



PROGRAMACIÓN FUNCIONAL

Modelo Funcional: Currificación



¿VOS SABÍAS QUE MI
MAMA' ES TRADUCTORA
DE FRANCÉS, MANOLITO?
YO TAMBIÉN SÉ
FRANCÉS; SÉ DECIR
"PAPA" EN FRANCÉS

¿SÍÍ? ¿CÓMO SE
DICE, A VER?

PAPÁ

¡AH, ES
FÁCIL!
¡SE DICE
IGUAL!

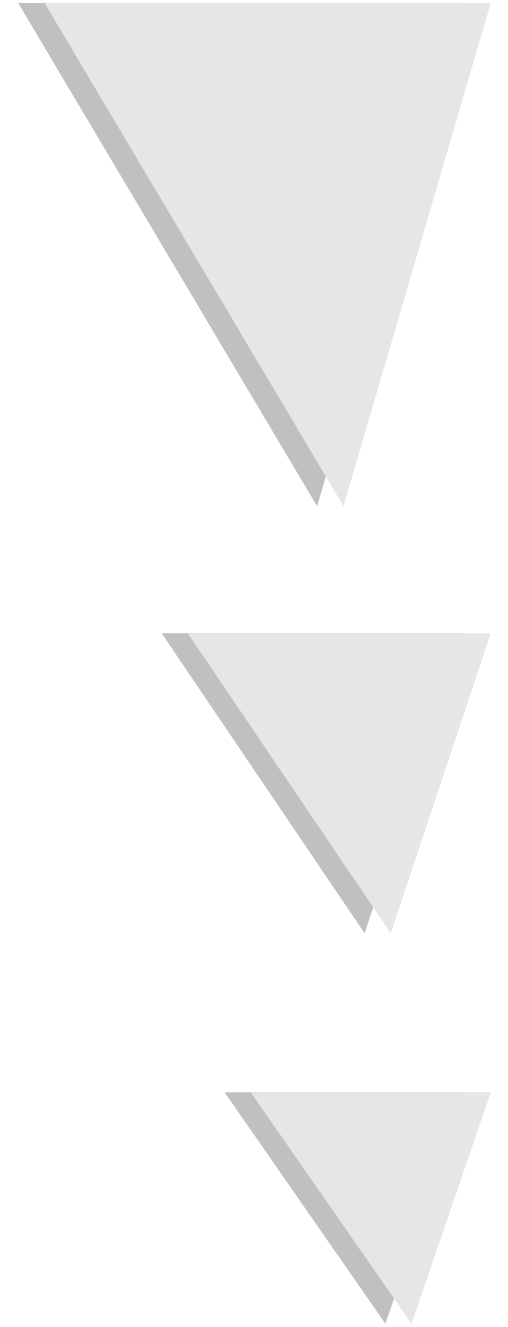
¿FÁCIL? ¿IGUAL?
¡NADA DE ESO, EL
ASUNTO ES PEN-
SAR EN FRANCÉS!
¡TRATA DE DECIR
"PAPA" PENSÁNDOLO
EN FRANCÉS! ¡DALE!
¿A VER? ¡DALE!



¡ES INÚTIL!
¡JAMÁS PODRÉ
HABLAR ESE
MALDITO IDIOMA!

Curricación

- ◆ Funciones como valores
- ◆ Aplicación: currificación.
- ◆ Notación. Ventajas.
- ◆ Ejemplos.



Funciones como valores

- ◆ Las funciones son valores, al igual que los números, las tuplas, etc.
 - ◆ pueden ser argumento de otras funciones
 - ◆ pueden ser resultado de otras funciones
 - ◆ pueden almacenarse en estructuras de datos
 - ◆ pueden ser estructuras de datos
- ◆ Funciones que manipulan funciones
 - ◆ Las llamamos “de alto orden”, abusando de esa nomenclatura

Funciones como valores

◆ Ejemplo

$\text{compose } (f,g) = h \text{ where } h \ x = f \ (g \ x)$

$\text{sqr } x = x * x$

$\text{twice } f = g \text{ where } g \ x = f \ (f \ x)$

$\text{aLaCuarta} = \text{compose } (\text{sqr}, \text{sqr})$

$\text{aLaOctava} = \text{compose } (\text{sqr}, \text{aLaCuarta})$

$\text{fs} = [\text{sqr}, \text{aLaCuarta}, \text{aLaOctava}, \text{twice } \text{sqr}]$

$\text{aLaCuarta } 2 \rightarrow ?$

◆ ¿Será cierto que $\text{aLaCuarta} = \text{twice } \text{sqr}$?

Aplicación del alto orden

- ◆ Considere las siguientes definiciones

$\text{suma}' :: ??$

$\text{suma}' (x,y) = x+y$

$\text{suma} :: ??$

$\text{suma } x = f \text{ where } f y = x+y$

- ◆ ¿Qué tipo tienen las funciones?
- ◆ ¿Qué similitudes observa entre suma y suma' ?
- ◆ ¿Qué diferencias observa entre ellas?

Aplicación del alto orden

- ◆ Similitudes

- ◆ ambas expresan la suma de dos enteros:
 $\text{suma}'(x,y) = (\text{suma } x) y$, para x e y cualesquiera

- ◆ Diferencias

- ◆ una toma un par y retorna un número;
la otra toma un número y retorna una *función*
- ◆ con suma se puede definir la función sucesor
sin usar variables extra:
 $\text{succ} = \text{suma } 1$

Curricación

- ◆ Correspondencia entre cada función de múltiples parámetros y una de alto orden que retorna una función intermedia que completa el trabajo.

- ◆ Por cada f' definida como

$$f' :: (a,b) \rightarrow c$$

$$f' (x,y) = e$$

siempre se puede escribir

$$f :: a \rightarrow (b \rightarrow c)$$

$$(f x) y = e$$

Curricación

- ◆ Correspondencia entre los tipos

$(a,b) \rightarrow c \quad y \quad a \rightarrow (b \rightarrow c)$

$\text{curry} :: ((a,b) \rightarrow c) \rightarrow (a \rightarrow (b \rightarrow c))$

$\text{curry} \dots$

$\text{uncurry} :: (a \rightarrow (b \rightarrow c)) \rightarrow ((a,b) \rightarrow c)$

$\text{uncurry} \dots$

- ◆ Se puede demostrar que

$\text{curry} (\text{uncurry } f) = f$

$\text{uncurry} (\text{curry } f') = f'$

Curricación - Sintaxis

- ◆ ¿Cómo escribimos una función curricada y su aplicación?
- ◆ Considerar las siguientes definiciones
 $\text{twice} :: (\text{Int} \rightarrow \text{Int}) \rightarrow (\text{Int} \rightarrow \text{Int})$
 $\text{twice}_1 f = g \text{ where } g\ x = f\ (f\ x)$
 $\text{twice}_2 f = \lambda x \rightarrow f\ (f\ x)$
 $(\text{twice}_3 f)\ x = f\ (f\ x)$
- ◆ ¿Son equivalentes? ¿Cuál es preferible?
 ¿Por qué?

Curricación

- ◆ ¿Qué pasa con un ejemplo más grande?
- ◆ Consideremos una función para sumar 5 números

$\text{sum5}' :: (\text{Int}, \text{Int}, \text{Int}, \text{Int}, \text{Int}) \rightarrow \text{Int}$
 $\text{sum5}' (x,y,z,v,w) = x+y+z+v+w$

vs.

$\text{sum5} :: ??$

$\text{sum5} = ??$

Curricación

◆ ¿Qué pasa con un ejemplo más grande? (cont.)

◆ Con nombres intermedios...

`sum5 :: Int -> (Int -> (Int -> (Int -> (Int -> Int))))`

`sum5 x = sum4`

`where sum4 y = sum3`

`where sum3 z = sum2`

`where sum2 v = sum1`

`where sum1 w = x+y+z+v+w`

Curricación

◆ ¿Qué pasa con un ejemplo más grande? (cont.)

◆ Con aplicación reiterada...

$\text{sum5} :: \text{Int} \rightarrow (\text{Int} \rightarrow (\text{Int} \rightarrow (\text{Int} \rightarrow (\text{Int} \rightarrow \text{Int}))))$
 $((((\text{sum5 } x) y) z) v) w = x+y+z+v+w$

vs.

$\text{sum5}' :: (\text{Int}, \text{Int}, \text{Int}, \text{Int}, \text{Int}) \rightarrow \text{Int}$
 $\text{sum5}' (x,y,z,v,w) = x+y+z+v+w$

Curricación

- ◆ ¿Cómo podemos evitar usar paréntesis?
Convenciones de notación

- ◆ La aplicación de funciones asocia a izquierda
- ◆ El tipo de las funciones asocia a derecha

`suma :: Int -> Int -> Int`

`suma x y = x+y`

`suma :: Int -> (Int -> Int)`

`(suma x) y = x+y`

Curricación

◆ ¿Qué pasa con un ejemplo más grande? (cont.)

◆ Consideremos una función para sumar 5 números

```
sum5 :: Int -> (Int -> (Int -> (Int -> (Int -> Int))))  
(((sum5 x) y) z) v) w = x+y+z+v+w
```

vs.

```
sum5 :: Int -> Int -> Int -> Int -> Int -> Int  
sum5 x y z v w = x+y+z+v+w
```

Curricación

- ◆ Por abuso de lenguaje

$\text{suma} :: \text{Int} \rightarrow \text{Int} \rightarrow \text{Int}$

$\text{suma } x \ y = x + y$

suma es una función que toma dos enteros y retorna otro entero.

en lugar de

$\text{suma} :: \text{Int} \rightarrow (\text{Int} \rightarrow \text{Int})$

$(\text{suma } x) \ y = x + y$

suma es una función que toma un entero y devuelve una función, la cual toma un entero y devuelve otro entero.

Curricación

- ♦ Ventajas.

- ♦ Mayor expresividad

- $\text{derive} :: (\text{Float} \rightarrow \text{Float}) \rightarrow (\text{Float} \rightarrow \text{Float})$

- $\text{derive } f \ x = (f \ (x+h) - f \ x) / h \quad \text{where } h = 0.0001$

- ♦ Aplicación parcial

- $\text{derive } f \quad (= \backslash x \rightarrow (f \ (x+h) - f \ x) / h)$

- ♦ Modularidad para tratamiento de código

- ♦ Al inferir tipos

- ♦ Al transformar programas

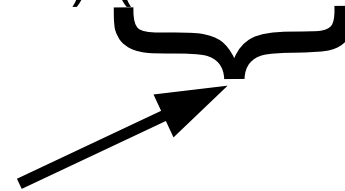
Aplicación Parcial

- ◆ Definir un función que calcule la derivada n-ésima de una función

`deriveN :: Int -> (Float -> Float) -> (Float -> Float)`

`deriveN 0 f = f`

`deriveN n f = deriveN (n-1) (derive f)`



Aplicación parcial de derive.

- ◆ ¿Cómo lo haría con `derive'`?

Expresividad

- ◆ Definir un función que calcule la aplicación n veces de una función

$\text{many} :: \text{Int} \rightarrow (\text{a} \rightarrow \text{a}) \rightarrow (\text{a} \rightarrow \text{a})$

$\text{many } 0 \text{ f } x = x$

$\text{many } n \text{ f } x = \text{many } (n-1) \text{ f } (\text{f } x)$

- ◆ Se pueden definir muchas ideas ya vistas...

$\text{twice} = \text{many } 2$

$\text{deriveN } n = \text{many } n \text{ derive}$

Curricación

- ◆ Decir que algo está currificado es una CUESTIÓN DE INTERPRETACIÓN

`movePoint :: (Int, Int) -> (Int, Int)`

`movePoint (x,y) = (x+1,y+1)`

`distance :: (Int, Int) -> Int`

`distance (x,y) = sqrt (sqr x + sqr y)`

- ◆ ¿Están currificadas? ¿Por qué?

Resumen

- ◆ Currificación.
- ◆ Ventajas: aumento de expresividad.

