КОЛЛЕКЦИИ

СОЗДАНИЕ КОЛЛЕКЦИЙ

```
val list = List(1, 2, 3, 2)

val vector = Vector(1, 2, 3, 2)

val stream = Stream(1, 2, 3, 2)

val set = Set(1, 2, 3, 2)

val map = Map("Москва" -> 12e6, "Питер" -> 5e6)
```

СОЗДАНИЕ КОЛЛЕКЦИЙ

```
val phrase1 = Vector("+")

val phrase2 = phrase1 +: "Stepik"

val phrase3 = "Scala" :+ phrase2

val phrase4 = Stream.empty[String]

val phrase5 = phrase4 :+ "=" :+ "♥"

val phrase = phrase3 ++ phrase5

phrase.mkString(" ")
```

СОЗДАНИЕ КОЛЛЕКЦИЙ

```
val cities1 = Map("Питер" -> 5e6, ("Москва", 12e6))

val cities2 = cities1 + ("Волгоград" -> 1e6)

val cities3 = List("Ростов-на-Дону" -> 1e6)

val cities4 = cities2 ++ cities3
```

ПОЛУЧЕНИЕ ЭЛЕМЕНТА

```
val cities = Vector("Москва", "Волгоград", "Питер")

cities(1) // Волгоград
cities.head // Москва
cities.last //Питер

val cityMap = Map("Москва" -> 12e6, "Питер" -> 5e6)

cityMap("Питер") // 5000000.0
cityMap.get("Москва") // Some(1.2E7)
cityMap.get("Петушки") //None

val citySet = Set("Москва", "Волгоград", "Питер")

citySet("Волгоград") // true
citySet("Петушки") // false
```

ИНФОРМАЦИЯ

```
val cities = Vector("Москва", "Волгоград", "Питер")

cities.size // 3
cities.contains("Москва") // true
cities.indices //0 until 3

val cityMap = Map("Москва" -> 12e6, "Питер" -> 5e6)

cityMap.size // 2
cityMap.contains("Питер") // true
cityMap.keySet // Set("Москва", "Питер")

val citySet = Set("Москва", "Волгоград", "Питер")

citySet.size // true
citySet.contains("Петушки") // false
```

СУБКОЛЛЕКЦИИ

```
val nums = Vector.range(1, 11)

nums.slice(3, 7)
nums.tail
nums.init
nums.take(3)
nums.drop(3)
nums.takeRight(3)
nums.dropRight(3)
```

СУБКОЛЛЕКЦИИ

```
val cityMap = Map(
  "Москва" -> 12e6,
  "Питер" -> 5e6,
  "Волгоград" -> 1e6
)

cityMap - "Питер"
cityMap -- List("Москва", "Волгоград")

val citySet = Set("Москва", "Волгоград", "Питер")

citySet - "Питер" // true
citySet -- List("Москва", "Волгоград")
```

УСЛОВНЫЙ ОТБОР

```
val nums = Vector.range(1, 21)

val odds = nums.filter(_ % 2 == 1)
val evens = nums.filterNot(_ % 2 == 1)
val (odds1, evens1) = nums.partition(_ % 2 == 1)

val small = nums.takeWhile(_ < 10)
val big = nums.dropWhile(_ < 10)
val (small1, big1) = nums.span(_ < 10)
```

ОТОБРАЖЕНИЯ

```
val nums = List.range(0, 10)

val alpha = 'A' to 'Z'

val nums2 = nums.map(i => alpha(i))
val nums3 = nums.map(alpha)
```

ОТОБРАЖЕНИЯ

```
val nums: List[Int] = List.range(0, 10)

val alpha = 'A' to 'Z'

val charLists: List[Char] =
  nums.collect{
    case i if i % 2 == 0 => alpha(i / 2 * 3)
    case 3 => '_'
    case 5 | 7 => '!'
}
```

ОТОБРАЖЕНИЯ

```
val nums: List[Int] = List.range(0, 10)

val alpha = 'A' to 'Z'

val charLists: List[List[Char]] =
  nums.map(i => List(alpha(i), alpha(i + 3)))

val charList: List[Char] = charLists.flatten

val chars: List[Char] =
  nums.flatMap(i => List(alpha(i), alpha(i + 3)))
```

В этом разделе
мы изучили коллекции

В следующем узнаем
о for-comprehension