部分答案来自于网络，有问题进作者创建iOS交流群：656315826，也欢迎广大iOS开发者进群交流，面试题资料分享，内推、招聘

面试题题目

1、iOS本地数据存储都有哪几种方式?

2、写出方法获取iOS内存使用情况。

3、深拷贝和浅拷贝的理解?

4.怎样实现一个singleton的类。

5、什么是安全释放?

6、RunLoop是什么?

7、简述OC中内存管理机制。与retain配对使用的方法是dealloc还是release,为什么?需要与alloc配对使用的方法是dealloc还是release,为什么?
readwrite,readonly,assign,retain,copy,nonatomic 、atomic、strong、weak属性的作用?

8、什么是序列化和反序列化,可以用来做什么?如何在OC中实现复杂对象的存储?

9、类变量的@protected ,@private,@public,@package,声明各有什么含义?

10、写一个标准宏MIN,这个宏输入两个参数并返回较小

的一个?

11、线程是什么?进程是什么?二者有什么区别和联系?

12、谈谈你对多线程开发的理解?ios中有几种实现多线程的方法?

13、iphone os有没有垃圾回收机制?简单阐述一下OC内存管理。

14、线程同步和异步的区别?IOS中如何实现多线程的同步?

15、假设有一个字符串aabcad,请写一段程序,去掉字符串中不相邻的重复字符串,即上述字符串处理之后的输出结果为:aabcd

16、UIImage初始化一张图片有几种方法?简述各自的优缺点。

17、写一个便利构造器。

18、使用UITableView时候必须要实现的几种方法?

19、获取一台设备唯一标识的方法有哪些?

20、iOS类是否可以多继承?如果没有,那可以用其他方法实现吗?简述实现过程。

21、为什么很多内置类如UITableViewControl的delegate属性都是assign而不是retain?请举例说明。

22、描述应用程序的启动顺序。

23、堆和栈的区别?

24、ViewController 的 alloc,loadView, viewDidLoad,viewWillAppear,viewDidUnload,dealloc、init分别是在什么时候调用的?在自定义ViewController的时候这几个函数里面应该做什么工作?

25、简述应用程序按Home键进入后台时的生命周期,以及从后台回到前台时的生命周期?

26：在KVO中，他是怎么知道监听的对象发生了变化?

27：字典的工作原理？怎100w个中是怎么快速去取value?

28：一个上线的项目，知道这个方法可能会出问题，在不破坏改方法前提下，怎么搞?

29：Block和函数指针的区别?

答案部分:

1、iOS本地数据存储都有哪几种方式?

NSKeyedArchiver

NSUserDefaults
Write写入方式（plist文件、txt文件等）
SQLite3
CoreData
(问题扩展：什么情况下使用什么样的数据存储)
（1）NSKeyedArchiver：采用归档的形式来保存数据，数据对象需要遵守NSCoding协议，对象对应的类必须提供encodeWithCoder:和initWithCoder:方法。缺点：只能一次性归档保存以及一次性解压。所以只能针对小量数

据，对数据操作比较笨拙，如果想改动数据的某一小部分，需要解压或归档整个数据。

（2）NSUserDefaults：用来保存应用程序设置和属性、用户保存的数据。用户再次打开程序或开机后这些数据仍然存在。NSUserDefaults可以存储的数据类型包括：NSData、NSString、NSNumber、NSDate、NSArray、NSDictionary。缺点：如果要存储其他类型，需要转换为前面的类型，才能用NSUserDefaults存储。

（3）Write写入方式：永久保存在磁盘中。第一步：获得文件即将保存的路径：第二步：生成在该路径下的文件：第三步：往文件中写入数据：最后：从文件中读出数据：

（4）SQLite：采用SQLite数据库来存储数据。SQLite作为一中小型数据库，应用ios中，跟前三种保存方式相比，相对比较复杂一些。

（5）CoreData：系统自带的数据库存储。

2、写出方法获取iOS内存使用情况。

（问题扩展：如何利用Xcode观察内存使用情况）

```
// 获取当前设备可用内存及所占内存的头文件
#import <sys/sysctl.h>
#import <mach/mach.h>

// 获取当前设备可用内存(单位：MB)
- (double)availableMemory
{
 vm_statistics_data_t vmStats;
 mach_msg_type_number_t infoCount = HOST_VM_INFO_COUNT;
 kern_return_t kernReturn = host_statistics(mach_host_self(),

                                            HOST_VM_INFO,
                                            (host_info_t)&vmSta
ts,
                                            &infoCount);

 if (kernReturn != KERN_SUCCESS)
 {
   return NSNotFound;
```

```
 }

 return ((vm_page_size *vmStats.free_count) / 1024.0) / 1024.0;

}


// 获取当前任务所占用的内存（单位：MB）
- (double)usedMemory
{
 task_basic_info_data_t taskInfo;
 mach_msg_type_number_t infoCount = TASK_BASIC_INFO_COUNT;
 kern_return_t kernReturn = task_info(mach_task_self(),
                                      TASK_BASIC_INFO,
                                      (task_info_t)&taskInfo,

                                      &infoCount);


 if (kernReturn != KERN_SUCCESS)
 {
   return NSNotFound;
 }

 return (taskInfo.resident_size / 1024.0 / 1024.0);
}
```

3、深拷贝和浅拷贝的理解?

对实例进行深拷贝时当前类需要实现NSCopying协议。
浅拷贝是复制出来一个跟原对象相同地址的对象
深拷贝时复制一个跟源对象不同地址的对象 改变源对象对新对象没有影响

4.怎样实现一个singleton的类。

问题扩展：单例的好处是什么？——节省内存

```
+ (AccountManager *)sharedManager
{
        static AccountManager *staticInstance = nil;
        static dispatch_once_t predicate;
        dispatch_once(&predicate, ^{
                staticInstance = [[self alloc] init];
        });
    return staticInstance;
}
```

5、什么是安全释放?

置nil 再释放

6、RunLoop是什么?

Run loops 是线程相关的的基础框架的一部分。
一个 run loop 就是一个事件处理 的循环,用来不停的调度工作以及处理输入事件。
使用 run loop的目的是让你的线程在有工作的时候忙于工作,而没工作的时候处于休眠状态。
Runloop还可以在loop在循环中的同时响应其他输入源，比如界面控件的按钮，手势等。

7、简述OC中内存管理机制。与retain配对使用的方法是dealloc还是release，为什么？需要与alloc配对使用的方法是dealloc还是release，为什么？readwrite，readonly，assign，retain，copy，nonatomic 、atomic、strong、weak属性的作用?

管理机制：使用了一种叫做引用计数的机制来管理内存中的对象。
OC中每个对象都对应着他们自己的引用计数，引用计数可以理解为一个整数计数器，当使用alloc方法创建对象的时候，持有计数会自动设置为1。当你向一个对象发送retain消息 时，持有计数数值会增加1。相反，当你像一个对象发送release消息时，持有计数数值会减小1。当对象的持有计数变为0的时候，对象会释放自己所占用的内存。

retain(引用计数加1)->release（引用计数减1）
alloc（申请内存空间）->dealloc(释放内存空间)
readwrite: 表示既有getter，也有setter (默认)
readonly: 表示只有getter，没有setter
nonatomic:不考虑线程安全
atomic:线程操作安全（默认）
线程安全情况下的setter和getter：

```
- (NSString *) value
{
  @synchronized(self) {
     return [[_value retain] autorelease];
  }
}


- (void)setValue:(NSString *)aValue
{
  @synchronized(self) {
   [aValue retain];
   [_value release];
   _value = aValue;
  }
}
```

retain: release旧的对象，将旧对象的值赋予输入对象，再提高输入对象的索引计数为1
assign: 简单赋值，不更改索引计数（默认）
copy: 其实是建立了一个相同的对象,地址不同（retain：指针拷贝 copy：内容拷贝）
strong:（ARC下的）和（MRC）retain一样（默认）
weak:（ARC下的）和（MRC）assign一样， weak当指向的内存释放掉后自动nil化，防止野指针
unsafe_unretained 声明一个弱应用，但是不会自动nil化，也就是说，如果所指向的内存区域被释放了，这个指针就是一个野指针了。autoreleasing 用来修饰一个函数的参数，这个参数会在函数返回的时候被自动释放。

8、什么是序列化和反序列化，可以用来做什么？如何在OC中实现复杂对象的存储?

序列化是把对象转化成字节序列的过程 反序列化是把字节序列恢复成对象
将对象写到文件或者数据库里，并且能读取出来
遵循NSCoding协议 实现复杂对象的存储 实现该协议后可以对其进行打包或解包，转化成NSData

9、类变量的@protected ,@private,@public,@package，声明各有什么含义?

@private：作用范围只能在自身类
@protected：作用范围在自身类和继承自己的子类（默认）
@public：作用范围最大，可以在任何地方被访问。
@package：这个类型最常用于框架类的实例变量,同一包内能用，跨包就不能访问

10、写一个标准宏MIN，这个宏输入两个参数并返回较小的一个?

#define MIN(X,Y) ((X)>(Y)?(Y):(X))
扩展：在定义宏的时候需要注意哪些问题?
宏全部大写 写在#import 下 @interface上 结尾无分号

11、线程是什么？进程是什么？二者有什么区别和联系?

一个程序至少有一个进程,一个进程至少有一个线程：
进程：一个程序的一次运行，在执行过程中拥有独立的内存单元，而多个线程共享一块内存
线程：线程是指进程内的一个执行单元。
联系：线程是进程的基本组成单位
区别：(1)调度：线程作为调度和分配的基本单位，进程作为拥有资源的基本单位
(2)并发性：不仅进程之间可以并发执行，同一个进程的多个线程之间也可并发执行
(3)拥有资源：进程是拥有资源的一个独立单位，线程不拥有系统资源，但可以访问隶属于进程的资源.

(4)系统开销：在创建或撤消进程时，由于系统都要为之分配和回收资源，导致系统的开销明显大于创建或撤消线程时的开销。
举例说明：操作系统有多个软件在运行（QQ、office、音乐等），这些都是一个个进程，而每个进程里又有好多线程（比如QQ，你可以同时聊天，发送文件等）

12、谈谈你对多线程开发的理解？ios中有几种实现多线程的方法?

好处：
（1）使用线程可以把占据时间长的程序中的任务放到后台去处理
（2）用户界面可以更加吸引人，这样比如用户点击了一个按钮去触发某些事件的处理，可以弹出一个进度条来显示处理的进度
（3）程序的运行速度可能加快
（4）在一些等待的任务实现上如用户输入、文件读写和网络收发数据等，线程就比较有用了。
缺点：
（1）如果有大量的线程,会影响性能,因为操作系统需要在它们之间切换。
（2）更多的线程需要更多的内存空间。
（3）线程的中止需要考虑其对程序运行的影响。
（4）通常块模型数据是在多个线程间共享的，需要防止线程死锁情况的发生。
实现多线程的方法：
NSObject类方法
NSThread
NSOperation
GCD

13、iphone os有没有垃圾回收机制？简单阐述一下OC内存管理。

iphone os没有垃圾回收机制 oc的内存管理是谁创建谁释放 程序中遇到retain 该对象引用计数+1 遇release该对象引用计数-1 retainCount为0时 内存释放

14、线程同步和异步的区别？IOS中如何实现多线程的同步?

异步：举个简单的例子 就是游戏，游戏会有图像和背景音乐
同步:是指一个线程要等待上一个线程执行完之后才开始执行当前的线程,上厕所
NSOperationQueue：maxcurrentcount
NSConditionLock
GCD->http://blog.csdn.net/onlyou930/article/details/8225906

15、假设有一个字符串aabcad，请写一段程序，去掉字符串中不相邻的重复字符串，即上述字符串处理之后的输出结果为：aabcd

```
NSMutableString * str = [[NSMutableString alloc]initWithFormat;
@“aabcad”];
for (int i = 0 ,i < str.length - 1 ;i++)
{
   unsigned char a = [str characterAtIndex:i];
   for (int j = i + 1 ,j < str.length ,j++)
   {
       unsigned char b = [str characterAtIndex:j];
       if (a == b )
    {
           if (j == i + 1)
        {

        }
        else
        {
                [str deleteCharactersInRange:NSMakeRange(j, 1)];

           }
       }
  }
}
NSLog(@“%@”,str);
```

16、UIImage初始化一张图片有几种方法？简述各自的

优缺点。

3种
imageNamed:系统会先检查系统缓存中是否有该名字的Image，如果有的话，则直接返回，如果没有，则先加载图像到缓存，然后再返回。
initWithContentsOfFile：系统不会检查系统缓存，而直接从文件系统中加载并返回。
imageWithCGImage：scale：orientation当scale=1

17、写一个便利构造器。

```
//id代表任意类型指针，这里代表Student *,类方法
+ (id)studentWithName:(NSString *)newName andAge:(int)newAge
{
   Student *stu=[[Student alloc]initName:newName andAge:newAge]
;
   return [stu autorelease];
   // 自动释放
}
```

18、使用UITableView时候必须要实现的几种方法?

```
- (NSInteger)tableView:(UITableView*)tableViewNumberOfRowsInSec
tion:(NSInteger)section;这个方法返回每个分段的行数，不同分段返回不同的
行数可以用switch来做，如果是单个列表就直接返回单个你想要的函数即可。-(UITa
bleViewCell*)tableView:(UITableView*)tableViewCellForRowAtIndex
Path:(NSIndexPath)indexPath;这个方法是返回我们调用的每一个单元格。通过
我们索引的路径的section和row来确定
```

19、获取一台设备唯一标识的方法有哪些?

(1)UDID
UDID的全称是Unique Device Identifier，顾名思义，它就是苹果IOS设备的唯一识别码，它由40个字符的字母和数字组成。
(2)UUID
UUID是Universally Unique Identifier的缩写,中文意思是通用唯一识别码.
(3)MAC Address
(4)OPEN UDID

(5)广告标识符
(6)Vindor标示符
Vendor是CFBundleIdentifier（反转DNS格式）的前两部分。来自同一个运营商的应用运行在同一个设备上，此属性的值是相同的；不同的运营商应用运行在同一个设备上值不同。
经测试，只要设备上有一个tencent的app，重新安装后的identifierForVendor值不变，如果tencent的app全部删除，重新安装后的identifierForVendor值改变。
*ios7以后使用keychain

20、iOS类是否可以多继承？如果没有，那可以用其他方法实现吗？简述实现过程。

不可以多继承，用protocol实现

21、为什么很多内置类如UITableViewControl的delegate属性都是assign而不是retain？请举例说明。

防止循环引用

22、描述应用程序的启动顺序。

（1）程序入口main函数创建UIApplication实例和UIApplication代理实例。
（2）在UIApplication代理实例中重写启动方法，设置第一ViewController。
（3）在第一ViewController中添加控件，实现应用程序界面。

23、堆和栈的区别?

堆需要用户手动释放内存，而栈则是编译器自动释放内存
问题扩展：要知道OC中NSString的内存存储方式

24、ViewController 的 alloc，loadView, viewDidLoad,viewWillAppear,viewDidUnload,dealloc、init分别是在什么时候调用的？在自定义ViewController的时候这几个函数里面应该做什么工作?

```
alloc // 申请内存时调用
loadView // 加载视图时调用
```

```
ViewDidLoad // 视图已经加载后调用
ViewWillAppear // 视图将要出现时调用
ViewDidUnload // 视图已经加载但没有加载出来调用
dealloc // 销毁该视图时调用
init // 视图初始化时调用
```

25、简述应用程序按Home键进入后台时的生命周期，以及从后台回到前台时的生命周期?

```
// 进入后台时
-(void)applicationWillResignActive:(UIApplication *)application
;
-(void)applicationDidEnterBackground:(UIApplication *)applicati
on;
// 进入前台时
-(void)applicationDidEnterForeground:(UIApplication *)applicati
on;
-(void)applicationWillResignActive:(UIApplication *)application
;

// 各个程序运行状态时代理的回调:
// 当进程启动但还没进入状态保存
- (BOOL)application:(UIApplication *)application willFinishLaun
chingWithOptions:(NSDictionary *)launchOptions
当进程启动基本完成程序准备开始运行
- (BOOL)application:(UIApplication *)application didFinishLaunc
hingWithOptions:(NSDictionary *)launchOptions
// 当应用程序将要入非活动状态执行，在此期间，应用程序不接收消息或事件，比如
来电话了
- (void)applicationWillResignActive:(UIApplication *)applicatio
n
当应用程序入活动状态执行，这个刚好跟上面那个方法相反
- (void)applicationDidBecomeActive:(UIApplication *)application

// 当程序被推送到后台的时候调用。所以要设置后台继续运行，则在这个函数里面设
```

```
置即可
- (void)applicationDidEnterBackground:(UIApplication *)applicat
ion
// 当程序从后台将要重新回到前台时候调用，这个刚好跟上面的那个方法相反
- (void)applicationWillEnterForeground:(UIApplication *)applica
tion
// 当程序将要退出是被调用，通常是用来保存数据和一些退出前的清理工作。这个需
要要设置UIApplicationExitsOnSuspend的键值。
- (void)applicationWillTerminate:(UIApplication *)application
// 当程序载入后执行
- (void)applicationDidFinishLaunching:(UIApplication*)applicati
on
```

26：在KVO中，他是怎么知道监听的对象发生了变化?

KVO底层封装了KVC, KVC最重要的原理就是isa-swizzling ,我们在利用KVO的时候就传入了观察者,对象,以及观察的属性.我们在底层就通过对象的方法名得到环境参数,isa结合环境参数直接得出方法接口(SEL),最后得到该方法的函数实现(IMP).我们对应属性的变化,就通过对应的settr方法,来到IMP,就会消息转发从动态子类转发给父类.同时会触发KVO的 -(void)observeValueForKeyPath:(NSString *)keyPath ofObject:(id)object change:(NSDictionary *)change context:(void *)context;方法.拿到变化

27：字典的工作原理？

1.NSDictionary（字典）是使用 hash表来实现key和value之间的映射和存储的，hash函数设计的好坏影响着数据的查找访问效率。

```
- (void)setObject:(id)anObject forKey:(id <NSCopying>)aKey;
```

2.Objective-C 中的字典 NSDictionary 底层其实是一个哈希表，实际上绝大多数语言中字典都通过哈希表实现，

28：一个上线的项目，知道这个方法可能会出问题，在不破坏改方法前提下，怎么搞?

利用runtime , method-swizzling 黑魔法,交换方法,通过改变我们原始的方法的IMP的指向.指向我们要处理正确逻辑的函数实现.这样的方式,还可以用作

来页面统计,接口记录,方法记录!
在知道会出问题的前提上,我们可以做相应宏控制是否开启监控或者异常回调处理

29：Block和函数指针的区别?

第一个区别，函数指针是对一个函数地址的引用，这个函数在编译的时候就已经确定了。而block是一个函数对象，是在程序运行过程中产生的。在一个作用域中生成的block对象分配在栈(stack)上，和其他所有分配在栈上的对象一样，离开这个作用域，就不存在了。
Block允许开发者在两个对象之间将任意的语句当做数据进行传递，往往这要比引用定义在别处的函数直观。
Block实体形式如下：

```
^(传入参数列){行为主体};
Block实体开头是“^”，接着是由小括号所包起来的参数列（比如 int a, int b, int
c），行为主体由大括号包起来，专有名字叫做block literal。行为主体可以用return回传值，类型会被compiler自动辨别。如果没有参数列要写成：
^(void)。
例如下面的一个例子：
^(int a){return a*a;};

这是代表Block会回传输入值的平方值（int a 就是参数列， return a*a;
就是行为主体）。记得行为主体里最后要加“;”，因为是叙述，而整个{}最后也要加“;”，因为Block是物件实体。
在ios开发中，blocks是对象，它封装了一段代码，这段代码可以在任何时候执行。Blocks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：blocks是inline的，并且它对局部变量是只读的。
Blocks的定义：
int (^myBlock) (int a,int b) = ^(int a,int b){
return a+b;
};
定义了一个名为myBlock的blocks对象，它带有两个int参数，返回int。等式右边就是blocks的具体实现，是不是有点像方法的定义?
```

Blocks可以访问局部变量，但是不能修改。比如下面的代码就会报编译错

```
int num = 0;
//使用block
int
(^myBlock) (int a,int b) = ^(int a,int b){
num = a+b;
return num;
};
```

如果要修改就要加关键字：__block

(注意，是两个下划线"_")

```
__block int num = 0;
//使用block
int (^myBlock) (int a,int b) = ^(int a,int b){
num = a+b;
return num;
};
```