kotlin学习笔记——常见高阶函数(map、flatMap、fold、reduce、joinToString、filter、takeWhile)

作者：崔兴旺
时间：2019/07/07 09:48
标签： Kotlin

高阶函数

1.map

```
//1.map:将List中每个元素转换成新的元素,并添加到一个新的List中,最后将新List返回
arrayOf(1, 2, 3).map { i: Int -> i * 10 }.forEach(::println)
/**
 * 打印输出
 * 10
 * 20
 * 30
 */
```

2.flatMap

```
//2.flatMap:将数组中全部元素按顺序组成一个list
//注意:lambda表达式中的参数类型可以不写.如:List<String>和IntRange
listOf(listOf("a", "b"), listOf("c", "d")).flatMap { i: List<String> -> i.asIterable() }.forEach(::print
println("----------")
arrayOf(1..3, 1..5).flatMap { i: IntRange -> i.asIterable() }.forEach(::println)
/**
 * 打印输出
 * a
 * b
 * c
 * d
 * ----------
 * 1
 * 2
 * 3
 * 1
 * 2
 * 3
 * 4
 * 5
 */
```

3.fold

```
/**
 * 3.fold:将集合中的元素依次冒泡组合,最终得到一个结果
 * 第一次执行时,由初始值10作为参数a,由集合中第0个元素作为参数b
 * 第二次执行时,第一次执行的返回值作为参数a,由集合中第1个元素作为参数b
 * 依次类推...
 * 最终将结果返回
 */
val foldResult1 = arrayOf(1, 2, 3).fold(10, { a, b -> a + b })//计算过程为10+1+2+3,等于16
val foldResult2 = arrayOf(1, 2, 3).fold(10, { a, b -> a * b })//计算过程为10*1*2*3,等于60
println(foldResult1)
println(foldResult2)
/**
 * 打印输出
 * 16
 * 60
 */
```

4.reduce

```
/**
 * 4.reduce:与fold类似,区别是reduce没有初始值
 */
val reduceResult1 = arrayOf(1, 2, 3, 4).reduce { acc, i -> acc + i }//计算过程为1+2+3+4,等于10
val reduceResult2 = arrayOf(1, 2, 3, 4).reduce { acc, i -> acc * i }//计算过程为1*2*3*4,等于24
val reduceResult3 = arrayOf("a", "b", "c", "d").reduce { acc, i -> "$acc,$i" }
println(reduceResult1)
println(reduceResult2)
println(reduceResult3)
/**
 * 打印输出
 * 10
 * 24

 * a,b,c,d
 */
```

5.joinToString

```
/**
 * 5.joinToString:为集合元素添加分隔符,组成一个新的字符串并返回
 */
val joinToStringResult1 = arrayOf("a", "b", "c", "d").joinToString { i -> i }
val joinToStringResult2 = arrayOf("a", "b", "c", "d").joinToString(separator = "#", prefix = "[前缀]", p
println(joinToStringResult1)
println(joinToStringResult2)
```

```
/**
 * 打印输出
 * a, b, c, d
 * [前缀]a#b#c#[省略号][后缀]
 */
```

6.filter

```
/**
 * 6.filter:将中的元素遍历,把符合要求的元素添加到新的list中,并将新list返回
 */
//根据元素值来过滤
arrayOf(1, 2, 3, 0, 4).filter { i -> i >= 2 }.forEach(::println)
println("----------")
//根据索引来过滤
arrayOf(1, 2, 3, 0, 4).filterIndexed { index, i -> index >= 2 }.forEach(::println)
/**
 * 打印输出
 * 2
 * 3
 * 4
 * ----------
 * 3
 * 0
 * 4
 */
```

7.takeWhile

```
/**
 * 7.takeWhile:遍历list中的元素,将符合要求的元素添加到新集合中
 * 注意:一旦遇到不符合要求的,直接终止
 */
//注意:返回的集合中只有4和3,没有5,因为遇到2时,不符合要求,程序直接终止
arrayOf(4, 3, 2, 5).takeWhile { i -> i > 2 }.forEach(::println)
/**
 * 打印输出
 * 4
 * 3
 */
```
