CURSO DE PROGRAMACION SCALA
Sesión 12

Sergio Couto Catoira

Índice

- Continuación de listas no-estrictas
- Gestión del estado en programación funcional
- Types

Función unfold

- Escribe las funciones map, take, takeWhile en función de unfold

- Escribe la función zipWith en función de unfold. Esta función está hecha en Lista, debe juntar dos listas aplicándoles una función.

- Escribe la función zipAll en función de unfold. ZipAll debe continuar recorriendo mientras alguna de las listas tenga elementos.

Función unfold

- Implementa la función tails que para un Stream, devuelva un Stream conteniendo los sufijos del stream de entrada como Stream:
  - Entrada: Stream(1,2,3)
  - Salida: Stream(Stream(1,2,3), Stream(2,3), Stream(3), Empty)

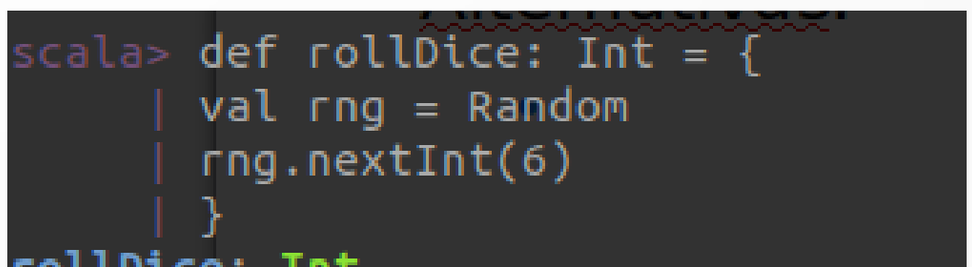
Estado

- Generación de números aleatorios. Clase Random de scala.util

```
scala> import scala.util.Random
import scala.util.Random

scala> val r = Random
r: scala.util.Random.type = scala.util.Random$@4ab926bc

scala> r.ne
ne              nextBytes     nextFloat       nextInt     nextPrintableChar
nextBoolean     nextDouble    nextGaussian    nextLong    nextString

scala> r.ne
   final def ne(x$1: AnyRef): Boolean

scala> r.nextInt
res0: Int = 1830252456

scala> r.nextInt
res1: Int = 1300677031

scala>
```

Estado

- Se mantiene información sobre el estado actual para devolver un valor diferente en cada llamada => Side effect
- Dificultad de testeo en funciones que lo usen
- Alternativas:
  - Pasarle el generador como parámetro => Si queremos asegurar el estado debemos saber cuántas veces llamamos
  - Construir un generador funcional



Estado

- La clave es hacer las actualizaciones de estado explícitas
- En lugar de devolver sólo el número aleatorio, devolveremos el número y el nuevo estado en el que se queda el generador.
- El trait RNG define un trait con una funcion nextInt que devuelve ambas cosas
- La case class RNG define esa función

Ejercicios

- Define una función que devuelva un entero no negativo entre 0 y Int.MaxValue (2147483647). Ten en cuenta que Int.MinValue (-2147483648) no tiene un equivalente no-negativo

- Escribe una función para generar un Double entre 0 y 1 (sin incluirlo). Usa Int.MaxValue y la función anterior

Ejercicios

- Define las siguientes funciones que generen tuplas de números aleatorios
  - intDouble: Debe generar una tupla Int, Double
  - doubleInt: Debe generar una tupla Double, Int
  - double3: Debe generar una tupla3 con 3 Doubles

- Escribe una función para generar una lista con los enteros indicados

Estado

- Todas las funciones reciben un RNG y devuelven un par (A, RNG)
- Es tedioso pasar el estado manualmente
- Definimos un alias de tipo
  - type Rand[+A] = RNG => (A, RNG)

- Definimos combinadores para evitar pasar el estado explícitamente.
- Ejercicio: Define las funciones unit y map.
  - Map debe tener el comportamiento habitual
  - Unit debe devolver el estado sin usarlo, con el parámetro recibido

Ejercicios

- Usa map para obtener un entero par no negativo
  - def nonNegativeEven: Rand[Int]
- Usa map para reimplementar double
  - def doubleMap: Rand[Double]
- ¿Sería posible implementar intDouble, doubleInt con map?

Ejercicios

- Implementa map 2, que combine dos acciones y una función para combinar sus resultados y devuelva una nueva acción con la combinación
  - def map2[A,B,C](ra: Rand[A], rb: [Rand[B])(f:(A, B) => C): Rand[C]
- Usa map2 para imlpemnetar un afunción both que devuelva dos valores como tupla
  - def both[A, B](ra:Rand[A], rb: Rand[B]): Rand[(A,B)]
- Usa la función both para implementar intDouble y doubleInt