## 第四章 元组和集合类型

### 4.1 元组

元组（tuple）把多个值组合成一个复合值。元组内的值可以使任意类型，并不要求是相同类型。如果定义一个学生包含2个属性姓名，年龄，可以用元组表示为：

let student = ("Swift", 19)

//student 的类型是 (String, Int)，值是 ("Swift", 19)

示例中，我们定义了一个(String, Int)元组，String表示姓名，Int表示年龄，所以这个元组可以被描述为“一个类型为(String, Int)的元组”。元组中的数据类型可以是任意类型，根据你的需求，你可以将元组定义成(Int, Int, Int)或是(String, Bool)或是其他任何你想要的组合。

你可以将元组的内容分解(decompose)成单独的变量和常量，然后正常使用它们了。

let (name, age) = student

print("name is \(name)")

//打印 “name is Swift”

print("age is \(age)")

//打印 “age is 19”

如果你只需要元组的一部分值，分解的时候可以用\_代替哟啊忽略的部分。

let (onlyName, \_) = student

print("name is \(onlyName)")

//打印 “name is Swift”

此外，你还可以通过下标直接访问元组的单个元素，下标从零开始。

print("name is \(student.0)")

//打印 “name is Swift”

print("age is \(student.1)")

//打印 “age is 19”

如果，在元组定义的时候给单个元素命名，之后调用的时候，就可以用名字来调用这些元素的值。

//定义时给元素命名

let otherStudent = (name:"iOS", age:18)

//直接用.名称调用

print("name is \(otherStudent.name)")

//打印 “name is iOS”

print("age is \(otherStudent.age)")

//打印 “age is 18”

作为函数返回值时，元组非常有用。一个用来获取网页的函数可能会返回一个(Int, String)元组来描述是否获取成功。和只能返回一个类型的值比较起来，一个包含两个不同类型值的元组可以让函数的返回信息更有用。

但是，提醒一下各位读者，虽然元组在临时组织值的时候很有用，但是并不适合创建复杂的数据结构。如果你的数据结构并不是临时使用，请使用类或者结构体而不是元组。

### 4.2 集合类型和可变性

Swift 语言提供Arrays、Sets和Dictionaries三种基本的集合类型用来存储集合数据。数组（Arrays）是有序数据的集。集合（Sets）是无序无重复数据的集。字典（Dictionaries）是无序的键值对的集。

Swift 语言中的Arrays、Sets和Dictionaries中存储的数据值类型必须明确。这意味着我们不能把不正确的数据类型插入其中。同时这也说明我们完全可以对取回值的类型非常自信。

如果创建一个Arrays、Sets或Dictionaries并且把它分配成一个变量，这个集合将会是可变的。这意味着我们可以在创建之后添加更多或移除已存在的数据项，或者改变集合中的数据项。如果我们把Arrays、Sets或Dictionaries分配成常量，那么它就是不可变的，它的大小和内容都不能被改变。在我们不需要改变集合的时候创建不可变集合是很好的实践。如此 Swift 编译器可以优化我们创建的集合。

### 4.3 数组(Arrays)

**数组的创建**

数组使用有序列表存储同一类型的多个值。相同的值可以多次出现在一个数组的不同位置中。Swift 数组会强制检测元素的类型，如果类型不同则会报错，Swift 数组应该遵循像Array<Element>这样的形式，其中Element是这个数组中唯一允许存在的数据类型。我们也可以使用像[Element]这样的简单语法。尽管两种形式在功能上是一样的，但是推荐较短的那种。

//创建一个空数组

var someInts = [Int]()

print("数组中有\(someInts.count)个元素")

//打印 "数组中有0个元素"

Swift 中的Array类型还提供**一**个可以创建特定大小并且所有数据都被默认的构造方法。我们可以把准备加入新数组的数据项数量（count）和适当类型的初始值（repeating）传入数组构造函数

//创建一个[Double]数组，等价于[0.0, 0.0, 0.0]

var threeDoubles = Array(repeating: 0.0, count: 3)

我们还可以使用数组字面量来进行数组构造，这是一种用一个或者多个数值构造数组的简单方法。数组字面量是一系列由逗号分割并由方括号包含的数值：[value 1, value 2, value 3]

//利用Swift类型推导机制，定义了包含3个元素的[String]数组

var shoppingList = ["Eggs", "Milk", "Bread"]

**数组的遍历**

我们可以使用上一章介绍的for-in循环来遍历数组中的数据项：

for item in shoppingList{

    print(item)

}

//打印 Eggs Milk Bread

如果我们同时需要每个数据项的值和索引值，可以使用enumerated()方法来进行数组遍历。enumerated()返回一个由每一**个数据项**索引值和数据值组成的元组。我们可以把这个元组分解成临时常量或者变量来进行遍历：

//利用元组访问

for (index, item) in shoppingList.enumerated() {

    print("item\(index) \(item)")

}

/\*

 打印：

 item0 Eggs item1 Milk item2 Bread

\*/

**数组的修改和访问**

我们可以使用append()或者赋值运算符+=在数组的末尾添加元素。

//为shoppingList添加元素"Tea"

shoppingList.append("Tea")

//+=将["Flowers","Orange"]添加到shoppingList末端

shoppingList += ["Flowers","Orange"]

现在shoppingList中就有6个元素了。分别是"Eggs","Milk","Bread"

,"Tea","Flowers","Orange"。我们还可以通过索引去访问和修改它们。

print(shoppingList[0])

//打印 Eggs

//把第0个元素修改为apples

shoppingList[0] = "apples"

print(shoppingList[0])

//打印 apples