Object-C

2016年5月12日 星期四　　上午8:17

Objective C

面向过程：数据结构+算法

面向对象：属性+行为

OC没有重载，可以重写

1、类只能在堆中定义变量，不能在栈区，struct可以在栈区

2、编译：clang -framework Foundation main.m -> a.out

3、访问权限：**private、protected、package、public**

@interface中默认protected,@implementation中一定是private

4、**@property int n;**(ios6) ->**self.n;**//用self访问成员变量

@synthesize n = _n;(ios4) 自动生成setter,getter

@synthesize n;(ios5)

5、*.m 中定义的方法是private私有方法，用self访问([self chkValid:x])

6、-(instancetype)init 无参初始化，NSObject方法

```
-(id)initWithX:(int)x andY:(int)y
```

instancetype：一般做工厂、初始化方法返回值类型，不能作为参数

id类型被定义为指向NSObject子类的实例对象的指针,OC的万能指针,eg:id s=nil;

[super init]：alloc->self(子类)->NSObject.init(self隐式参数,清零)->self(返回self)

7、**self** 指向当前对象的指针(引用) 只能在类中方法函数体中使用，谁调用就指谁

注意：**self在类方法中不是指对象，而是指当前类本身**

8、**super** 表示父类，指向对象中继承过来的父类部分的指针，调用父类的方法

9、类方法&实例方法

类方法：静态方法，由类名直接调用，用于创建各种对象　eg:+(id)point;

实例方法：对象方法，定义在对象之后由对象调用，用于类的某种行为　eg:-(void)show;

10、工厂方法

```
+(id)point;//工厂方法的方法名一定以类名开头，首字母小写
+(id)pointWithX:(int)x andY:(int)y;
```

11、单例模式：只生成唯一对象，用于应用程序的资源共享控制　Singleton

12、组合：两对象之间是整体和部分的强关系”contains-a”,部分的生命周期不能超越整

```
/**
 Window被创建时调用init方法，此时创建并初始化Button和Edit对象，
 所以同时被创建，@autoreleasepool结束后是释放w指针，alloc创建的
 window堆上的空间被释放(系统自动调用dealloc)，此空间上又有button
 和edit的成员变量，也被销毁，则所指向的对象引用计数为0，故被销毁
 */
Window *w = [[Window alloc]init];
w = nil;//不用等自动释放，会导致对象提前被释放
/**
 指针变量w和a都在栈中，则遵循先进后出原则，故组合Animal后被创建而先被销毁
 */
Animal *a = [[Animal alloc]init];
```

聚合：两对象之间是整体和部分的弱关系”has　a”,部分生命周期可以超越整体

注意：组合用成员变量(没有setter,getter)，聚合用@property

13、封装、继承派生、(参数、数组、返回值)多态

14、自动释放池(在工厂方法中加入此关键字，将工厂对象交由自动释放池管理)

当自动释放池的作用域结束时，池中所有对象自动被释放

MRC：autorelease

ARC：__autoreleasing　直到@autoreleasepool才释放->嵌套使用

15、协议(Protocol)，可以多继承，可以多重采纳　类单继承多协议

@required(默认) @synthesize a = _a;

@optional:声明可选遵守的属性的方法

16、分类：类的补充和扩展　分解大的代码降低耦合度，给已有的类添加新方法

*.h @interface 主类类名(分类类名)　　*.m @implemention 主类类名(分类类名)

1、分类中不能创建实例变量和属性，可以访问主类属性，但不能访问主类实例变量

2、可以给系统类和没有源代码的类添加分类 NSObject,NSString...

3、分类会覆盖主类中的同名方法

17、扩展：是没有名字的分类，用来声明私有的属性和方法

1、*.m文件中中声明的实例变量、方法、属性都是私有的，只能在.m文件中使用

2、扩展没有.m文件，方法必须在主类的.m文件中实现

3、分类&扩展：是否可声明成员变量和属性

4、协议&扩展：扩展只依附于主类，协议可同时被多个类采纳

18、__strong(默认)：根据引用和作用域范围自动隐式retain/release

__weak：当指向的对象被销毁时，若引用本身会自动赋值为空(nil)

__autoreleasing：在ARC下将对象交由自动释放池管理

__unsafe__unretain：等同于assign,__weak,没有自动清零功能

```
__weak TRPoint *p = [[TRPoint alloc]initWithX:1 andY:9];
NSLog(@"%p", p);//不能给弱指针分配空间，堆上的空间赋值给弱指针后会马上释
```

内存管理

标记清除：当变量进入环境，就标记为“进入环境”，永远不能释放进入环境的变量所占用的内存

垃圾收集器会给存储在内存中的所有变量都加上标记，然后会去掉环境中的变量以及被环境中

变量引用的变量的标记，在此之后再被加上标记的变量将被视为准备删除的变量。

引用计数：(alloc、copy、new)跟踪记录每个值被引用的次数，当声明了一个变量并将一个引用

赋给该变量则引用次数为1，如果同一个值又被赋给另一个变量，则+1；如果包含对这个值引用

变量又取得了另外一个值，则引用计数-1。

(ARC中存在，MRC中不存在)缺点：循环引用意味着永远不能被回收，用完后要p=nil,断开连

```
Point1 *p = [[Point1 alloc]init];//引用计数-1
```

存。

用类型值

的

连接

```
Point1 *p = [[Point1 alloc]init];//引用计数=1
[p release];//调用dealloc释放
//私有堆->共有堆(指针p和堆对象p内容都没有改变)
NSLog(@"引用计数=%ld", [p retainCount]);//1,此处是垃圾值
p = nil;//为防止使用垃圾值，将p☞空,引用计数=0
p = [[Point1 alloc]init];//count=1
Point1 *p1 = p;
[p1 retain];//引用计数+1,count=2
[p1 release];//引用计数-1!=0,不会调用dealloc而直接返回
NSLog(@"%ld", [p1 retainCount]);
```

声明式属性

```
@property(retain) Point1 *center;
```

retain(引用类型)、assign(基本类型)、copy(深拷贝)、readonly、atomic、nonatomic

MRC setter

```
-set:(NSString  s){
if(s == self.s) return;
[self.s release];
self.s = s;
[self.s retain];
}
```

重写setter方法，当属性值发生变化做相应的Action，可以实现Observer观察者模式

OC：动态类型(id泛型)、动态绑定(SEL)、动态加载(Runtime)

iOS7～9API

Xcode6、7版本

OC语言基础-1

OC语言基础-2

OC语言基础-3

OC语言基础-4

OC语言基础-5

OC语言基础-6

常用网站：
http://www.code4app.com/