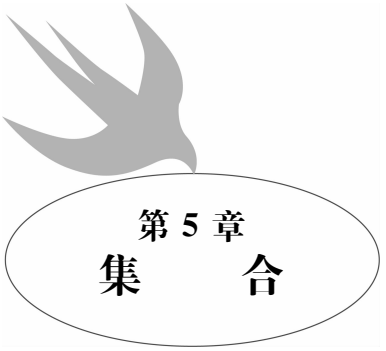

第5章 集　合

集合就是用来存储一组数据的容器。Swift 中有 3 种常用的集合类型：数组、集合和字典。

数组是一种按顺序存储相同类型数据的集合，相同的值可以在数组中的不同位置重复出现。集合和字典类型也是存储了相同类型数据的集合，但是数据之间是无序的。集合不允许值重复出现。字典中的值可以重复出现，但是每个值都有唯一的键值与其对应。本章将对这 3 种非常重要的集合类型做详细介绍。

5.1 数组

当要按照一定的次序存储一组数据，并通过数据所在位置的序号检索时，就要采用数组实现。数组是类型安全的，因此数组中包含的数据类型必须是显式声明的。

数组的声明格式为 Array<DataType>或者 [DataType]。后一种声明方式比较简洁，用得较多。

5.1.1 数组的初始化

数组有多种初始化方法，这里介绍 3 种。

第一种初始化方法是先将数组声明为一个空数组，然后再逐步往数组中增加元素。如图 5.1 所示，首先定义一个字符串型的空数组变量 animalArray，然后调用数组类型的方法 isEmpty()判断数组是否为空。如果为空，则开始以此向数组中添加元素。

添加元素用到数组类型的方法 append()。append()方法用来向数组的末端添加一个元素。需要注意的是,添加的元素必须和数组声明的数据类型一致,否则会导致报错。另外,append()方法每次都是向数组的末端添加元素,因此数组中的元素是按照添加的次序顺序排列的。

```
var animalArray = [String]()
if animalArray.isEmpty {
    print("animalArray is empty!")
}
animalArray.append("tiger")
animalArray.append("lion")
```

```
[]

"animalArray is empty!\n"

["tiger"]
["tiger", "lion"]
```

图 5.1　isEmpty()方法和 append()方法

第二种数组初始化的方法是在声明数组的时候直接赋值为一个实际的数组。图 5.2定义了一个整型数组变量 oneBitNumberArray,并在声明语句中直接将一个含有 10 个整型数的数组赋值给它。另外,还用同样的方式定义了两个字符串型数组 botanyArray1 和 botanyArray2。最后,调用数组类型的 count()方法统计并打印数组 botanyArray1 的长度(即数组所含元素的个数)。

```
var oneBitNumberArray : Array<Int> =
    [0,1,2,3,4,5,6,7,8,9]
var botanyArray1 : [String] =
    ["rosemary","parsley","sage","thyme"]
var botanyArray2 =
    ["rosemary","parsley","sage","thyme"]
print("There are \(botanyArray1.count) kinds
    of botany.")
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

["rosemary", "parsley", "sage", "thyme"]

["rosemary", "parsley", "sage", "thyme"]

"There are 4 kinds of botany.\n"
```

图 5.2　数组初始化

第 3 种数组初始化的方法叫作快捷初始化方法,即一次性将数组的所有元素初始化为同一个值。图 5.3 定义了两个整型数组变量 twoBitNumberArray 和 threeBitNumberArray,在声明语句中同时定义了数组的长度(count)分别为 6 和 3,以及每个元素的初值(repeating)分别为 0 和 11。

```
var twoBitNumberArray = [Int](repeating: 0,
    count: 6)
var threeBitNumberArray = [Int](repeating:
    11, count: 3)
```

```
[0, 0, 0, 0, 0, 0]

[11, 11, 11]
```

图 5.3　数组的相加

5.1.2 数组的相加和累加运算

数组的相加和累加运算分别使用运算符“＋”和“＋＝”，从而由已知数组快速得到新数组。图 5.4 定义了一个数组变量 theAddedNumberArray，该数组由两个整型数组 twoBitNumberArray 和 threeBitNumberArray 相加而得。字符串数组 animalArray 中含有两个元素，然后通过两次累加分别将含有一个元素和两个元素的数组添加到 animalArray 数组中。这里需要注意的是，相加或累加的数组必须是相同数据类型的。

```
var theAddedNumberArray = twoBitNumberArray +
    threeBitNumberArray
                                                  [0, 0, 0, 0, 0, 0, 11, 11, 11]

animalArray += ["hawk"]                           ["tiger", "lion", "hawk"]
animalArray += ["sheep","horse"]                  ["tiger", "lion", "hawk", "sheep", "horse"]
```

图 5.4 数组的累加

5.1.3 数组的下标操作

数组中的元素是有序排列的，通过下标可以找到特定位置的元素，获取该元素的值或者对其进行修改。注意，数组中第一个元素的下标序号是 0，而不是 1。如图 5.5 所示，要取出字符串数组 animalArray 中第一个元素的值，只通过 animalArray[0]即可。要修改数组中第二个元素的值，直接对 animalArray[1]进行赋值即可。另外，还可以通过下标批量修改数组元素的值。

```
var theFirstAnimal = animalArray[0]
animalArray[1] = "hen"
animalArray[2...4] =
    ["goat","rat","cock","rabbit"]
print(animalArray)
```

```
"tiger"
"hen"
["goat", "rat", "cock", "rabbit"]

"["tiger", "hen", "goat", "rat", "cock", "rabbit"]\n"
```

图 5.5 数组的下标操作

5.1.4 插入与删除元素

数组的插入和删除操作分别用 insert()和 remove()方法。如图 5.6 所示，向数组 animalArray 中下标为 3 的位置插入一个新元素字符串"snake"。如果删除数组中

第一个元素"tiger"或者最后一个元素"rabbit"，可以直接调用方法 removeFirst()和 removeLast()。如果要删除特定位置的元素，可以调用方法 remove(at: Int)，并给出该位置的具体参数。

```
print(animalArray)
animalArray.insert("snake", at: 3)

animalArray.removeFirst()
print(animalArray)

animalArray.removeLast()
print(animalArray)

animalArray.remove(at: 2)
print(animalArray)
```

```
"["tiger", "hen", "goat", "rat", "cock", "rabbit"]\n"
["tiger", "hen", "goat", "snake", "rat", "cock", "rabbit"]

"tiger"
"["hen", "goat", "snake", "rat", "cock", "rabbit"]\n"

"rabbit"
"["hen", "goat", "snake", "rat", "cock"]\n"

"snake"
"["hen", "goat", "rat", "cock"]\n"
```

图 5.6　数组的插入和删除

5.1.5　数组的遍历

数组的遍历一般采用 for-in 语句。如图 5.7 所示，第一个 for-in 循环对数组 animalArray 中的元素进行遍历并打印。第二个 for-in 循环以元组的方式对数组中的下标和相应元素遍历并打印。

```
for animal in animalArray {
    print(animal)
}

for (index,animal) in
    animalArray.enumerated() {
    print("No.\(index) animal is \(animal)")
}
```

```
(4 times)

(4 times)
```

图 5.7　数组的遍历

5.1.6　数组片段

通过数组的下标操作可以获取数组中的某一个元素，如果需要获取数组的一部分或者多个元素，就要用到获取子数组的方法。

获取数组 Array 的片段 arraySlice 的格式为

```
arraySlice=Array[startIndex...endIndex]
```

其中 startIndex 为子数组起始元素的索引，endIndex 为结束元素的索引。注意，startIndex 的最小值为 0，即 Array 的第一个元素的索引，endIndex 的最大值为 Array.count－1，即 Array 的最后一个元素的索引。

数组片段是由原数组中部分连续的元素组成的子集。数组片段是原数组的一部分，共用相同的存储空间。换言之，如果数组片段中元素的值发生变化，那么原数组中相应元素的值也会同步改变。

如图 5.8 所示，数组 animalArray 中含有 4 个元素。常量 myZoo 赋值为数组 animalArray 的一个片段，包含下标 1～2 的两个元素。但是，myZoo 不是一个新的数组，myZoo[0]元素并不存在。myZoo 只是原数组的一部分，它的类型是 ArraySlice。myZoo 中元素的下标和数组 animalArray 中相应元素的下标一致。

```
print(animalArray)

let myZoo = animalArray[1...2]
print(myZoo)
myZoo[1]
myZoo[2]
//myZoo[0]
```

```
"["hen", "goat", "rat", "cock"]\n"

["goat", "rat"]
"["goat", "rat"]\n"
"goat"
"rat"
```

图 5.8　数组片段

如果要根据数组片段生成一个新的数组，可以将数组片段类型转换为数组。如图 5.9 所示，将数组片段 animalArray[1...2]类型转换为数组后，数组 newZoo 中的元素就从下标 0 开始排列了。

```
let newZoo = Array(animalArray[1...2])
print(newZoo)
newZoo[0]
newZoo[1]
```

```
["goat", "rat"]
"["goat", "rat"]\n"
"goat"
"rat"
```

图 5.9　由数组片段生成新数组

5.1.7　元素交换位置

在数组中，如果要交换两个元素的位置，可以使用方法 swapAt(_:_:)，参数表中提供需要的元素位置。如图 5.10 所示，对数组 animalArray 中下标为 2 和 3 的元素交换位置。

```
print(animalArray)
animalArray.swapAt(2, 3)
```

```
"["hen", "goat", "rat", "cock"]\n"
["hen", "goat", "cock", "rat"]
```

图 5.10　数组元素交换位置

5.1.8　数组排序

如果要对数组中的所有元素进行排序，可以使用方法 sort()。如图 5.11 所示，对数组 animalArray 中的所有元素按照字母先后顺序排序。

```
print(animalArray)
animalArray.sort()
```

```
"["hen", "goat", "cock", "rat"]\n"
["cock", "goat", "hen", "rat"]
```

图 5.11　数组元素排序

5.1.9　检索特定元素

如果要检索数组或数组片段中是否含有特定的元素，可以使用方法 contains(_:)。该方法根据检索结果返回布尔值。图 5.12 分别检索了数组片段 animalArray[1...3] 和数组 animalArray 中是否含有字符串"hen"，检索结果为 true 或 false。

```
animalArray[1...3].contains("hen")
animalArray.contains("hen")
```

```
false
true
```

图 5.12　检索数组中是否含有特定元素

练习题

1. 定义 3 个字符串数组，分别用来表示已经选修的专业必修课、专业选修课以及通识课。分别用 3 种方法对这 3 个数组进行初始化，其中专业必修课包含数据结构、计算机组成、计算机网络；专业选修课包含 iOS 应用开发实践、Swift 程序设计、人工智能；通识课包含音乐、美术、文学。

2. 定义一个字符串数组，用来表示所有的选修课程，由专业必修课、专业选修课和通识课组成。

3. 在所有选修课程数组中，将专业必修课“数据结构”修改为“离散数学”，增加一门新的专业选修课“移动应用开发实践”，删除通识课“美术”。

4. 将所有选修课程数组中的元素及其下标依次打印输出。

5. 在第4题所有选修课程数组中，生成下标3～6元素的数组片段，由该数组片段生成一个新的数组。该数组片段和这个新数组是否完全相同？请通过代码说明两者的相同和不同之处。

6. 对第5题中新数组的第一个元素和最后一个元素交换位置，并打印输出。

7. 对新数组中的所有元素进行全排序，并打印输出。

8. 检索新数组中是否含有字符串"Swift"和"Swift Programming"，并打印输出检索结果。

5.2 集合

集合是由一组相同数据类型的元素组成的，每个元素的值都是唯一的。集合中的元素是无序的。

集合类型的声明格式为Set<DataType>。

5.2.1 集合的初始化

集合类型的初始化既可以从创建一个空的集合开始，也可以通过直接赋值的方法。第一种方法需要显式地指出集合中的元素类型。第二种方法可以通过被赋的初始值推断集合中的元素类型。如图5.13所示，通过两种初始化方法定义了集合类型变量weatherOfSanya和weatherOfBj。

```
var weatherOfSanya = Set<String>()
weatherOfSanya = ["rainy","sunny","stormy"]

var weatherOfBj : Set = ["dry","windy","frogy"]
```

图5.13 集合的初始化

5.2.2 集合的为空判断和插入元素

集合类型和数组类型一样，也提供了isEmpty()和insert()方法，可以便捷地操作集合。如图5.14所示，通过方法isEmpty()判断集合weatherOfSanya是否为空，并打印提示信息。然后，通过方法insert(_:)向集合中添加一个元素"cloudy"。

```
if weatherOfSanya.isEmpty {
    print("The set of weather is empty!")
} else {
    print("There are \(weatherOfSanya.count) kinds weather!")
}

weatherOfSanya.insert("cloudy")
```

图 5.14　集合的 isEmpty()和 insert()方法

5.2.3　删除元素

要删除集合中的一个元素，可以使用方法 remove(_:)。在 remove()方法的参数表中要提供被移除元素的值，如果集合中有这个元素，则删除后返回这个元素的值；如果集合中不存在该元素，则返回一个 nil。要删除集合中的所有元素，可以直接调用方法 removeAll()。如图 5.15 所示，集合变量 weatherOfSanya 中含有元素"stormy"，删除该元素后，返回值为"stormy"。然后删除元素"dry"时，由于集合 weatherOfSanya 中不含有此元素，因此返回值为 nil。

```
weatherOfSanya.remove("stormy")
weatherOfSanya.remove("dry")
```

图 5.15　集合中删除元素

5.2.4　检索特定元素

要检查集合中是否包含某个特定的元素，可以直接调用方法 contains(:_)，如图 5.16 所示。

```
if weatherOfSanya.contains("sunny") {
    print("Sanya is sunny sometimes.")
}
```

图 5.16　集合中检索特定元素

5.2.5　遍历集合

通过 for-in 循环可以对集合进行遍历，如图 5.17 所示。

```
for weather in weatherOfSanya {
    print("\(weather)")
}
```

图 5.17　集合的遍历

5.2.6 集合排序

集合中的元素是乱序排列的。如果要有序地输出集合元素，可以通过 sorted()函数对集合中的元素按照值先排序，然后再输出。图 5.18 打印了集合变量 weatherOfSanya 排序后的结果。

```
print("the result of unsorted : ")
for weather in weatherOfSanya {
    print("\(weather)")
}
```

图 5.18 集合的排序

5.2.7 集合间的运算

Swift 提供各种方法支持两个集合之间的操作，包括 intersect()、union()、subtract()。其中，intersect()方法计算两个集合的交集，结果为一个新的集合；union()方法计算两个集合的并集，结果为一个新的集合；subtract()方法计算一个集合中不属于另一个集合的部分，结果为该部分元素形成的新集合。图 5.19 对两个集合变量 weatherOfSanya 和 weatherOfBj 分别进行交集、并集和相减运算。

```
var weatherOfSanya = Set<String>()
weatherOfSanya = ["rainy","sunny","stormy"]

var weatherOfBj : Set = ["dry","windy","frogy","sunny"]

weatherOfBj.intersection(weatherOfSanya)
weatherOfBj.union(weatherOfSanya)
weatherOfBj.subtracting(weatherOfSanya)
weatherOfBj.subtract(weatherOfSanya)
```

图 5.19 集合的实例

练习题

1. 定义一个空的字符串型集合 gradeOfTheory，用来表示理论课成绩的分档集合，然后判断 gradeOfTheory 是否为空。如果不为空，则打印提示信息；如果为空，则依次向集合中添加元素 Fail、Pass、Common、Good、Excellent。

2. 定义一个字符串型集合 gradeOfExperiment，用来表示实验课成绩的分档集

合，并直接将 gradeOfTheory 赋值给 gradeOfExperiment。然后，将其中的元素 Common、Good、Excellent 从集合中删除。最后，判断集合 gradeOfExperiment 中是否含有元素 Fail 和 Pass，并打印出提示信息。

3. 分别遍历集合 gradeOfTheory 和 gradeOfExperiment，并打印出其中的元素。

4. 对集合 gradeOfTheory 和 gradeOfExperiment 进行排序后，重新打印。

5. 对集合 gradeOfTheory 和 gradeOfExperiment 进行交集、并集、相减运算，并将结果打印出来。集合的相减运算相关的方法有 subtracting()和 subtract()，请说明两者的差别。

5.3　字典

字典通过键值对(key-value pair)的形式存储数据，每个值(value)都有唯一的键(key)与其对应。字典中数据的组织是无序的。对字典中元素的操作，一般都是基于 key 实现的。

声明字典类型的格式为 Dictionary<KeyType，ValueType>。其中，KeyType 为 key 的类型，ValueType 为 value 的类型。

声明字典类型的简化格式为[KeyType ： ValueType]。

5.3.1　字典的初始化

字典的初始化从声明并创建一个空字典开始。如图 5.20 所示，分别通过两种声明格式定义并初始化了两个字典类型变量。其中，字典变量 ascIIDictChar 的键为整型，值为字符型；字典变量 ascIIDictNum 的键为整型，值为整型。

```
var asciIDictChar = Dictionary<Int, Character>()
var asciIDictNum = [Int:Int]()
```

图 5.20　字典的声明

声明字典变量的时候，如果知道字典中的元素，可以直接初始化为具体的值。如图 5.21 所示，在定义两个字典类型变量时直接赋初始值。其中，ascIIDictChar 字典中包含 6 个元素，键为十进制的 ASCII 码值，值为码值对应的字符。ascIIDictNum 字典中也包含 6 个元素，键为十进制的 ASCII 码值，值为码值对应的整型数字。