PROGRAMACIÓN FUNCIONAL

Modelo Funcional: Currificación

Currificación

- 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
- (ABUSANDO DEL LENGUAJE) Función de alto orden:
  - una función que recibe otra función como argumento, o la retorna como resultado

Funciones como valores

- Ejemplo

```
compose (f,g) = h  where h x = f (g x)
sqr x = x*x
twice f = g   where g x = f (f x)
aLaCuarta = compose (sqr,sqr)
aLaOctava = compose (sqr,aLaCuarta)
fs = [ sqr, aLaCuarta, aLaOctava, twice sqr ]
```

aLaCuarta 2 $\rightarrow$?

- ¿Será cierto que aLaCuarta = twice sqr?

Ejercicio

- Consigna

  Usted trabajó en primer año con al menos dos funciones de alto orden. Intente identificar por lo menos una de ellas, junto con un ejemplo de uso.

- Características
  - individual
  - tiempo: 5 minutos

Aplicación del alto orden

- Considere las siguientes definiciones

```
suma' :: ??
suma' (x,y) = x+y

suma :: ??
suma x = f   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 retornan la suma de dos enteros:
    suma' (x,y) = (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:
    succ = suma 1

Currificación

- 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) -> c
f' (x,y) = e
```

  siempre se puede escribir

```
f :: a -> (b -> c)
(f x) y = e
```

Currificación - Sintaxis

- ¿Cómo escribimos una función currificada y su aplicación?
- Considerar las siguientes definiciones

```
twice :: (Int->Int) -> (Int -> Int)
twice1 f = g where g x = f (f x)
twice2 f = \x -> f (f x)
(twice3 f) x = f (f x)
```

- ¿Son equivalentes? ¿Cuál es preferible? ¿Por qué?

Currificación

- ¿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 :: Int -> (Int ->Int)
suma x y = x+y                 (suma x) y = x+y
```

Currificación

- Por abuso de lenguaje

  ```
  suma :: Int ->Int ->Int
  suma x y = x+y
  ```

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

en lugar de

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

Currificación

- Ventajas.
  - Mayor expresividad

```
derive :: (Int -> Int) -> (Int -> Int)
derive f x = (f (x+h) - f x) / h   where h = 0.0001
```

  - Aplicación parcial

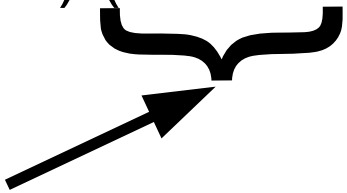
```
derive f          ( = \x -> (f (x+h) - f x) / h )
```

  - 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 -> (Int -> Int) -> (Int -> Int)
deriveN 0 f = f
deriveN n f = deriveN (n-1) (derive f)
```



Aplicación parcial de derive.

- ¿Cómo lo haría con derive'?

Currificación

- 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

- Asignación de tipos. Sistema de tipos.
- Polimorfismo.
- Currificación.

Ejercicio: respuesta

- Integral en un intervalo $[a,+\infty]$

  $$\text{int } f = \backslash a \text{ -> } \int_a^{+\infty} f(x)\, dx$$

  Toma la función f y devuelve otra función que dado a, calcula el área bajo la curva dada por f en el intervalo $[a,+\infty]$.

- Derivada genérica

  $$\text{deriv } f = \backslash x \text{ -> } f'(x)$$

  Toma la función f y devuelve otra función que dado un punto x, calcula la pendiente de la recta tangente a f en x.

Currificación - Sintaxis

- ¿Cómo escribimos una función currificada y su aplicación?
- Considerar las siguientes definiciones

```
twice :: (Int->Int) -> (Int -> Int)
twice1 f = g where g x = f (f x)
twice2 f = \x -> f (f x)
(twice3 f) x = f (f x)
```

- ¿Son equivalentes? ¿Cuál es preferible? ¿Por qué?