Introducción a la Programación Funcional

Universidad Autónoma de Entre Ríos

Lazy evaluation o call by name

Lazy evaluation o call by name

- Si buscamos en la wikipedia “lazy evaluation”, podemos encontrar este concepto como lazy evaluation o call-by-need. Los dos nombres le quedan muy bien pero personalmente opino que call-by-need explica mejor lo que sucede. Lazy evaluation permite que un bloque de código sea evaluado luego o mejor dicho sólo cuando se lo necesite, esto nos permite realizar código que tiene un rendimiento superior en ciertas situaciones.

Lazy evaluation o call by name

Veamos un ejemplo, supongamos que tengo la función multiplicación:

mult(a, b) = a * a //ojo no utiliza b

Si llamara a esta función con los parámetros 8 y 5+2 de forma tradicional o eager sucedería lo siguiente :

mult(8, 5+2)

mult(8,7)

8*8

64

Pero si ejecutamos el código de forma perezosa:

mult(8, 5+2)

8 * 8

64

Lazy evaluation en los lenguajes

- Haskell por defecto es de ejecución perezosa.
- Lazy evaluation en R: R es un lenguaje funcional y a la vez tiene la característica de evaluación perezosa.
- En C, C++, java o javascript existen operadores perezosos por cuestiones de performance.

Lazy evaluation en scala

- Scala es un lenguaje que por defecto es eager pero se puede indicar que un parámetro sea manejado de forma perezosa, para esto se debe utilizar “=>” cuando se declara el parámetro. Vamos a hacer una función “and” en Scala.
- A la vez se puede indicar que un inmutable es lazy con el modificador lazy antes del val
- Que sucede cuando hacemos:
  - val x = { println("se imprimo x"); 1}
  - lazy val y = {println("se imprimo y"); 2}
  - x + y

Stream

- Es similar a las listas pero el tail utiliza lazy evaluation
- Stream.empty = Nil
- Stream.cons = ::
- StreamRange(1,10).take(3) cuantos elementos va a generar?
-