概览

- 平台相关

  一、c 语言语法，常用数据结构

  二、objc 语法

  三、iOS api

  四、项目相关

- 平台无关

  一、多线程和网络编程

  二、数据结构 和 算法

  三、设计模式

平台相关

一、C语言语法，常用数据结构

什么是指针

type *p;
一类数据类型，用来存储内存单元的编号
指针 不完全等于 地址，有类型的标识
很多类型的指针，指向数据、指向方法、void 型
不同的编译环境，sizeof 可能不同

指针和数组有什么区别

char *p = “hello”; char a[] = “hello”;
变量名代表的意义:
p：只是一个指向某个内存块的一个变量
数组名：1、sizeof 的时候代表整个数组 2、代表数组首个元素的首地址，在值上等于数组的地址
对变量名的修改：
数组相当于 chat * const p; 常型指针
对代表数据的操作：
修改、复制、sizeof、予以区分，参考《指针和数组的区别》

写出标准宏MIN，这个宏输入两个参数并返回较小的一个

```
#define MIN(a,b) ((a)>(b)?(b):(a))
```

延伸：

```
#define RADIANS_TO_DEGREES(radians) ((radians) * (180.0 / M_PI))
#define DEGREES_TO_RADIANS(angle) ((angle) / 180.0 * M_PI)
```

需要了解下 BuildSetting 的 DEBUG_MODE

如何引用一个已经定义过的全局变量

多个.m文件定义同名的引起link 错误

```
int global_Value = 1;
extern int global_Value;
```

编译原理：.h、.m 怎么编译，怎么链接，怎么打包

static 全局变量和普通变量的区别、局部变量呢？函数

Static变量 限定了作用域的全局变量
C语言中的static函数是限定作用域的全局函数
C++中的 static 函数是相对于成员函数而言，调用主体是类

降低模块间的耦合度
静态全局变量的作用域局限于一个源文件内，可以避免在其它源文件中引起已定义错误

队列和栈的区别

先进先出　先进后出

堆和栈

堆内存：自己申请开辟的内存空间
栈内存：系统自动管理的内存空间
比如：方法块内部的变量，当方法执行结束后，栈内存自动回收

```
-(void)methods
{
    int i = 4;//栈内存
    int* ptr = &i;//栈内存
    ptr = malloc(100);//堆内存，得到的是否是连续可操作的内存？
}
```

递归 的栈内存 一直没有释放，导致性能低下

声明一个有10个指针的数组，该指针指向一个函数，该函数有一个整形参数并返回一个整型数

函数
```
int func (int)
```

函数指针
```
int （*pFunc）（int）
```

指针数组 ：一个包含10个int*型指针的指针数组
```
int *p[10];
```

数组指针 ：指向一个包含10个int型值的数组的指针
```
int （*p）[10];
```

结果：有10个指向函数的指针的指针数组
```
int （*arrFunc[10]）（int）
```

交换两个变量的值，不使用第三个变量

$$a = a + b;\ b = a - b;\ a = a - b;$$

$$a = a \wedge b;\ b = a \wedge b;\ a = a \wedge b;$$

计算sizeof的值

```
void *p = malloc(100);sizeof(p)= ?;

void Func(char str[100])

{

    sizeof(str) = ?;

}
```

扩展：结构体的内存对齐

二、objc 语法

你平时使用过哪些框架做开发？你是怎么理解UIKit的

IOS层次架构
cocoaTouch/Media/Core Services/Core OS

对象间传递消息：回调

delegate

notification

block 基本使用

performSelector

delegate、notification、block、performSelector 的合理使用

什么是函数回调？把函数实现好，等待适当的时候调用

如：实现dealloc ，系统调用该方法，实现tableView的代理方法，等待系统调用 ... ...

1对1：delegate、block

1对多：notification

代码可复用，其他地方调用：delegate、notification

代码封装，不开放：block

耦合性高：delegate、block

耦合性低：notification

效率高：delegate、block

效率低：notification

调试难度难：notification

调试难度易：delegate、block

代码可读性、后期维护 等等 各抒己见

参考Demo：TestSendMessageBetweenObject

可能考问题：

1、委托代理和通知中心的区别

2、实现函数回调有哪几种方式？

3、写一个 delegate 的声明和使用

4、使用block有什么优点

5、使用delegate 有什么优点

... ...

内存管理

引用计数

对象生命周期的标识

retainCount：alloc、new、retain ： +1，release -1；

最后一次release 触发 dealloc，对象销毁

强引用、弱引用、retain cycle

强引用：retainCount +1

弱引用：简单的指针地址的拷贝

弱引用可以避免引用循环

注意MRC环境下弱引用的使用，避免崩溃

非ARC (MRC) 使用原则

每一次对retain,alloc或者new的调用，需要对应一次release或autorealse

需要持有一个对象，那么对其发送retain

不再使用该对象，那么需要对其发送release（或者autorealse）

MRC环境下的 NSAutoReleasePool的使用

优化以下代码：

```
NSAutoreleasePool *pool = [[NSAutoreleasePool alloc] init];
    for (int i = 1;i < 10000000 ;i++) {
        NSMutableString *str = [[[NSMutableString alloc]
initWithString:@"FFFF"] autorelease];
        if(i % 1000 == 0)
        {//优化
            [pool release];
            pool = [[NSAutoreleasePool alloc] init];
        }
    }
[pool release];
```

ARC 和非ARC (MRC) ： 混合

-fno-objc-arc

-fobjc-arc

ARC 实现原理

在程序预编译阶段，将ARC 的代码转换为非ARC的代码，自动加入release、autorelease、retain

深拷贝、浅拷贝

Copy、Mutablecopy（注意不是所有对象都有Mutablecopy）

系统的非容器类 ：string、number（只有copy）、date（只有copy）、等

系统的容器类 ：array、dictionary 等（主要是复制后容器内对象的变化）

用户自定义类：copy、mutablecopy的实现，参考demo：TestCopyAndMutableCopy

继承、多态

有无多继承，多继承 的替代方案

多继承：对象多态(属性、方法)

Protocol（接口）、Category （接口；extension：匿名Category，可增加属性）

参考Demo：TestMultiInherit

protocol

protocol：一堆减方法的集合

extension

扩展：增加属性、方法

category

类别：只能增加方法；可访问私有方法，在越狱开发中常用

其他

私有方法 和 私有变量

私有变量：`@private` 来声明私有变量

私有方法：没有私有方法这个语法，在.m文件中声明来模拟私有方法

没有绝对的私有

import 和 include 的区别

使用#import可以避免重复包含头文件

objc 中的集合类

NSSet、NSMutableSet：无序、通过hush查找，效率高于遍历

KVC KVO 的理解

KVC：通过数据成员的名字来访问到它的值，它是很多技术的基础：UI Binding、KVO

KVO:监听 属性值 是否发生变动，变动的

参考Demo：KVC和KVO

文档：iPhone开发KVC_and_KVO、KVO和KVC

三、iOS api

iOS 中线程使用

创建一个线程，有多少种方式创建

层次越高的抽象程度越高，使用起来也越方便，也是苹果最推荐使用的方法

perform:selector:OnBackground:

NSThread

NSOpertion子类、NSOperationQueue

Dispatch

参考Demo：TestMultiTask

创建一个不会结束的线程

1、通过：while+sleep

2、通过：runloop+nstimer

参考Demo：TestMultiTask

子线程怎么刷新UI

回到主线程、做UI得刷新

多种方式回到主线程

perform:selector:OnMainThread:

dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0), ^{dispatch_async(dispatch_get_main_queue(), ^{ }); });

多线程对数据的写操作，最好加锁：对一个数据的操作需要经历多步完成：比如对同一个数组做　添加、删除某个对象的操作

对一个数据的操作需要经历多步完成：比如对同一个数组做　添加、删除某个对象的操作，即在某个块内的操作可能造成冲突时，应该加锁

atomic、@synchronized、NSLock、Dispatch 信号，NSCondition

视图特性

UIViewController 中的一些关键方法，谈谈你得使用心得

Init、loadView、ViewDidLoad、viewwillappear、viewdidappear、viewwilldisappear、`viewDidUnload`、`dealloc`

Load cycle 是怎样的?

viewdidload执行几次?

一般情况，ViewDidLoad只执行一次

`6.0`之前调用`viewDidUnload`,则再次调用ViewDidLoad

self.view的属性访问，进入ViewDidLoad

frame 、 bounds 、center

Frame: 以父视图的坐标原点作为参考系

bounds: 以自身坐标原点作为参考系

frame：当view做了transform的时候，该值不准确!

通过加载xib创建一个 view ，使用哪个方法；在加载完xib之后 需要设置 界面属性，需要重写view 的哪个方法?

```
 [[NSBundle mainBundle] loadNibNamed:@"QFView" owner:self options:nil]
```

-(void)`awakeFromNib`

drawRect 怎么调用

```
[viewobj setNeedsDisplay];
```

UIView 设置圆角的方法

```
viewobj.layer.cornerRadius = 5
```

UIImageView 怎么响应用户点击

```
Gesture + userInteractionEnabled
```

数据处理

字符串 "asdfd#test" 获取#号之前的字符串

```
NSString *str = @"asdfd#test";
NSRange range = [str rangeOfString:@"#"];
NSString *substr = [str substringToIndex:range.location];
```

输出一个小数，实现四舍五入，精确到小数点后1位，

```
float rvalue = roundf(fvalue);
```

需要注意：类似4.45 的四射五入取一位小数 @"%.1f"，其结果是4.4

数据存储

那些数据持久存储的方式?

归档、数据库、xml（plist、userdefaults）、

什么时候用数据库？小说怎么存？什么时候用userDefaults？什么时候用归档?

数据排序、检索、修改 、数据与数据有关联性，使用数据库

轻量级数据（用户配置，登陆数据等一些配置数据）

页面的缓存?

总之：根据开发的效率高低、使用的复杂度 来选择评估

扩展：文件系统的了解：沙箱盒的目录 ，多用户的目录操作 等

各种 持久存储 支持的类型，以及实现

NSUserDefaults支持：NSNumber（Integer、Float、Double），NSString，NSDate，NSArray，NSDictionary，NSData，BOOL类型

用户自定义数据的持久存储

转NSData,再使用

```
@protocol NSCoding
- (void)encodeWithCoder:(NSCoder *)aCoder;
- (id)initWithCoder:(NSCoder *)aDecoder;
```

UIImage 该怎么存储?

转NSData,再使用

存文件，存储文件路径

其他

写一个同步的HTTP请求，写出主要逻辑结构

NSString * str = [NSString stringWithContentsOfURL:[NSURL URLWithString:@""] encoding:NSUTF8StringEncoding error:nil];

NSData、NSDictionary，同步请求 阻塞线程

```
NSURLResponse *response ;
    [NSURLConnection sendSynchronousRequest:[[NSURLRequest alloc] initWithURL:[NSURL URLWithString:@"http://www.baidu.com"]] returningResponse:&response error:Nil];
```

ios8 新特性

healthKit、数据同步和分享、 第三方输入、iTunes的视频展示、swift

appdelegate 有哪些关键方法，分别使用在什么场景下

```
//程序加载完
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions

//远程通知注册
- (void)application:(UIApplication *)application
didRegisterForRemoteNotificationsWithDeviceToken:(NSData
*)deviceToken

//程序进入后台和回到前台
- (void)applicationDidEnterBackground:(UIApplication
*)application
- (void)applicationWillEnterForeground:(UIApplication
*)application

//程序完全退出
- (void)applicationWillTerminate:(UIApplication *)application
```

如何实现横竖屏切换

```
//6.0以及6.0以后
- (BOOL)shouldAutorotate
- (NSUInteger)supportedInterfaceOrientations

//6.0以前
- (NSUInteger)supportedInterfaceOrientations
```

ipad 开发应注意什么

```
主要是两个：UIPopOverController  UISplitViewController
```

推送的实现

```
[[UIApplication sharedApplication]
registerForRemoteNotificationTypes:UIRemoteNotificationTypeBadg
e|UIRemoteNotificationTypeAlert|UIRemoteNotificationTypeSound];
- (void)application:(UIApplication *)application
didRegisterForRemoteNotificationsWithDeviceToken:(NSData
*)deviceToken
```

四、项目相关

UDID解决办法

7.0以前 Macaddress + 6.0以后 IDFA

SVN的使用

详情参考：Versions SVN 技术专题

IOS多语言发布

应用名称的多语言：infoPlist.strings：设置 CFBundleDisplayName

应用内语言：Localizable.strings： NSLocalizedString(key, comment)

MRC 环境下使用 delegate（assign）崩溃的解决办法

将delegate置nil

- 平台无关

一、多线程和网络编程

进程和线程区别?

进程：分配资源的最小单位 线程：独立运行的最小单位

死锁的概念，如何解决?

资源的抢占，尽量避免锁的嵌套

异步下载与同步下载的优缺点，与应用场景

应用场景：数据量特别小是不是要同步下载?

代码在子线程，不需要下载进度，直接同步下载

同步下载，开发效率高，线程阻塞

异步下载，线程不阻塞，获取下载进度等需求

TCP、UDP、HTTP的概念 与应用场景

网络层：TCP\UDP 传输控制协议、用户数据报协议

应用层：HTTP 基于TCP实现 超文本传输协议

各自应用场景：文件传输、聊天、游戏、看视频

socket是什么？怎么建立一个TCP的socket链接

socket：

1、网络层通信 开发包

2、一个结构体 socket

步骤：

1、创建一个socket

2、初始化 ：确定IP、端口、协议簇

3、建立连接 -- connect 同步等待

4、建立链接之后

a、send 发送Buf 、BufSize 同步等待

b、监听返回数据 recv 同步等待

5、关闭socket，销毁

二、数据结构 和 算法

选择排序

```
-(void)bunbleSort:(NSMutableArray *)aData
{
    int count = 0;
    for(int i = 0; i < [aData count]-1;i++)
    {
        for(int j = i+1; j < [aData count];j++)
        {
            if([[aData objectAtIndex:i] integerValue] < [[aData
objectAtIndex:j]integerValue])
            {
                NSNumber *temp = [aData objectAtIndex:i];
                [aData replaceObjectAtIndex:i withObject:[aData
objectAtIndex:j]];
                [aData replaceObjectAtIndex:j withObject:temp];
                count ++;
            }
        }
    }
```

冒泡排序

```
-(void)sort2:(NSMutableArray *)resource
{
    //NSNumber  小 -> 大
    int count = [resource count];
    for(int i = 0; i < count-1; i ++)
    {
        for(int j = 0; j < count-i-1; j ++)
        {
            if([[resource objectAtIndex:j] integerValue]>
[[resource objectAtIndex:j+1] integerValue])
            {
                NSNumber *temp = [resource objectAtIndex:j];
                [resource replaceObjectAtIndex:j
withObject:[resource objectAtIndex:j+1]];
                [resource replaceObjectAtIndex:j+1
withObject:temp];
            }
        }
    }
}
```

写一个单链表，要求可以插入数据 和 删除 单个数据

```
struct QFInfo
{
    int num;
    struct QFInfo *next;
};

struct QFInfo *qfinfo;

//链表头
void insert_AtFirst(struct QFInfo *head,struct QFInfo *insert)
{
    insert->next = head->next;
    head->next = insert;
}

//链表尾
void insert_AtEnd(struct QFInfo *head,struct QFInfo *insert)
{
    struct QFInfo *temp = head->next;
    while (temp->next != NULL) {
        temp = temp->next;
    }

    insert->next = NULL;
    temp->next = insert;
}

//删除
void delete_1(struct QFInfo *head,struct QFInfo *del)
{
    struct QFInfo *temp = head->next;
    while (temp->next != NULL && temp->next != del) {
        temp = temp->next;
    }

    if(temp->next != NULL)
    {
        temp->next = temp->next->next;
    }
}
```

三、设计模式

什么是单例模式？实现一个单例模式的类

全局只有一个该类的对象

```
+(id)shareInstance
{
    static QFMutableArray *qfm = nil;

    //注意多线程调用时的枷锁
    @synchronized(qfm)
    {
        if(qfm == nil)
        {
            qfm = [[super allocWithZone:nil]init];
        }
    }
    return qfm;
}
```

注意 +(id)allocWithZone: 的使用

```
+(id)allocWithZone:(struct _NSZone *)zone
{
    return [QFSigObject shareInstance];
}

-  (id)init

{  //数据的设置 }
```

UIScrollView 用到了什么设计模式？列举一些系统中其他类似模式的类

代理模式

UITableView、UIAlertView、UIActionView、… …

说说你做过某个项目的架构设计

前后台架构、MVC架构

谈谈cocoa里面的MVC的理解

ViewController、View（代码、Xib）

C对M：API

C对V：Outlet

V对C：Target-action， Delegate，Datasource

M对C：Notification，KVO