从零开始学iOS7开发系列3-我的地盘我做主-Cha2

原文及示例代码来自raywenderlich store中的iOS Apprentice 系列3教程，经过翻译和改编。

小伙子们，看看谁来了？欢迎继续我们的学习。

很高兴你看了本系列教程的介绍仍然有兴趣继续学习下去。

在正式开启我们的地图之行前，首先要稍微回顾一下Objective-C这门编程语言。

在之前的系列教程中，我们已经接触了Objective-C语言的很多特性，但并不是全部。为了让你不至于中途退缩，我隐藏了一些技术细节，然后说了点善意的谎言来安慰你。因为前两个系列是标准入门阶段，最主要的目的是让你对iOS开发产生兴趣，所以这么做是有必要的。但现在不同了，为了真正掌握iOS开发的精髓，显然我们需要对相关理论知识加以补充。

在学习Objective-C的新知识之前，首先让我们来回顾一下以往所学的内容。

变量和类型(Variables&types)

你可以把变量看做用来存储某种特殊类型数值的临时容器。

比如：

```
int count;
BOOL shouldRemind;
NSMutableArray *list;
NSString *text;
```

变量的数据类型(datatype或者说type)决定了它里面能保存什么类型的数值。有些变量可以保存基本数据类型的数值，比如int和BOOL，而另一些变量则可以保存对象，比如NSMutalbeArray和NSString。要区分这两种类型的变量很简单，只需要看变量前面有没有*星号就知道了。很快我们将介绍基本数据类型和对象区别的更多细节。

我们目前已经接触到的基本数据类型包括：用于整数的int，用于小数（计算机术语浮点数）的float，以及用于布尔类型值的BOOL(YES和NO)。

除了以上几种数据类型，Objective-C中还提供了其它的几种基本数据类型：

1.double

double和float比较类似，都是用来保存小数（浮点数）的，只不过它的精度更高。很快我们将用到double来保存地理位置的精度和纬度信息。

2.char和unichar

这两种数据类型都是用来保存单个字符的（和字符串不同，字符串可以保存多个字符）。通常情况下char用来表示一个byte(字节）,byte是计算机存储的最小计量单位（比如你的内存通常用mega-bytes兆字节或者giga-bytes G字节来表示，我的macbook air内存 4G，硬盘只有可怜的128G，该升级了~）。
你可能听说过最近很火爆的比特币(bitcoins)，里面用到了另外一种单位bit（位）。刚才说byte是计算机存储的最小计量单位，bit要表示不满了。就好比说学过物理的人如果说质子和中子是构成物质的最小单位，夸克就会不满了。

在计算机的二进制世界中，所有都是由0和1组成的，每个0或1都是1个位（bit)。通常来说（注意是一般情况下），一个byte由8个bit组成。为什么1个byte由8个bit组成？而不是9个或10个？好吧，这是因为TCP/IP协议里面就是这样约定的。至于TCP/IP协议是什么东东，后面有空可以聊聊。
回到刚才的说法，大家公认byte是计算机存储的最小计量单位，虽然bit表示强烈不满，但既定事实就是这样的。原因在于，目前bit的主要用途不是衡量计算机存储单元的容量大小，而是用来衡量网络通讯或者数据传输的速率。比如大家都熟知的bps就有bps和Bps的不同。bps 是bits per second（多少位/秒），而Bps是（bytes per second)。
比如USB 2.0 标准接口传输速率为“480Mbps”，你可能会误解为 480 兆字节/秒。事实上，“480Mbps”应为“480 兆比特/秒”或“480 兆位/秒”，它等于“60 兆字节/秒”，即传输速度为“60 兆/秒”（每秒传输速度为 60MB）。也就是说，“480Mbps”中的“b”，指的是 bit。byte 与 bit 两者换算是 1：8 的关系。所以，480Mbps 的传输速度等于 60MB/秒。一般的网络传输速率也是用bit衡量。
所以下次去电信宽带咨询的时候，别忘了这种单位换算。刚才突然觉得网络有点慢，于是去网上的自助测速平台上检测了下，心里顿时哇凉哇凉的。
顺便问一下，移动iPhone5S的4G LTE下载速度到底是多少啊？最近有意入手，求大家帮忙算算真实网速。

还记得黑客帝国(Matrix)里面的The One Neo吗。知道你精通计算机语言，就可以在虚拟世界中成为新的主宰~

3.short,long,long long

别想歪了，这些都是int（整数）类型变量的变种。它们的区别在于可以存储数值的字节数（也就是容量）。如果某个类型可以存储更多的字节数，显然就能够保存更大的数值。通常我们默认会使用int，它可以保存4个字节的数值（好吧，很快你就会忘掉这个事实，不过也不会有太大影响）。这种说法太抽象了，实际上你只需要知道它可以保存从负的20亿到正的20亿之间的数值就好了。
至于几种变种可以保存的数值范围是神马？难道你不会用谷哥、度娘或者维基吗？

4.unsigned

这是int的另一个变种，unsigned意味着它只能保存非负数（0或者正数），术语党喜欢叫无符号数，是一回事。在Objective-C中，一个unsigned int变量可以保存从0到正40亿之间的数值，但不能保存负数，只需要知道这一点就够了。

在iOS SDK还会碰到以下几个基本数据类型的同义词：

1.NSInteger
和int是一回事，你就把它当int看。通常我们更喜欢用NSInteger，这样就无需考虑你用的硬件设备是32位还是64位的。

2.NSUInteger
和unsigned int是一回事

3.CGFloat
和float是一回事,它只是对float或double的typedef定义，在苹果官方文档中的定义如下：

typedef float CGFloat;// 32-bit
typedef double CGFloat;// 64-bit
也就是说在64位的设备上CGFloat实际上就是double 类型，而在32位的设备上就是float。
还是那句话，嫌麻烦的话用CGFloat就好了，苹果会帮你自动解决类型定义问题。

所以，虽然很多NS开头的变量类型属于类对象，但NSInteger和NSUInteger其实是基本数据类型，千万别看到NS开头的东东就以为一定是一个对象。

当你看到这些类型名称时可能会觉得有点害怕，Don’t panic，编译器会轻松的在这些类型间进行转换。比如当你在代码中使用int的时候，SDK通常会自动把它看做一个NSInteger。

注意这里用的是通常，因为在32位设备上int和NSInteger就是完全的同义词。但是自从iPhone 5S推出以来，首次在移动设备上采用了64位芯片。此时int仍然是32位，而NSInteger则成了64位。为了避免麻烦，我们还是统一用NSInteger好了。

当然，如果你在编写代码的时候如果类型不符，Xcode会用Warning给你温馨提示的。
记得之前的教程中强调的事情吗？千万别忽视任何的Warning!虽然它们只是黄牌警告，但很有可能在关键时刻升级进化成红牌，把你拖到大坑里面被千万人唾骂（比如著名的全民产品12306!)

对于基本数据类型变量，我们可以直接使用。
比如：

```
count = 10;
shouldRemind = YES;
```

而对于对象，则必须首先创建并初始化：

```
list = [[NSMutableArray alloc] initWithCapacity:10];
```

某些特殊的对象可以直接使用，比如字符串：

```
NSString *text = @"Hello, world";
```

整数10，布尔变量YES和字符串@“Hello,world”又被术语党称为字面值常量(literal constants)，或者叫字面值(literals)。程序猿的思维很变态，所以看到这种千奇百怪的术语还是从了吧。

现在我们已经接触了两种类型的变量：
本地变量(local variables)-
它们只活在定义它们的方法中，如冬虫夏草，不知寒暑岁月变迁。
实例变量（instance variables,或者变态的缩写ivars）-
它们可以用在整个对象的世界中，然后被该世界中的任何方法访问。

对于变量的生命周期，术语党给了一个名词叫scope（砖家们将其取名为作用域，建议大家还不如记住scope这个说法更直观，还是把英语学好吧，我表示看到这些翻译过来的术语很头大，理解不能）一个本地变量的作用域要小于实例变量的作用域，因为它离了创建自己的对象就根本活不了，是百分百的寄生生命，就好比我朝的猿类，离开政府机构就没法生存了。）

```
@implementation MyObject {

int _count;
- (void)myMethod

} {

// an instance variable

// a local variable
temp = _count; // OK to use the instance variable here

int temp;

}
```

个人建议在.m的@implementation语句后声明实例变量。当然，你可能看到不少项目代码把实例变量的声明放到.h文件的@interface语句后。那是之前的惯用方法，目前仍然有效，但并非最佳实践。如果我们有一个本地变量的名称和实例变量名称相同，Xcode就会发出警告，说它将要shaow（或hide-隐藏）那个同名的实例变量。我们应该尽可能避免出现这种情况，否则你回碰到一些莫名其妙的bug，为此通宵熬夜而错失女神的邀约。

```
@implementation MyObject {
int count; // an instance variable

}

- (void)myMethod {

int count = 0; // local variable shadows instance variable }
```

建议可以在实例变量名称的前面加个下划线，比如_count，这样一看到前面有下划线的变量就知道是实例变量了。
在任何时刻，一个变量都只能保存一个单一数值。好吧，这一点并非永恒不变的宇宙真理。就我所知，等今后量子计算机大行其道的时候这句话肯定就变成老古董的固执言论了。但至少目前来说还真是这样。为了保存多个对象，我们需要用到collection（集合）类对象。说到集合类对象，你可能就想到了上一系列教程中用到的NSArray数组和NSDictionary词典。
数组可以用来保存一系列的对象，这些对象的保存是按照某种特定顺序的，因此我们可以用编号来获取其中的任何一个对象。
词典用来保存键对值。在键对值中，一个对象（通常是字符串）是用来获取另一个对象的key（钥匙，键）。
在iOS中还有其它类型的集合对象，不过数组和词典是最常用的。

顺便提一下，我们都知道Objective-C是从C语言演化而来的，而在C语言中有自己内置的数组类型，不过在iOS 开发中我们几乎没有机会用到。为了让你以后看到了不陌生，可以稍微展示 下：

```
int oldSchoolArray[10];

oldSchoolArray[0] = 100;
 oldSchoolArray[1] = 200;
```

C语言原生数组的问题在于灵活度太差。如果我们声明了一个C语言的数组，其中可以保存10个项目，那么这个数组也就固定只能保存10个项目了，当我们需要添加更多项目时，数组并不会自动扩展。虽然在某些情况下可能会用到C样式的数组，不过只有那些硬（过）核（时）程序猿才会在iOS开发中这样用。
关于变量就先了解这些知识吧，希望这些理论知识不至于太枯燥，因为这个系列中用花比较多的篇幅来帮你打好基础~
今天就到这里吧，送上今日福利（2014年1月18日）不知道春节前还有没有时间写，先送个红包表表心意吧。