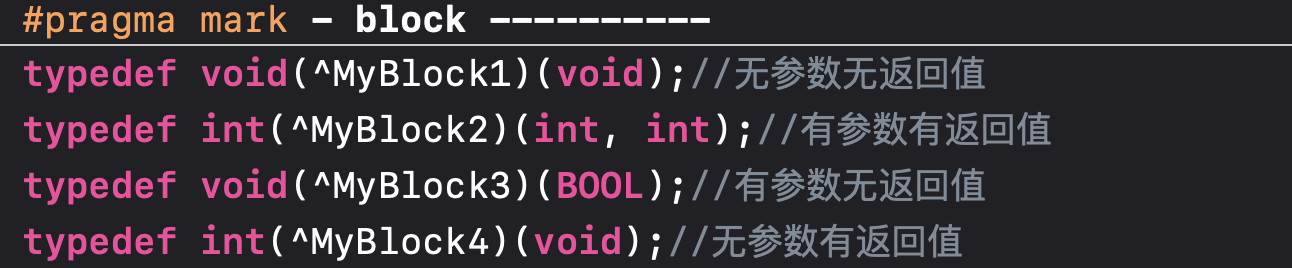
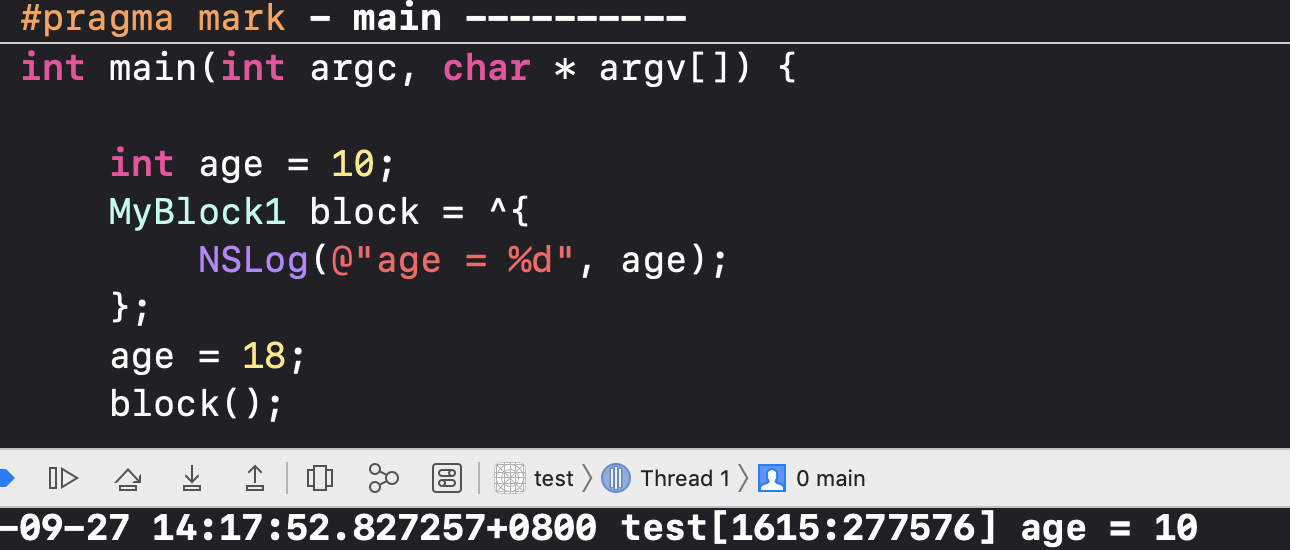
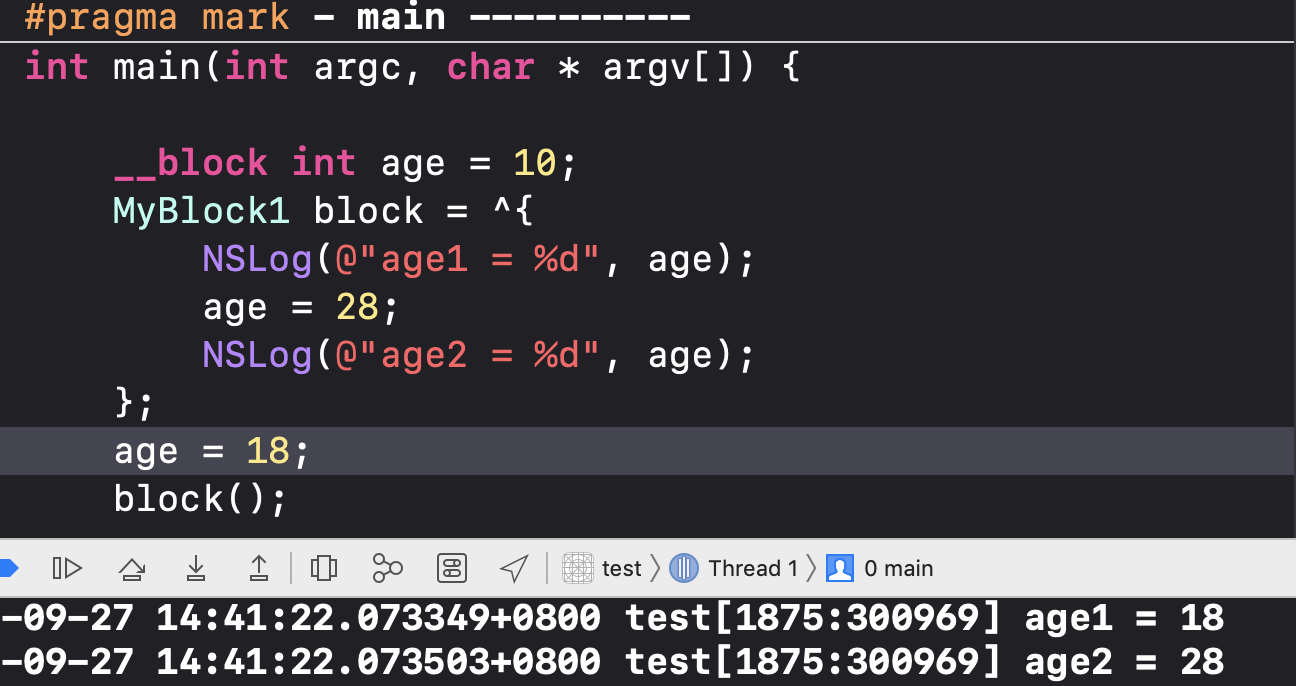
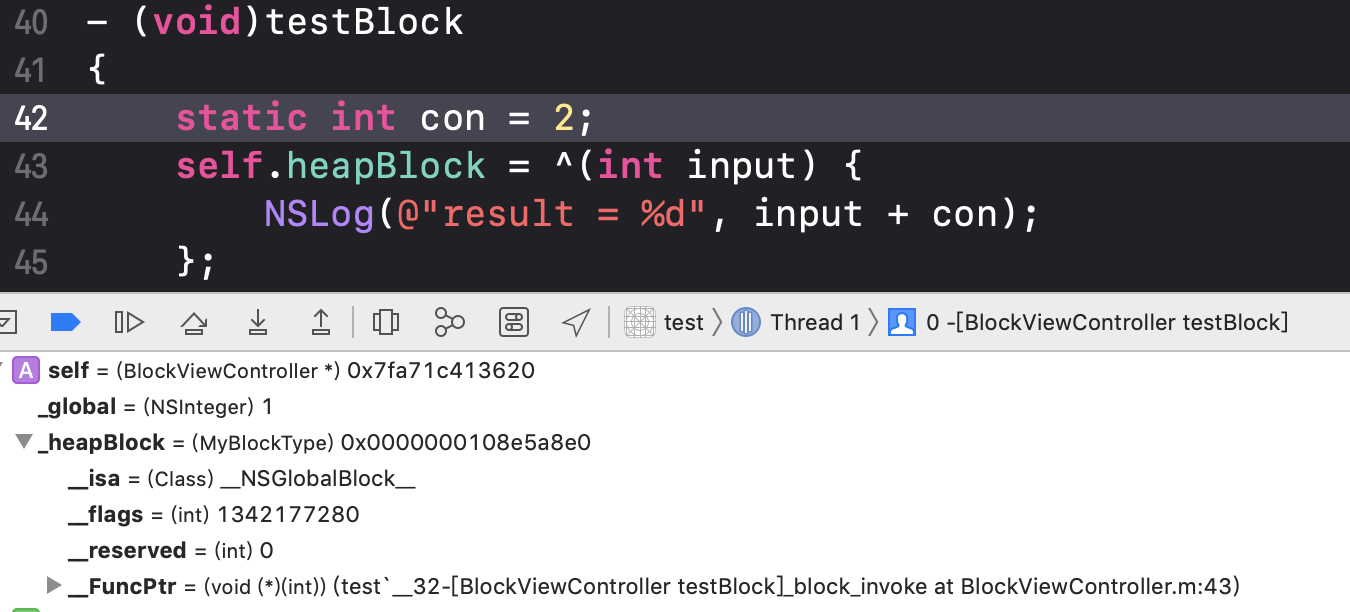
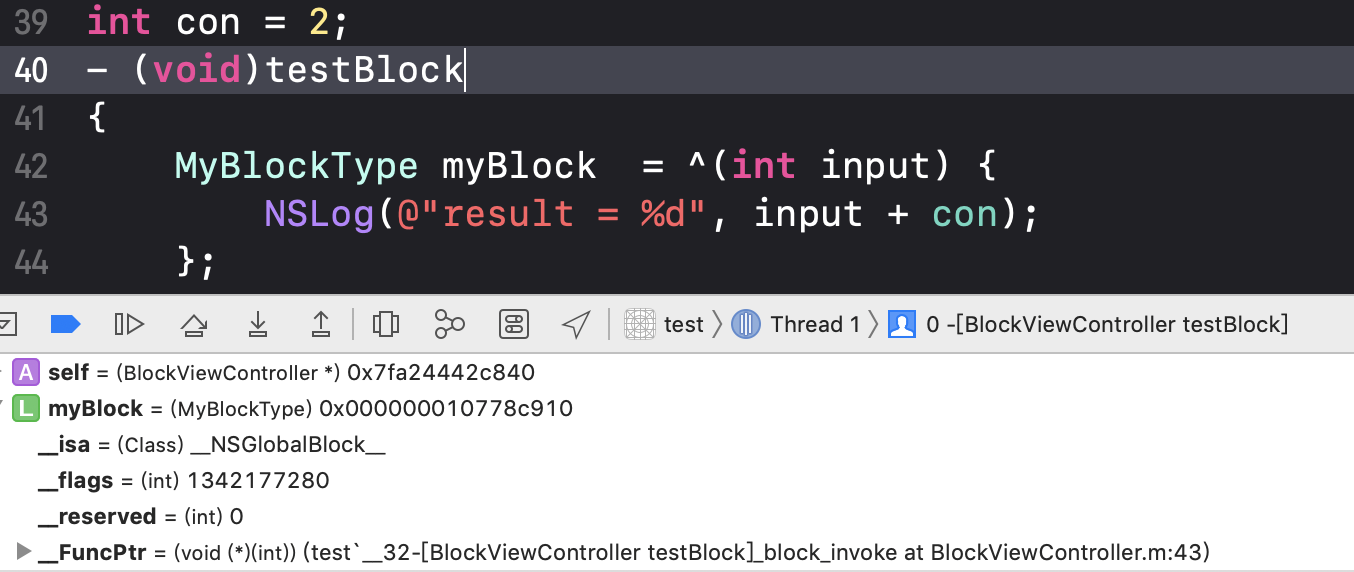
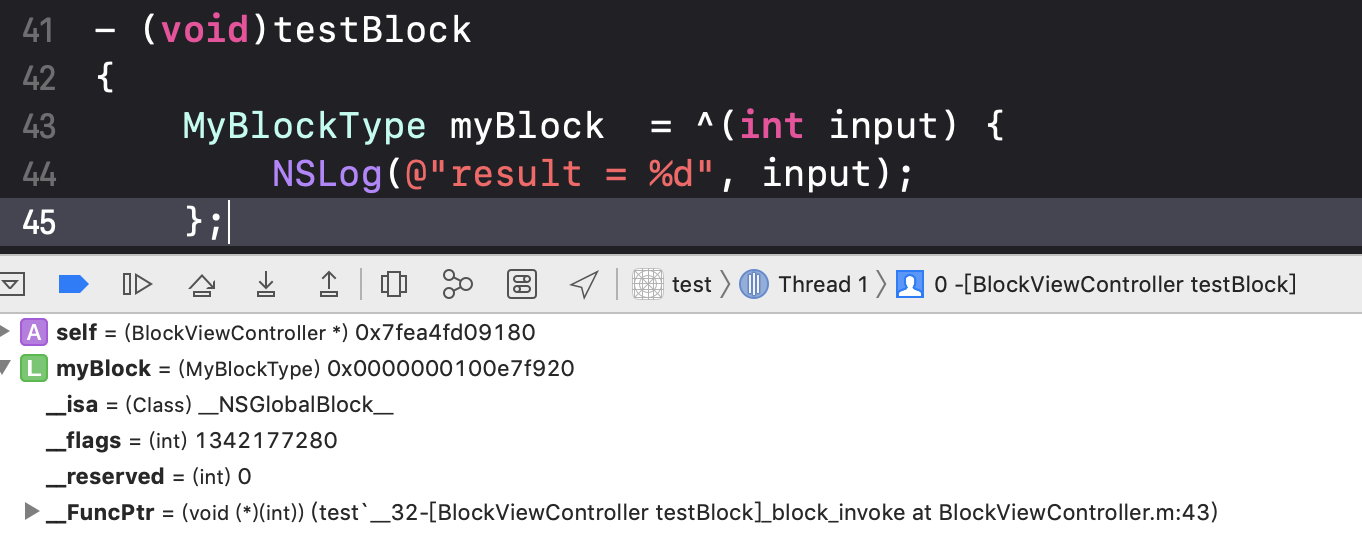
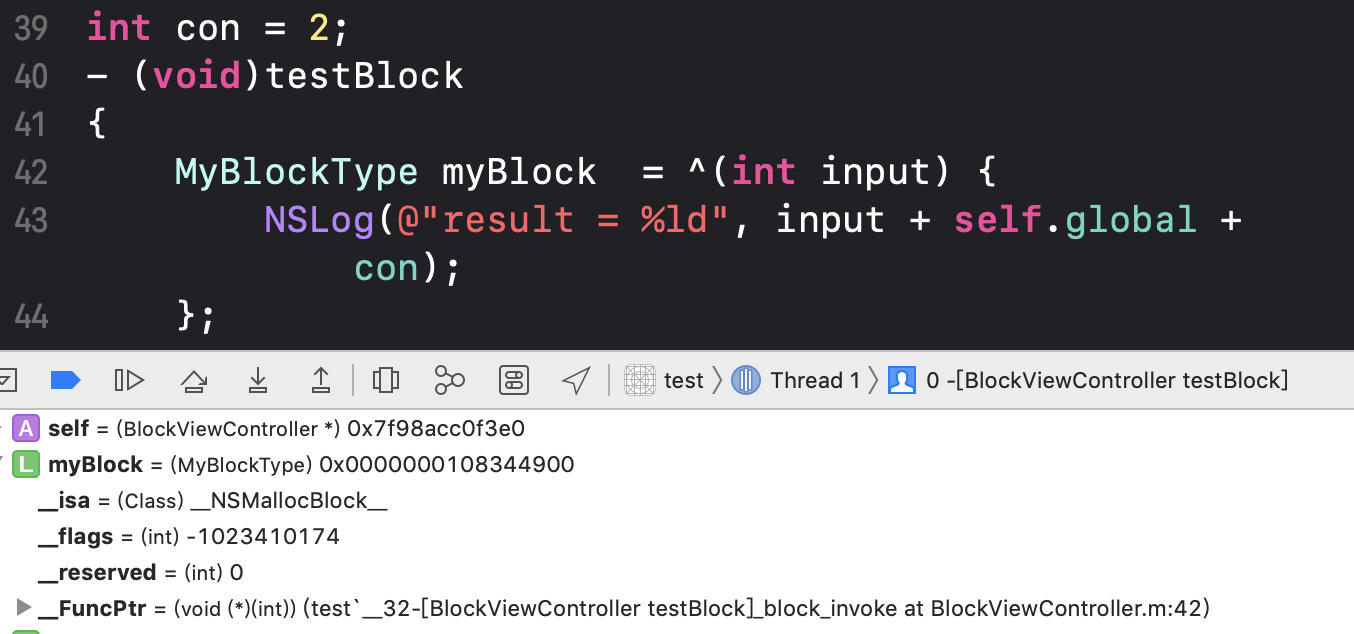
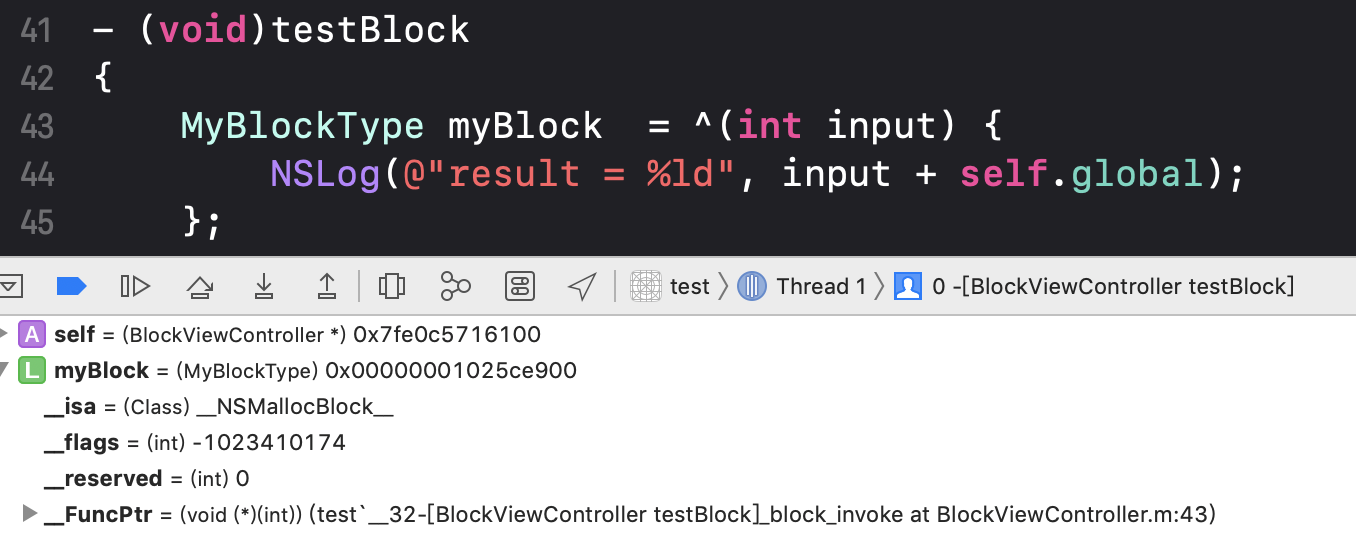
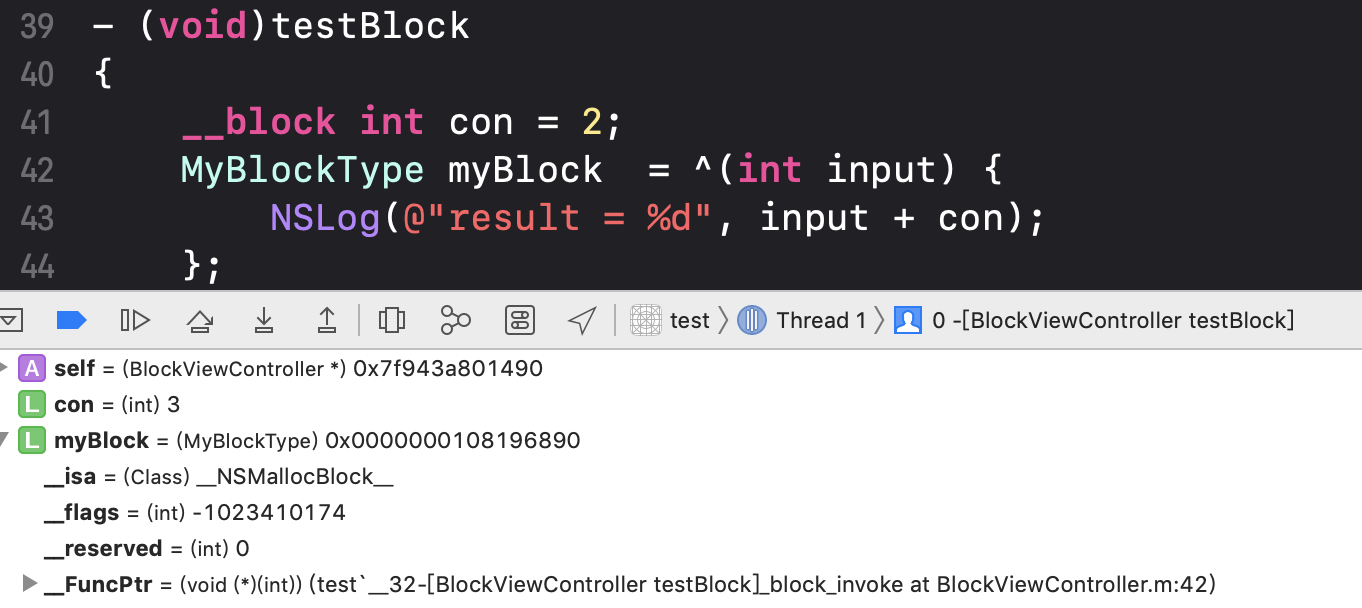
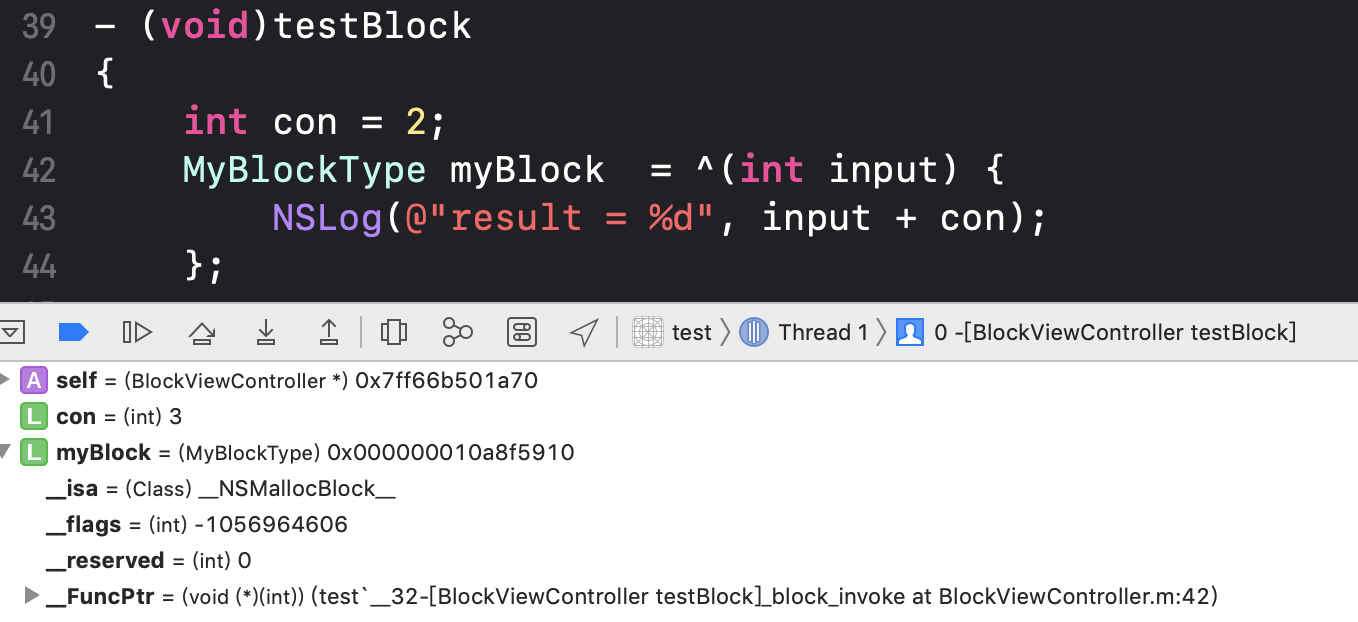
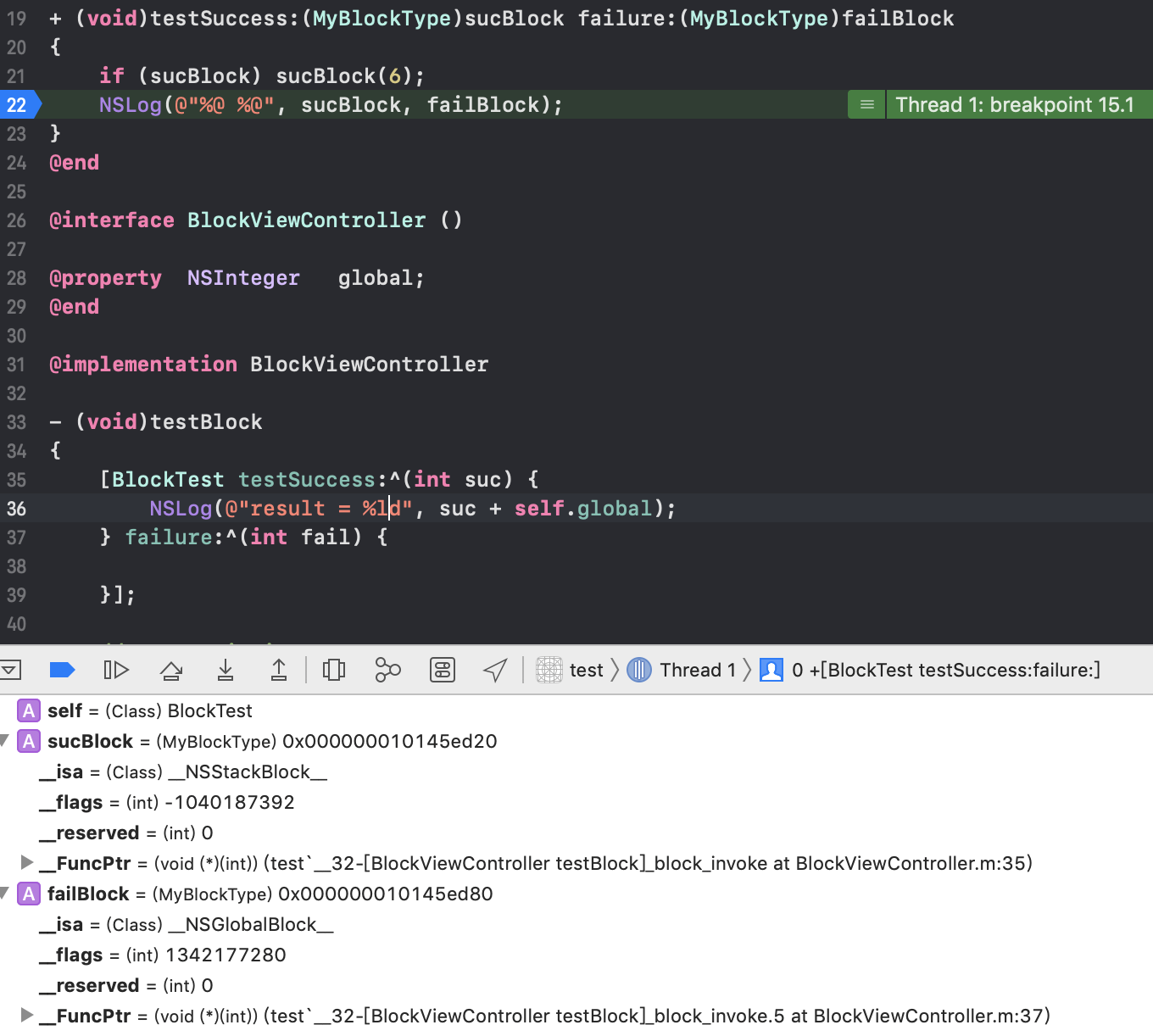
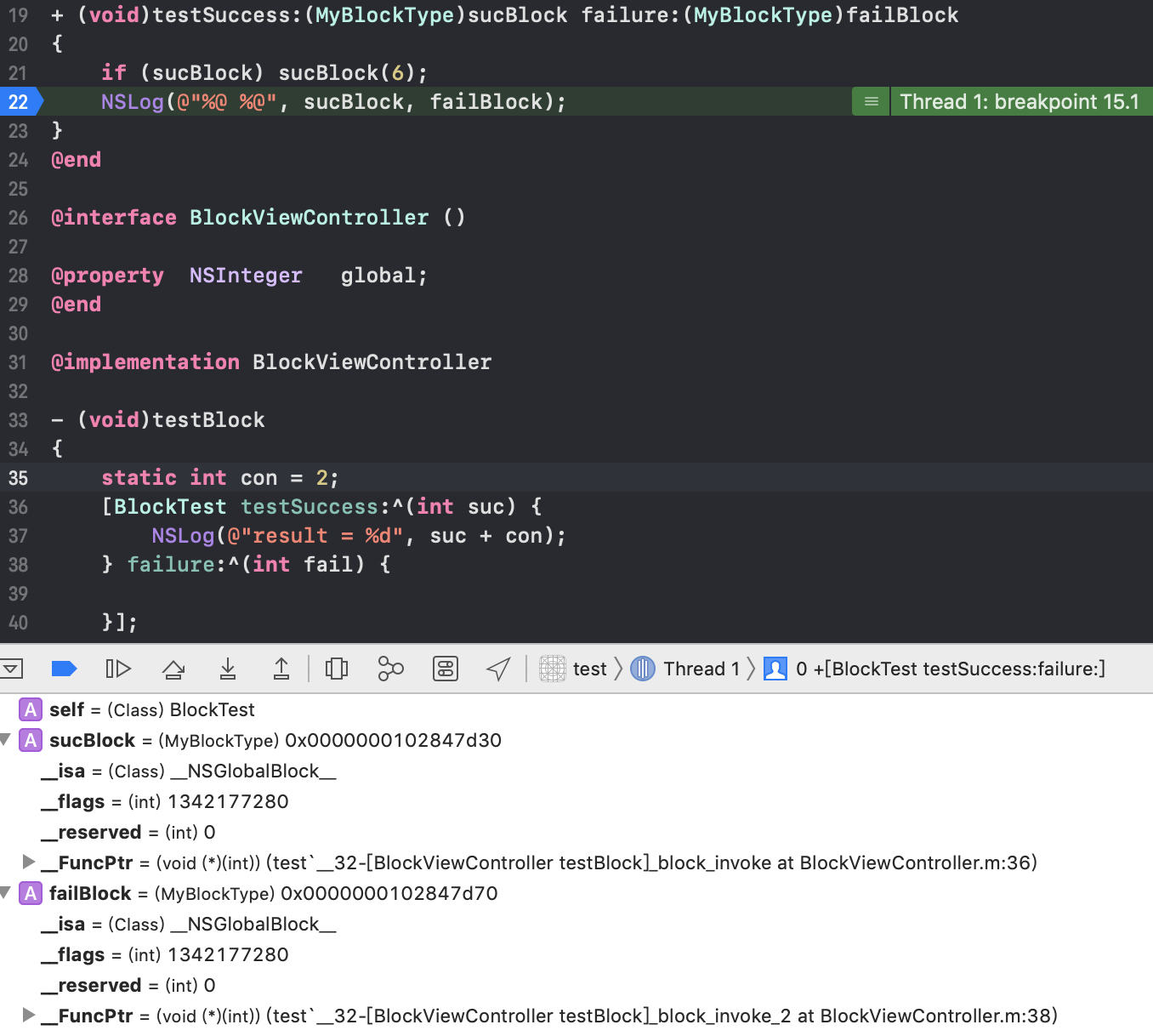
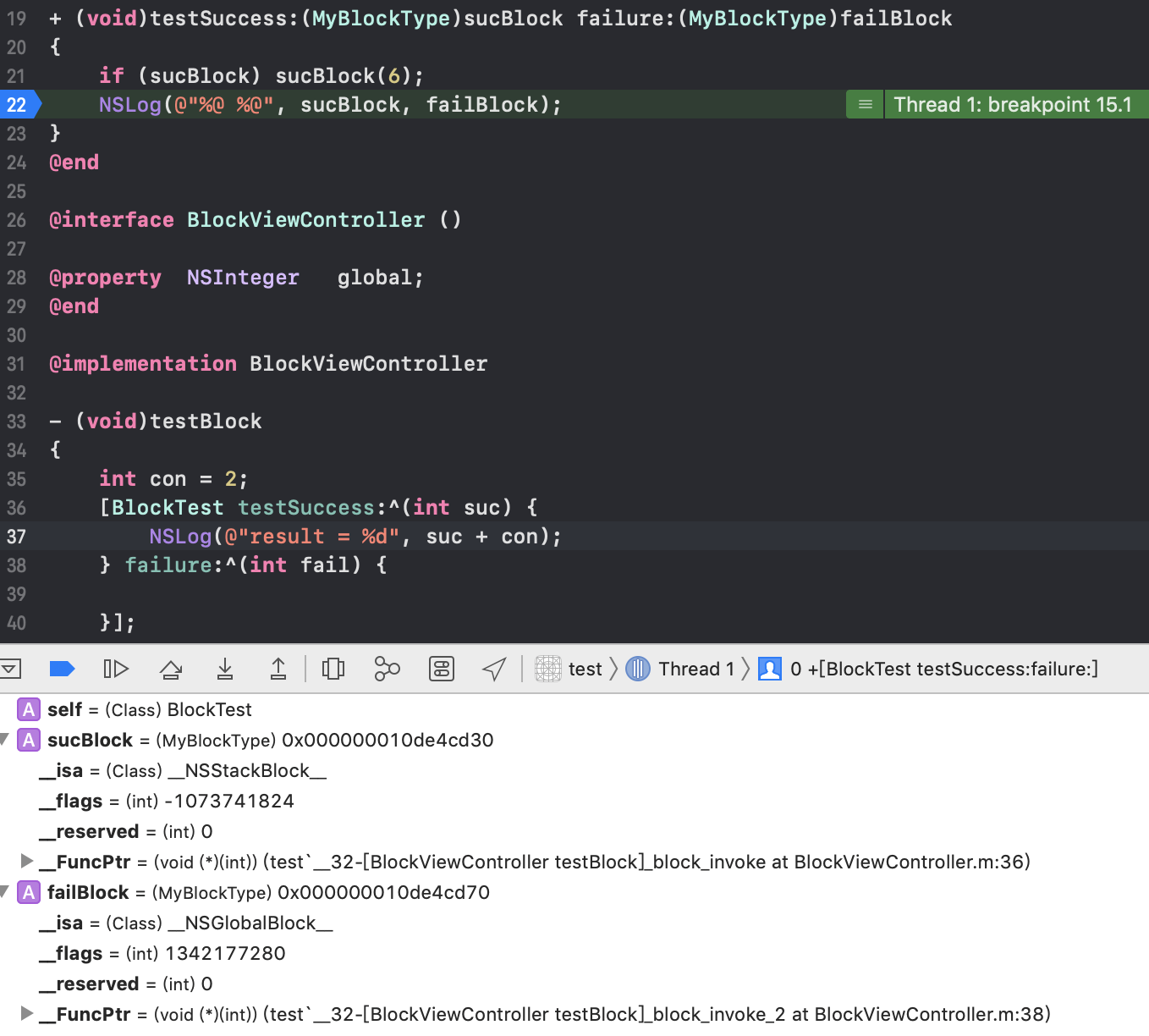
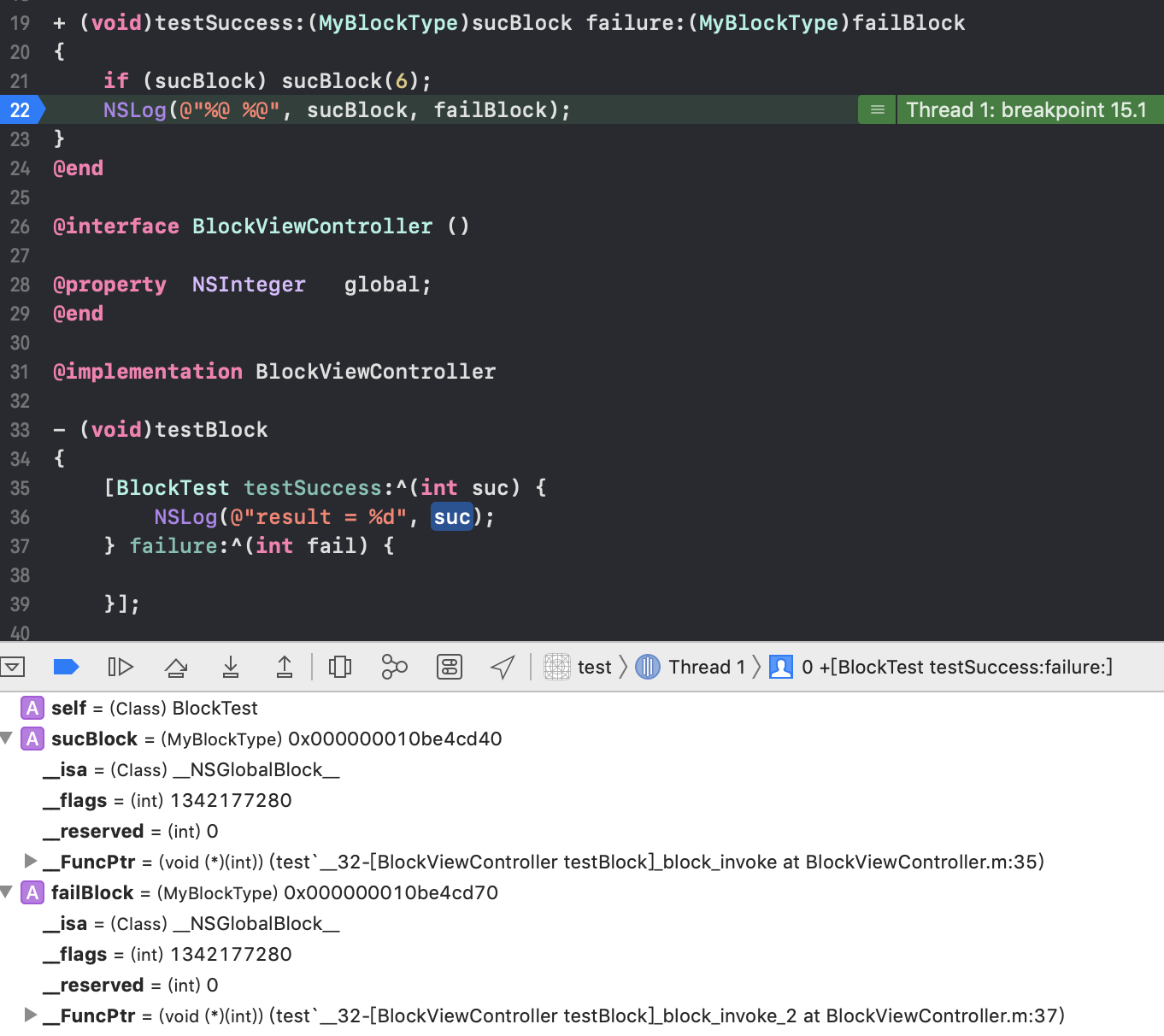
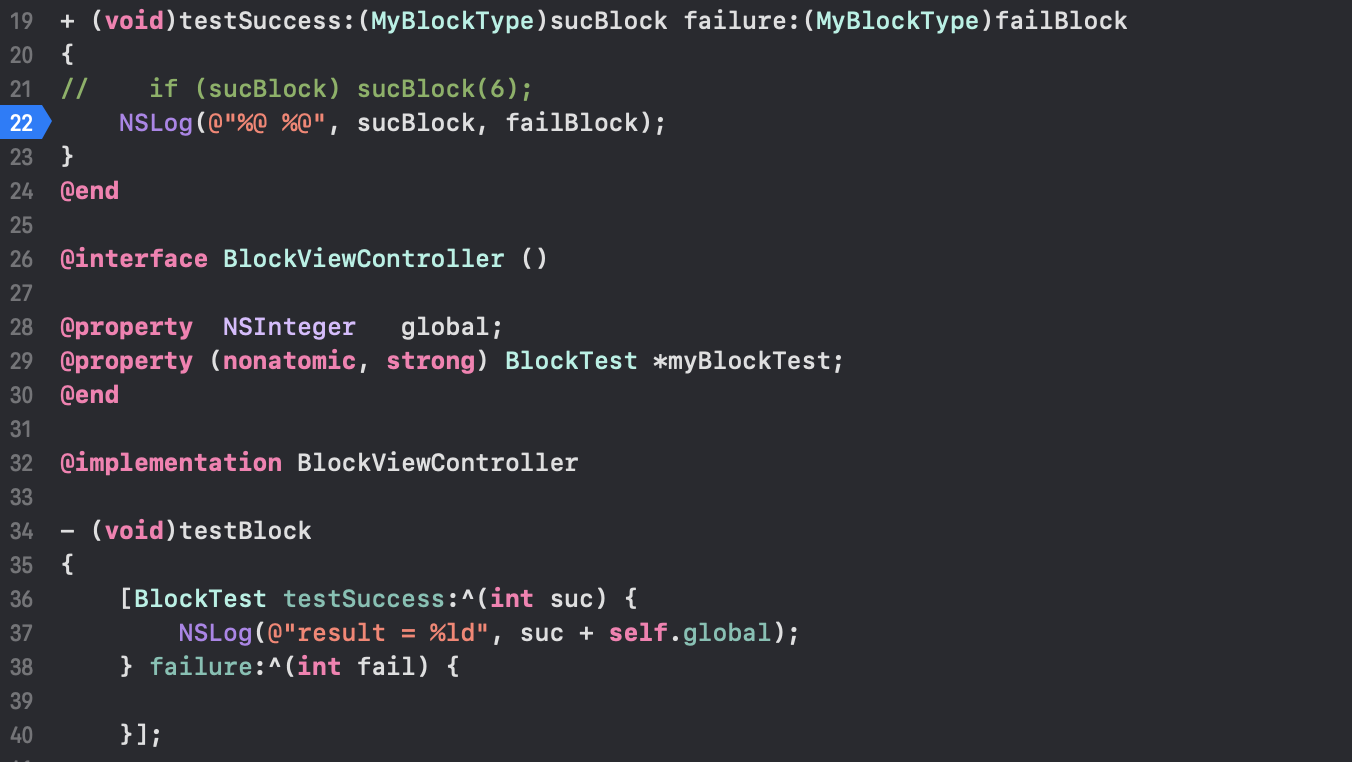
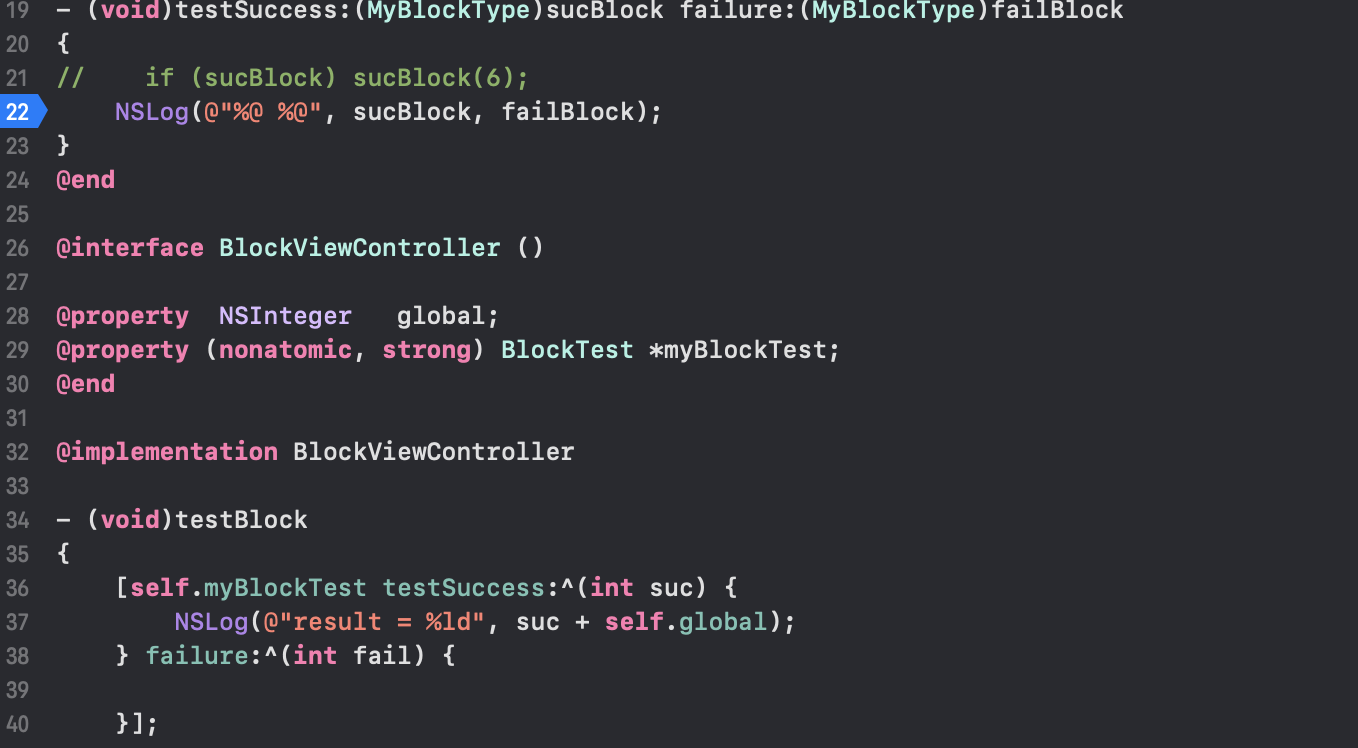
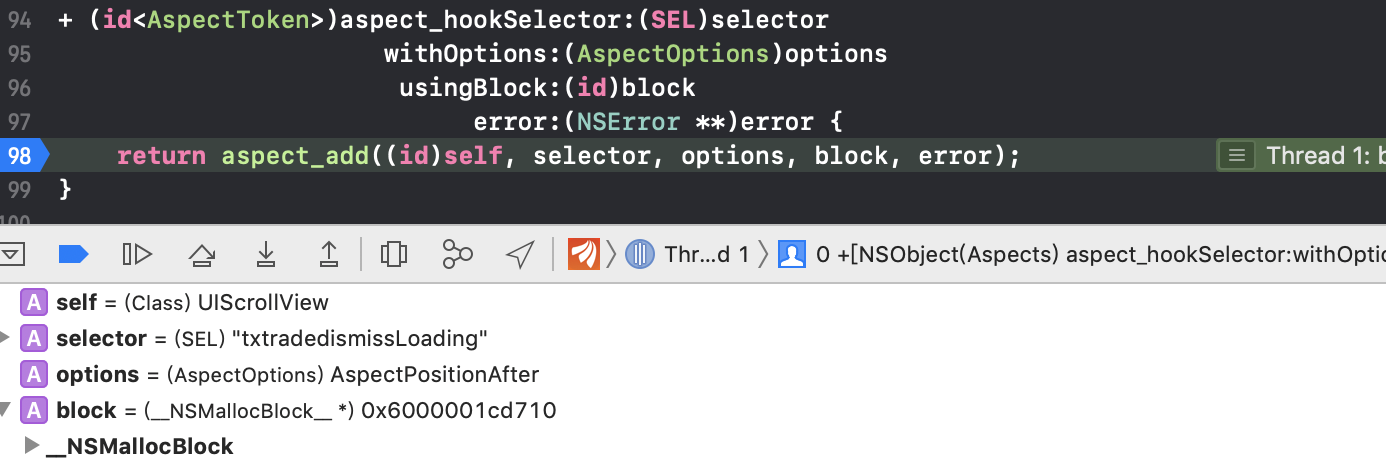
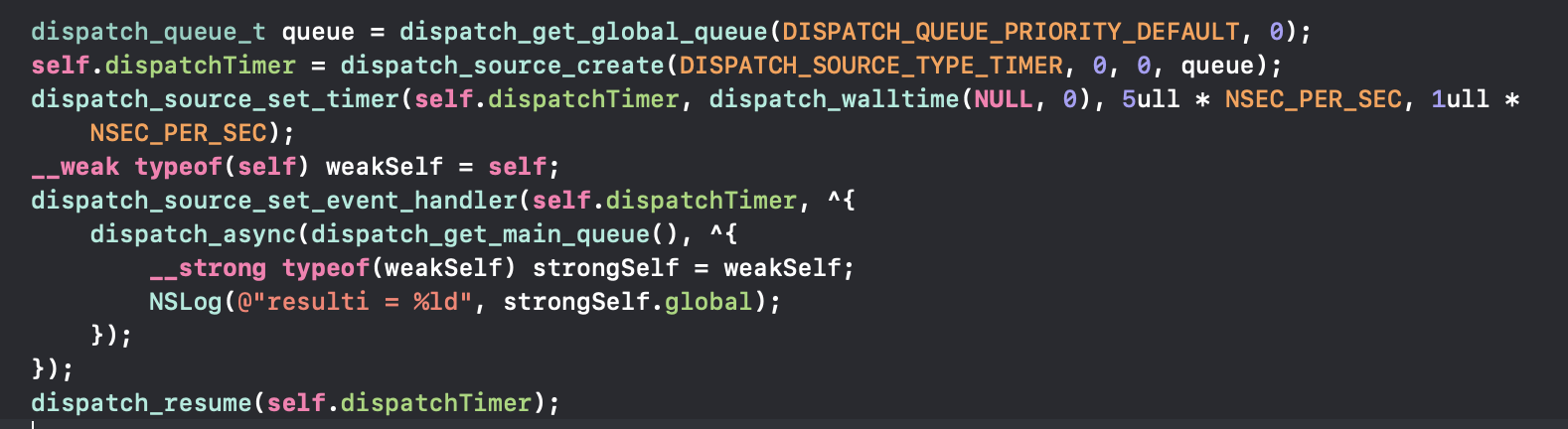
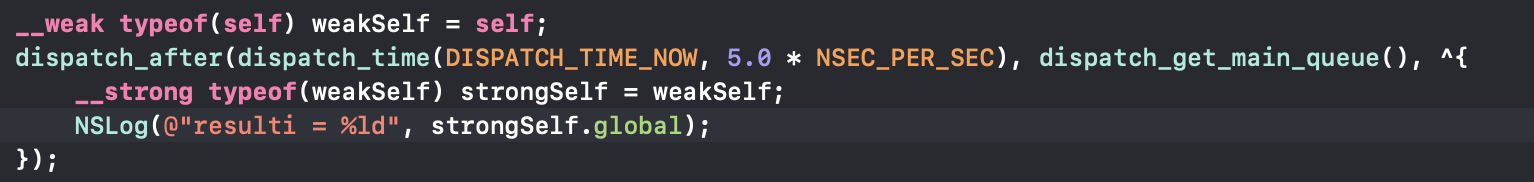
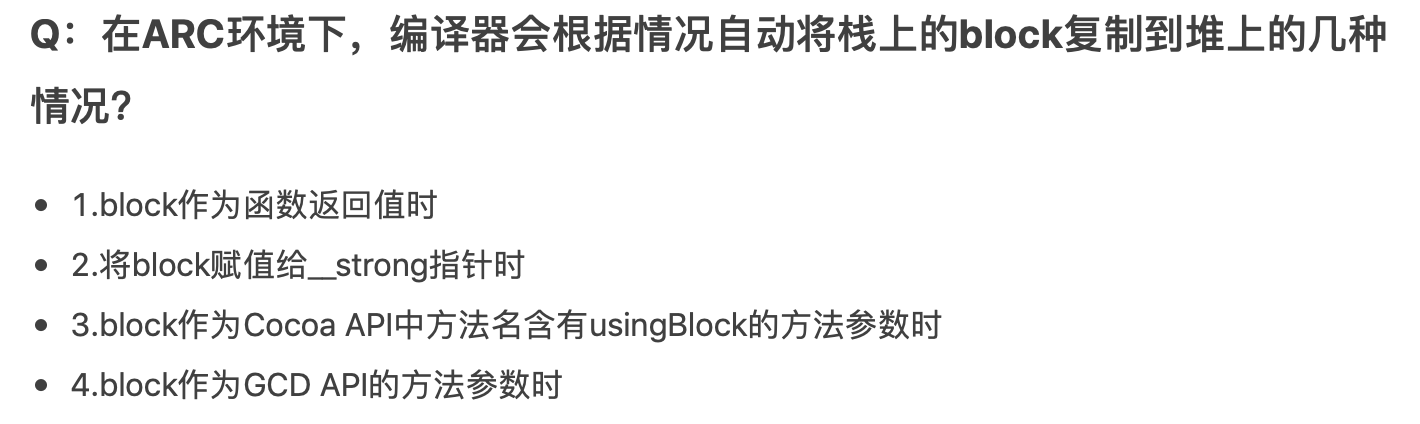
# Block

1. Block定义声明 
2. Block截获局部变量值(基本数据类型截获值;对象类型连同所有权修饰符一截获;静态变量以指针形式截获); 不截获全局变量
   1. 将block内部使用的局部变量的值复制到block数据结构中以实现访问(只能访问,不能修改) 
   2. 将block内部使用的局部变量的地址复制到block数据结构中以实现访问(可访问,可修改)
3. Block的存储域
   1. 全局块(\_NSConcreteGlobalBlock) //存在全局内存中,相当于单例
   2. 栈块(\_NSConcreteStackBlock)//存在栈内存中,超出作用域马上被销毁
   3. 堆块(\_NSConcreteMallocBlock)//存在堆内存中,是一个带引用计数的对象,需自行管理内存
4. 判断block的存储位置
   1. Block不访问栈中和堆中的变量, 或只访问全局变量, 存储在全局内存中 
   2. 访问外界变量的block实际存放在栈区, ARC环境下自动拷贝到堆区,自动释放
   3. 不访问栈中和堆中变量的block作为函数参数时存放在全局区, 访问外部变量的block作为函数参数时存放在栈区 
5. 循环引用实例
   1. 未发生循环引用(栈内存, 方法调用结束后, block持有的self也会被释放, 如果方法调用比较慢, 比如网络不好时请求回调比较慢, 这时self需要等到请求方法执行结束才会释放) 
   2. 未发生循环引用 
   3. 可能发生循环引用(传入的block如果有截获self, 必须先转化成弱引用) 
   4. 循环引用(需使用weakSelf) 
   5. 不会循环引用, 但是如果block不执行的话, self就一直不释放, 所以还是使用weakSelf
   6. 不会循环引用(虽然是堆block, 但执行完了自动释放) 
6. 总结
   1. 全局block不会发生循环引用, 因为不会截获任何外部变量, 包括self;
   2. 栈block也不会发生循环引用, 但如果方法执行很慢, 必须等到方法执行完毕才能释放其参数block截获的self 变量, 所以为了及时释放, 可以将self转化成弱引用, 这样有可能导致方法回调时, 其block使用self时, self已经被释放;
   3. 堆block可能会发生循环引用, 发生循环引用时必须使用self的弱引用.
7. 例如:
   1. UIView动画方法block内使用self不会导致内存泄漏, 哪怕是正在调用过程中;
   2. Dispatch\_async的block使用self要区分有没有延时, 有延时最好使用weakSelf, 否则一旦运行到该代码段, 却不能及时返回, 则会造成内存泄漏, 不运行没有问题, 运行后立即返回, 也没有问题;
   3. 单例的block需要使用weakSelf, 否则运行到该代码段则一定会内存泄漏(不管是否立即返回, 都无法释放), 不运行没关系;
   4. 循环引用肯定会内存泄漏(无法释放), 此时需要使用weakSelf破除循环;
   5. 对于定时器和观察者, 也要选择合适的方法避免内存泄漏;
   6. 一般情况静态方法不会造成内存泄漏, 但如果方法内部包含请求, 请求完后回调block, 这时block也需要使用weakSelf, 因为延时会导致对象不能立即释放;
   7. …

总之, 我们只需要从block最终被调用的位置去判断属于以上哪种情况, 就可以轻松的决定是否需要使用weakSelf.

## 要了解更多信息并获取 OneNote，请访问 [www.onenote.com](http://go.microsoft.com/fwlink/?LinkID=523891)。