Swift 构造过程 Swift 自动引用计数（ARC）

Swift 析构过程

在一个类的实例被释放之前，析构函数被立即调用。用关键字`deinit`来标示析构函数，类似于初始化函数用`init`来标示。析构函数只适用于类类型。

析构过程原理

Swift 会自动释放不再需要的实例以释放资源。

Swift 通过自动引用计数（ARC）处理实例的内存管理。

通常当你的实例被释放时不需要手动地去清理。但是，当使用自己的资源时，你可能需要进行一些额外的清理。

例如，如果创建了一个自定义的类来打开一个文件，并写入一些数据，你可能需要在类实例被释放之前关闭该文件。

语法

在类的定义中，每个类最多只能有一个析构函数。析构函数不带任何参数，在写法上不带括号：

```
deinit {
    // 执行析构过程
}
```

实例

```
var counter = 0;  // 引用计数器
class BaseClass {
    init() {
        counter += 1;
    }
    deinit {
        counter -= 1;
    }
}

var show: BaseClass? = BaseClass()
print(counter)
show = nil
print(counter)
```

以上程序执行输出结果为：

```
1
0
```

当 show = nil 语句执行后，计算器减去 1，show 占用的内存就会释放。

```
var counter = 0;  // 引用计数器

class BaseClass {
    init() {
        counter += 1;
    }

    deinit {
        counter -= 1;
    }
}

var show: BaseClass? = BaseClass()

print(counter)
print(counter)
```

以上程序执行输出结果为：

```
1
1
```

Swift 构造过程 Swift 自动引用计数（ARC）