

# PROGRAMACIÓN FUNCIONAL

## Trabajo Práctico Nro. 3

**Temas:** Currificación. Alto orden. Reducción. Ordenes de evaluación.

### Bibliografía relacionada:

- Simon Thompson. The craft of Functional Programming. Addison Wesley, 1996. Cap. 6 y 7.
- L.C. Paulson. ML for the working programmer. Cambridge University Press, 1996. Cap. 2 y 5.

1. a) Cuando es posible, dar al menos dos ejemplos de funciones totales que tengan cada uno de los siguientes tipos:

- |   |  |
|---|--|
| 1) $\text{Bool} \rightarrow \text{Bool} \rightarrow \text{Bool}$                      | 4) $\text{Int} \rightarrow \text{Int} \rightarrow \text{Int}$                          |
| 2) $a \rightarrow b \rightarrow a$  | 5) $(\text{Int} \rightarrow \text{Int}) \rightarrow \text{Int} \rightarrow \text{Int}$ |
| 3) $(a \rightarrow b) \rightarrow (a \rightarrow c) \rightarrow a \rightarrow (b, c)$ | 6) $(a \rightarrow b \rightarrow c) \rightarrow b \rightarrow a \rightarrow c$         |

b)  $\otimes$  Igual al anterior, pero con funciones parciales.

2. Decidir si las siguientes funciones son de alto orden o no, y en qué casos están currificadas. Ejemplificar.

- a)  $\text{fUno} :: ((\text{Int}, \text{Int}) \rightarrow \text{Int}) \rightarrow \text{Int} \rightarrow \text{Int} \rightarrow \text{Int}$   
 $\text{fUno } f \ x \ y = f \ (x, y)$
- b)  $\text{fDos} :: (\text{Char} \rightarrow \text{Char} \rightarrow \text{Bool}) \rightarrow (\text{Char} \rightarrow \text{Char}) \rightarrow (\text{Int} \rightarrow \text{Char}) \rightarrow \text{Char} \rightarrow \text{Int} \rightarrow \text{Bool}$   
 $\text{fDos } f1 \ f2 \ f3 \ a \ b = f1 \ (f2 \ a) \ (f3 \ b)$
- c) 1)  $\text{fTres} :: (\text{Char}, \text{Char}, \text{Char}) \rightarrow \text{Bool}$   
 $\text{fTres } (c1, c2, c3) = (c1 == c2) \ \&\& \ (c2 == c3)$   
2) ¿Qué sucede con la función anterior si definimos  
 $\text{type Pers} = (\text{Char}, \text{Char}, \text{Char})$  y  $\text{fTres} :: \text{Pers} \rightarrow \text{Bool}$ ?

3. Sea  $(\#\#)$  un operador binario. Proponer una función equivalente a  $(\#\#x)$  que no use sección de operadores.
4. Enumere las diferencias y ventajas de cada uno de los órdenes de aplicación vistos en clase.
5. Pasar de notación Haskell a notación lambda.

- a) `twice f x = f (f x)`
- b) `flip f x y = f y x`
- c) `inc = (+1)`

6. ⓧ Dar los tipos de las siguientes expresiones y verificar el resultado en Hugs:

- a) `fix`, donde  
`fix f x = f (fix f) x`
- b) `fork (fork,fork) (fork,fork)`, donde  
`fork (f,g) x = (f x,g x)`
- c) `apply apply apply`, donde  
`apply f x = f x`
- d) `curry`, donde  
`curry f x y = f (x,y)`

### Ejercicios complementarios

7. Reducir las siguientes expresiones tanto como sea posible utilizando los órdenes de aplicación vistos en clase:

- a) `square (square (3+7))`
- b) `apply first (const 3 4, (/0) (seven seven))`

8. ¿Qué función es `(+ -2)`?

9. Defina una función de tipo `Int -> Int` que no devuelva ningún valor definido.

10. Sea `h x y = f (g x y)`. Decidir cuáles de las siguientes afirmaciones son verdaderas:

- a) `h = f . g`
- b) `h x = f . (g x)`
- c) `h x y = (f . g) x y`

11. Definir `sumDigit :: Char -> Int -> Int`, que suma un dígito con un entero, utilizando la función `compose`.

Por ejemplo `sumDigit '8' 1 = 9`.