第5章

Scala 函数

函数是组织好的、可重复使用的、用来实现单一或相关联功能的代码段，其能提高应用的模块性和代码的重复利用率。本章主要介绍定义函数的方法、匿名函数和高阶函数。

5.1 定义函数

Scala 声明函数的语法格式如下所示。

```
def 函数名([参数列表]):[函数的返回值类型] ={
    函数体
    return [返回值表达式]
}
```

在 Scala 中，用户可以使用 def 关键字定义函数，定义函数时需要注意以下几个事项。

(1) 以 def 关键字开头，表示定义函数。

(2) def 之后是函数名，这个名字由用户自行指定，def 和函数名中间以空格分隔。

(3) 函数名后跟圆括号，圆括号之后是一个冒号“:”[函数的返回值类型]，再后面就是一个等号“=”，最后是函数体以及 return 语句。圆括号内的是函数参数，被称为形式参数，简称形参，参数是可选的，函数可以没有参数。参数名后面应紧跟着冒号和参数类型，Scala 要求必须指明参数类型。如果有多个参数，那么参数之间应用逗号隔开。只要函数不是递归的，就不需要指定函数返回值类型。

(4) 函数体的作用是指定函数应当完成什么操作，其由若干语句组成，如果最后没有 return 语句，那么函数体中最后一个表达式的值就是函数的返回值。用户也可以像 Java 那样使用 return 来带回返回值，不过在 Scala 中这种做法并不常见。

注意：Scala 允许函数的嵌套定义，即在一个函数定义里再定义另外一个函数。

定义函数举例如下。

1. 定义有返回值的函数

(1) 标准形式(包含函数形参、返回值类型、return 语句)示例如下。

```
def addInt(a: Int, b: Int): Int ={
    var total: Int =a +b
    return total
}
```

圆括号里是形参，圆括号后面的冒号之后是函数返回值的数据类型，花括号里是函数体。

(2) 定义函数时可以省略函数返回值类型和 return 语句，示例如下。

```
def addInt(a: Int, b: Int) ={
     a +b
}
```

当函数末尾的表达式值有效且可以作为函数返回值时，可以隐式地定义返回值类型，Scala 会自动判断。同时 return 也可以省略。

Scala 会自动返回函数体中最后一个表达式的值并判断类型。在 Scala 中赋值语句返回的是空值，所以，如果想将某个表达式作为返回值使用，应确保其值有效。

(3) 省略花括号。

当函数体只有一行语句时，可以省略花括号。上面的函数可以再简写成以下形式。

```
def addInt(a: Int, b: Int) =a +b
```

2. 定义无返回值的函数

(1) 显式标识无返回值的函数，示例如下。

```
def retrunNone(a: Int,b: Int): Unit ={
    print(a +b)
}
```

Unit 关键字表示该函数无返回值。

(2) 省略 Unit。与有返回值类似，这里也可以省略 Unit 关键字，让 Scala 推断这个函数无返回值。那么它是怎么知道的呢？就是省略等号。当定义函数中没有等号时，无论函数内部有没有返回值，Scala 都认为这个函数无返回值，示例如下。

```
def retrunNone(a: Int,b: Int){
    a +b
}
scala>retrunNone(1,2)          //执行后没有返回值
```

3. 调用函数

Scala 提供了多种不同的函数调用方式，以下是调用函数的标准格式。

```
functionName(参数列表)
```

【例 5-1】 编写一个函数，将整数数组中相邻的元素置换，例如，将 Array(1,2,3,4,5) 置换为 Array(2,1,4,3,5)，示例如下。

```
scala>: paste
// Entering paste mode (ctrl-D to finish)
def reorderArray(arr: Array[Int]): Array[Int]={
  val t =arr.toBuffer
```

```
    for(i <-1 until (t.length,2);tmp =t(i);j =i -1){
      t(i) =t(j)
      t(j) =tmp
    }
    t.toArray
  }
println(reorderArray(Array(1,2,3,4,5)).mkString(","))
// Exiting paste mode, now interpreting
```

按 Ctrl+D 组合键退出 paste 模式并执行代码,结果如下。

```
2,1,4,3,5
```

【例 5-2】 给定一个整数数组,以其为基础创建一个新的数组,新数组中包括原数组中的全部正值,以原有顺序排列,之后的元素是原数组中的零或负值,也以原有顺序排列,示例如下。

```
scala>: paste
// Entering paste mode (ctrl-D to finish)
val a =Array(1,3,-3,-5,-7,3,2)
def reorderArray(arr: Array[Int]) ={
    val b =arr.filter(_ >0)
    val c =arr.filter(_ <=0)
    val newarr =b ++c
    print(newarr.toBuffer.toString())
  }
reorderArray(a)
// Exiting paste mode, now interpreting
```

按 Ctrl+D 组合键退出 paste 模式并执行代码,结果如下。

```
ArrayBuffer(1, 3, 3, 2, -3, -5, -7)
```

【例 5-3】 编写函数“largest(fun:(Int)=>Int,inputs:Seq[Int])”,输出在给定输入序列中给定函数的最大值。举例来说,“largest(x=>10x-x×x,1 to 10)”应该返回 25,示例如下。

```
scala>def largest(fun: (Int)=>Int, inputs: Seq[Int]) =inputs.map(fun(_)).max
largest: (fun: Int =>Int, inputs: Seq[Int])Int
scala>println(largest(x =>10 * x -x * x, 1 to 10))
25
```

5.2 匿名函数

Scala 匿名函数是由箭头操作符“=>”定义的,箭头的左边是参数列表,箭头的右边是表达式,表达式的值即为匿名函数的返回值,示例如下。

```
scala>def f1(a: Int,b: Int)={a+b}          //声明一个普通函数
f1: (a: Int, b: Int)Int
scala>(a: Int,b: Int)=>a+b                 //声明了一个匿名函数
res14: (Int, Int) =>Int =$$Lambda$858/371976262@503358e6
```

“=>”操作符指明这个函数把左边的整数 a、b 转变成了右边的 a+b。所以，这个函数可以把任意整数 a、b 映射为 a+b，其调用方式如下。

```
scala>((a: Int,b: Int)=>a+b)(1,2)          //声明匿名函数并调用
res15: Int =3
scala>val f2=(a: Int,b: Int)=>a+b          //声明了一个匿名函数，并赋值给一个变量
f2: (Int, Int) =>Int =$$Lambda$861/1399992133@6c8f2e4e
scala>f2(1,2)                              //调用匿名函数
res16: Int =3
```

如果想让匿名函数包含多条语句，则可以用花括号包住函数体，一行放一条语句，这样就组成了代码块。当函数被调用时，所有的语句将被执行，而函数的返回值就是最后一行表达式所产生的值，示例如下。

```
scala>: paste
// Entering paste mode (ctrl-D to finish)

val f3=(x: Int)=>{
    println("Who")
    println("am")
    println("I")
    x+100
}

// Exiting paste mode, now interpreting
```

按 Ctrl+D 组合键退出 paste 模式并执行代码，结果如下。

```
scala>val y=f3(1)
Who
am
I
y: Int =101
scala>y
res3: Int =101
```

此外，也可以把多条语句写在同一行中，语句之间用“;”隔开，示例如下。

```
scala>val f4=(x: Int)=>{println("Who");println("am");println("I");x+100}
f4: Int =>Int =$$Lambda$802/1847252568@486dd616
scala>val w=f4(10)
Who
am
I
w: Int =110
scala>print(w)
110
```

5.3　高阶函数

高阶函数是指使用其他函数作为参数或者返回一个函数作为结果的函数，示例如下。

```
scala>def f3(a: Int,b: Int,f: (Int,Int)=>Int)={f(a,b)}        //定义一个高阶函数
```

```
f3: (a: Int, b: Int, f: (Int, Int) =>Int)Int
scala>f3(2,3,(a: Int,b: Int)=>{a*b})                    //调用高阶函数
res17: Int =6
scala>f3(2,3,(a: Int,b: Int)=>{a+b})                    //调用高阶函数
res18: Int =5
```

如果想让匿名函数体中的语句更简洁,则可以把下画线“_”当作函数表达式中的自变量占位符,前提是每个参数在函数表达式内仅出现一次,举例如下。

```
scala>val a =Array(1,3,5,7,9,10)
a: Array[Int] =Array(1, 3, 5, 7, 9, 10)
scala>a.filter(_>5)
res5: Array[Int] =Array(7, 9, 10)
```

这里可以把下画线看作表达式里需要被“填入”的“空白”,这个空白在每次函数被调用时用函数的参数填入。a.filter()方法会把_>5 里的下画线“_”首先用 1 替换,就如 1>5,然后用 3 替换,如 3>5,这样直到 a 的最后一个值替换处理为止。_>5 与 x=>x>5 的功能相同,演示如下所示。

```
scala>a.filter(x=>x>5)
res6: Array[Int] =Array(7, 9, 10)
```

当有多个下画线时,第 1 个下画线代表函数的第 1 个参数,第 2 个下画线代表函数的第 2 个参数,……,以此类推。

5.4 习　　题

1. 编写 WordCount()函数,统计传入的字符串中单词的个数。

2. 编写一个函数 minmax(values:Array[Int]),返回数组中最小值和最大值的对偶。

3. 编写一个函数 indexes()。根据给定字符串,创建一个包含所有字符下标的映射。举例来说:indexes("acbbibbiddi")应返回一个映射,让"a"对应集{0},"b"对应集{2,3,5,6},以此类推。