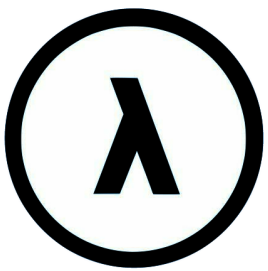
Funciones de Orden Superior

Ignacio Ballesteros
Luis Eduardo Bueso

https://github.com/edububa/haskell_course

6 de noviembre de 2017



Índice

Introducción

Curried functions

Las funciones en Haskell reciben **un único argumento**.

```
Prelude> max 4 5
5
Prelude> (max 4) 5
5
```

Si observamos el tipo de max:

```
max :: (Ord a) => a -> a -> a
```

que es equivalente a:

```
max :: (Ord a) => a -> (a -> a)
```

Introducción

Curried functions

Las siguientes funciones son equivalentes:

```haskell
multThree :: (Num a) => a -> a -> a -> a
multThree x y z = x * y * z

multThree' :: (Num a) => a -> (a -> (a -> a))
multThree' x y z = x * y * z
```

Entonces... ¿Qué sucede cuando llamamos a una función con menos argumentos de los que deberíamos?

```
*Main> mult2y4 = multThree 2 4
*Main> mult2y4 3
24
*Main>
```

Introducción

Definición

Una **función de orden superior** es aquella que recibe otra función como argumento y/o devuelve otra función como resultado.

Esto es posible gracias a que las funciones son **ciudadanos de primera clase**

Ejemplos:

```haskell
applyTwice :: (a -> a) -> a -> a
applyTwice f x = f (f x)

flip :: (a -> b -> c) -> (b -> a -> c)
flip f = g
  where g x y = f y x
```