

# ESTRUCTURA DE DATOS

TEMA 1  
02/10/2017

Estructura de una función

$$\left\{ \begin{array}{l} \text{factorial} :: \text{Integer} \xrightarrow{\text{Entra un...}} \text{Integer} \xrightarrow{\text{Sale un...}} \\ \text{factorial } 0 = 1 \\ \text{factorial } n \mid n > 0 = n * \text{factorial } (n-1) \end{array} \right\} \begin{array}{l} \text{Cabecera} \\ \text{Cuerpo} \end{array}$$

Un operador con paréntesis es una función:

Prelude> 3 + 2 = (+) 3 2 = 5

Puede expresarse una condición mediante la estructura "if condición then acción1 else acción2", pero la estructura no puede emplearse como sentencia, ya que siempre debe llevar "(...) else acción2".

Como alternativa, pueden usarse guardas "!" para aplicar una sentencia:

$\text{factorial } n \mid n > 0 = n * \text{factorial } (n-1)$

Si  $n > 0$ , entonces  $n * \text{factorial } (n-1)$ .

Si no, ignora la línea y pasa a la siguiente.

Ejemplo (pizarra):

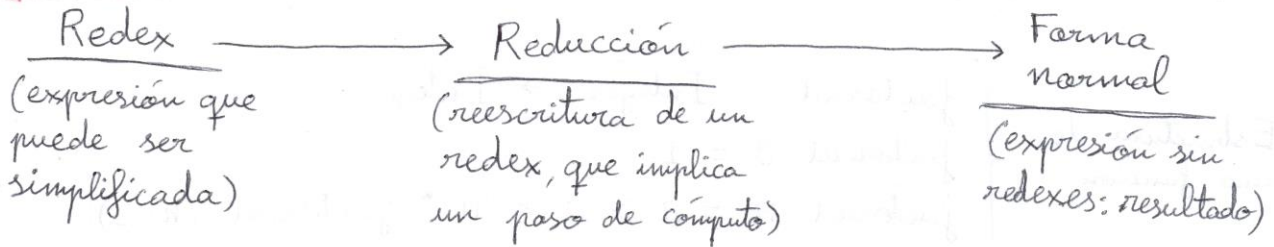
"Función en el prelude que devuelva el número de 0s del factorial de 1000."

Prelude> takeWhile (== '0').reverse.show \$ factorial 1000

Construye un String mientras que se cumpla la condición, recorriendo Char a Char el String argumento.

Invierte su String argumento.

Convierte su argumento en un String.



Se dice que un método es normalizante cuando es seguro que llegue a la forma normal.

- El orden aplicativo no es normalizante.
- El orden normal sí es normalizante.

Teorema de la estandarización: Resolver un problema por varios métodos distintos siempre lleva a la forma normal.

Las operaciones aritméticas primitivas (+, -, \*, /) requieren que sus argumentos sean conocidos, por lo que a veces no es un redex.

La evaluación perezosa (lazy) evalúa en orden normal hasta que encuentra un argumento complejo, que lo evalúa en orden aplicativo.

$$\text{twice } (10+2) = (10+2) + (10+2) = 12 + 12 = 24$$

$$\begin{array}{ccc} \underbrace{\phantom{a}} & \underbrace{\phantom{a}} & \underbrace{\phantom{a}} \\ a & a & a \\ \downarrow & \downarrow & \downarrow \\ a = (10+2) = 12 & 12 & 12 \end{array}$$

Una tupla es un tipo de variable.  $(\text{tipo } a, \text{tipo } b)$  <sup>Tupla</sup>

Para hacer tipos genéricos, se usan variables de tipo en la cabecera de la función (letras minúsculas).

Función polimórfica restringida pura: funciona de forma genérica para cualquier tipo de variable.

Función polimórfica restringida sobrecargada: funciona de forma diferente según el tipo de variable.

Se dice que un tipo está habitado, si es posible crear una función con dicho tipo. Además, si un tipo tiene habitante, este cumple ser una tautología.

$$f :: \underbrace{a \rightarrow (a \rightarrow b)}_{\text{función (variable)}} \rightarrow \underbrace{b}_{\text{resultado}} \quad \Rightarrow \quad f \times g = g \times \underbrace{f}_{\underbrace{a \rightarrow b}_b}$$

$$f :: a \rightarrow (a \rightarrow c) \rightarrow b \quad \Rightarrow \quad f \times g = \text{ERROR (no es posible)}$$

$$f :: a \rightarrow b \rightarrow ((a, b) \rightarrow c) \rightarrow c \quad \Rightarrow \quad f \times g = g(x, y)$$

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

$$f :: a \rightarrow (a \rightarrow b) \rightarrow a \rightarrow b \rightarrow c \quad \Rightarrow \quad f \times g \times h = h \times (g \times)$$

$$f \times g \times h = h \left( \underbrace{\underbrace{(g \times)}_b \underbrace{x}_a}_c \right) \quad \Rightarrow \quad f :: a \rightarrow (a \rightarrow b) \rightarrow (b \rightarrow a \rightarrow c) \rightarrow c$$

Ejemplo de función polimórfica restringida:

$$(+), (-), (*) :: \underbrace{(\text{Num } a)}_{\text{restricción}} \Rightarrow a \rightarrow a \rightarrow a$$

Sobrecargada, ya que para cada operador la función es diferente (varía según el tipo del argumento).

$$f \times y = \underbrace{x == y}_{\text{Devuelve un Bool}} \xrightarrow{\text{cTipo?}} f :: \underbrace{\text{Eq } a}_{\substack{\text{Las variables deben} \\ \text{ser igualables, su} \\ \text{tipo debe contemplar} \\ \text{la igualdad}}} \Rightarrow a \rightarrow a \rightarrow \text{Bool}$$

Dicho tipo debe contemplar el orden

Deben ser del mismo tipo

$$g \times y \mid x > y = z + 1 \xrightarrow{\text{cTipo?}} g :: (\text{Ord } a, \text{Num } b) \Rightarrow a \rightarrow a \rightarrow b \rightarrow b$$

| otherwise =  $2 * z$

Los valores deben ser del tipo de  $z$  ← Deben ser números

error :: String → a ⇒ error "Fuera Dominio"

Función parcial: aquella que no cubre todos los valores de un dominio.

En Haskell pueden declararse variables y funciones de forma local (las variables son funciones de 0 argumentos).

Evaluar con ecuaciones:

Ejecución ↓

```
fact :: (...)
fact 0 = 1
fact x = x * fact (x-1)
```

Patrón: función con un dato dado. En la ejecución, se evalúa si el argumento escrito es el dato dado.



## Definición del operador "&&":

$(\&\&) :: \text{Bool} \rightarrow \text{Bool} \rightarrow \text{Bool}$

$\text{True} \&\& x = x$   
 $\text{False} \&\& \_ = \text{False}$

$$\begin{cases} \text{True} \&\& \text{True} = \text{True} \\ \text{True} \&\& \text{False} = \text{False} \\ \text{False} \&\& \text{True} = \text{False} \\ \text{False} \&\& \text{False} = \text{False} \end{cases}$$

$(\parallel) :: \text{Bool} \rightarrow \text{Bool} \rightarrow \text{Bool}$

$\text{True} \parallel \_ = \text{True}$

$\text{False} \parallel x = x$

## Declarar un operador:

$\text{infixL}^{(\text{iz})}$       Valor de      Símbolo del  
 $\text{infixR}^{(\text{der})}$       + preferencia      + operador  
 $\text{infix}^{(\text{null})}$       (tabla)      (no empezar por ".")

Ejemplo:  $\text{infix } 4 \ (\sim =)$   
 $(\dots) \sim = (\dots) = (\dots)$   
 $(\dots) \sim = (\dots) = (\dots)$

Al crear una función, Haskell infiere su tipo. Si el tipo que tú le asignas es el mismo o un caso particular del tipo inferido por Haskell, el tipo es admitido.

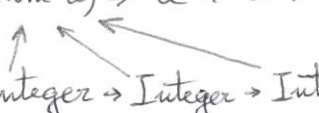
Ejemplo:

Para " $f \ x \ y = x + y$ " tú indicas que el tipo es " $\text{Integer} \rightarrow \text{Integer} \rightarrow \text{Integer}$ ". Haskell infiere el tipo " $(\text{Num } a) \Rightarrow a \rightarrow a \rightarrow a$ ".

Como  $\text{Integer} \in \text{Num}$ , el tipo " $\text{Integer} \rightarrow \text{Integer} \rightarrow \text{Integer}$ " es aceptado.

$f :: (\text{Num } a) \Rightarrow a \rightarrow a \rightarrow a$

$f :: \text{Integer} \rightarrow \text{Integer} \rightarrow \text{Integer}$



10/10/2017

$$f :: (\text{Ord } a) \Rightarrow a \rightarrow a \rightarrow a$$

Contexto  $\uparrow$   
 → Actúa como  
 restricción de  
 tipos

Un contexto de tipo solo se emplea para tipos genéricos, ya que los tipos concretos (Int, Integer...) ya tienen los contextos especificados.

16/10/2017

## Corrección del Cuestionario

1

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

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

2

$$\text{función } f \times y = f \ y \times$$

3

$$f :: (\text{Eq } a, \text{Ord } b) \Rightarrow a \rightarrow a \rightarrow b \rightarrow b \rightarrow \text{Bool}$$

4

Bien

5

Bien