# 1 集合

## 2.1 集合简介

Scala中的集合(collection)分为两种，一种是可变集合，另一种是不可变的集合。可变集合可以被添加、修改、删除；不可变集合执行添加、删除、更新操作后返回的是新集合，原来的集合保持不变。Scala中所有的集合都来自于Scala 默认采用不可变集合，对于几乎所有的集合类，Scala 都同时提供了可变(mutable)和不可变(immutable)的版本。

Scala 的集合有三大类：序列 Seq(有序的,Linear Seq)、集 Set、映射 Map【key->value】，所有的集合都扩展自 Iterable 特质，在 Scala 中集合有可变（mutable）和不可变（immutable）两种类型。

## 2.2 数组

数组为相同数据类型的元素按一定顺序排列的整合，在Scala语言中数组是最常用、最重要的数据结构。Scala中的数组分为定长数组和变成数组，定长数组在定义时长度被确定，在运行时数组长度不会发生改变，而变长数组内存空间长度会随程序运行的需要动态扩容。

<https://www.scala-lang.org/api/2.12.18/scala/Array.html>

import scala.collection.mutable.ArrayBuffer  
  
object ArrayScala {  
 def main(args: Array[String]): Unit = {  
 //1.创建数组对象  
 //定长数组，默认值为0  
 val a1 = new Array[Int](10)  
 //变成数组，直接进行初始化  
 val a2 = Array(1,2,3,4,5,6)  
 println(a2(1))  
 a2.foreach(a => println(a))  
 a2.foreach(print)  
 println()  
 println(a2.mkString(","))  
 a2(3)=10  
 println(a2.mkString(","))  
 println(a2.length)  
 println(a2.size)  
  
 //变长数组  
 val a3 = ArrayBuffer(1,2,3,4,5)  
 println(a3.mkString(","))  
 a3.append(10,20)  
 println(a3.mkString(","))  
 a3 += (30,40,50)  
 println(a3.mkString(","))  
 //可变数组与不可变数组之间的转换  
 a1.toBuffer  
 a3.toArray  
  
 }  
  
}

## 1.3 列表(List)

List常用方法:

<https://www.scala-lang.org/api/2.12.18/scala/collection/immutable/List.html>

List操作示例:

import scala.collection.mutable.ListBuffer

object ListExample {

def main(args: Array[String]): Unit = {

//1.创建List对象

val v1 = List(1,2,3,4,7,1,4)

println(v1(1))

println(v1)

//2.空列表

val v2\_1 = Nil

val v2\_2 = List()

//3.列表的连接操作

val v3 = List(33,55,44)

//3.1 将V1作为一个元素添加到1,2前面

val v3\_1 = v1 :: List(1,2)

println(s"::$v3\_1")

//3.2 +: 用于在List头部添加元素

val v3\_2 = v1 +: v3

println(s"+:$v3\_2")

//3.3 :+ 用于在List尾部追加元素

val v3\_3 = v1 :+ v3

println(s":+$v3\_3")

//4.创建重复元素

val v4 = List.fill(5)("hello")

println(s"fill:$v4")

//5.集合反转

val v5 = v1.reverse

println(v5)

//6.排序(升序/降序)

println(v5.sorted)

//7.遍历

v1.foreach(print)

println()

for(i<- v1){

println(i)

}

//8.可变列表

val v8 = ListBuffer(1,2,3,4)

v8.append(33,44,55)

println(v8)

}

}

ListAPI应用示例

object ListAPIExample {

def main(args: Array[String]): Unit = {

/\*val nums = List(1,2,3)

val square = (x:Int) => x \* x

val squareNums3 = nums.map(square) // List(1,4,9)

val squareNums3 = nums.map(x=>square(x)) // List(1,4,9)

val squareNums1 = nums.map(num => num \* num) //List(1,4,9)

val squareNums2 = nums.map(math.pow(\_,2)) //List(1,4,9)

val squareNums2 = nums.map(x=>math.pow(x,2)) //List(1,4,9)

val text = List("A,B,C","D,E,F")

val textMapped = text.map(\_.split(",").toList) // List(List("A","B","C"),List("D","E","F"))

val textFlattened = textMapped.flatten // List("A","B","C","D","E","F")

val textFlatMapped = text.flatMap(\_.split(",").toList) //// List("A","B","C","D","E","F")\*/

/\* val nums = List(1,2,3)

val sum1 = nums.reduce((a,b) => a + b) //6

val sum2 = nums.reduce(\_+\_) //6

val sum3 = nums.sum //6\*/

/\*val nums = List(2.0,2.0,3.0)

val resultLeftReduce = nums.reduceLeft(math.pow) // pow(pow(2.,2.0),3.0) = 64.0

val resultRightReduce = nums.reduceRight(math.pow) // pow(2.0,pow(2.0,3.0)) = 256.0\*/

/\*val nums = List(2,3,4)

val sum = nums.fold(1)(\_+\_) // 1+2+3+4 = 9

val nums2 = List(2.0,3.0)

val result1 = nums2.foldLeft(4.0)(math.pow) // pow(pow(4.0,2.0),3.0) = 4096

val result2 = nums2.foldRight(1.0)(math.pow) // pow(2.0,pow(3.0,1.0)) = 8.0\*/

/\*val nums = List(1,3,2,4)

val sorted = nums.sorted // List(1,2,3,4)

val users = List(("zhangmin",25),("wangfei",23))

val sortedByAge = users.sortBy{

case (user,age) => age

} // List(("wangfei",23),("zhangmin",25))\*/

/\*val nums = List(1,2,3,4)

val odd = nums.filter( x=>x % 2 !=0) //List(1,3)

val even = nums.filterNot( \_ % 2 != 0) // List(2,4)\*/

val nums = List(-1,-2,0,1,2)

val plusCnt1 = nums.count (\_ >0)

val plusCnt1 = nums.count (x=>x >90)

println(plusCnt1)

val pusCnt2 = nums.filter(\_ >0).length

val nums1 = List(1,2,3)

val nums2 = List(2,3,4)

val diff1 = nums1 diff nums2 // List(1)

val diff2 = nums2.diff(nums1) //List(4)

val union1 = nums1 union nums2 // List(1,2,3,2,3,4)

val union2 = nums2 ++ nums1 //List(2,3,4,1,2,3)

val intersection = nums1 intersect nums2 // List(2,3)

val list = List("A","B","C","A","B")

val distincted = list.distinct //List("A","B","C")

val data = List(("zhangmin","Male"),("wangxiao","Femail"),("Liming","Male"))

val group1 = data.groupBy(\_.\_2)

val group2 = data.groupBy(x=>x.\_2)

//Map(Male -> List((zhangmin,Male), (Liming,Male)), Femail -> List((wangxiao,Femail)))

println(group1)

val group2 = data.groupBy{case(name,sex) => sex} //Map(Male -> List((zhangmin,Male), (Liming,Male)), Femail -> List((wangxiao,Femail)))

println(group2)

}

}

## 1.4 映射(Map)

Map是一种键值对的集合。

Map的操作示例如下所示:

object MapScala {  
 def main(args: Array[String]): Unit = {  
 //1.创建Map对象  
 val m1 = Map("zhangsan"->1,"lisi"->2,"wangming"->3)  
 println(m1)  
 //2.增加数据  
 val m2 = m1 + ("liufei"->4)  
 println(m2)  
 // 返回值false  
 println(m1 == m2)  
 //3.删除数据  
 val m3 = m1 - "zhangsan"  
 println(m3)  
 //4.修改数据  
 val m4\_1 = m1.updated("zhangsan",5)  
 println(m4\_1.mkString(","))  
 //5.取值  
 //方法返回Option类型的对象，此类有两个对象:Some、None  
 println(m4\_1.get("zhangsan"))  
 //根据键值获取Value值，如果集合中没有此键，则返回默认值。  
 println(m4\_1.get("zhangsan").getOrElse(0))  
 //6.可变的Map  
 val m6 = mutable.Map("1001"->20,"1002"->18,"1003"->19)  
 m6   
 println(m6)  
  
 }  
  
}

# 练习作业

1.统计一篇英文文章英语单词的个数

2.根据如下数组中的初始数据统计每个城市的平均气温

var day01 = Array(("taiyuan",10.0),("beijing",12.0),("shanghai",9.0),("guangzhou",20.0))

var day02 = Array(("taiyuan",12.0),("beijing",15.0),("shanghai",7.0),("guangzhou",21.0))

var day03 = Array(("taiyuan",8.0),("beijing",6.0),("shanghai",5.0),("guangzhou",22.0))