内存管理

1. iOS5.0 之前，OC 内存管理遵循“谁创建，谁释放；谁引用，谁管理”的原则。
2. 当创建或引用一个对象的时候，需向她发送 alloc、copy、new、retain 消息；当释放对象时需发送 release 消息；当对象引用计数为 0 时，系统将释放该对象，这是 OC 的手动管理机制 MRC
3. Ios5.0 之后，引用自动管理机制——自动引用计数(ARC)。管理原则一样，知识不再需要手动调用 retain、release、autorelease
4. ARC 编译特性：在适当位置自动插入 release 和 autorelease
5. ARC 引入 strong 和 weak 关键字：strong 是强引用，只要引用存在，对象就不能被销毁。weak 是弱引用，当被引用对象消失时，弱引用会自动设置为 nil。二者区别是，当一个对象不再有强引用指针指向时候它就会释放，即使还有 weak 的指针指向
6. 谁创建，谁释放：如果使用 alloc、copy(mutableCopy)或 retain 一个对象时，你就有义务，向它发送一条 release 或 autorelease 消息。其他方法创建的对象，不需由你来管理内存
7. 向一个对象发送一条 autorelease 消息，这个对象不会立即被销毁，而是将这个对象放入自动释放池，待池子释放时，它会向池中每个对象发一个 release 消息，以此来释放对象
8. 向一个对象发送 release 消息，并不意味这个对象被销毁了，而是当这个对象的引用计数为 0 时系统才会调用对象的 dealloc 方法，释放该对象和对象本身所拥有的实例。
9. 如果一个对象有一个 strong 类型的指针指向着，这个对象就不会被释放(强引用某对象)引用计数＋1
10. 如果一个指针指向超出了它的作用域，就会被指向 nil
11. 如果一个指针被指向 nil，那么它原来指向的对象就被释放了
12. 当一个视图控制器被释放时，它内部的全局指针都会被指向 nil
13. 局部变量：出了作用域，指针会被置为 nil
14. 方法内部创建对象，外部需要添加 autorelease
15. 连线时用 weak 描述
16. block 中为了避免循环引用问题，使用 weak 描述

17. 如果需使用自动释放池，用@autoreleasepool{};

18. 在非 ARC 工程中才用 ARC 去编译某些类:在 compile sources 双击，-fobjc-arc

19. 在 ARC 工程中才用 MRC 去编译某些类:在 compile sources 双击，-fno-objc-arc

20. iOS 定义@propery 属性时需遵循内存管理语义，添加修饰符：
assign,weak,retain,strong,copy，以下分别进行介绍：

1) assign：用于基本数据类型进行赋值操作，不更改引用计数，也可以修饰对象，但 assign 修饰对象在释放后指针地址还在，指针并没有置为 nil，指向被销毁的内存，称为野指针。后续往该对象发消息会 crash 基础类型一般分配在栈上，栈内存由系统自动处理，不会造成野指针

2) weak：修饰 object 类型，对象释放后，指针会置为 nil，是一种弱引用。与 strong 不同的是，当一个对象不再有强类型指针指向它时，就会被释放，即使有其他 weak 指针指向，也会被清除

3) strong：相当于 retain，但有些类型如 NSString，使用 strong 相当于使用 copy

4) copy：建立一个索引计数为 1 的对象后释放旧对象，赋值的是一份深拷贝，即内容拷贝。注：若不可变对象，则为浅拷贝，指针拷贝。

5) retain：释放旧值，新值 retainCount＋1，只指针拷贝(浅拷贝)

6) block 用 copy 属性修饰符:block 分配在栈上随时会被回收,copy 一份到堆上面，就不会了。在 ARC 下无修饰也会自动进行 copy 的

7) __block：在 ARC、MRC 下均可用，修饰对象或基本数据类型。__block 修饰对象可以在 block 中重新被赋值，但__weak 不行

8) __weak 只在 ARC 中使用，只能修饰对象，不能修饰基本数据类型

9) ARC 下，为避免 block 出现循环引用，经常会__weak typeof(self) weakSelf = self;

10) Parent-child 相互持有、委托模式引起循环引用：让 child 只有 parent 对象为 weak 类型

11) 类 property 用 strong、copy 等持有了 block 对象，如果 block 再持有 self 引起循环引用；用 9)中方法可以避免

12) NSTimer 会持有对象，若类 property 持有了 nstimer，timer 又持有了 self 引起循环引用：在删除 timer 前调[timer invalidate]; _timer = nil;