打造最受名企欢迎的iOS程序员（上）

本文编辑：竹_子

整理的对你有帮助可以关注作者简书：小猿员

本文分为上下两篇，此篇是上篇（1-50），下篇（51-110），此文档由一个人整理，如果有语法错误，排版错误，请联系作者，谢谢！

（下篇免费获取方式：加iOS技术交流群624212887，私聊群主，免费获取下篇；同时欢迎大家进群多多交流）

面试题合集：

1、堆和栈什么区别？

答：管理方式：对于栈来讲，是由编译器自动管理，无需我们手工控制；对于堆来说，释放工作由程序员控制，容易产生memory leak。

2、数组和链表什么区别？

答:数组是将元素在内存中连续存放,由于每个元素占用内存相同，可以通过下标迅速访问数组中任何元素。

链表恰好相反，链表中的元素在内存中不是顺序存储的，而是通过存在元素中的指针联系到一起。

3、delegate和notification什么区别，什么情况使用？

答：Delegate:

消息的发送者(sender)告知接收者(receiver)某个事件将要发生，delegate同意然然后发送者响应事件，

delegate机制使得接收者可以改变发送者的行为。

通常发送者和接收者的关系是直接的一对多的关系。

Notification:

消息的发送者告知接收者事件已经发生或者将要发送，仅此而已，接收者并不能反过来影响发送者的行为。

通常发送者和接收者的关系是间接的多对多关系。

4、什么是MVC，为什么使用MVC，有什么好处？

答： 分别为： 模型(Model),视图(View)和控制Controller)。

模型（Model） “数据模型”（Model）用于封装与应用程序的业务逻辑相关的数据以及对数据的处理方法。

“模型”有对数据直接访问的权力，例如对数据库的访问。

视图（View） 视图层能够实现数据有目的显示。

控制器（Controller） 控制器起到不同层面间的组织作用，用于控制应用程序的流程

5、从一个数组中找出重复的元素打印出来

```
NSArray *arr = [NSArrayarrayWithObjects:@"1",@"2",@"1",@"7",@"4",@"5",@"2",@"6",@"5",nil];
NSMutableArray *arrmu = [[NSMutableArrayalloc]init];//过滤
NSMutableArray *sameArray = [[NSMutableArrayalloc]init];//找出相同的
for (int i = 0 ; i < [arrcount]; i++) {

    id str = [arr objectAtIndex:i];
      if ([arrmu count] == 0)
       {
            [arrmuaddObject:str];
       }
else{
            BOOL flag = NO;
            for (int j = 0; j < [arrmucount]; j++ ) {
                  if ([strisEqual:[arrmu objectAtIndex:j]])
                   {
                        [sameArrayaddObject:str];
                        flag =YES;
```

```
                    break;
                }
                else{
            flag =NO;
            }
            }
            if (flag == NO) {

                [ arrmuaddObject:str];
            }
        }
            }
    NSLog(@"sameArray : %@",sameArray);
```

还有两种直接找出的方法，上代码：

一：

```
NSArray *arr = [NSArrayarrayWithObjects:@"1",@"2",@"1",nil];
     NSSet *set = [NSSetsetWithArray:arr];
    NSLog(@"%@",[setallObjects]);
```

二：

```
NSArray *arr =@[@1,@2,@1];
    NSMutableDictionary *dict =
[NSMutableDictionarydictionary];
    for (NSNumber *numberin arr) {
        [dictsetObject:numberforKey:number];
     }
            NSLog(@"%@",[dictallValues]);
```

6、UITableView能否绑定多个数据源？

答：不能

7、一个UIViewController能否管理多个UITableView ？

答：可以

8.说一下响应链的响应机制

9、同步请求和异步请求什么区别？
答：发送同步请求，程序将停止用户交互，直至服务器返回数据完成，才可以进行下一步操作，

异步请求不会阻塞主线程，而会建立一个新的线程来操作，用户发出异步请求后，依然可以对UI进行操作，程序可以继续运行

10、iOS中的多线程操作、多线程方式
答：iOS提供了方便的多线程调用机制:NSOperation和NSOperationQueue。它的使用方法也很简单，

一般都是将NSOperation对象加入到NSOperationQueue队列中，加入后队列就开始处理，知道任务操作完成。

11、UIViewController的生命周期
答：当一个视图控制器被创建，并在屏幕上显示的时候。 代码的执行顺序

// 视图显示在屏幕上的顺序

1、 alloc 创建对象，分配空间。

2、init (initWithNibName) 初始化对象,初始化数据。

3、loadView 从nib载入视图,这一步不需要去干涉。除非没有使用xib文件创建视图。

4、viewDidLoad 加载完毕，可以进行自定义数据以及动态创建其他控件。

5、viewWillAppear 视图将出现在屏幕之前，马上这个视图即将显示在屏幕上。

6、viewDidAppear 视图已在屏幕上渲染完成。

// 视图将被从屏幕上移除的顺序

1、viewWillDisappear 视图将被从屏幕上移除之前执行。

2、viewDidDisappear 视图已经被从屏幕上移除。

3、dealloc 视图被销毁，此时需要在init和viewDidLoad中创建的对象进行释放。

4、viewDidUnload 出现内存警告在内存不足时执行，并对所有非当前显示的controller执行。

本视图的所有子视图将被销毁，以释放内存,此时开发者需要手动对viewLoad、viewDidLoad中创建的对象释放内存。

因为当这个视图再次显示在屏幕上的时候，viewLoad、viewDidLoad 再次被调用，以便再次构造视图

12、Autorerelease对象什么时候释放？

答：autorelease实际上只是把对release的调用延迟了，对于每一个Autorelease，系统只是把该Object放入了当前的Autorelease pool中，当该pool被释放时，该pool中的所有Object会被调用Release。

13、iOS数据持久化方式

答：四种：属性列表、对象归档、SQLite3和Core Data

14、Object－c的类可以多重继承么？可以实现多个接口么？Category是什么？重写一个类的方式用继承好还是分类好？为什么？

答：Object-c的类不可以多重继承；可以实现多个接口，通过实现多个接口可以完成C++的多重继承；Category是类别，一般情况用分类好，用Category去重写类的方法，仅对本Category有效，不会影响到其他类与原有类的关系。

15. #import 跟#include 又什么区别，@class呢, ＃import<> 跟#import”"又什么区别？

答：#import是Objective-C导入头文件的关键字，#include是C/C++导入头文件的关键字,使 用#import头文件会自动只导入一次，不会重复导入，相当于#include和#pragma once；@class告诉编译器某个类的声明，当执行时，才去查看类的实现文件，可以解决头文件的相互包含；#import<>用来包含系 统的头文件，#import”“用来包含用户头文件。

16.属性readwrite，readonly，assign，retain，copy，nonatomic 各是什么作用，在那种情况下用？

答：readwrite 是可读可写特性；需要生成getter方法和setter方法时

readonly 是只读特性　只会生成getter方法 不会生成setter方法 ;不希望属性在类外改变

assign 是赋值特性，setter方法将传入参数赋值给实例变量；仅设置变量时；

retain 表示持有特性，setter方法将传入参数先保留，再赋值，传入参数的retaincount会+1;

copy 表示赋值特性，setter方法将传入对象复制一份；需要完全一份新的变量时。

nonatomic 非原子操作，决定编译器生成的setter getter是否是原子操作，atomic表示多线程安全，一般使用nonatomic

17.常见的object-c的数据类型有那些， 和C的基本数据类型有什么区别？如：NSInteger和int

答：object-c的数据类型有 NSString，NSNumber，NSArray，NSMutableArray，NSData等等，这些都是class，创建后便是对象，而C语言的基本数据类型int，只是一定字节的内存空间，用于存放数值；而object-c的NSNumber包含有父类NSObject的方法和 NSNumber自己的方法，可以完成复杂的操作。

18.Objective-C如何对内存管理的,说说你的看法和解决方法?

答：Objective-C的内存管理主要有三种方式ARC（自动内存计数）、手动内存计数、内存池。

解决方法的话： 谁持有，谁释放。

19.如何对iOS设备进行性能测试?

答：Profile-> Instruments ->Time Profiler

20.Object C中创建线程的方法是什么？如果在主线程中执行代码，方法是什么？如果想延时执行代码、方法又是什么？

答：线程创建有三种方法：使用NSThread创建、使用 GCD的dispatch、使用子类化的NSOperation,然后将其加入NSOperationQueue;在主线程执行代码，方法是performSelectorOnMainThread，如果想延时执行代码可以用performSelector:onThread:withObject:waitUntilDone:

21.描述一下iOS SDK中如何实现MVC的开发模式

答：MVC是：模型--视图--控制　开发模式，对于iOS SDK，所有的View都是视图层的，它应该独立于模型层，由视图控制层来控制。所有的用户数据都是模型层，它应该独立于视图。所有的ViewController都是控制层，由它负责控制视图，访问模型数据。

22.定义属性时，什么情况使用copy、assign、retain?

答：assign用于简单数据类型，如NSInteger,double,bool, 其实还有后面的block等;

retain和copy用于对象，copy用于当a指向一个对象，b也想指向同样的对象的时候，如果用assign，a如果释放，再调用b会crash,如果用copy 的方式，a和b各自有自己的内存，　就可以解决这个问题。

retain 会使计数器加一，也可以解决assign的问题。

另外:atomic和nonatomic用来决定编译器生成的getter和setter是否为原子操作。在多线程环境下，原子操作是必要的，否则有可能引起错误的结果。

加了atomic，setter函数会变成下面这样：

```
if (property != newValue) {
[property release];
property = [newValue retain];
```

}

23. Object-C有私有方法吗？私有变量呢？

答：objective-c - 类里面的方法只有两种，静态方法和实例方法，所有实例变量默认都是私有的，所有实例方法默认都是公有的。

24.浅复制和深复制的区别？//浅拷贝和深拷贝

答案：

浅层复制（copy）：只复制指向对象的指针，而不复制引用对象本身。//通过对象的指针来访问这个对象深层复制（mutableCopy）：复制引用对象本身意思就是有个A对象，复制一份后得到A_copy对象后，对于浅复制来说，A和A_copy指向的是同一个内存资源，复制的只不过是是一个指针，对象本身资源还是只有一份，那如果我们对A_copy执行了修改操作,那么发现A引用的对象同样被修改,这其实违背了我们复制拷贝的一个思想。深复制就好理解了,内存中存在了两份独立对象本身。//当修改A时，A copy不变。

打个比喻：1、浅拷贝就是：你挂了，你妈妈喊你回家吃饭时找不到人了，她很伤心。2、深拷贝就是：你克隆了一个你自己：你挂了、你兄弟还在，你妈妈喊你回家吃饭时能找到人。所以、孩子，安全起见、深拷贝吧，记得内存管理就是了。

25.自动释放池是什么,如何工作

答： 当您向一个对象发送一个autorelease消息时，Cocoa就会将该对 象的一个引用放入到最新的自动释放池。

它仍然是个正当的对象,因此自动释放池定义的作用域内的其它对象可以向它发送消息。当程序执行到作用域结束的位置时，

自动释放池就会被释放，池中的所有对象也就被释放

26.单件实例是什么

答： Foundation 和 Application Kit 框架中的一些类只允许创建单件对象，即这些类在当前进程中的唯一实例。

举例:NSFileManager 和NSWorkspace类在使用时都是基于进程进行单件对象的实例化。

当向这些类请求实例的时候，它们会向您传递单一实例的一个引用，如果该实例还不存在，则首先进行实例的分配 和初始化。

27.类别的作用？继承和类别在实现中有何区别？

答：category 可以在不获悉，不改变原来代码的情况下往里面添加新的方法，只能添加，不能删除修改。

并且如果类别和原来类中的方法产生名称冲突，则类别将覆盖原来的方法，因为类别具有更高的优先级。 类别主要有3个作用：

(1)将类的实现分散到多个不同文件或多个不同框架中。

(2)创建对私有方法的前向引用。

(3)向对象添加非正式协议。

继承可以增加，修改或者删除方法，并且可以增加属性

28.类别和类扩展的区别。

答：category和extensions的不同在于 后者可以添加属性。另外后者添加的方法是必须要实现的。 extensions可以认为是一个私有的Category

29.KVO and KVC?

答：kvc:键 - 值编码是一种间接访问对象的属性，使用字符串来标识属性，而不是通过调用存取方法，直接或通过实例变量访问的机制。 很多情况下可以简化程序代码。apple文档其实给了一个很好的例子。 kvo:键值观察机制，他提供了观察某一属性变化的方法，极大的简化了代码。 具体用看到用到过的一个地方是对于按钮点击变化状态的的监控。 比如我自定义的一个button [cpp]

```
  [self addObserver:self forKeyPath:@"highlighted" options:0
context:nil];
    #pragma mark KVO
  - (void)observeValueForKeyPath:(NSString *)keyPath
ofObject:(id)object change:(NSDictionary *)change
context:(void *)context   {
           if ([keyPath isEqualToString:@"highlighted"] )
{
                [self setNeedsDisplay];
     }
      }
```

对于系统是根据keypath去取的到相应的值发生改变，理论上来说是和kvc机制的道理是一样的。 对于kvc机制如何通过key寻找到value：

“当通过KVC调用对象时，比如：[self valueForKey:@”someKey”]时，程序会自动试图通过几种不同的方式解析这个调用。首先查找对象是否带有 someKey 这个方法，如果没找到，会继续查找对象是否带有someKey这个实例变量（iVar），如果还没有找到，程序会继续试图调用
-(id) valueForUndefinedKey:这个方法。如果这个方法还是没有被实现的话,程序会抛出一个NSUndefinedKeyException异常错误。

(cocoachina.com注：Key-Value Coding查找方法的时候，不仅仅会查找someKey这个方法，还会查找getsomeKey这个方法，前面加一个get，或者_someKey以及_getsomeKey这几种形式。同时，查找实例变量的时候也会不仅仅查找someKey这个变量，也会查找_someKey这个变量是否存在。） 设计valueForUndefinedKey:方法的主要目的是当你使用-(id)valueForKey方法从对象中请求值时，对象能够在错误发生前，有最后的机会响应这个请求。这样做有很多好处，下面的两个例子说明了这样做的好处。“ 来至cocoa，这个说法应该挺有道理。

因为我们知道button却是存在一个highlighted实例变量.因此为何上面我们只是add一个相关的keypath就行了，

可以按照kvc查找的逻辑理解，就说的过去了

30.代理的作用？

答：代理的目的是改变或传递控制链。允许一个类在某些特定时刻通知到其他类，而不需要获取到那些类的指针。可以减少框架复杂度。 另外一点，代理可以理解为java中的回调监听机制的一种类似。

31.说说响应链

答： 事件响应链。包括点击事件，画面刷新事件等。在视图栈内从上至下，或者从下之上传播

32.frame和bounds有什么不同？

答：frame指的是：该view在父view坐标系统中的位置和大小。（参照点是父亲的坐标系统） bounds指的是：该view在本身坐标系统中 的位置和大小。（参照点是本身坐标系统）

33.方法和选择器有何不同？

答：selector是一个方法的名字，method是一个组合体

34.Object－c的类可以多重继承么？可以实现多个接口么？重写一个类的方式用继承好还是分类好？为什么？

答：Objective-c只支持单继承，如果要实现多继承的话，可以通过类别和协议的方式来实现，cocoa 中所有的类都是NSObject 的子类，多继承在这里是用protocol 委托代理 来实现的。

35.ARC自动引用技术

答:1.ARC是编译特性，不是运行时特性，只是在编译的时候，编译器会自动加上释放代码

2.不能调用release、retain、autorelease、retainCount

3.dealloc注意

1> 不能在dealloc中调用[super dealloc]

2> 不能在dealloc中释放资源

4.@property参数说明

1> retain 改为 strong

2> 基本数据类型(int\float)还是用assign

3> copy 还是 copy

4> 如果2个对象循环引用，一端用strong，一端用weak

5> weak是用在对象上，weak其实作用跟assign相当

5.ARC中只允许使用通过@autoreleasepool {}创建自动释放池

36 GCD技术

答:Grand Central Dispatch简称GCD 解决多核并行运算的一种方案

看代码就行：

```
//   Grand Central Dispatch简称GCD技术

// Do any additional setup after loading the view.

//      dispatch_queue_t newDispath =
dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT,
0);
//      dispatch_async(newDispath, ^{
//            [self downloadImage];
//      });
//      #defineDISPATCH_QUEUE_PRIORITY_HIGH   2
//      #defineDISPATCH_QUEUE_PRIORITY_DEFAULT   0
//      #defineDISPATCH_QUEUE_PRIORITY_LOW (-2)
//      #defineDISPATCH_QUEUE_PRIORITY_BACKGROUNDINT16_MIN
```

/*dispatch queue分为下面三种：

* Serial:又称为private dispatch queues，同时只执行一个任务。Serial queue通常用于同步访问特定的资源或数据。当你创建多 个 Serial queue时，虽然它们各自是同步执行的，但Serial queue与Serial queue之间是并发执行的。

* Concurrent: 又称为global dispatch queue，可以并发地执行多个任务，但是执行完成的顺序是随机的。

```
      * Main dispatch queue它是全局可用的serial queue，它是在应用程序主线程上执行任务的
      */

     //   一般GCD 可以如下操作

     dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT,0), ^{
            // 耗时的操作
           dispatch_async(dispatch_get_main_queue(), ^{
                 // 更新界面
            });
      });

   [selfexampleDispatch];



      /*
        *系统给每一个应用程序提供了三个concurrent dispatch queues。
        *这三个并发调度队列是全局的，它们只有优先级的不同。
        *因为是全局的，我们不需要去创建。我们只需要通过使用函数dispath_get_global_queue去得到队列
        */

     dispatch_queue_t globalQ =dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT,0);

      NSLog(@"global:%p",globalQ);
      dispatch_queue_t mainQ =dispatch_get_main_queue();
        NSLog(@"mainQ:%p",mainQ);
      /*
```

```
        *虽然dispatch queue是引用计数的对象，但是以上两个都是全局的队列，不用retain或release。
            */

       /*
        *dispatch_group_async可以实现监听一组任务是否完成，完成后得到通知执行其他的操作。
        *这个方法很有用，比如你执行三个下载任务，当三个任务都下载完成后你才通知界面说完成的了。
        */
       timeInt = 0;
       [NSTimerscheduledTimerWithTimeInterval:1
                                                    targe
t:self
                                                  selector:
@selector(checkingTime)
                                                  userInfo:
nil
                                                    repeat
s:YES];
          [selfexampleDispath_group];

   /*dispatch_barrier_async的使用
        *dispatch_barrier_async是在前面的任务执行结束后它才执行，而且它后面的任务等它执行完成之后才会执行
        */

       [selfexampleDispatch_barrier];



      /*dispatch_apply
        *执行某个代码片段N次。
        */
```

```
dispatch_apply(5, globalQ, ^(size_t index) {
    // 执行5次
});
```

类别的作用？继承和类别在实现中有何区别？

答案：category 可以在不获悉，不改变原来代码的情况下往里面添加新的方法，只能添加，不能删除修改。// category:类、种类并且如果类别和原来类中的方法产生名称冲突，则类别将覆盖原来的方法，因为类别具有更高的优先级。//类别跟类的优先级类别主要有3个作用：(1)将类的实现分散到多个不同文件或多个不同框架中。(2)创建对私有方法的前向引用。(3)向对象添加非正式协议。 继承可以增加，修改或者删除方法，并且可以增加属性。//非正式协议:是使用类别category来实现，非正式协议是NSObject的一个类别，这样任何类的对象都可以作为委托对象来使用，它可以列出对象能够执行的所有方法，这样用来实现委托， 我们可以使用选择器来判断该非正式协议中是否有这个方法。正式协议:是一个命名的方法列表，与非正式协议相比不同的是，它要求显示的采用协议，采用协议的方法是在类的@ｉｎｔｅｒｆａｃｅ声明中列出协议的名称，此时，实现协议的类应该遵守协议，承诺实现协议中的所有方法。

37.代理的作用？

答案：代理的目的是改变或传递控制链。允许一个类在某些特定时刻通知到其他类，而不需要获取到那些类的指针。可以减少框架复杂度。

38.我们说的oc是动态运行时语言是什么意思？

答案：多态。 主要是将数据类型的确定由编译时，推迟到了运行时。这个问题其实浅涉及到两个概念，运行时和多态。运行时机制使我们直到运行时才去决定一个对象的类别，以及调用该类别对象指定方法。多态：不同对象以自己的方式响应相同的消息的能力叫做多态。

//都用有一个相同的方法，不同的对象以自己的方式响应了相同的消息.因此也可以说，运行时机制是多态的基础

39.通知和协议的不同之处？

答案：协议有控制链(has-a)的关系，通知没有。//通知:它可以一对多，一条消息可以发送给多个消息接受者,但是不会处理消息控制链：单一拥有和可控制的对应关系。

40.关于多态性

答案：多态，子类指针可以赋值给父类。对象不仅仅可以已本身的类型存在，也可以作为其父类类型存在。 多态性是允许将父对象设置成为和一个或多个它的子对象相等的技术， 多态性使得能够利用同一类(基类)类型的指针来引用不同类的对象,以及根据所引用对象的不同,以不同的方式执行相同的操作.

41. NSOperation队列

操作和操作队列，基本可以看成java中的线程和线程池的概念。用于处理ios多线程开发的问题。网上部分资料提到一点是，虽然是queue，但是却并不是带有队列的概念，放入的操作并非是按照严格的先进现出。这边又有个疑点是，对于队列来说，先进先出的概念是Afunc添加进队列，Bfunc紧跟着也进入队列，Afunc先执行这个是必然的，但是Bfunc是等Afunc完全操作完以后，B才开始启动并且执行，因此队列的概念离乱上有点违背了多线程处理这个概念。但是转念一想其实可以参考银行的取票和叫号系统。因此对于A比B先排队取票但是B率先执行完操作，我们亦然可以感性认为这还是一个队列。但是后来看到一票关于这操作队列话题的文章，其中有一句提到“因为两个操作提交的时间间隔很近，线程池中的线程，谁先启动是不定的。”瞬间觉得这个queue名字有点忽悠人了，还不如pool~综合一点，我们知道他可以比较大的用处在于可以帮组多线程编程就好了。

42.是否在一个视图控制器中嵌入两个tableview控制器？

答案：一个视图控制只提供了一个View视图，理论上一个tableViewController也不能放吧，只能说可以嵌入一个tableview视图。而是宏观的表示视图控制者，那我们倒是可以把其看成一个视图控制者，它可以控制多个视图控制器，比如TabbarController

43. 什么是id类型

id类型的变量可以存放任何数据类型的对象。在内部处理上，这种类型被定义为指向对象的指针，实际上是一个指向这种对象的实例变量的指针。例如：id number将number声明为id类型的变量。可声明的方法使其具有id类型的返回值，如下：-（id）newObject;(int)type;这个程序声明了一个名为newObject的实例方法，它具有名为type的单个整型参数并有id类型的返回值。应该注意的是，对返回值和参数类型声明来说，id是默认的类型。id类型是objetive-c中非常中药店额数据类型，它是多态和动态绑定的基础。

44.请简要说明viewDidLoad和viewDidUnload何时调用
答：
viewDidLoad在view从nib文件初始化时调用，
loadView在controller的view为nil时调用。
此方法在编程实现view时调用，view控制器默认会注册memory warning notification，
当view controller的任何view没有用的时候，
viewDidUnload会被调用，在这里实现将retain的view release，如果是retain的IBOutlet view 属性则不要在这里release，IBOutlet会负责release 。

45.打印结果

```
main()
  {
     int a[5]={1,2,3,4,5};
```

```
    int *ptr=(int *)(&a+1);
    printf(“%d,%d”,*(a+1),*(ptr-1));
}
```

答：2,5

(a+1) 就是a[1]，(ptr-1)就是a[4],执行结果是2，5
&a+1不是首地址+1，系统会认为加一个a数组的偏 移，是偏移了一个数组的大小（本例是5个int）
int *ptr=(int *)(&a+1);
则ptr实际 是&(a[5]),也就是a+5
原因如下：
&a是数组指针，其类型为 int (*)[5];
而 指针加1要根据指针类型加上一定的值，不同类型的指针+1之后增加的大小不同。
a是长度为5的int数组指针，所以要加 5*sizeof(int)
所以ptr实际是a[5]
但是prt与(&a+1)类型是不一样的(这点很重要)
所以prt-1只会减去sizeof(int*)
a,&a的地址是一样的，但意思不一样
a是数组首地址，也就是a[0]的地址，&a是对象（数组）首地址，
a+1是数组下一元素的地址，即a[1],&a+1是下一个对象的地址，即a[5].

```
void Func ( char str[100] )
{
sizeof(str ) = ?
}
void*p = malloc( 100 ); sizeof( p ) = ?
```

这题 很常见了,Func (char str[100])函数中数组名作为函数形参时，在函数体内，数组名失去了本身的内涵，仅仅只是一个指针；在失去其内涵的同时，它还失去了其常量特性，可以作自增、自减等操作，可以被修改。Windows NT 32位平台下，指针的长度（占用内存的大小）为4字节，故sizeof(str) 、sizeof(p) 都为4。

46.写一” 标准”宏MIN ，这个宏输入两个参数并返回较小的一个
答：#define MIN(A,B) （（A） <= (B) ? (A) : (B))
这个测试是为下面的目的而设的：
标识#define在宏中应用的基本知识。这是很重要的，因为直到嵌入(inline)操作符变为标准C的一部分，宏是方便产生嵌入代码的唯一方
法，对于嵌入式系统来说，为了能达到要求的性能，嵌入代码经常是必须的方法。
三重条件操作符的知识。这个操作符存在C语言中的原因是它使得编译器能产生比 if-then-else 更优化的代码，了解这个用法是很重要的。 懂得在宏中小心地把参数用括号括起来 我也用这个问题开始讨论宏的副作用，例如：当你写下面的代码时会发生什么
事？ least = MIN(*p++, b);
结果是：
((*p++) <= (b) ? (*p++) :(*p++))
这个表达式会产生副作用，指针p会作三次++自增操作。

47.数组和指针的区别
（1）数组可以申请在栈区和数据区；指针可以指向任意类型的内存块
（2）sizeof作用于数组时，得到的是数组所占的内存大小；作用于指针时，得到的都是4个字节的大小
（3）数组名表示数组首地址，值不可以改变，如不可以将＋＋作用于数组名上；普通指针的值可以改变，如可将＋＋作用于指针上
（4)用字符串初始化字符数组是将字符串的内容拷贝到字符数组中；用字符串初始化字符指针是将字符串的首地址赋给指针，也就是指针指向了该数组

48.static的作用
（1）函数体内static 变量的作用范围为该函数体，不同于 auto 变量，该变量的内存只被分配一次，
因此其值在下次调用时仍维持上次的值；

（2）在模块内的static 全局变量可以被模块内所用函数访问，但不能被模块外其它函数访问；
（3）在模块内的static 函数只可被这一模块内的其它函数调用，这个函数的使用范围被限制在声明
它的模块内；
（4）在类中的static 成员变量属于整个类所拥有，对类的所有对象只有一份拷贝；
（5）在类中的static 成员函数属于整个类所拥有，这个函数不接收this 指针，因而只能访问类的static 成员变量。

49.简述内存分区情况
（1）代码区：存放函数二进制代码
（2）数据区：系统运行时申请内存并初始化，系统退出时由系统释放。存放全局变量、静态变量、常量
（3）堆区：通过malloc等函数或new等操作符动态申请得到，需程序员手动申请和释放
（4）栈区：函数模块内申请，函数结束时由系统自动释放。存放局部变量、函数参数

50.const char *p;　charconst*p;　char*const p;　const char* const p;四个修饰指针有什么区别
答：　（1）定义了一个指向不可变的字符串的字符指针
（2）和（1）一样
（3）定义了一个指向字符串的指针，该指针值不可改变，即不可改变指向
（4）定义了一个指向不可变的字符串的字符指针，且该指针也不可改变指向