回看以往的面试题，总结整理一番

iOS 交流群：624212887，欢迎入驻，交流探讨，更多技术书籍，面试资料尽在此群！

*面试心声:总结起来就是把基础的东西弄好,复杂的东西了解就 ok 了!

*此题库是北上广深杭各大小公司面试题。

*注:如今社会还是得靠本事,看面试题只是多了一个机会,珍惜机会的同时提高自己硬实力才是真理!

题库：

1.给定一个字符串,输出本字符串中只出现一次并且最靠前的那个字符的位置?
比如“abaccddeeef”则是 b,输出 2
答:

```
 int main()

{

char a[80] = "abaccddeeef\0";

char ch;

int i, m, b[80];

int flag = 0;

ch = getchar();//获取一个字符

m = strlen(a);

for (i = 0; i < m; ++i){

if (a[i] == ch){//找到了，直接判断是否相等

b[flag] = i+1;//记录位置

flag += 1;
```

```
}

}

if (flag == 0)printf ("no");

else {

printf ("%d\n", flag);

for (i = 0; i < flag; i++){//对位置进行输出，用循环

printf ("%d ", b[i]);

}

printf ("\n");

}

return 0;

}
```

2.实现一个冒泡排序或者快速排序
答:冒泡排序:

```
intarray[5] = { 28,27,36,45,8};

for (int i = 0; i < 4; i++) {

for(int j = 0; j < 4; j++) {

if (array[j] > array [j + 1]){

int temp = array[j];

array[j] = array[j + 1];

array[j + 1] = temp;

}}}

for(int i = 0; i < 5; i++) {
```

```
printf("%d\n",array[i]);}
```

3.请编写一个函数用于计算阶乘
答:

```
 int f(int i)

{intt=1,j;

for(j=1;j<=i;j++)

t=t*j;

return t;

}
```

4.Cocoa Touch 提供了几种 Core

Animation 过渡类?

答:Cocoa Touch 提供了 4 种 Core

Animation 过渡类型，分别为：交叉淡化、推挤、显示和覆盖。

5.iOS 平台怎么做数据的持久化?coredata 和 sqlite 有无必然联系?coredata 是一个关系型数据吗?

答:数据的持久化本质上都是就是写文件，但从逻辑上又分成很多种，比如写入沙盒，比如存到网络上，比如写入数据库。

core data 是对 sqlite 的封装，因为 sqlite 是 c 语言的 api，然而有人也需要 obj-c 的 api，所以有了 core data ,另外，core data 不仅仅是把 c 的 api 翻译成 oc 的 api，还提供了一些管理的功能，使用更加方便。

•App 升级之后数据库字段或者表有更改会导致 crash，CoreData 的版本管理和数据迁移变得非常有用，手动写 sql 语句操作还是麻烦一些。

•CoreData 不光能操纵 SQLite，CoreData 和 iCloud 的结合也很好，如果有这方面需求的话优先考虑 CoreData。

•CoreData 并不是直接操纵数据库，比如：使用 CoreData 时不能设置数据库的主键，目前仍需要手动操作。

6.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。

7.#import 跟#include 有什么区别,@class 呢?#import<>跟#import””有什么区别?

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。

8.属性 readwrite,readonly,assin,retain,copy,nonatomic 各是什么作用,在哪种情况下用?

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 表示拷贝特性，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

9.写一个 setter 方法用于完成@property(nonatomic,retain)NSString *name;写一个 setter 方法用于完成@property(nonatomic, copy)NSString *name;

答:

```
 -(void)setName:(NSString *) str

{

[str retain];

[name release];

name = str;

}

- (void)setName:(NSString *)str

{

id t = [str copy];

[name release];

name = t;  }
```

10.对于语句 NSString *obj =[[NSData alloc] init]; obj 在编译时和运行时分别是什么类型的对象?

答:编译时是 NSString 的类型；运行时是 NSData 类型的对象

11.当前已经编程实现函数:int

rand100().该函数可返回 0~99 的随机整数,且可以保证等概率.请利用该函数实现 int rand10000(),要求等概率返回 0~9999 的随机数.(不可使用其他的系统函数)

12.汤姆现在要在家里举行宴会,他虽然有很多筷子,但这些筷子的长度并不完全相同,先已知每根筷子的长度,要求每位客人都能拿到两根长度相同的筷子,求最多可邀请的客人数.

编程实现:int getMax(int arrLength[N])

13.现有一个整数序列,你可以交换其中的任意两个数以得到一个新序列.求共能得到多少种可能结果.(注意:3,3,3,3 无论怎么交换,只能得到一个序列)

编程实现:int getTotal(int arrOrigin[N])

14.现有一个 M 行 N 列的数组,要求安装反向斜对角线(右上->左下)的方式,打印该数组.编程实现:intprintMatrix[int arrMatrix[M][N]]
下面样例的打印顺序为:

```
0->1->4->2->5->8->3->6->9->7->10->11

123

4567

8910 11
```

15.在 UIKit 中,frame 与 bounds 的关系是(C)

A. frame 是 bounds 的别名

B. frame 是 bounds 的继承类

C. frame 的参考系是父规图坐标, bounds 的参考系是自身的坐标

D.frame 的参考系是自身坐标,bounds 的参考系是父规图的坐标

16.一个类的 delegate(代理)的作用不正确的是(D)

A.delegate 中的函数在其他类中实现

B.主要用于不同类型的对象之间一对一传递消息

C.没有指派则不会触发

D.可以一个对象的 delegate 指派给多个其他类型的对象

17.下面关于 Objective-C 内存管理的描述错误的是(A)

A.当使用 ARC 来管理内存时,对象的 retain,dealloc 方法不会被调用

B.autoreleasepool 在 drain 的时候会释放在其中分配的对象

C.当使用 ARC 来管理内存时,在线程中大量分配对象而不用 autoreleasepool 则可能会造成内存泄露

D.在使用 ARC 的项目中不能使用 NSZone

18.下面 block 定义正确的是(A)

A.tyoedef void(^SuccessBlock)(BOOLsuccess);

B. tyoedef void(^SuccessBlock)(NSStringvalue,BOOL success);

C. tyoedef void^(SuccessBlock)(NSStringvalue,BOOL success);

D. tyoedef void^(SuccessBlock)(NSString*value);

19.UIButton 从子类到父类一次继承自:(D)

A. UIView-> UIViewController->UIController

B. UIResponder-> UIControl-> UIView

C. UIControl-> UIResponder->UIViewController

D. UIControl-> UIView-> UIResponder

20.下列关于 iOS 开发中类方法的使用描述,错误的是:(C)

A.类方法可以调用类方法

B.类方法不可以调用实例方法,但是类方法可以通过创建对象来访问实例方法

C.类方法不可以使用实例变量,包括 self(可以使用 self)

D.类方法作为消息,可以被发送到类或者对象里面去

21.什么情况下使用关键字 weak 和 assign 有何不同?

答:assign 指针赋值,不对引用计数操作,使用之后如果没有置为 nil,可能就会产生野指针;而 weak 一旦不进行使用后,永远不会使用了,就不会产生野指针!

22.Object-C 的类可以多重继承么?可以实现多个接口么?Category 是什么?重写一个类方法的方法用继承好还是分类好?为什么?*8

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。

23.如何用 iOS 设备进行性能测试?

答: Profile-> Instruments ->Time Profiler

24.我们说的 oc 是动态运行时语音是什么意思?

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。简单来说，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。意思就是假设生物类（life）都用有一个相同的方法-eat;那人类属于生物，猪也属于生物，都继承了 life 后，实现各自的 eat，但是调用是我们只需调用各自的 eat 方法。也就是不同的对象以自己的方式响应了相同的消息（响应了 eat 这个选择器）。因此也可以说，运行时机制是多态的基础。

25.你的项目什么时候选择使用 GCD,什么时候选择 NSOperation?

答:项目中使用 NSOperation 的优点是 NSOperation 是对线程的高度抽象，在项目中使用它，会使项目的程序结构更好，子类化 NSOperation 的设计思路，是具有面向对象的优点(复用、封装)，使得实现是多线程支持，而接口简单，建议在复杂项目中使用。项目中使用 GCD 的优点是 GCD 本身非常简单、易用，对于不复杂的多线程操作，会节省代码量，而 Block 参数的使用，会是代码更为易读，建议在简单项目中使用。

26.读文件是输入流还是输出流?

东西读入内存就是输入流东西从内存写到记录存储输出流而我们本身就以记录存储为原点所有会有不解的感觉～java io 流按照 java io 流的方向可以分为输入流和输出流输入流是将资源数据读入到缓冲 Buffer 中，输出流是将缓冲 Buffer 中的数据按照指定格式写出到一个指定的位置，所以这两个流一般同时使用，才有意义。例如你要做文件的上传，你要先用输入流将待上传文件读入缓冲，然后用输出流将文件写出到网络服务器的一个位置，则上传成功；若是文件下载，则先获得输入流，来读取网络服务器中的一个文件，然后用输出流写到本地的一个文件中；还有例如文件的拷贝，也是先用输入流读再用输出流写出去的很好的例子，你可以先做一个小例子试试，对你理解 java io 有帮助

27.简述 CALayer 和 UIView 的关系

答:UIView 和 CALayer 是相互依赖的关系。UIView 依赖与 calayer 提供的内容，CALayer 依赖 uivew 提供的容器来显示绘制的内容。归根到底 CALayer 是这一切的基础，如果没有 CALayer，UIView 自身也不会存在，UIView 是一个特殊的 CALayer 实现，添加了响应事件的能力。

结论：

UIView 来自 CALayer，高于 CALayer，是 CALayer 高层实现与封装。UIView 的所有特性来源于 CALayer 支持。

28.声明一个静态方法和一个实例方法

答:先说实例方法，当你给一个类写一个方法，如果该方法需要访问某个实例的成员变量时，那么就将该方法定义成实例方法。一类的实例通常有一些成员变量，其中含有该实例的状态信息。而该方法需要改变这些状态。那么该方法需要声明成实例方法。

静态方法正好相反，它不需要访问某个实例的成员变量，它不需要去改变某个实例的状态。我们把该方法定义成静态方法。

29.常见的 Object-C 的数据类型有哪些?和 Cd 基本数据类型有什么区别?

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 nsobject 的方法和 nsnumber 自己的方法,可以完成复杂的操作。

30.UIView 的动画效果有哪些

如 UIViewAnimationOptionCurveEaseInOut

UIViewAnimationOptionCurveEaseIn

UIViewAnimationOptionCurveEaseOut

UIViewAnimationOptionTransitionFlipFromLeft

UIViewAnimationOptionTransitionFlipFromRight

UIViewAnimationOptionTransitionCurlUp

UIViewAnimationOptionTransitionCurlDown

31.你了解 svn,cvs 等版本控制工具么?

答:了解.

32.静态链接库(了解一下)

答：静态库是程序代码的集合，是共享代码的一种方式

静态库是闭源的存在形式.a 和.framework

连接时，静态库会被完全的复制到可执行文件中，被多次使用就会有冗余拷贝，相当于 java 里的 jar 包，把一些类编译到一个包中，在不同的工程中如果导入此文件就可以使用里面的类，

33.什么是沙箱模型?哪些操作是属于私有 api 范畴?

答:1、应用程序可以在自己的沙盒里运作，但是不能访问任何其他应用程序的沙盒。

2、应用程序间不能共享数据，沙盒里的文件不能被复制到其他应用程序文件夹中,也不能把其他应用程序文件夹中的文件复制到沙盒里。

3、苹果禁止任何读、写沙盒以外的文件，禁止应用程序将内容写到沙盒以外的文件夹中。

4、沙盒根目录里有三个文件夹：Documents，一般应该把应用程序的数据文件存到这个文件夹里，用于存储用户数据或其他应该定期备份的信息。
Library，下有两个文件夹，Caches 存储应用程序再次启动所需的信息，Preferences 包含应用程序偏好设置文件，不过不要在这里修改偏好设置。
temp，存放临时文件，即应用程序再次启动不需要的文件。

沙盒根目录里有三个文件夹分别是：documents，tmp，Library。

1、Documents 目录：您应该将所有 de 应用程序数据文件写入到这个目录下。这个目录用于存储用户数据或其它应该定期备份的信息。

2、AppName.app 目录：这是应用程序的程序包目录，包含应用程序的本身。由于应用程序必须经过签名，所以您在运行时不能对这个目录中的内容进行修改，否则可能会使应用程序无法启动。

3、Library 目录：这个目录下有两个子目录：Caches 和 Preferences

Preferences 目录：包含应用程序的偏好设置文件。您不应该直接创建偏好设置文件，而是应该使用 NSUserDefaults 类来取得和设置应用程序的偏好.

Caches 目录：用于存放应用程序专用的支持文件，保存应用程序再次启动过程中需要的信息。

4、tmp 目录：这个目录用于存放临时文件，保存应用程序再次启动过程中不需要的信息。

iOS 沙盒(sandbox)中的几个目录获取方式：

```
//获取沙盒主目录路径

NSString *homeDir = NSHomeDirectory();

//获取 Documents 目录路径

NSArray *paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDoma
inMask, YES);

NSString *docDir = [paths objectAtIndex:0];

//获取 Caches 目录路径

NSArray *paths = NSSearchPathForDirectoriesInDomains(NSCachesDirectory, NSUserDomain
Mask, YES);

NSString *cachesDir = [paths objectAtIndex:0];

//获取 tmp 目录路径

NSString *tmpDir =  NSTemporaryDirectory();

//获取当前程序包中一个图片资源（apple.png）路径

NSString *imagePath = [[NSBundle mainBundle] pathForResource:@"apple" ofType:@"png
"];
```

```
UIImage *appleImage = [[UIImage alloc] initWithContentsOfFile:imagePath];

例子：

NSFileManager* fm=[NSFileManagerdefaultManager];

if(![fm fileExistsAtPath:[selfdataFilePath]]){

//下面是对该文件进行制定路径的保存

[fm createDirectoryAtPath:[selfdataFilePath] withIntermediateDirectories:YES attributes:nil error:nil];

//取得一个目录下得所有文件名

NSArray *files = [fm subpathsAtPath: [selfdataFilePath] ];

//读取某个文件

NSData *data = [fm contentsAtPath:[selfdataFilePath]];

//或者

NSData *data = [NSDatadataWithContentOfPath:[self dataFilePath]];

}
```

34.协议是什么?有什么作用?

协议：声明一系列的方法，可由任何类实施，即使遵守该协议的类没有共同的超类。协议方法定义了独立于任何特定类的行为。简单的说，协议就是定义了一个接口，其他类负责来实现这些接口。如果你的类实现了一个协议的方法时，则说该类遵循此协议。

协议的作用：

1.定义一套公用的接口（Public）

@required：必须实现的方法

@optional：可选实现的方法（可以全部都不实现）

2.委托代理（Delegate）传值：

它本身是一个设计模式，它的意思是委托别人去做某事。

比如：两个类之间的传值，类 A 调用类 B 的方法，类 B 在执行过程中遇到问题通知类 A，这时候我们需要用到代理（Delegate）。

又比如：控制器（Controller）与控制器（Controller）之间的传值，从 C1 跳转到 C2，再从 C2 返回到 C1 时需要通知 C1 更新 UI 或者是做其它的事情，这时候我们就用到了代理（Delegate）传值。

35.你在开发大型项目时,如何进行内存泄露检测的?

可以通过 xcode 的自带工具 run---start with performance tool 里有 instruments 下有个 leaks 工具，

启动此工具后，运行项目，工具里可以显示内存泄露的情况，双击可找到源码位置，可以帮助进行内存泄露的处理。

36.你实现过一个框架或者库以供别人使用么?如果有,请谈一谈构建框架或者库是的经验;如果没有,请设想和设计框架的 public 的 API,并指出大概需要如何做,需要注意一些什么方面,来方便别人容易地使用你的框架.

37.app 从创建应用到上架过程(appstore)
在你开始将程序提交到 App Store 之前，你需要有一个 App ID，一个有效的发布证书，以及一个有效的 Provisioning profile。
在 itunesconnect 网站上,创建 app 应用，设置对应信息，上传 app 打包文件，提交等待审核

38.用你熟悉的语音,编程实现 Fibonacci 数列:int F(intn);

```
Fibonacci 数列递推式 F(n) = F(n-1) +F(n-2)

F(1) = 1

F(2) = 2

F(3) = 3

F(4) = 5

F(5) = 8

int F(int n){
```

```
if(n == 1){

return1;

}

return f(n-1)+f(n-2);

}
```

39.给定两个排好序的数组 A,B,请写一个函数,从中找出他们的公共元素:findCommon(A,B)并列举其他可能的查找方法,越多越好
例如:

```
Array A = [1, 3, 5, 6, 9]

Array B = [2, 3, 6, 8, 10]

返回结果= [3, 6]

void FindCommon(int* a, int* b, int n)

{

int i = 0;

int j = 0 ;

while(i < n && j < n){

if (a[i] < b[j])

++i ;

else if(a[i] == b[j])

{

cout << a[i] << endl ;

++i ;

++j ;
```

```
}

else// a[i] > b[j]

++j ;

}
```

40.KVO 的实现原理?

答:KVO:当指定的对象的属性被修改了，允许对象接收到通知的机制。

41.如何给一个对象的私有属性赋值?

答:利用 KVC 即键值编码来给对象的私有属性赋值.

42.block 的本质是什么?为啥在 block 里面更改外面变量的值,要给外面的变量加_block 修饰,加_block 修饰的原理是什么?

答: (1) block 本质是一个数据类型,多用于参数传递,代替代理方法, (有多个参数需要传递或者多个代理方法需要实现还是推荐使用代理方法),少用于当做返回值传递. block 是一个 OC 对象,它的功能是保存代码片段,预先准备好代码,并在需要的时候执行.

(2)因为使用 block 代码块可能会引起内部循坏引用,所以应在 block 定义前加上修饰

43.block 在哪种情况下会造成循环引用,如何解决?

答:(1)从两方面分析造成循环引用问题

当 self 拥有一个 block 的时候，在 block 又调用 self 的方法(或者 self 所拥有的某个属性)。形成你中有我，我中有你，这种时候会造成循环引用

把某个实例变量变成本地临时变量,强引用将直接指向这个本地临时变量,但本地临时变量一般都会很快释放,所以一般考虑第一种情况

(2)解决方案:对 block 进行修饰__weak(arc)或__block(mrc)

44.NSURLSession 在什么情况下回存在循环引用的问题,怎么解决?

答: (1)在使用 NSURLSession 签订其代理的时候会存在循环引用问题，因为其代理是 retain 强引用

(2)解决方案

（1）在下载完成后取消 NSURLSession 会话并释放 Session，赋值为 nil。

（2）再视图将要消失时也执行同样的操作。为了防止没有下载完成就跳转控制器。

具体如下：

```
/**视图将要消失的时候，取消 session*/

- (void)viewWillDisappear:(BOOL)animated

{

[superviewWillDisappear:animated];

//任务完成，取消 NSURLSession

[self.sessioninvalidateAndCancel];

//释放会话

self.session =nil;

}
```

45.如何自己实现 GET 缓存?

答:1.使用 GET 请求数据

2.iOS 系统 SDK 已经做好了缓存。需要的仅仅是设置下内存缓存大小、磁盘缓存大小、以及缓存路径,代码如下

NSURLCache *urlCache = [[NSURLCache alloc] initWithMemoryCapacity:4 * 1024 * 1024 diskCapacity:20 * 1024 * 1024 diskPath:nil];

[NSURLCache setSharedURLCache:urlCache];

46.在使用 SQLite 过程中,如果多条线程同时操作同一数据库会造成什么问题,怎么解决?

答:(1)容易造成系统崩溃

(2)解决方案：开启第 3 种串行模式，使用一个类（单例方式）操作数据库。

47.如果提交一个 Json 格式的数据给后台服务器,后台服务器返回的是一段普通文字,用 NSURLConnection/NSURLSession/AFN 分别如何实现?

答:1.使用 NSURLConnection 发送请求的步骤很简单

（1）创建一个 NSURL 对象，设置请求路径（设置请求路径）

（2）传入 NSURL 创建一个 NSURLRequest 对象，设置请求头和请求体（创建请求对象）

（3）使用 NSURLConnection 发送 NSURLRequest（发送请求）

2.使用 NSURLSession 发送请求的步骤很简单

1）确定请求路径（一般由公司的后台开发人员以接口文档的方式提供），GET 请求参数直接跟在 URL 后面

2）创建请求对象（默认包含了请求头和请求方法【GET】），此步骤可以省略

3）创建会话对象（NSURLSession）

4）根据会话对象创建请求任务（NSURLSessionDataTask）

5）执行 Task

6）当得到服务器返回的响应后，解析数据（XML|JSON|HTTP）

48.请描述一下 SDWebImage 内部实现的原理

答:SDWebImage 底层实现有沙盒缓存机制，主要由三块组成

1、内存图片缓存

2、内存操作缓存

3、磁盘沙盒缓存

49.你对 runtime 都有哪些了解,你在实现开发过程中,或是你在所使用的第三方框架中,有没有使用过 runtime 的,如果有,请你描述一下其内部实现机制

答:Runtime:runtime 是一套比较底层的纯 C 语言 API,属于 1 个 C 语言库,包含了很多底层的 C 语言 API。在我们平时编写的 OC 代码中,程序运行过程时,其实最终都是转成了 runtime 的 C 语言代码, runtime 算是 OC 的幕后工作者.

50.线程间怎么通信?
（1）GCD：

```
dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT,0), ^{

//下载图片

UIImage *image = nil;

dispatch_async(dispatch_get_main_queue(),^{

//回到主线程

});

（2）NSThread 的线程通信

dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT,0), ^{

//下载图片

UIImage *image = nil;

[selfperformSelector:@selector(settingImage:) onThread:[NSThread mainThread]withObje
ct:image waitUntilDone:YES modes:nil];

}
```

这种情况也适用于子线程之间的通信。

51.网络图片处理问题中怎么解决一个相同的网络地址重复请求的问题?

答案:利用字典图片地址为 key，下载操作为 value

52.自动释放池底层怎么实现?

答:自动释放池以栈的形式实现:当你创建一个新的自动释放池时,它将被添加到栈顶.当一个对象收到发送 autorelease 消息时,他被添加到当前线程的处于栈顶的自动释放池中,当自动释放池被回收时,他们从栈中被删除,并且会给池子里面所有的对象都会做一次 release 操作

53.不用中间变量,用两种方法交换 A 和 B 的值

A = A+B;

B = A - B;

A = A - B;

54.简单描述一下客户端的缓存机制?

答案:无法简述,详细了解下,明白了够装逼就好
http://www.cnblogs.com/wendingding/p/3950198.html

55.控制器 View 的生命周期及相关函数是什么?你在开发中是如何用的?

1.在视图显示之前调用 viewWillAppear;该函数可以调用多次;
2.视图显示完毕,调用 viewDidAppear;
3.在视图消失之前调用 viewWillDisAppear;该函数可以调用多次(如需要);
4.在布局变化前后,调用 viewWill/DidLayoutSubviews 处理相关信息;

56.NSRunLoop 的实现机制,及在多线程中如何使用?

答案:NSRunLoop 是 iOS 的消息机制的处理模式

1NSRunloop 的主要作用:控制 runloop 里面线程的执行和休眠,在有事情做的时候使挡墙 NSRunloop 控制的线程工作,没有事情做让当前 runloop 的控制线程休眠.
2.runloop 就是一直在循环检测,从线程 start 到线程 end,检测 inputsourse(如点击,双击等操作)异步时间,检测 timesourse 同步事件,见到检测到输入源会执行处理函数，首先会产生通知，corefunction 向线程添加 runloop observers 来监听事件，意在监听事件发生时来做处理。
3.runloopmode 是一个集合，包括监听：事件源，定时器，以及需通知的 runloop observers

1.只有在为你的程序创建次线程的时候，才需要运行 run loop。对于程序的主线程而言，run loop 是关键部分。Cocoa 提供了运行主线程 run loop 的代码同时也会自动运行 run loop。IOS 程序 UIApplication 中的 run 方法在程序正常启动的时候就会启动 run loop。如果你使用 xcode 提供的模板创建的程序，那你永远不需要自己去启动 run loop
2.在多线程中，你需要判断是否需要 run loop。如果需要 run loop，那么你要负责配置 run loop 并启动。你不需要在任何情况下都去启动 run loop。比如，你使用线程去处理一个预先定义好的耗时极长的任务时，你就可以毋需启动 run loop。Run loop 只在你要和线程有交互时才需要

57.简单说一下 APP 的启动过程,从 main 文件开始说起

进入 main 函数，在 main.m 的 main 函数中执行了 UIApplicationMain 这个方法，这是 ios 程序的入口点!

int UIApplicationMain(int argc, char argv[], NSString principalClassName, NSString *delegateClassName)

argc、argv：ISO C 标准 main 函数的参数，直接传递给 UIApplicationMain 进行相关处理即可

principalClassName：指定应用程序类，该类必须是 UIApplication(或子类)。如果为 nil,则用 UIApplication 类作为默认值

delegateClassName：指定应用程序类的代理类，该类必须遵守 UIApplicationDelegate 协议

此函数会根据 principalClassName 创建 UIApplication 对象，根据 delegateClassName 创建一个 delegate 对象，并将该 delegate 对象赋值给 UIApplication 对象中的 delegate 属性

lUIApplication 对象会依次给 delegate 对象发送不同的消息，接着会建立应用程序的 main runloop(事件循环)，进行事件的处理(首先会调用 delegate 对象的 application:didFinishLaunchingWithOptions:)

程序正常退出时这个函数才返回。如果进程要被系统强制杀死，一般这个函数还没来得及返回进程就终止了

58.第三方 API 你是怎么用的?

cocoa pod 导入

59.用预处理指令#define 声明一个常数,用以表明一年中有多少秒?(忽略闰年问题)

答:#define second 365*24*60*60

60.UITableView 需要实现哪些代理?列出 UITableView 代理中必须实现的与其他一些常用的函数.

答:

-(NSInteger)tableView:(UITableView *)tableViewnumberOfRowsInSection:(NSInteger)section;

一组有多少行

-(UITableViewCell *)tableView:(UITableView *)tableViewcellForRowAtIndexPath:(NSIndexPath *)indexPath;

每行中的 cell 的实现以上两个方法为必须要实现的

常用的有

- (void)tableView:(UITableView *)tableViewdidSelectRowAtIndexPath:(NSIndexPath*)indexPath

选中以后事件设置

-(CGFloat)tableView:(UITableView *)tableViewheightForRowAtIndexPath:(NSIndexPath*)indexPath

设置 cell 的高度

等等。。。。。

61.在 iOS 上开发一个应用程序时怎么做的?

答:首先，要有一个 MAC 系统（买一台苹果电脑，苹果本或者 MACmini），没有这个条件可以装一个黑苹果的 mac 系统或者装一个虚拟机。然后装一个 X-CODE 开发环境。要是学习 ios 开发的话，这些就可以了。如果要开发、上线的话，就得准备 iphone/ipod、ipad 做为测试机，到苹果申请一个开发者账号，

每年的年费 99 美元。再然后接着就可以开发你的程序了，开发完毕之后，发布到 App store 上面，通过审核就可以了。

62.C++和 Objective-C 的混合使用,以下描述错误的是()

A. cpp 文件只能使用 C/C++代码

B. cpp 文件 include 的头文件中,可以出现 objective-C 的代码

C. mm 文件中混用 cpp 直接使用即可

D. cpp 使用 objective-C 的关键是使用接口,而不能直接使用代码

63.以下哪一段代码不会抛出异常(C& D)

A. NSArray *array = @[1, 2, 3];NSNumber*number = array[3];// @[@1,@ 2,@ 3]

B. NSDictionary *dict = @{@”key”:

nil};//value 不能为空

C. NSString *str = nil; NSString *str2 =[str substringFromIndex:3];

D. NSString *str = @”hi”;NSString *str2 =[str substringFromIndex:3];

64.在没有 navigationController 的情况下,要从一个 ViewController 切换到另一个 ViewController 应该()

A.{self.navigationControllerpushViewController:nextViewController animated:YES};

B.{self .viewaddSubview:nextViewController}

C. {selfpresentModalViewController:nextViewController animated:YES};

D. {selfpushViewController:nextViewController animated:YES};

分析:A、C 都需要有 navigationController,B 一个控制器的 view 是无法加载另一个控制器的 view 的，所以选 C！

65.关于下面线程管理错误的是()

//不确定

A.GCD 在后端管理着一个线程池

B.NSOperationQueue 是对 NSthread 的更高层的封装,对

C.NSThread 需要自己管理线程的生命周期

D.GCD 可以根据不同优先级分配线程,对

66.iOS 中的数据持久化方式(D)

A.属性列表

B.对象归档

C.SQLite 和 CoreData

D.以上全部+对象归档

67.设有一下宏定义:

defineN4

defineY(n)((N + 1) * n)

则执行语句: Y(5 + 1)为:(26)

68.如下程序用于把 " blue " 字符串返回,请指出其中的错误.

//不确定

```
char *GetBlue()

{

char *pcColor;
```

```
char*pcNewColor;

pcColor = “blue”;

pcNewColor =(char*)malloc(strlen(pcColor));

strcpy(pcNewColor, pcColor);

return pcNewColor;

}
```

答:strcpy 是一个字符串拷贝的函数,它的函数原型为 strcpy(char *dst, c*****t char *src);将 src 开始的一段字符串拷贝到 dst 开始的内存中去,结束的标志符号为'\0',由于拷贝的长度不是由我们控制的,所以这个字符串拷贝很容易出错

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

答:object 的数据类型由 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，NSInteger 是基本的数据类型，并不是 NSNumber 的子类，当然也不是 NSObject 的子类。NSInteger 是基本数据类型 int 或者 Long 的别名（NSInteger 的定义 typedef long NSInteger）它的区别在于，NSInteger 会根据系统是 32 位还是 64 位来决定是本身 int 还是 long.

70.iOS 有垃圾回收机制吗?它是以怎样的机制来工作的?

答: OC 是支持垃圾回收机制的(Garbage collection 简称 GC),但是 apple 的移动终端中,是不支持 GC 的,Mac 桌面系统开发中是支持的.

移动终端开发是支持 ARC（Automatic

Reference Counting 的简称）,ARC 是在 IOS5 之后推出的新技术,它与 GC 的机制是不同的。我们在编写代码时,不需要向对象发送 release 或者 autorelease 方法,也不可以调用 delloc 方法,编译器会在合适的位置自动给用户生成 release 消息(autorelease),ARC 的特点是自动引用技术简化了内存管理的难度.

71.请使用 gcd 完成如下任务,执行并发任务 task1,task1 完成后 update UI.
答:

```
dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT,0), ^{

//task1:

NSLog(@"执行 task1");

//更新 UI

dispatch_async(dispatch_get_main_queue(), ^{

NSLog(@"更新 UI");

});

});
```

72.为什么在主线程中更新 UI?子线程中想要更新 UI 怎么做?

答：（1）在子线程中不能更新 UI，除了极少数 UI 外，其他 UI 更新要等到子线程执行完毕后回到主线程中进行更新。如果子线程一直在运行，则子线程中 UI 更新的函数栈主线程无法得知，即 UI 无法更新；

（2）回到主线程中进行 UI 更新；

73.简述通过 Storyboard 实现一个 tableView

(自定义 cell 的关键步骤).

答:首先创建自己的自定义 cell 的类，我们叫做 CustomCell，要继承于 UITableViewCell。在这个类中定义自己所需要的控件。

然后，打开 storyboard，选择自己要添加自定义 cell 的 UIViewController，我们叫它为 ViewController。在 UITableView 里面添加一个 cell（或者修改原有的 cell）。将 cell 的 style 改为 custom，将 cell 的类改为 CustomCell，将 identifier 改为 CustomCellIdentifier。然后，可以在 cell 中添加控件，将控件和刚才在 CustomCell 中定义的控件连起来。

最后，在 ViewController 的 UITableView 的 tableView:cellForRowAtIndexPath:代理方法中添加以下代码：

[plain]

CustomCell*cell=[tableViewdequeueReusableCellWithIdentifier:@"CustomCellIdentifier"];

这样，就创建了一个 cell，可以在这句代码之后对自己添加的控件进行设置。

74.如何生成同时支持多个架构(simulator,arm7,arm64)的通用静态库?

答:ValidArchitectures 设置为：armv7 | armv7s | arm64 | i386 | x86_64；

Architectures 设置不变（或根据你需要）:armv7 | arm64；

然后分别选择 iOS 设备和模拟器进行编译，最后找到相关的.a 进行合包，使用 lipo -create 真机库.a 的路径模拟器库.a 的的路径－output 合成库的名字.a；

这样就制作了一个通用的静态库.a；

75.请写出一个 xml 文件,用于描述一个书架,书架上有 2 本书,书本的类别(category)分别是 cooking,children.要求 tag 中包含书名(title),作者(author).类别(category)要用属性表示.
答:

```
书名 1<\title>

作者 1<\author>

<\book>

书名 2<\title>

作者 2<\author>

<\book>
```

76.strcpy 和 memcpy 的最大区别是什么?

答:1、复制的内容不同。strcpy 只能复制字符串，而 memcpy 可以复制任意内容，例如字符数组、整型、结构体、类等。

2、复制的方法不同。strcpy 不需要指定长度，它遇到被复制字符的串结束符"\0"才结束，所以容易溢出。memcpy 则是根据其第

3 个参数决定复制的长度。

3、用途不同。通常在复制字符串时用 strcpy，而需要复制其他类型数据时则一般用 memcpy

108.g++, ld 是什么?声明编译选项–DSUPPORT_BLUETOOTH =

1 有什么作用?

答: g++是 GNU 的 c++编译器;

77.@class 用途

答:@class 一般用于头文件中声明某个类的实例变量的时候用到.它只是声明,至于内部的实现是没有告诉编译器的.

78.delegate 使用 assign or retain 简述理由.

答：assign，防止出现循环引用；

79.NSString 与 NSData 之间的转换过程中要特别注意的事项是什么?
解:

```
NSString 转换成 NSData 对象

NSData* xmlData = [@"testdata"dataUsingEncoding:NSUTF8StringEncoding];

NSData 转换成 NSString 对象

NSData * data;

NSString *result = [[NSString alloc] initWithData:data  encoding:NSUTF8StringEncodin
g];

NSData 转换成 char*

NSData *data;

char *test=[data bytes];

char*转换成 NSData 对象
```

```
byte* tempData = malloc(sizeof(byte)*16);

NSData *content=[NSData dataWithBytes:tempData length:16];
```

转换过程中要注意 NSData 的编码格式问题.

解决方法:

先设置断点然后在控制台 po 出 NSData 的变量，看看会显示什么。

如果 po 出的 NSData 是可阅读的，直接能看到文本的内容，则使用[NSString stringWithFormat:NSData] (这里的 NSData 指的是你需要转换成 NSString 的 NSData 变量)即可。

如果 po 出的 NSData 是不可阅读的乱码，那一般都是有编码格式的，最常用的是 NSUTF8StringEncoding，另外还有 NSASCIIStringEncoding 等，你可以在 Apple 文档里找到编码格式的那个枚举类型，挨个尝试。

80.请用代码如何判断某个对象 obj 是否支持某个 method.
解:

```
if ([srespondsToSelector:@selector(print:)]) {

[s print:@"支持这个方法"];

}
```

81.请用简单的代码展示@protocol 的定义及实现.
解:

```
#warning 代理第一步:声明协议

@protocol MarryMe

-(void)makeMoney;

@end

#warning 代理第二步:声明代理

@property(nonatomic,assign)id myDeleget;
```

```
.m 文件中

#warning 代理第三步:代理人执行协议方法

[self.myDeleget makeMoney];

代理人.m 文件中

#warning 代理第四步:签订协议

@interface Boy : NSObject

Girl *girl = [[Girl alloc] init];

#warning 代理第五步:成为代理人

girl.myDeleget = self;

[girl getMessage:message];

#warning 协议代理第六步:实现协议方法

-(void)makeMoney{

NSLog(@"aaa");

}
```

82.请描述应聘岗位的未来职业规划

解:答案不唯一,如有需要请自行规划活着百度.

83.3 升的杯子一个,5 升的杯子一个,杯子的形状不规则,问怎么才能得到 4 升的水,水无限多.(请写出推理过程)

解:先将 5 升的杯子倒满,然后把 5 升的杯子中的水倒入 3 升的杯子,倒满后 5 升的杯子剩下 2 升.再把 3 升杯子中的水倒掉,把 5 升的杯子中剩余的 2 升水倒入 3 升的杯子中,然后把 5 升的杯子倒满.再用 5 升的杯子中的水给 3 升的杯子添满,则 5 升的杯子中剩余 4 升的水.

84.数据持久化存储方案有哪些?

解:所谓的持久化，就是将数据保存到硬盘中，使得在应用程序或机器重启后可以继续访问之前保存的数据。在 iOS 开发中，数据持久化的方案有 5 种方案：

plist 文件（属性列表）

preference（偏好设置）

NSKeyedArchiver（归档）

SQLite 3

CoreData

85.网络通信用过哪些方式？

解: ios 设备的网络通信的方法，有如下两个大类：

1、使用 socket 的方式进行通信。

2、使用 asynsocket 类库进行通信。

86.如何处理多个网络请求并发的情况？

解: //了解(并发)当有多个线程在操作时,如果系统只有一个 CPU,则它根本不可能真正同时进行一个以上的线程，它只能把 CPU 运行时间划分成若干个时间段,再将时间段分配给各个线程执行，在一个时间段的线程代码运行时，其它线程处于挂起状。.这种方式我们称之为并发(Concurrent)。

遇到这种情况建议使用第三方的网络库。比如 AFNetworking。也可以通过 GCD 和 NSOperationQueue 来控制并发

87.简单介绍一下 KVC 和 KVO，他们都可以应用在哪些场景?

解: KVO:键值监听,观察某一属性的方法

KVC:键值编码,是一种间接访问对象的属性

88.讲述一下 runtime 的概念，messagesend 如果寻找不到响应的对象，会如何？

Objc Runtime 其实是一个 Runtime 库，它基本上是用 C 和汇编写的，这个库使得 C 语言有了面向对象的能力。

89. iOS 能否嵌入其他语言?如何实现?

90. iOS 移动开发最终生成的是什么文件?其结构如何?

最后打包完成是一个.ipa 文件可以通过 iTunes 和其他工具对有测试资格的手机进行安装

91. UINavigationController 如何要使用 push/pop 功能的话,需要怎么实现

1.用 UINavigationController 的时候用 pushViewController:animated

----返回之前的视图[[selfnavigationController] popViewControllerAnimated:YES];

---ps：push 以后会在 navigation 的 left bar 自动添加 back 按钮，它的响应方法就是返回。所以一般不需要写返回方法，点 back 按钮即可。

2.其他时候用 presentModalViewController:animated

[selfpresentModalViewController:controller animated:YES];//YES 有动画效果

-----返回之前的视图[selfdismissModalViewControllerAnimated:YES];

3.切换视图一般用不到 addSubview

UINavigationController 是导航控制器，如果 pushViewController 的话，会跳转到下一个 ViewController，点返回会回到现在这个 ViewController；

如果是 addSubview 的话，其实还是对当前的 ViewController 操作，只是在当前视图上面又“盖”住了一层视图，其实原来的画面在下面呢，看不到而已。

92. UIView 如何需要重新绘制整个界面,需要调用什么方法?

UIView setNeedsDisplay 和 setNeedsLayout 方法。首先两个方法都是异步执行的。而 setNeedsDisplay 会调用自动调用 drawRect 方法，这样可以拿到 UIGraphicsGetCurrentContext，就可以画画了。而 setNeedsLayout 会默认调用 layoutSubViews，就可以处理子视图中的一些数据。

综上所述：setNeedsDisplay 方便绘图，而 layoutSubViews 方便出来数据

setNeedDisplay 告知视图它发生了改变，需要重新绘制自身，就相当于刷新界面.

93. Plist 文件?结构是?

Plist 文件通常用于储存用户设置，也可以用于存储捆绑的信息，该功能在旧式的 Mac OS 中是由资源分支提供的。

Plist 主要有 Core Fundation 类型构成,也可以将这些类型放入 NSDictionary 和 NSArray 以便后膛更复杂的数据类型

94. iOS 里面的二进制数据类型是什么?和 NSString 如何互相转换?

NSData:用于存储二进制的数据类型

NSData 类提供了一种简单的方式，它用来设置缓冲区、将文件的内容读入缓冲区，或将缓冲区的内容写到一个文件。

不变缓冲区（NSData 类），也可定义可变的缓冲区（NSMutableData 类）。

NSData、NSString 互转：

NSData * data = [str dataUsingEncoding:NSUTF8StringEncoding];

//NSString 转换成 NSData 类型

NSString * newStr = [[NSString alloc]initWithData:data encoding:NSUTF8StringEncoding];

95. iOS 里面是否有 GBK 的字符编码描述?即 NSUTF8StringEncoding 如果有,是怎样的?

96. iOS 里面的手势是如何实现的?

97.谈谈你了解的设计模式,你用过哪些,他们的缺点

1.MVC:优点：

1、开发人员可以只关注整个结构中的其中某一层；

2、可以很容易的用新的实现来替换原有层次的实现；

3、可以降低层与层之间的依赖；

4、有利于标准化；

5、利于各层逻辑的复用。

缺点:

1、降低了系统的性能。这是不言而喻的。如果不采用分层式结构，很多业务可以直接造访数据库，以此获取相应的数据，如今却必须通过中间层来完成。

2、有时会导致级联的修改。这种修改尤其体现在自上而下的方向。如果在表示层中需要增加一个功能，为保证其设计符合分层式结构，可能需要在相应的业务逻辑层和数据访问层中都增加相应的代码。

2.观察者模式优点：

1、观察者模式在被观察者和观察者之间建立一个抽象的耦合。被观察者角色所知道的只是一个具体观察者列表，每一个具体观察者都符合一个抽象观察者的接口。被观察者并不认识任何一个具体观察者，它只知道它们都有一个共同的接口。

由于被观察者和观察者没有紧密地耦合在一起，因此它们可以属于不同的抽象化层次。如果被观察者和观察者都被扔到一起，那么这个对象必然跨越抽象化和具体化层次。

2、观察者模式支持广播通讯。被观察者会向所有的登记过的观察者发出通知，

观察者模式缺点：

1、如果一个被观察者对象有很多的直接和间接的观察者的话，将所有的观察者都通知到会花费很多时间。

2、如果在被观察者之间有循环依赖的话，被观察者会触发它们之间进行循环调用，导致系统崩溃。在使用观察者模式是要特别注意这一点。

3、如果对观察者的通知是通过另外的线程进行异步投递的话，系统必须保证投递是以自恰的方式进行的。

4、虽然观察者模式可以随时使观察者知道所观察的对象发生了变化，但是观察者模式没有相应的机制使观察者知道所观察的对象是怎么发生变化的。

3.单例模式:主要优点：

1、提供了对唯一实例的受控访问。

2、由于在系统内存中只存在一个对象，因此可以节约系统资源，对于一些需要频繁创建和销毁的对象单例模式无疑可以提高系统的性能。

3、允许可变数目的实例。

3.单例模式:主要缺点：

1、由于单利模式中没有抽象层，因此单例类的扩展有很大的困难。

2、单例类的职责过重，在一定程度上违背了“单一职责原则”。

3、滥用单例将带来一些负面问题，如为了节省资源将数据库连接池对象设计为的单例类，可能会导致共享连接池对象的程序过多而出现连接池溢出；如果实例化的对象长时间不被利用，系统会认为是垃圾而被回收，这将导致对象状态的丢失.

98.数据持久化存储方案有哪些?
答:
(附网址:http://www.cocoachina.com/industry/20130328/5908.html)

iOS 中的数据持久化方式，基本上有以下四种：属性列表、对象归档、SQLite3 和 Core Data

1.属性列表(NSUserDefaults，用于存储配置信息)

涉及到的主要类：NSUserDefaults,一般[NSUserDefaults standardUserDefaults]就够用了

2.对象归档

要使用对象归档,对象必须实现 NSCoding 协议.大部分 Object C 对象都符合 NSCoding 协议,也可以在自定义对象中实现 NSCoding 协议,要实现 NSCoding 协议,实现两个方法

3.SQLite3

SQLite 的数据库权限只依赖于文件系统，没有用户帐户的概念。SQLite 有数据库级锁定，没有网络服务器。它需要的内存，其它开销很小，适合用于嵌入式设备。你需要做的仅仅是把它正确的编译到你的程序。

4.Core Data

Core Data 本质上是使用 SQLite 保存数据，但是它不需要编写任何 SQL 语句。

要使用 Core Data，需要在 Xcode 中的数据模型编辑器中设计好各个实体以及定义好他们的属性和关系。之后，通过操作这些对象，结合 Core Data 完成数据的持久化：

99.网络通信用过哪些方式?

(附网址:http://blog.csdn.net/chang6520/article/details/7967698)

同样也是代码解释

iOS 设备的网络通信的方法，有如下两个大类：

1、使用 socket 的方式进行通信。

以 TCP 为利，对于 TCP 来说，是要区分服务端和客户端的。服务端：通常的方法是服务端启动后监听，是否有客户端连接，如果有连接，则建立与客户端的通信。客户端的方法通常是连接服务端，当连接成功之后，就希望发送数据了。

2、使用 asynsocket 类库进行通信。

100.如何处理多个网络请求并发的情况?
答:
(附网址:http://www.cnblogs.com/yanhuaxuanlan/p/4683557.html)

答案都是代码,大家可以打开网址仔细阅读

1.并发当有多个线程在操作时,如果系统只有一个 CPU,则它根本不可能真正同时进行一个以上的线程，它只能把 CPU 运行时间划分成若干个时间段,再将时

间段分配给各个线程执行，在一个时间段的线程代码运行时，其它线程处于挂起状。.这种方式我们称之为并发(Concurrent)。

2.并行当系统有一个以上 CPU 时,则线程的操作有可能非并发。当一个 CPU 执行一个线程时，另一个 CPU 可以执行另一个线程，两个线程互不抢占 CPU 资源，可以同时进行，这种方式我们称之为并行(Parallel)。

3.区别并发和并行是即相似又有区别的两个概念，并行是指两个或者多个事件在同一时刻发生；而并发是指两个或多个事件在同一时间间隔内发生。

101.简单介绍一下 KVC 和 KVO,他们都可以应用在哪些场景?
答:
(附网址:http://blog.csdn.net/zhaozy55555/article/details/8598374

http://www.cnblogs.com/kenshincui/p/3871178.html)

KVC:NSKeyValueCoding 的简称，是一种可以直接通过字符串的名字（key）来访问类属性的机制，而不是通过调用的 Setter、Getter 方法访问。

KVC 的操作方法由 NSKeyValueCoding 协议提供，NSObject 就实现了这个协议，也就是说如果对象是 NSObject 的子对象那么就支持 KVC 操作，KVC 有两种操作方法，一种是设值，一种是取值，可以理解为 getter 和 setter，不过稍微有所不同的是，设置对象值的方法中有两个，setValue:属性值 forKey:属性名（一般的设置，比如说是说设置 NSString,NSNumber 等基本类类型，setetValue:属性值 forKeyPath:属性路径

2.KVO:NSKeyValueObserving 的简称，当指定的对象的属性被修改了，允许对象接受到通知的机制。每次指定的被观察对象的属性被修改的时候，KVO 都会自动的去通知相应的观察者，相当于设计模式中的观察者模式。

Key-Value Observing (KVO)建立在 KVC 之上，能够观察一个对象的 KVC key path 值的变化，接下来的做的实例是在 iOS 中视图的 ViewDidLoad 中实现的，跟 KVC 类似，不过可以监听值的变化，实现起来很简单 addObserver 添加观察，observeValueForKeyPath 观察变化之后的事件，最后需要销毁以下监听事件，

102.实现多线程有哪些方法,分别有什么区别?

答: (http://www.cnblogs.com/hanjun/p/3667874.html)

1.NSThread

2.NSOperationQueue

3.GCD

区别:

Thread 是这三种范式里面相对轻量级的，但也是使用起来最负责的，你需要自己管理 thread 的生命周期，线程之间的同步。线程共享同一应用程序的部分内存空间，它们拥有对数据相同的访问权限。你得协调多个线程对同一数据的访问，一般做法是在访问之前加锁，这会导致一定的性能开销。在 iOS 中我们可以使用多种形式的 thread:

Cocoa threads:使用 NSThread 或直接从 NSObject 的类方法 performSelectorInBackground:withObject:来创建一个线程。如果你选择 thread 来实现多线程，那么 NSThread 就是官方推荐优先选用的方式。

Cocoa operations 是基于 Obective-C 实现的，类 NSOperation 以面向对象的方式封装了用户需要执行的操作，我们只要聚焦于我们需要做的事情，而不必太操心线程的管理，同步等事情，因为 NSOperation 已经为我们封装了这些事情。NSOperation 是一个抽象基类，我们必须使用它的子类。iOS 提供了两种默认实现：NSInvocationOperation 和 NSBlockOperation。

Grand Central Dispatch (GCD): iOS4 才开始支持，它提供了一些新的特性，以及运行库来支持多核并行编程，它的关注点更高：如何在多个 cpu 上提升效率。

103.The ios/osx's graphics is bassed onOpenGL . what is OpenGL?
iOS 的/ OS X 的图形是基于 OpenGL。什么是 OpenGL?

(附网址:https://developer.apple.com/opengl/)

官方的解释:OpenGL 是硬件基础图形加速在 OS X 的权力核心动画，核心形象，和石英的极端和给你的应用程序访问惊人的 3D 图形处理能力。使用工业标准的图形 API 创建一系列应用程序，包括游戏，动画制作软件，以及医疗成像解决方案。

百度的解释:

OpenGL:(Open Graphics Library)是指定义了一个跨编程语言、跨平台的编程接口规格的专业的图形程序接口。它用于三维图像（二维的亦可），是一个功能强大，调用方便的底层图形库。计算机三维图形是指将用数据描述的三维空间通过计算转换成二维图像并显示或打印出来的技术。OpenGL 就是支持这种

转换的程序库，它源于 SGI 公司为其图形工作站开发的 IRIS GL，在跨平台移植过程中发展成为 OpenGL。OpenGL 被设计成独立于硬件、独立于窗口系统，在各种操作系统的计算机上都可用的，并能在网络环境下以客户/服务器模式工作，是专业图形处理、科学计算等高端应用领域的标准图形库

104.What is CoreFoundation framework,andwhat is Foundation framework

什么是框架的 CoreFoundation，什么是 Foundation 框架

答:

(附:文顶顶网址 http://www.cnblogs.com/wendingding/p/3710820.html 和 http://blog.csdn.net/annkey123/article/details/8271867)

Core Foundation 框架(CoreFoundation.framework)是一组 C 语言接口，它们为 iOS 应用程序提供基本数据管理和服务功能。

Foundation—基础框架。框架中包含了很多开发中常用的数据类型，如结构体，枚举，类等，是其他 ios 框架的基础。

如果要想使用 foundation 框架中的数据类型，那么包含它的主头文件就可以了。

即#import

补充：core foundation 框架相对底层，里面的代码几乎都是 c 语言的，而 foundation 中是 OC 的。

105.How do you save data for you app

你如何保存你的应用程序数据

答:

(附网址:应用数据存储方式

（XML 属性列表-
plist）: http://www.cnblogs.com/ wendingding/p/3773867.html

(偏好设置):http://www.cnblogs.com/wendingding/p/3775178.html

(归档):http://www.cnblogs.com/wendingding/p/3775293.html)

ios 应用数据存储方式（XML 属性列表-plist）ios 应用常用的数据存储方式

1.plist（XML 属性列表归档）

2.偏好设置

3.NSKeydeArchiver 归档（存储自定义对象）

4.SQLite3(数据库，关系型数据库，不能直接存储对象，要编写一些数据库的语句，将对象拆开存储)

5.Core Data（对象型的数据库，把内部环节屏蔽）

106.Do you use GIT version control?What isthe difference between merge and rebase ? If you are not using GIT ,are youusing any distributed version control system?

您是否使用 Git 版本控制？什么是合并和重订之间的区别？如果你没有使用 Git，您使用的分布式版本控制系统？

答:建议大家使用百度翻译打开网址以下均是英文解释

(附网址:什么是版本控制:https://git-scm.com/book/zh/ch1-1.html

国外网友的解释:http://translate.baiducontent.com/transpage?query=http%3A%2F%2Fserve.3ezy.com%2Fstackoverflow.com%2Fquestions%2F16666089%2Fwhats-the-difference-between-git-merge-and-git-rebase&from=en&to=zh&source=url

合订和重定的解释:http://translate.baiducontent.com/transpage?cb=translateCallback&ie=utf8&source=url&query=http%3A%2F%2Fserve.3ezy.com%2Fgit-scm.com%2Fbook%2Fen%2FGit-Branching-Rebasing&from=en&to=zh&token=&monLang=zh

视频的网址:http://www.git-tower.com/learn/git/videos/)

107.Storyboard or Xib, which do youprefer?And why

Storyboard 和 Xib 你喜欢哪个为什么?

答:

(附网址:http://blog.mobilejazz.com/storyboards-xibs-best-practices/)

喜欢哪个大家可以选择:以下是 Storyboard 和 Xib 的优缺点可以参考以下

以下解释是英文翻译过来的(建议大家可以看网址原文章)

xibs 是 XML 文件定义和配置的一组对象，并专门操纵主要观点（*UIView* 子类）。Xcode 具有友好的编辑器，可以显示这些意见，它是一个运行的应用程序，使得它的配置和设计布局非常容易（节省很多行代码）。

即使大多数开发商关联一个 xib 文件“屏幕”或“视图控制器”，是通用的容器 xibs 对象和可以一起使用，与其它类型的对象作为 *nsobjects UIViews*，或者只是一个人。

Storyboard

这是一个 XML 文件代表统一一套 xibs 为了定义导航之间的一组视图控制器。有一个故事一个开发者可以定义多个“屏幕”（或和导航之间的 *UIViewController* 类）他们。

作为一个差的 XIB，故事是由定义的视图和导航*处理*他们之间。

xibs VS 代码的好处

视图的可视化配置。

自动布局的视觉形态。

大小班的视觉形态。

节省时间和代码，而“创建实例”。

节省时间和代码，而“配置对象”。

快速 *UITableViewCell* 原型。

快速配置控制动作（*ibactions*）。

Storyboard 与 xibs 效益

可以直接在脚本本身原型*的行*。

可以定义静态*表格*部分和行。

可以使用自动布局约束添加到 *toplayoutguide* 和 *bottomlayoutguide*。

可以指定导航和过渡（这是一个主要的目的！）。

可以定义多个“屏幕”（*处理*的）在一个地方（不需要多 xibs）。

对 xibs VS 代码不便

容易破裂合并时（GIT）。

不复杂的自动布局定义。

不能引用（或包括）其他 xibs。

Storyboard 与代码 xibs 不便

故事情节是：大文件的加载速度慢，有时。

需要一个大屏幕显示所有（或部分）的内容。

容易破裂合并时（GIT）。

高概率打破它时，合并（GIT）作为故事包含很多信息。

许多警告当支持 iOS 7 和 iOS 8（利润，大小班）。

不能引用（或包括）其他 xibs。

108. Aside from regular tasks likereading/writing element, getting the count of an array, can you write somethingelse you can do to an NSArray with only the built-in SDK? E.g, how do youfilter（过滤器）, map, an NSArray?

109.Do you use SDWebImage? If yes, why doyou choose this library? If no, how do you load an image from Internet to anUIImageView?

翻译:你使用 SDWebImage 吗？如果使用，你为什么使用这个库，如果不使用，你是怎样加载一张网络图片的？

答案：这个类库提供一个 UIImageView 类别以支持加载来自网络的远程图片。具有缓存管理、异步下载、同一个 URL 下载次数控制和优化等特征。

110.看下面的程序,第一个 NSLog 会输出什么?这时 str 的 retainCount 是多少?第二个和第三个呢?为什么

```
NSMutableArray* ary = [[NSMutableArrayarray] retain]; //2

NSString *str = [NSStringstringWithFormat:@"test"];//1

[str retain];// 2

[ary addObject:str]; // 3

NSLog(@"%d", [str retainCount]);

[str retain];// 4

[str release];// 3

[str release];// 2

NSLog(@"%d", [str retainCount]);//2

[ary removeAllObjects]; // 1

NSLog(@"%d", [strretainCount]);

结果：3、2、1
```

111.Storyboard or Xib, which do you prefer?And why?

你喜欢使用 Storyboard 还是 Xib?为什么？

详细解释：“http://stackoverflow.com/questions/13834999/storyboards-vs-the-old-xib-way”

答案: The reasons for liking Storyboard:There are things you can do with

a storyboard that you can't do with a nib. A storyboard lets you create segues

between view controllers, and it lets you design table view cells in-place. One

big advantage of Storyboard is the ability to view your entire application's

GUI（图形用户界面）flow. Zoom out and you can see how everything interconnects andflows.

The reasons for liking Xib : There are

things you can do with a nib that you can't do with a storyboard. In a nib, you

can create references to the File's Owner placeholder（占位符）. You

can create multiple top-level views, edit them, and create connections between

them. See this answer for an example of why you'd want to do that. You can add

external（外部的）object placeholders (a rarely-used feature 很少使用的功能).

1.Apartfrom the historical value in the xib approach, xib's also provide modularity.Perhaps you have a library of code or wish to share a useful widget you made.Taking the xib approach would facilitate that sharing and reuse.

2.Thexib approach also allows you some greater flexibility in terms of your owncode.

3.Withxibs, while the modularity is nice, it's tougher to envision how everythingconnects and flows together. This can be a useful feature for yourself, or ifyou have a larger team to share with, to allow others to see how the app flows.

112. if you’ve started using Swift,writedown one or two features that is presenting Swift which is not Objective-C, andhow it helps you

英语翻译：如果你已经使用了 Swift,写出一个或两个 Swift 有而 OC 没有的功能，它是怎样帮助你的？

答案：可选类型（optionals）、元组（tuples）、泛型（generics），类型推断（type inference）以及其他等等。（答案不全）

113. How do you do abackground task（后台任务）in iOS,indifferent ways ? And which one doyou prefer

翻译：你怎样用一种不同的方式做一个后台任务，你喜欢哪一种方式？

答案：http://www.2cto.com/kf/201402/278626.html

114.static 关键字的作用?

在 C 语言中，关键字 static 有三个明显的作用：

1).在函数体，一个被声明为静态的变量在这一函数被调用过程中维持其值不变。

2).在模块内（但在函数体外），一个被声明为静态的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。它是一个本地的全局变量。

3).在模块内，一个被声明为静态的函数只可被这一模块内的其它函数调用。那就是，这个函数被限制在声明它的模块的本地范围内使用。

115.堆和栈的区别?

堆和栈的区别：

一、堆栈空间分配区别：

1、栈（操作系统）：由操作系统自动分配释放，存放函数的参数值，局部变量的值等。其操作方式类似于数据结构中的栈；

2、堆（操作系统）：一般由程序员分配释放，若程序员不释放，程序结束时可能由 OS 回收，分配方式倒是类似于链表。

二、堆栈缓存方式区别：

1、栈使用的是一级缓存，他们通常都是被调用时处于存储空间中，调用完毕立即释放；

2、堆是存放在二级缓存中，生命周期由虚拟机的垃圾回收算法来决定（并不是一旦成为孤儿对象就能被回收）。所以调用这些对象的速度要相对来得低一些。

三、堆栈数据结构区别：

堆（数据结构）：堆可以被看成是一棵树，如：堆排序；

栈（数据结构）：一种先进后出的数据结构。

116.目标~动作机制

目标是动作消息的接收者。一个控件，或者更为常见的是它的单元，以插座变量（参见"插座变量"部分）

的形式保有其动作消息的目标。

动作是控件发送给目标的消息，或者从目标的角度看，它是目标为了响应动作而实现的方法。

程序需要某些机制来进行事件和指令的翻译。这个机制就是目标-动作机制

117.自动释放池是什么,如何工作?

一、什么是自动释放池

1、Autorelease pool

自动释放池（Autorelease pool）是 OC 的一种内存自动回收机制，可以将一些临时变量通过自动释放池来回收统一释放

自动释放池本事销毁的时候，池子里面所有的对象都会做一次 release 操作

2、autorelease

任何 OC 对象只要调用 autorelease 方法，就会把该对象放到离自己最近的自动释放池中（栈顶的释放池）。

二：O-C 当中的内存释放，并不是像 java/.net 那样有一个自动的释放池，开发人员不用去关心有关内存释放的问题，O-C 里面的自动释放池比 c 语言的手动内存管理要好一些，但是相对于 java/.net 来说又弱一些，所以说 O-C 当中的释放属于半自动的释放池。

三、如何创建一个自动释放池

```
//ios5.0 新方式

@autoreleasepool

{

}

//ios5.0 之前的老方式

NSAutoreleasePool *pool=[[NSAutoreleasePoolalloc]init];

[pool release];
```

四、自动释放池如何释放对象内存

黄金法则：如果对一个对象使用了 alloc,[mutable]copy,retain,那么必须使用相应的 release 或者 autorelease.

118.obj-c 的优点

objc 优点：

1. Cateogies

2. Posing

3)动态识别

4)指标计算

5）弹性讯息传递

6)不是一个过度复杂的 C 衍生语言

7. Objective-C 与 C++可混合编程

缺点:

1)不支援命名空间

2)不支持运算符重载

3）不支持多重继承

4）使用动态运行时类型，所有的方法都是函数调用，所以很多编译时优化方法都用不到。（如内联函数等），性能低劣。

OC 没有多重继承，Java 也没有 C++才有

OC 使用协议来实现多重继承

119.什么时候用 delegate,什么时候用 Notification

1.参数的不同

使用 delegate 参数更加直观，使用 notification 参数不那么直观，所以能使用 delegate 的地方，尽量使用 delegate

2.传递的长度

有时候你的页面会一次跳好几个页面，那么你想把这个页面的数据传回到底层是很麻烦的事情，因为你要把 delegate 指针从底层界面一直传上来。

3.传递多个数据

当你在同一个对象传递给多个对象，用 delegate 就不可行了。

120.什么是 KVC 和 KVO?

1、KVC，即是指 NSKeyValueCoding，一个非正式的 Protocol，提供一种机制来间接访问对象的属性。而不是通过调用 Setter、Getter 方法访问。KVO 就是基于 KVC 实现的关键技术之一。

2、KVO 的是 KeyValue

Observe 的缩写，中文是键值观察。这是一个典型的观察者模式，观察者在键值改变时会得到通知。iOS 中有个 Notification 的机制，也可以获得通知，但这个机制需要有个 Center，相比之下 KVO 更加简洁而直接。

121.类别的作用

作用：

1.对系统的类或自己定义的类的扩充（只是指，不能声明属性 Instance variable）e.g. base64 MD5.但是不能保证所有的方法都能被调用。尤其是在指定一个与系统的方法相同的方法时一定要注意。

文件名：NSString+additions.h

NSString+additions.m

@interface NSString (Base64)

2.可以声明私有方法。

```
ZYViewController.m

@interfaceZYViewController (additions)

//{

//BOOL isOK;

//}

- (void)ok;

@property

@end
```

3.可以声明分散类的方法（NSIndexPath）

分散类体现出类别不仅可以扩充系统的方法，而且可以扩充自己定义的类。

由第 3 个作用可知：不管是否使用的分散类在任何的.h 文件中，我只考虑类别中的方法包裹在@interface className ..@end,那么我只明确将来类别中的方法就属于 className 这个类。

122.浅复制和深复制的区别

简单的来说就是，在有指针的情况下，浅拷贝只是增加了一个指针指向已经存在的内存，而深拷贝就是增加一个指针并且申请一个新的内存，使这个增加的指针指向这个新的内存，采用深拷贝的情况下，释放内存的时候就不会出现在浅拷贝时重复释放同一内存的错误！

我列举一个例子来说吧：

你正在编写 C++程序中有时用到，操作符的重载。最能体现深层拷贝与浅层拷贝的，就是‘=’的重载。

看下面一个简单的程序：

```
class string{

char *m_str;

public:

string(char *s) {

m_str=s;

}

string()

{};

String & operator=(const string s){

m_str=s.m_str;

return *this

}

};
```

```
int main(){

string s1("abc"),s2;

s2=s1;

cout<
```

上面的=重载其是就是实现了浅拷贝原因。是由于对象之中含有指针数据类型.s1,s2 恰好指向同一各内存。所以是浅拷贝。而你如果修改一下原来的程序：

```
string&operator=(const string&s){

if(strlen(m_str)!=strlen(s.m_str))

m_str=new char[strlen(s.m_str)+1];

if(*this!=s)

strcopy(m_str,s.m_str);

return *this;

}
```

这样你就实现了深拷贝，原因是你为被赋值对象申请了一个新的内存所以就是深拷贝。

123.代理的作用

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。

124.我们说的 OC 是动态运行时语言是什么意思?

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。

简单来说，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。

意思就是假设生物类（life）都用有一个相同的方法-eat;那人类属于生物，猪也属于生物，都继承了 life 后，实现各自的 eat，但是调用是我们只需调用各自的 eat 方法。

也就是不同的对象以自己的方式响应了相同的消息（响应了 eat 这个选择器）。

因此也可以说，运行时机制是多态的基础。

125.什么是 ARC？请简述一下 ARC 的原理。

1）ARC 是 iOS 5 推出的新功能，全称叫 ARC(Automatic Reference Counting)。简单地说，就是代码中自动加入了 retain/release，原先需要手动添加的用来处理内存管理的引用计数的代码可以自动地由编译器完成了

2）、ARC 的规则就是只要对象没有强指针引用，就会被释放掉，换而言之只要还有一个强引用指针变量指向对象，那么这个对象就会存在内存中。弱指针指向的对象，会被自动变成空指针（nil 指针），从而不会引发野指针错误。

126.简述视图控制器的生命周期。

1）、init 函数(init;initWithFrame;initWithCoder;等)－－初始化

2）、awakeFromNib－－在 loadView 之前的工作放在这里

3）、viewDidLoad－－注意，一个 ViewController 一个生命周期内这个函数只会调用一次

4）、viewWillAppear－－view 将要出现，每次 View 消失再出现都会调用

5）、viewWillLayoutSubviews－－简要对子试图进行布局

6）、viewDidLayoutSubivews－－完成对子试图布局

7）、viewDidAppear－－视图将要出现在屏幕上

－－－上述代码不含部分

8）、viewWillDisappear－－View 将要消失

9）viewDidDisappear－－View 已经消失

127.请描述一下线程的生命周期。

新建（new Thread）、就绪（runnable）、运行（running）、死亡（dead）、堵塞（blocked）

128.请至少列举 5 个常用的设计模式。

1）、代理模式 2）、观察者模式 3）、MVC 模式 4）、单例模式 5）工厂模式

129.如何增强 iOS 应用程序的性能。

初级

1、使用 ARC 进行内存管理、2.在适当的情况下使用 reuseIdentifier

3.尽可能将 View 设置为不透明（Opaque）4.避免臃肿的 XIBs 5.不要阻塞主线程 6.让图片的大小跟 UIImageView 一样 7.选择正确的集合 8.使用 GZIP 压缩

中级：

9.重用和延迟加载 View 10.缓存、缓存、缓存 11.考虑绘制 12.处理内存警告 13.重用花销很大的对象 14.使用 Sprite

Sheets 15.避免重新处理数据 16.选择正确的数据格式 17.设置适当的背景图片 18.降低 Web 内容的影响 19.设置阴影路径 20.优化 TableView 21.选择正确的数据存储方式

高级

22.加速启动时间 23.使用 Autorelease

Pool 24.缓存图片—或者不缓存 25.尽量避免 Date 格式化

130.请列举至少五个 iOS 中常用的第三方类库。

1）.AFNetworking

2）. SDWebImage

3）. shareKit

4）. FMDatebase

5）. MMDrawerController

131.队列和栈有什么区别。

栈（Stack）：是限定只能在表的一端进行插入和删除操作的线性表

队列（Queue）是限定只能在表的一段进行插入和在另一端进行删除操作的的线性表

1）、队列是先进先出，栈是先进后出

2）、遍历数据速度不同，队列遍历速度要快得多

132.常用的 XML 文件的解析方式有哪些？它们各自的区别是什么？

1）、有两种解析方式：DOM 解析与 SAX 解析

2）、DOM 解析必须先完成 DOM 树的创建，在处理规模较大 XML 文档时就很耗内存，占用资源较多

3）与 DOM 不同，SAX 是用事件驱动模型，解析 XML 时每遇到一个开始或结束标签、或者属性、或者一条指令时，程序就会产生一个事件进行相应的处理，因此，SAX 相对于 DOM 来说更适合操作较大的 XML 文档

133.请介绍几个常用的 git 命令。

git branch 查看本地所有分支、git status 查看当前状态、git commit 提交、git branch -a 查看所有的分支、git

branch -r 查看本地所有分支

134.请简单描述一下自己的职业生涯规划。

135.static 关键字的作用

答案:

（1）设置变量的存储域，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）限制变量的作用域，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）限制函数的作用域，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 成员变量意味着它为该类的所有实例所共享，也就是说当某个类的实例修改了该静态成员变量，其修改值为该类的其它所有实例所见；

（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。

136.堆和栈的区别?

答:答:栈完全是由系统管理的,堆是由程序员自己控制管理的,包括内存空间的开辟和释放.栈是先进后出.

137.目标-动作机制?

答:目标是动作消息的接收者。一个控件，或者更为常见的是它的单元，以插座变量（参见"插座变量"部分）

的形式保有其动作消息的目标。

动作是控件发送给目标的消息，或者从目标的角度看，它是目标为了响应动作而实现的方法。

程序需要某些机制来进行事件和指令的翻译。这个机制就是目标-动作机制

138.自动释放池是什么,如合工作?

答:当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。

139. obj-c 的优缺点

答: objc 优点：

1. Cateogies
2. Posing

3)动态识别

4)指标计算

5）弹性讯息传递

6)不是一个过度复杂的 C 衍生语言

7. Objective-C 与 C++可混合编程

缺点:

1)不支援命名空间

2)不支持运算符重载

3）不支持多重继承

4）使用动态运行时类型，所有的方法都是函数调用，所以很多编译时优化方法都用不到。（如内联函数等），性能低劣。

OC 没有多重继承，Java 也没有 C++才有

OC 使用协议来实现多重继承

140.什么时候用 delegate,什么时候用 Notification?

答: Delegate

消息的发送者(sender)告知接受者(receiver)某个事件将要发生, delegate 同意后发送者响应事件,delegate 机制使得接受者可以改变发送者的行为.

1/传值

b 把自己的数据和对象传给 a,让 a 去展示或处理

2/传事件

delegate 的优势：

1.非常严格的语法。所有将听到的事件必须是在 delegate 协议中有清晰的定义。

2.如果 delegate 中的一个方法没有实现那么就会出现编译警告/错误

3.协议必须在 controller 的作用域范围内定义

4.在一个应用中的控制流程是可跟踪的并且是可识别的；

5.在一个控制器中可以定义定义多个不同的协议，每个协议有不同的 delegates

6.没有第三方对象要求保持/监视通信过程。

7.能够接收调用的协议方法的返回值。这意味着 delegate 能够提供反馈信息给 controller

Notification

消息的发送者告知接受者事件已经发生或者将要发送(接受者不能影响发送者的行为)

消息接受者通过 keyPath 的方式指定需要接受的消息类型,通常在对象初始化完成之后声明开始接收消息在对象被销毁前注销接收消息.

notification 优势：

1.不需要编写多少代码，实现比较简单；

2.对于一个发出的通知，多个对象能够做出反应，即 1 对多的方式实现简单

Delegate 和 Notification 区别

1/二者都用于传递消息.

delegate 针对 one - to - one 关系 receiver 可以返回值给 sender.

notification 用于 one - to - one /

many / none 关系 receiver 无法返回值给 sender.

2/delegate 两者之间必须建立联系否则没办法调用代理方法.

notification 不需要两者之间有联系.

3/delegate 用于希望 sender 接受到 receiver 的某个功能值反馈

notification 用于通知多个 object 某个事件

4/notification 通过维护一个 array，实现一对多消息的转发

141.什么是 KVC 和 KVO?

答: Key value coding，Key value observer.

Kvc 是路径访问的规范，kvo 是观察某个变量的变化过程

KVO 可以观察某个对象的变量变化过程，KVC 是满足被观察的编

码规范。

KVC/KVO 类似于代理，通知中心。都是一种通讯方法。

142.类别的作用?

答: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

143.浅复制和深复制的区别?

答:简单的来说就是，在有指针的情况下，浅拷贝只是增加了一个指针指向已经存在的内存，而深拷贝就是增加一个指针并且申请一个新的内存，使这个增加的指针指向这个新的内存，采用深拷贝的情况下，释放内存的时候就不会出现在浅拷贝时重复释放同一内存的错误!

144.代理的作用?

答:作用有两个，一个是传值，一个是传事件

145.我们说的 OC 是动态运行时语言是什么意思?

多态。

答: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。

简单来说，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。

意思就是假设生物类（life）都用有一个相同的方法-eat;那人类属于生物，猪也属于生物，都继承了 life 后，实现各自的 eat，但是调用是我们只需调用各自的 eat 方法。

也就是不同的对象以自己的方式响应了相同的消息（响应了 eat 这个选择器）。

因此也可以说，运行时机制是多态的基础。

146.Object–C 的类可以多重继承吗?可以实现多个接口吗?Category 是什么?重写一个类的方法用继承好还是分类好?为什么?

答案：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。

147.属性 readwrite , readonly , assign , retain , copy , nonatomic ,各是什么作用,在哪种情况下用?

答:

assign 用于简单数据类型，如 NSInteger,double,bool,

retain 和 copy 用于对象，

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；需要一份新的变量时。

nonatomic 非原子操作，atomic 原子性操作。原子性指的是一个操作不可以被中途 cpu 暂停然后调度,即不能被中断,要不就执行完,要不就不执行，就是为了多线程安全的。

一般使用 nonatomic。

148.内存管理的几条原则是什么?按照默认法则,哪些关键字生成的对象需要手动释放,在和 property 结合的时候怎样有效的避免内存泄露?

答:当使用 new、alloc 或 copy 方法创建一个对象时，该对象引用计数器为 1。如果不需要使用该对象，可以向其发送 release 或 autorelease 消息，在其使用完毕时被销毁。

如果通过其他方法获取一个对象，则可以假设这个对象引用计数为 1，并且被设置为 autorelease，不需要对该对象进行清理，如果确实需要 retain 这个对象，则需要使用完毕后 release。

如果 retain 了某个对象，需要 release 或 autorelease 该对象，保持 retain 方法和 release 方法使用次数相等。

使用 new、alloc、copy 关键字生成的对象和 retain 了的对象需要手动释放。设置为 autorelease 的对象不需要手动释放，会直接进入自动释放池。

149.堆和栈什么区别？

答:一、堆栈空间分配区别：

1、栈（操作系统）：由操作系统自动分配释放，存放函数的参数值，局部变量的值等。其操作方式类似于数据结构中的栈；

2、堆（操作系统）：一般由程序员分配释放，若程序员不释放，程序结束时可能由 OS 回收，分配方式倒是类似于链表。

二、堆栈缓存方式区别：

1、栈使用的是一级缓存，他们通常都是被调用时处于存储空间中，调用完毕立即释放；

2、堆是存放在二级缓存中，生命周期由虚拟机的垃圾回收算法来决定（并不是一旦成为孤儿对象就能被回收）。所以调用这些对象的速度要相对来得低一些。

三、堆栈数据结构区别：

堆（数据结构）：堆可以被看成是一棵树，如：堆排序；

栈（数据结构）：一种先进后出的数据结构。

150.描述一下 iOS SDK 中如何实现 MVC 设计模式？

答:MVC 是模型、试图、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。

151.iOS 数据持久化方式有哪些？

152.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。

153. if you can Android/Windows Phone,write something about the comparison between it and iOS
翻译如果你有 Android /Windows Phone 的手机，写一些关于它与 iOS 设备之间的比较

答案:自己发挥

154. Write something about the differencesbetween Objective-C and C++.

翻译写一些关于 Objective-C 和 C++之间的差异

答案:

1、Objective C 基本上是在 C +

Smalltalk 基础上衍生出来的，C++就是 C++。除了都是从 C 语言分化出来的以外，完全没有任何共性。

2、最大的区别就是 Objective C 的成员函数调用是基于动态绑定的，类本身并没有限制对象可以有什么函数。相对于 C++类会规定好成员有什么函数。这使得 Objective C 的灵活性很大，但是执行效率也很低。

3、在 NeXT 系统的时代 C++和 Objective C 都不是那么大众的东西，C++也还没完善，而且那个时代编译器的有优化能力也没现在那么变态。所以选择 Objective C 也不是很奇怪的事。但是现在 Objective C 的劣势就越来越明显了，所以苹果开发了 Swift，而且就内部消息苹果很可能会准备放弃 OC。

**155. __unsafe_unretained vs__weak; purposeof __block; NSString const * vs NSString const*

unsafe_unretained 往往都是用来声明属性的，如果想声明临时变量就得用 __strong,__weak,__unsafe_unretained,__autoreleasing;

block 闭包就是能够读取其它函数内部变量的函数;

NSString const *HSCoder = @"汉斯哈哈哈";

"*HSCoder"不能被修改，"HSCoder"能被修改

NSString * const HSCoder = @"汉斯哈哈哈";

"HSCoder"不能被修改，"*HSCoder"能被修改

156. Write something about what you achievedby and what you learned from the UITableView class

首先，Controller 需要实现两个 delegate，分别是 UITableViewDelegate 和 UITableViewDataSource 然后 UITableView 对象的 delegate 要设置为 self,注册 cell,之后给定分区数和行数,注意 cell 的重用机制,可以设置系统 cell 或者自定义 cell

157. 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?

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: afterDelay: 或者使用 GCD 的函数：
dispatch_after(dispatch_time(DISPATCH_TIME_NOW,(int64_t)(2.0 * NSEC_PER_SEC)), dispatch_get_main_queue(), ^{

// 2 秒后异步执行这里的代码...

});

158.描述一下 iOS SDK 中如何实现 MVC 的开发模式

MVC 是模型、视图、控制器开发模式，对于 iOS

SDK，所有的 View 都是视图层的，它应该独立于模型层，由视图器来控制。所有的用户数据都是模型层，它应该独立于视图。所有的 ViewController 都是视图器，由它负责控制视图，访问模型数据。

159.简述浅拷贝和深拷贝

答案：浅层复制：只复制指向对象的指针，而不复制引用对象本身。

深层复制：复制引用对象本身。

意思就是说我有个 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。

用网上一哥们通俗的话将就是：

浅复制好比你和你的影子，你完蛋，你的影子也完蛋

深复制好比你和你的克隆人，你完蛋，你的克隆人还活着。

160.在 iPhone 应用中如何保存数据?

XML 属性列表（plist）归档

Preference(偏好设置)

NSKeyedArchiver 归档(NSCoding)

SQLite3

Core Data

161.ViewController 的 didReceiveMemoryWarning 怎么被调用:

1、当程序收到内存警告时候 ViewController 会调用 didReceiveMemoryWarning 这个方法。

2、调用了这个方法之后，对 view 进行释放并且调用 viewDidUnload 方法

3、从 iOS3.0 开始，不需要重载这个函数，把释放内存的代码放到 viewDidUnload 中去。

162.写一个委托的 interface

1.声明一个协议（只有.h 文件）

import

@protocal 协议名 superMan<父协议 NSObject>

@property NSString* name;

-(void)method;

@end

2.使一个类遵守协议

@interface TRMan:NSObject

多个协议要用，隔开

3.使用协议

id<协议名 superMan>obj=[[TRMan alloc]init];

@ protocol MyDelegate

@interface MyClass : NSObject

@property(nonatomic,weak)id delegate;

@end

协议也可以没有独立文件，写在类里

import

@class MyClass;

@protocol MyClassDelegate

```
-(void)protocolMethod;

@end

@interface MyClass : NSObject

@property(nonatomic,weak)id delegate;

@end
```

163.线程与进程的区别与联系

进程，是并发执行的程序在执行过程中分配和管理资源的基本单位，是一个动态概念，竞争计算机系统资源的基本单位。每一个进程都有一个自己的地址空间，即进程空间或（虚空间）。进程空间的大小只与处理机的位数有关，一个 16 位长处理机的进程空间大小为 216，而 32 位处理机的进程空间大小为 232。进程至少有 5 种基本状态，它们是：初始态，执行态，等待状态，就绪状态，终止状态。

线程，在网络或多用户环境下，一个服务器通常需要接收大量且不确定数量用户的并发请求，为每一个请求都创建一个进程显然是行不通的，——无论是从系统资源开销方面或是响应用户请求的效率方面来看。因此，操作系统中线程的概念便被引进了。

线程，是进程的一部分，一个没有线程的进程可以被看作是单线程的。线程有时又被称为轻权进程或轻量级进程，也是 CPU 调度的一个基本单位。

164.mvc 设计模式是什么?你还熟悉什么设计模式

答:系统分为三个部分: Model. View. Controller.在 cocoa 中,你的程序中的每一个 object

(对象)都将明显地仅属于这三部分中的一个,而完全不属于另外两个.MVC 课一帮助确保帮助实现程序最大程度的可重用性.各 MVC 元素彼此独立运作,通过分开这些元素,可以构建可维护,可独立更新的程序组建.

Delegate 设计模式

Target-action 设计模式

单例模式

165.什么是沙箱模式?哪些操作属于私有 api 范畴

某个 iphone 工程进行文件操作有此工程对应的指定的位置，不能逾越。

iphone 沙箱模型的有四个文件夹，分别是什么，永久数据存储一般放在什么位置，得到模拟器的路径的简单方式是什么.

documents，tmp，app，Library。

（NSHomeDirectory()），

手动保存的文件在 documents 文件里

Nsuserdefaults 保存的文件在 tmp 文件夹里

Documents 目录：您应该将所有 de 应用程序数据文件写入到这个目录下。这个目录用于存储用户数据或其它应该定期备份的信息。

AppName.app 目录：这是应用程序的程序包目录，包含应用程序的本身。由于应用程序必须经过签名，

所以您在运行时不能对这个目录中的内容进行修改，否则可能会使应用程序无法启动。Library 目录：这个目录下有两个子目录：Caches 和 Preferences

Preferences 目录包含应用程序的偏好设置文件。您不应该直接创建偏好设置文件，而是应该使用 NSUserDefaults 类来取得和设置应用程序的偏好.

Caches 目录用于存放应用程序专用的支持文件，保存应用程序再次启动过程中需要的信息。tmp 目录：这个目录用于存放临时文件，保存应用程序再次启动过程中不需要的信息。获取这些目录路径的方法：

1，获取家目录路径的函数：

NSString*homeDir=NSHomeDirectory();

2，获取 Documents 目录路径的方法：

NSArray*paths=NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,NSUserDomainMask,YES);

NSString*docDir=[pathsobjectAtIndex:0];

3，获取 Caches 目录路径的方法：

NSArray*paths=NSSearchPathForDirectoriesInDomains(NSCachesDirectory, NSUserDomainMask,YES);

NSString*cachesDir=[pathsobjectAtIndex:0];

4，获取 tmp 目录路径的方法：

NSString*tmpDir=NSTemporaryDirectory();

5，获取应用程序程序包中资源文件路径的方法：

例如获取程序包中一个图片资源（apple.png）路径的方法：

NSString*imagePath=[[NSBundlemainBundle]pathForResource:@"apple"ofType:@"png"];UIImage*appleImage=[[UIImagealloc]initWithContentsOfFile:imagePath];

代码中的 mainBundle 类方法用于返回一个代表应用程序包的对象。

文件 IO 写入

1，将数据写到 Documents 目录：

-(BOOL)writeApplicationData:(NSData*)datatoFile:(NSString*)fileName{

NSArray*paths=

NSUserDomainMask,YES);NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,

NSString*docDir=[pathsobjectAtIndex:0];

if(!docDir){

NSLog(@"Documentsdirectorynotfound!");returnNO;

}

NSString*filePath=[docDirstringByAppendingPathComponent:fileName];

```
return[datawriteToFile:filePathatomically:YES];

}
```

2，从 Documents 目录读取数据：

```
-(NSData*)applicationDataFromFile:(NSString*)fileName{

NSArray*paths=NSSearchPathForDirectoriesInDomains(NSDocumentDirecto
ry,NSUserDomainMask,YES);

NSString*docDir=[pathsobjectAtIndex:0];

NSString*filePath=[docDirstringByAppendingPathComponent:fileName];

NSData*data=[[[NSDataalloc]initWithContentsOfFile:filePath]autorelease];

returndata;

}
```

NSSearchPathForDirectoriesInDomains 这个主要就是返回一个绝对路径用来存放我们需要储存的文件。

```
-(NSString*)dataFilePath{

NSArray*paths=NSSearchPathForDirectoriesInDomains(NSDocumentDirecto
ry,NSUserDomainMask,YES);

NSString*documentsDirectory=[pathsobjectAtIndex:0];

return[documentsDirectorystringByAppendingPathComponent:@"shoppingCa
r.plist"];}

NSFileManager*fm=[NSFileManagerdefaultManager];

if(![fmfileExistsAtPath:[selfdataFilePath]]){

//下面是对该文件进行制定路径的保存
```

[fmcreateDirectoryAtPath:[selfdataFilePath]withIntermediateDirectories:YESattributes:nilerror:nil];

//取得一个目录下得所有文件名

NSArray*files=[fmsubpathsAtPath:[selfdataFilePath]];

//读取某个文件

NSData*data=[fmcontentsAtPath:[selfdataFilePath]];

//或者

NSData*data=[NSDatadataWithContentOfPath:[selfdataFilePath]];

}

iphone 常见私有 api 的应用（比如直接发送短信，访问沙箱之外的磁盘文件）.

166.描述一下 iOS SDK 中如何实现 MVC 的开发模式

MVC 是模型、视图、控制器开发模式，对于 iOS

SDK，所有的 View 都是视图层的，它应该独立于模型层，由视图器来控制。所有的用户数据都是模型层，它应该独立于视图。所有的 ViewController 都是视图器，由它负责控制视图，访问模型数据。

167.简述浅拷贝和深拷贝

//浅拷贝就比如像引用类型，而深拷贝就比如值类型。

浅拷贝是指源对象与拷贝对象共用一份实体，仅仅是引用的变量不同（名称不同）。对其中任何一个对象的改动都会影响另外一个对象。举个例子，一个人一开始叫张三，后来改名叫李四了，可是还是同一个人，不管是张三缺胳膊少腿还是李四缺胳膊少腿，都是这个人倒霉。

深拷贝是指源对象与拷贝对象互相独立，其中任何一个对象的改动都不会对另外一个对象造成影响。举个例子，一个人名叫张三，后来用他克隆（假设法律允许）了另外一个人，叫李四，不管是张三缺胳膊少腿还是李四缺胳膊少腿都不会影响另外一个人。比较典型的就是 Value（值）对象，如预定义类型 Int32，Double，以及结构（struct），枚举（Enum）等。

C#中有两种类型变量，一种是值类型变量，一种是引用类型变量。对于前者，copy 是属于全盘复制；而对于后者，一般的 copy 只是浅 copy，相当于只传递一个引用指针一样。因此对于后者进行真正 copy 的时候，也是最费事的，具体的说，必须为其实现 ICloneable 接口中提供的 Clone 方法。

浅拷贝(影子克隆):只复制对象的基本类型,对象类型,仍属于原来的引用.

深拷贝(深度克隆):不紧复制对象的基本类,同时也复制原对象中的对象.就是说完全是新对象产生的.

168.在 iPhone 应用中如何保存数据

//ios 程序中数据数据存储有下列 5 种方式

XML 属性列表（plist）归档、Preference(偏好设置)、NSKeyedArchiver 归档(NSCoding)、SQLite3、Core Data

每个 iOS 应用都有自己的应用沙盒(应用沙盒就是文件系统目录)，与其他文件系统隔离。应用的数据必须待在自己的沙盒里，其他应用不能访问该沙盒

169.假设有一个双向循环列队，每个节点保存了一个自然数，目前呈 10，9，4，11 的队列，代码写出一个向其中插入数字 20 的算法

170.描述下 tableView cell 的重用机制,谈谈你是如何优化 UITableView

//重用机制简单的说意思一行一行的 cell 都是在复用的，滑动 tableview 的时候，刚离开视图的 cell 会被放到复用池中，等下一个 cell 需要显示时，会先看复用池中有没有 cell 如果有的时候，就从复用池中拿出来 cell，没有的话就重新创建 cell

使用不透明视图。

不透明的视图可以极大地提高渲染的速度。因此如非必要，可以将 table cell 及其子视图的 opaque 属性设为 YES（默认值）。

其中的特例包括背景色，它的 alpha 值应该为 1（例如不要使用 clearColor）；图像的 alpha 值也应该为 1，或者在画图时设为不透明。

不要重复创建不必要的 table

cell。

前面说了，UITableView 只需要一屏幕的 UITableViewCell 对象即可。因此在 cell 不可见时，可以将其缓存起来，而在需要时继续使用它即可。

而 UITableView 也提供了这种机制，只需要简单地设置一个 identifier 即可：

staticNSString *CellIdentifier = @"xxx"; UITableViewCell *cell = [tableViewdequeueReusableCellWithIdentifier:CellIdentifier];if (cell == nil) { cell =[[[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefaultreuseIdentifier:CellIdentifier] autorelease]; }值得一提的是，cell 被重用时，它内部绘制的内容并不会被自动清除，因此你可能需要调用 setNeedsDisplayInRect:或 setNeedsDisplay 方法。

此外，在添加 table cell 的时候，如果不需要动画效果，最好不要使用 insertRowsAtIndexPaths:withRowAnimation:方法，而是直接调用 reloadData 方法。因为前者会对所有 indexPaths 调用 tableView:cellForRowAtIndexPath:方法，即便该 cell 并不需要显示（不知道是不是 bug），这就可能创建大量多余的 cell。勘误：只是在模拟器上测试如此，真机调试时没有这种 bug。

减少视图的数目。

UITableViewCell 包含了 textLabel、detailTextLabel 和 imageView 等 view，而你还可以自定义一些视图放在它的 contentView 里。然而 view 是很大的对象，创建它会消耗较多资源，并且也影响渲染的性能。

如果你的 table

cell 包含图片，且数目较多，使用默认的 UITableViewCell 会非常影响性能。奇怪的是，使用自定义的 view，而非预定义的 view，明显会快些。

当然，最佳的解决办法还是继承 UITableViewCell，并在其 drawRect:中自行绘制：

-(void)drawRect:(CGRect)rect { if (image) { [image drawAtPoint:imagePoint];self.image = nil; } else { [placeHolder drawAtPoint:imagePoint]; } [textdrawInRect:textRect withFont:font lineBreakMode:UILineBreakModeTailTruncation];}不过这样一来，你会发现选中一行后，这个 cell 就变蓝了，其中的内容就被挡住了。最简单的方法就是将 cell 的 selectionStyle 属性设为 UITableViewCellSelectionStyleNone，这样就不会被高亮了。

此外还可以创建 CALayer，将内容绘制到 layer 上，然后对 cell 的 contentView.layer 调用 addSublayer:方法。这个例子中，layer 并不会显著影响性能，但如果 layer 透明，或者有圆角、变形等效果，就会影响到绘制速度了。解决办法可参见后面的预渲染图像。

不要做多余的绘制工作。

在实现 drawRect:的时候，它的 rect 参数就是需要绘制的区域，这个区域之外的不需要进行绘制。

例如上例中，就可以用 CGRectIntersectsRect、CGRectIntersection 或 CGRectContainsRect 判断是否需要绘制 image 和 text，然后再调用绘制方法。

预渲染图像。

你会发现即使做到了上述几点，当新的图像出现时，仍然会有短暂的停顿现象。解决的办法就是在 bitmap context 里先将其画一遍，导出成 UIImage 对象，然后再绘制到屏幕，详细做法可见《利用预渲染加速 iOS 设备的图像显示》。

不要阻塞主线程。

做到前几点后，你的 table

view 滚动时应该足够流畅了，不过你仍可能让用户感到不爽。常见的现象就是在更新数据时，整个界面卡住不动，完全不响应用户请求。

出现这种现象的原因就是主线程执行了耗时很长的函数或方法，在其执行完毕前，无法绘制屏幕和响应用户请求。其中最常见的就是网络请求了，它通常都需要花费数秒的时间，而你不应该让用户等待那么久。

解决办法就是使用多线程，让子线程去执行这些函数或方法。这里面还有一个学问，当下载线程数超过 2 时，会显著影响主线程的性能。因此在使用 ASIHTTPRequest 时，可以用一个 NSOperationQueue 来维护下载请求，并将其 maxConcurrentOperationCount 设为 2。而 NSURLRequest 则可以配合 GCD 来实现，或者使用 NSURLConnection 的 setDelegateQueue:方法。

当然，在不需要响应用户请求时，也可以增加下载线程数，以加快下载速度：

-(void)scrollViewDidEndDragging:(UIScrollView *)scrollViewwillDecelerate:(BOOL)decelerate { if (!decelerate)

{queue.maxConcurrentOperationCount = 5; } } - (void)scrollViewDidEndDecelerating:(UIScrollView *)scrollView {queue.maxConcurrentOperationCount = 5; } - (void)scrollViewWillBeginDragging:(UIScrollView *)scrollView {queue.maxConcurrentOperationCount = 2; }此外，自动载入更新数据对用户来说也很友好，这减少了用户等待下载的时间。例如每次载入 50 条信息，那就可以在滚动到倒数第 10 条以内时，加载更多信息：

-(void)tableView:(UITableView *)tableView willDisplayCell:(UITableViewCell*)cell forRowAtIndexPath:(NSIndexPath *)indexPath { if (count - indexPath.row< 10 && !updating) { updating = YES; [self update]; } }// update 方法获取到结果后，设置 updating 为 NO 还有一点要注意的就是当图片下载完成后，如果 cell 是可见的，还需要更新图像：

NSArray*indexPaths = [self.tableView indexPathsForVisibleRows];for (NSIndexPath *visibleIndexPathin indexPaths) { if (indexPath == visibleIndexPath) { MyTableViewCell *cell =(MyTableViewCell *)[self.tableView cellForRowAtIndexPath:indexPath]; cell.image= image; [cell setNeedsDisplayInRect:imageRect]; break; } }//也可不遍历，直接与头尾相比较，看是否在中间即可。最后还是前面所说过的 insertRowsAtIndexPaths:withRowAnimation:方法，插入新行需要在主线程执行，而一次插入很多行的话（例如 50 行），会长时间阻塞主线程。而换成 reloadData 方法的话，瞬间就处理完了。

171.做过 IM 开发么,谈谈对 XMPPFramework 的了解

//XMPPFramework 是一个 OS X/iOS 平台的开源项目，使用 Objective-C 实现了 XMPP 协议（RFC-3920），同时还提供了用于读写 XML 的工具，大大简化了基于 XMPP 的通信应用的开发。

XMPP 中常用对象们

XMPPStream：xmpp 基础服务类

XMPPRoster：好友列表类

XMPPRosterCoreDataStorage：好友列表（用户账号）在 core

data 中的操作类

XMPPvCardCoreDataStorage：好友名片（昵称，签名，性别，年龄等信息）在 coredata 中的操作类

XMPPvCardTemp：好友名片实体类，从数据库里取出来的都是它

xmppvCardAvatarModule：好友头像

XMPPReconnect：如果失去连接,自动重连

XMPPRoom：提供多用户聊天支持

XMPPPubSub：发布订阅

源码地址：http://code.google.com/p/xmppframework/，目前需要使用 git 才能 download 到源码。

172.你是如何实现多屏幕适配的

//一、iOS 屏幕适配发展历程

设备

适配技术

4 及以前（iPad 未出）

直接用代码计算

有了 iPad

autoResizing

有不同屏幕的 iPhone 后

autoLayout

有更多不同屏幕的 iPhone 后

sizeClass

二、各个技术的特性

1、autoLayout

帮我们确定在不同设备、不同（父 view）环境下，同一个可视单元所应具有合适的位置和尺寸(任何两个视图的关系都可以确定)

1. autoLayout 的用法：

ï 直接建立约束条件

•[self.viewaddConstraint: [NSLayoutConstraintconstraintWithItem:blueView attribute:NSLayoutAttributeLeftrelatedBy:NSLayoutRelationEqual toItem:redView attribute:NSLayoutAttributeLeftmultiplier:1 constant:0]];

这样虽然代码量比较大，但是是绝对可行的办法，也是使用 autoLayout 最根本的办法之一。

ï 使用 VFL 语言

- (void)viewDidLoad {

[super viewDidLoad];

UIButton *button=[[UIButtonalloc]init];

[button setTitle:@"点击一下"forState:UIControlStateNormal];

button.translatesAutoresizingMaskIntoConstraints=NO;

[button setBackgroundColor:[UIColorblackColor]];

[self.view addSubview:button];

NSArray *constraints1=[NSLayoutConstraint

constraintsWithVisualFormat:@"H:|-[button]-|"options:0 metrics:nil

views:NSDictionaryOfVariableBindings(button)];

NSArray *constraints2=[NSLayoutConstraint

constraintsWithVisualFormat:@"V:|-20-[button(==30)]"options:0metrics:nil

views:NSDictionaryOfVariableBindings(button)];

[self.viewaddConstraints:constraints1];

[self.viewaddConstraints:constraints2];

}

ï 使用使用第三方库，如：Masonry、UIView+AutoLayout……

autoLayout 的好处：

ï 你基本上可以不用考虑 3.5 寸和 4 寸以及即将上市的 x.x 寸屏幕不同分辨率的问题，你终于可以不用在 viewDidLoad 方法里判断不同分辨率下，不同控件应该放在哪里，或者针对不同分辨率写不同的 storyboard 和 xib；

ï 你可以抛弃那些根据不同文字来计算 tableViewCell、UILabel 高度的代码了，因为 autolayout 会帮你自动计算好；

ï 如果你的布局在横屏竖屏下变化不是特别大，你不用再为横着竖着写两套代码或者写两个 storyboard/xib 了；

2.sizeClass

在 iOS8 中，新增了 Size Classes 特性，它是对当前所有 iOS 设备尺寸的一个抽象。那我们就只把屏幕的宽和高分别分成三种情况：Compact:紧凑、Regular:宽松、Any:任意。

这样宽和高三三一整合，一共 9 中情况。如下图所示，针对每一种情况。我们可以在每种情况下设置不同的布局（包括控件的约束，甚至是控件是否显示）

sizeClass.png

对 sizeClass 的理解：sizeClass 的实质是将 iOS 屏幕分成了不同的抽象概念，这些不同的抽象组合，对应着不同的设备屏幕。所以，利用 sizeClass 可以针对同一套 UI，来适配所有的屏幕。注意：这些所有的适配，都是利用 autoLayout 来实现的，sizeClass 只是负责提供不同的屏幕尺寸。

173.谈谈你了解的设计模式,你用过哪些,他们的优缺点

//（一）代理模式

应用场景：当一个类的某些功能需要由别的类来实现，但是又不确定具体会是哪个类实现。

优势：解耦合

敏捷原则：开放-封闭原则

实例：tableview 的数据源 delegate，通过和 protocol 的配合，完成委托诉求。

列表 row 个数 delegate

自定义的 delegate

（二）观察者模式

应用场景：一般为 model 层对，controller 和 view 进行的通知方式，不关心谁去接收，只负责发布信息。

优势：解耦合

敏捷原则：接口隔离原则，开放-封闭原则

实例：Notification 通知中心，注册通知中心，任何位置可以发送消息，注册观察者的对象可以接收。

kvo，键值对改变通知的观察者，平时基本没用过。

（三）MVC 模式

应用场景：是一中非常古老的设计模式，通过数据模型，控制器逻辑，视图展示将应用程序进行逻辑划分。

优势：使系统，层次清晰，职责分明，易于维护

敏捷原则：对扩展开放-对修改封闭

实例：model-即数据模型，view-视图展示，controller 进行 UI 展现和数据交互的逻辑控制。

（四）单例模式

应用场景：确保程序运行期某个类，只有一份实例，用于进行资源共享控制。

优势：使用简单，延时求值，易于跨模块

敏捷原则：单一职责原则

实例：[UIApplication sharedApplication]。

注意事项：确保使用者只能通过 getInstance 方法才能获得，单例类的唯一实例。

java，C++中使其没有公有构造函数，私有化并覆盖其构造函数。

object c 中，重写 allocWithZone 方法，保证即使用户用 alloc 方法直接创建单例类的实例，

返回的也只是此单例类的唯一静态变量。

（五）策略模式

应用场景：定义算法族，封装起来，使他们之间可以相互替换。

优势：使算法的变化独立于使用算法的用户

敏捷原则：接口隔离原则；多用组合，少用继承；针对接口编程，而非实现。

实例：排序算法，NSArray 的 sortedArrayUsingSelector；经典的鸭子会叫，会飞案例。

注意事项：1，剥离类中易于变化的行为，通过组合的方式嵌入抽象基类

2，变化的行为抽象基类为，所有可变变化的父类

3，用户类的最终实例，通过注入行为实例的方式，设定易变行为

防止了继承行为方式，导致无关行为污染子类。完成了策略封装和可替换性。

（六）工厂模式

应用场景：工厂方式创建类的实例，多与 proxy 模式配合，创建可替换代理类。

优势：易于替换，面向抽象编程，application 只与抽象工厂和易变类的共性抽象类发生调用关系。

敏捷原则：DIP 依赖倒置原则

实例：项目部署环境中依赖多个不同类型的数据库时，需要使用工厂配合 proxy 完成易用性替换

注意事项：项目初期，软件结构和需求都没有稳定下来时，不建议使用此模式，因为其劣势也很明显，

增加了代码的复杂度，增加了调用层次，增加了内存负担。所以要注意防止模式的滥用。

174.网络通信用过哪些方式?

答

1、使用 socket 的方式进行通信。

2、使用 asynsocket 类库进行通信。

175.现有 100 个数字的乱序数组,请用一种方法将它排序.只需写出过程就行.如果乱序数组里的元素增加到 10000 个,请再次将它排序.
答:

```
int n = array.Length;

for (int i = 0; i < n - 1; i++){

for (int j = 0 ; j > n - i - 1; j++){

if (array[j] > array[j + 1]) {

int temp = array[j ];

array[j] = array[j + 1];

array[j + 1] = temp;

}
```

```
}

}
```

176.用你擅长的语言,写出获取当前系统时间的代码。

答:

NSDateFormatter *formatter = [[NSDateFormatter alloc] init];

[formatter setLocale:[[NSLocale alloc]initWithLocaleIdentifier:@"en_US"]];

[formatter setDateFormat:@"yy-MM-dd HH:mm"];

NSString *currentTime = [formatter stringFromDate:[NSDate date]];

NSLog(@"%@",currentTime);

177.请描述下“极客帮 SOHO”是如何替程序员赚到钱的。

答:

极客邦 SOHO 是独立程序员兼职任务协同平台，专业服务程序员，IT 技术的兼职，众包服务，交易服务，兼职服务，帮助程序员赚钱 looip.cncoding.net，外快。

178.写一个标准的宏 MIN,这个宏输入两个参数并返回较小的一个.

答

define MIN(X,Y)((X)>(Y)?(Y):(X))

179.Obj-c 有多重继承吗?不是的话有什么替代方法?

答:

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。你不用去考虑繁琐的多继承，虚基类的概念。多态特性在 obj-c 中通过委托来实现。

180.Static 全局变量与普通的全局变量有什么区别?static 局部变量和普通局部变量有什么区别?static 函数与普通函数有什么区别?

答:

static 全局变量与普通的全局变量有什么区别：static 全局变量只初使化一次，防止在其他文件单元中被引用;

static 局部变量和普通局部变量有什么区别：static 局部变量只被初始化一次，下一次依据上一次结果值；

static 函数与普通函数有什么区别：static 函数在内存中只有一份，普通函数在每个被调用中维持一份拷贝

181.MVC 模式的理解

答:

MVC，全称 Model（模型）-View（视图）-Controller（控制器），这是一种开发模式，他的好处是可以将界面和业务逻辑分离。

Model（模型），是程序的主体部分，主要包含业务数据和业务逻辑。在模型层，还会涉及到用户发布的服务，在服务中会根据不同的业务需求，更新业务模型中的数据。

View(视图），是程序呈现给用户的部分，是用户和程序交互的接口，用户会根据具体的业务需求，在 View 视图层输入自己特定的业务数据，并通过界面的事件交互，将对应的输入参数提交给后台控制器进行处理。

Controller（控制器），Controller 是用来处理用户输入数据，已经更新业务模型的部分。控制器中接收了用户与界面交互时传递过来的数据，并根据数据业务逻辑来执行服务的调用和更新业务模型的数据和状态。

182.用变量 a 给出下面的定义

a)一个整型数（An integer）

b)一个指向整型数的指针（A pointer to an integer）

c)一个指向指针的的指针，它指向的指针是指向一个整型数（A pointer to a pointer to an integer）

d)一个有 10 个整型数的数组（An array of 10 integers）

e)一个有 10 个指针的数组，该指针是指向一个整型数的（An array of 10 pointers to integers）

f)一个指向有 10 个整型数数组的指针（A pointer to an array of 10 integers）

g)一个指向函数的指针，该函数有一个整型参数并返回一个整型数（A pointer to a function that takes an integer as an argument and returns an integer）

h)一个有 10 个指针的数组，该指针指向一个函数，该函数有一个整型参数并返回一个整型数（An array of ten pointers to functions that take an integer argument and return an integer）

答案：

a) int a; // An integer

b) int *a; // A pointer to an integer

c) int **a; // A pointer to a pointer to an integer

d) int a[10]; // An array of 10 integers

e) int *a[10]; // An array of 10 pointers to integers

f) int (*a)[10]; // A pointer to an array of 10 integers

g) int (*a)(int);

// A pointer to a function a that takes an integer argument and returns an integer

h) int (*a[10])(int);

// An array of 10 pointers to functions that take an integer argument and return an integer

183、在一个数组中存在一万条以上的字符串，现在要对数组中所有字符串进行拼接操作，请写出拼接方法（要考虑到性能及内存占用情况，ARC 模式）。

NSString *string = [arraycomponentsJoinedByString:@","];

184、请举例说明代理和通知两种方式分别适合在什么情况下使用？

代理：一般控件用的比较多，其实也可以用 block 实现，如果实现的接口比较多的话，建议用代理，如 UITableview。

通知：这东西是全局的，而且是同步的，如果你要全局发送消息，并且做的事情时间不长，不会阻塞线程的话，建议使用。

185、是否使用过 SQLite 或者 FMDataBase 对数据库进行操作，并试述对事务概念的理解。

FMDatabase 是 IOS 中 Sqlite 数据库操作类

186、以下两种 GCD 队列创建有什么不同？

dispatch_queue_t queue =
dispatch_queue_create("MyQueue",DISPATCH_QUEUE_SERIAL);

dispatch_queue_t queue =dispatch_queue_create(@“MyQueue",
DISPATCH_QUEUE_CONCURRENT);

//生成一个串行队列，队列中的 block 按照先进先出（FIFO）的顺序去执行，实际上为单线程执行。第一个参数是队列的名称，在调试程序时会非常有用，所有尽量不要重名了。

//生成一个并发执行队列，block 被分发到多个线程去执行

187、运行以下代码会打印什么结果？为什么？

dispatch_queue_t queue =dispatch_queue_create("MyQueue",
DISPATCH_QUEUE_SERIAL);

dispatch_apply(3, queue, ^(size_t i) {

NSLog(@"apply loop:%zu", i);

dispatch_apply(3, queue, ^(size_t j) {

NSLog(@"apply loop inside:%zu", j);

});

});

apply loop:0

188、简单说明你对 block 的理解以及使用 block 有什么好处。

答:block 是对象，它封装了一段代码，这段代码可以在任何时候执行。block 可以作为函数参数或者函数的返回值，而其本身又可以带输入参数或返回值。它和传统的函数指针很类似，但是有区别：block 是 inline 的，并且它对局部变量是只读的。

好处: Blocks 更清晰。比如一个 viewController 中有多个弹窗事件，Delegate 就得对每个事件进行判断识别来源。而 Blocks 就可以在创建事件的时候区分开来了。这也是为什么现在苹果 API 中越来越多地使用 Blocks 而不是 Delegate。

189、setValue：forKey 和 setObject：forKey 的区别是什么？

答: 1, setObject：forkey：中 value 是不能够为 nil 的，不然会报错。

setValue：forKey：中 value 能够为 nil，但是当 value 为 nil 的时候，会自动调用 removeObject：forKey 方法

2, setValue：forKey：中 key 的参数只能够是 NSString 类型，而 setObject：forKey：的可以是任何类型

190、try-catch-finally 的作用和使用方法。

Java:

1，一个方法内可以有多个 try…catch…finally 语句块，还可以彼此嵌套，比如下面这个方法：

2，如果一个有返回值的方法内有多个 try…catch…finally 语句块，return 语句要么写在任意一个 try…catch 内，要么写在方法的最后，否则编译无法通过，如果 return 语句写在方法的最后，那么以上 try…catch…finally 语句中的每一个 finally 块内的代码都将会执行；

3，无论方法内是否发生异常（jvm 可以处理的异常），finally 块内的代码都将会执行。

191、请写出同步网络请求和异步网络请求函数。

答:首先在 ios 模拟器上创建一个 text 窗口（我起名叫 tongbu）：

```
//同步网络请求函数

//获取名叫 tongbu 文本框的内容

NSString *txt=self.tongbu.text;

//创建 url 对象

NSURL *url=[NSURLURLWithString:txt];

//创建请求对象

NSURLRequest*req=[NSURLRequest requestWithURL:url];

//发起同步，赶回数据给 data

NSData*data=[NSURLConnection sendSynchronousRequest:req returningResponse:nilerror:nil];

//异步网络请求

在 ViewController.m 文件上的- (void)viewDidLoad 方法里

NSString*txt2=self.tongbu.text;

//创建 url 对象

NSURL *url2=[NSURLURLWithString:txt2];

//创建请求对象

NSURLRequest*req2=[NSURLRequest requestWithURL:url2];

//发送请求并建立一个代理

[NSURLConnectionconnectionWithRequest:req2 delegate:self];
```

```
//因为代理人是自己所以让自己遵守协议

协议在 ViewController.h 文件里

@interface ViewController :

UIViewController//因为代理对象是对象所以让自己遵守协议

//同时创建一个 NSMutableData 类型的对象来接从网络上接收的数据，同时创建 3 个协议方法来进行接收数据

@property(retain,nonatomic)NSMutableData*data;

//协议方法

//1 连接接收响应，表示成功建立连接

-(void)connection:(NSURLConnection*)connection didReceiveResponse:(NSURLResponse *)response{

self.data=[[NSMutableData alloc]init];//创建代理对象，并初始化数据

}

//2 连接接收数据

//形参(NSData *)表示接收到的数据

-(void)connection:(NSURLConnection*)connection didReceiveData:(NSData *)data{

[self.data

appendData:data];//向 data 反复添加数据

}

//3 连接成功

-(void)connectionDidFinishLoading:(NSURLConnection*)connection{

NSLog(@"连接成功");

}
```

192、从用户体验角度举例说明同步和异步。

答:1.同步意为着线程阻塞，在主线程中使用此方法会不响应任何用户事件。所以，在应用程序设计时，大多被用在专门的子线程增加用户体验，或用异步请求代替。

2.异步请求的好处是不阻塞当前线程，但相对于同步请求略为复杂，至少要添加两个回调方法来获取异步事件

答:从用户的体验来说,异步请求数据的 APP 比同步请求的 APP 操作更加流畅,快捷,

193、声明 NSString 类型的属性时，用 copy 和 strong 的区别是什么？

copy 修饰的 NSString,在初始化时,如果来源是 NSMutableString 的话,会对来源进行一次深拷贝,将来源的内存地址复制一份,这样,两个对象就一点关系就没有了,无论你怎么操作来源,都不会对自己的 NSString 有任何影响

194、谈谈 Object-C 的内存管理方式及过程？

从一段内存被申请之后，就存在一个变量用于保存这段内存被使用的次数，我们暂时把它称为计数器，当计数器变为 0 的时候，那么就是释放这段内存的时候，比如说，当在程序 A 里面一段内存被成功申请完成之后，那么这个计数器就从 0 变成了 1（我们把这个过程叫做 alloc）然后程序 B 也需要使用这个内存，那么计数器就从 1 变成了 2（我们把这个过程叫做 retain）紧接着程序 A 不再需要这段内存了，那么程序 A 就把这个计数器减 1（我们把这个过程叫做 release）程序 B 也不再需要这段内存的时候，那么也把计数器减 1（这个过程还是 release）当系统（也就是 Foundation）发现这个计数器变成了 0，那么就会调用内存回收程序把这段内存回收（我们把这个过程叫做 dealloc）

195、static 全局变量与普通的全局变量有什么区别？static 普通函数有什么区别？

全局变量(外部变量)的说明之前再冠以 static 就构成了静态的全局变量。全局变量本身就是静态存储方式，静态全局变量当然也是静态存储方式。这两者在存储方式上并无不同。这两者的区别虽在于非静态全局变量的作用域是整个源程序，当一个源程序由多个源文件组成时，非静态的全局变量在各个源文件中都是有效的。而静态全局变量则限制了其作用域，即只在定义该变量的源文件内有效，在同一源程序的其它源文件中不能使用它。由于静态全局变量的作用域局限于一个源文件内，只能为该源文件内的函数公用，因此可以避免在其它源文件中引起错误。从以上分析可以看出，把局部变量改变为静态变量后是改变了它的存储方式即改变了它的生存期。把全局变量改变为静态变量后是改变了

它的作用域，限制了它的使用范围。static 函数与普通函数有什么区别？只在当前源文件中使用的函数应该说明为内部函数(static)，内部函数应该在

当前源文件中说明和定义。对于可在当前源文件以外使用的函数，应该在一个头文件中说明，要使用这些函数的源文件要包含这个头文件。

196、Objective-C 堆和栈的区别？

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。

申请大小：

栈：在 Windows 下,栈是向低地址扩展的数据结构，是一块连续的内存的区域。这句话的意思是栈顶的地址和栈的最大容量是系统预先规定好的，在 WINDOWS 下，栈的大小是 2M（也有的说是 1M，总之是一个编译时就确定的常数），如果申请的空间超过栈的剩余空间时，将提示 overflow。因此，能从栈获得的空间较小。

堆：堆是向高地址扩展的数据结构，是不连续的内存区域。这是由于系统是用链表来存储的空闲内存地址的，自然是不连续的，而链表的遍历方向是由低地址向高地址。堆的大小受限于计算机系统中有效的虚拟内存。由此可见，堆获得的空间比较灵活，也比较大。

碎片问题：对于堆来讲，频繁的 new/delete 势必会造成内存空间的不连续，从而造成大量的碎片，使程序效率降低。对于栈来讲，则不会存在这个问题，因为栈是先进后出的队列，他们是如此的一一对应，以至于永远都不可能有一个内存块从栈中间弹出

分配方式：堆都是动态分配的，没有静态分配的堆。栈有 2 种分配方式：静态分配和动态分配。静态分配是编译器完成的，比如局部变量的分配。动态分配由 alloca 函数进行分配，但是栈的动态分配和堆是不同的，他的动态分配是由编译器进行释放，无需我们手工实现。

分配效率：栈是机器系统提供的数据结构，计算机会在底层对栈提供支持：分配专门的寄存器存放栈的地址，压栈出栈都有专门的指令执行，这就决定了栈的效率比较高。堆则是 C/C++函数库提供的，它的机制是很复杂的。

197、__block 和__weak 修饰符的区别是什么？

答：1，在 MRC 时代，__block 修饰，可以避免循环引用；ARC 时代，__block 修饰，同样会引起循环引用问题；

2，__block 不管是 ARC 还是 MRC 模式下都可以使用，可以修饰对象，还可以修饰基本数据类型；

3，__weak 只能在 ARC 模式下使用，也只能修饰对象，不能修饰基本数据类型；

4，__block 对象可以在 block 中被重新赋值，__weak 不可以；

198、如何解决 TableView 卡顿的问题？

1.答：使用不透明视图。

2.不要重复创建不必要的 table

cell。

3.减少视图的数目。

4.不要做多余的绘制工作。

5.预渲染图像。

6.不要阻塞主线程。当然，在不需要响应用户请求时，也可以增加下载线程数，以加快下载速度：

199、简要说下 Http 通信协议的原理，与 Socket 协议的区别有哪些？

答：HTTP 协议：简单对象访问协议，对应于应用层，HTTP 协议是基于 TCP 连接的

tcp 协议：对应于传输层

ip 协议：对应于网络层

TCP/IP 是传输层协议，主要解决数据如何在网络中传输；而 HTTP 是应用层协议，主要解决如何包装数据。

Socket 是对 TCP/IP 协议的封装，Socket 本身并不是协议，而是一个调用接口（API），通过 Socket，才能使用 TCP/IP 协议。

http 连接：http 连接就是所谓的短连接，即客户端向服务器端发送一次请求，服务器端响应后连接即会断掉；

socket 连接：socket 连接就是所谓的长连接，理论上客户端和服务器端一旦建立起连接将不会主动断掉；但是由于各种环境因素可能会是连接断开，比如说：服务器端或客户端主机 down 了，网络故障，或者两者之间长时间没有数据传输，网络防火墙可能会断开该连接以释放网络资源。

200、MVC 是什么？有什么特性？你还熟悉哪些设计模式，请简要说明。

他们有重复的

201、autorelease 和垃圾回收机制（gc）有什么关系？

答：autorelease 只是延迟释放,gc 是每隔一段时间询问程序,看是否有无指针指向的对象,若有,就将它回收。他们两者没有什么关系。

**202、假设类名为 MyManager 的类有以下单例方法，请写出单例方法的实现：+（MyManager ）sharedManager；*
答:

```
+ (MyManager*)sharedManager

{

static MyManager *sharedMyManagerInstance = nil;

static dispatch_once_t predicate;

dispatch_once(&predicate, ^{

sharedMyManagerInstance =[[self alloc] init];

});

return sharedMyManagerInstance;

}
```

203、全局变量可不可以定义在可被多个.C 文件包含的头文件中？为什么？

答:可以.在不适用 static 或者 const(隐式 static)情况下,变量的定义只能出现一次，否则会导致重复定义。但却可以声明多次。因此全局变量不可以定义在头文件中。因为当该头文件被多个 c 文件包含的话，会导致重复定义。因此一般做法是在某个特定的头文件中声明，而在另外一个特定的 c 文件中定义。需要使用就包含前者。

204、TCP/IP 通信建立的过程怎样，端口有什么作用？

答:发出将建立通信会话的第一个数据包之前，发送方主机上的 TCP/IP 协议执行以下四个不同的步骤：

1.TCP/IP 将主机名或 NetBIOS 名称解析为 IP 地址。

2.使用目标 IP 地址和 IP 路由表，TCP/IP 确定要使用的接口和下一跃点 IP 地址。

3.对于共享访问技术（例如，以太网、令牌环和分布式光纤数据接口(FDDI)）上的单播 IP 流量，地址解析协议(ARP)将下一跃点 IP 地址解析为媒体访问控制(MAC)地址（也称为数据链接层地址）。

对于以太网和 FDDI 上的多播 IP 流量，目标多播 IP 地址会被映射到相应的多播 MAC 地址。对于令牌环上的多播 IP 流量，使用功能地址 0xC0-00-00-04-00-00。对于共享访问技术上的广播流量，MAC 地址会被映射到 0xFF-FF-FF-FF-FF-FF。4.之后，IP 数据报会被发送到通过 ARP 解析的 MAC 地址、多播映射或 MAC 级广播地址。

网络访问要通过不同的协议进行，各种协议要通过不同的端口进行访问，如 25 端口是邮件端口，3389 超级终端（就是木马程序最想打开的端口），8000=腾讯 OICQ 服务器端等等很多很多，记住常用的几个就可以了。

端口:说白了就相当于门,每个门都对应着相对的 TCP/IP