**闭包**

闭包是自包含的函数代码块，可以在代码中被传递和使用。闭包可以捕获和存储其所在上下文中任意常量和变量的引用。 这就是所谓的闭合并包裹着这些常量和变量，俗称闭包。Swift 会为您管理在捕获过程中涉及到的所有内存操作。在[函数](http://www.yiibai.com/swift/functions.html" \t "_blank) 章节中介绍的全局和嵌套函数实际上也是特殊的闭包，闭包采取如下三种形式之一：

* 全局函数是一个有名字但不会捕获任何值的闭包
* 嵌套函数是一个有名字并可以捕获其封闭函数域内值的闭包
* 闭包表达式是一个利用轻量级语法所写的可以捕获其上下文中变量或常量值的匿名闭包

Swift 的闭包表达式拥有简洁的风格，并鼓励在常见场景中进行语法优化，主要优化如下：

* 利用上下文推断参数和返回值类型
* 隐式返回单表达式闭包，即单表达式闭包可以省略return关键字
* 参数名称缩写
* 尾随(Trailing)闭包语法

**闭包表达式**

闭包表达式是一种利用简洁语法构建内联闭包的方式。闭包表达式的语法形式如下：

{(参数列表) -> 返回值类型 in

语句

}

其中，参数可以是常量、变量和输入-输出参数，但没有默认值。开发者也可以在参数列表最后使用可变参数。远足也可以作为参数和返回值。关键字in表示必报的参数和返回值类型已经完成，闭包函数体即将开始。闭包表达式和函数一样，可以分为无参闭包表达式和有参闭包表达式。

具有参数的闭包表达式语法形式如下：

{(参数名1：数据结构，参数名2：数据结构，参数名3：数据结构，…) -> 返回值类型 in

语句

}

定义的语法形式如下：

let/var 闭包表达式常量名称/闭包表达式常量名称=具有参数的闭包表达式

调用形式如下:

闭包表达式常量名称/闭包表达式变量名称（参数值1，参数值2...）

无参闭包表达式的应用与具有参数的闭包表达式差不多，只是没有参数。

示例 以下实现两个整数的相除和输出字符串“Swift”

let divide = {(val1: Int, val2: Int) -> Int in   //有参有返回值闭包

    return val1 / val2

}

let result = divide(200, 20)

print (result)

let printstr={() in         //无参无返回值闭包

    print("Swift")

}

printstr()

此时运行程序会出现如下结果：

**10**

**Swift**

闭包表达式最常用于其他函数中，并不单独使用

示例 以下将闭包表达式作为函数的一部分，实现判断在数组中是否有大于500的值。代码如下：

func compare(arr:[Int],\_ value:Int,\_ cb:(\_ num:Int,\_ val:Int)->Bool)->Bool{   //闭包

    for item in arr{

        if(cb(item,value)){

            return true

        }

    }

    return false

}

let ray=[20,80,60,100]

var vl=compare(arr: ray, 500, {(num,value) in  num>value})

print(vl)

此时运行程序会出现如下结果：

**false**

**(1)推断类型和**省略return

Compare（）函数的第三个参数是闭包表达式，它的类型为（） 。由于swift可以推断其参数和返回值的类型，所以->和数值类型以及返回值类型都可以省略，单行表达式闭包可以通过隐藏return关键字来隐式返回单行表达式的结果，如以下代码所示：

var vl=compare(arr: ray, 500, {(num,value) in  num>value})

（2）简写参数名与用运算符函数

Swift为内联函数提供了参数名缩写功能，开发者可以通过$0,$1,$2来顺序的调用闭包的参数。

var vl=compare(arr: ray, 500, {$0>$1})

也可以省略参数，运用运算符函数大于号（>）来实现

var vl=compare(arr: ray, 500, >)

## 尾随闭包

尾随闭包是一个书写在函数括号之后的闭包表达式，函数支持将其作为最后一个参数调用。

func someFunctionThatTakesAClosure(closure: () -> Void) {

    // 函数体部分

}

// 以下是不使用尾随闭包进行函数调用

someFunctionThatTakesAClosure({

    // 闭包主体部分

})

// 以下是使用尾随闭包进行函数调用

someFunctionThatTakesAClosure() {

    // 闭包主体部分

}