Haskell – Guia del laboratori 2

Passar funcions i condicions com a paràmetre

En Haskell és molt senzill passar funcions i condicions com a paràmetre, doncs es fa de la mateixa manera que es passa per paràmetre qualsevol valor.

A continuació es mostra un exemple de funció que a partir d'una condició, dues operacions i dues llistes que tinguin el mateix nombre d'elements, aplica la primera operació a les parelles que compleixen la condició i aplica la segona operació a les parelles que no la compleixen:

```
combina _ _ _ [] _ = []
combina _ _ _ _ [] = []
combina cond f1 f2 (x:rest1) (y:rest2) = (if (cond x y) then (f1 x y) else (f2 x y)) : combina cond f1 f2 rest1 rest2
```

Nota:

- Es pot utilitzar la barra baixa (_) per indicar que el valor d'un paràmetre d'entrada ens és indiferent

La capçalera d'una funció que rep condicions i/o funcions com a paràmetre inclourà un parèntesi amb els detalls de cada funció i/o condició que rebi. Aquests parèntesis han de detallar el tipus i ordre dels paràmetre d'entrada que espera la funció o condició, així com el tipus de la sortida.

Per a la funció "combina" vista anteriorment, la capçalera seria la següent:

```
combina::(Int -> Int -> Bool) -> (Int -> Int -> Int) -> (Int -> Int -> Int) -> [Int] -> [Int] -> [Int]
```

condició amb dos enters d'entrada i sortida booleana

funcions que operen dos enters i retornen un altre enter

D'acord a la capçalera de la funció, les següents crides serien vàlides:

Crida	Sortida esperada
combina (>) (+) (*) [5,2,3] [4,5,6]	[9,10,18]
combina (>) (-) (\x y -> y-x) [2,4,6,8] [7,5,3,1]	[5,1,3,7]
combina (\x y -> x>y) (\x y -> x-y) (\x y -> y-x) [2,4,6,8] [7,5,3,1]	[5,1,3,7]

Ús de les funcions map, filter i foldr

Primerament veurem el propòsit de les funcions, per utilitzar-les posteriorment.

Funció map

Aplica una operació a cada element d'una llista

La seva capçalera és la següent:

map :: (a->b) -> [a] -> [b]

Exemples de crides:

Crida	Sortida esperada
map (*5) [1,2,3]	[5,10,15]
map (>5) [1,2,3]	[False,False,False]
map (++" a tots") ["hola","adéu"]	[“hola a tots”, “adéu a tots”]

Funció filter

Donada una llista, retorna només aquells elements que compleixen una determinada condició

La seva capçalera és la següent:

filter :: (a->Bool) -> [a] -> [a]

Exemples de crides:

Crida	Sortida esperada
filter (>5) [1,2,3,4,5,6]	[6]
filter (=="hola") ["hola","adéu","hola"]	[“hola,”hola”]

Funció foldr

Combina entre elles els elements d'una llista, mitjançant una operació determinada i començant per l'element neutre especificat.

La seva capçalera és la següent:

foldr :: (a->b->b) -> b -> [a] -> b

Exemples de crides:

Crida	Sortida esperada
foldr (+) 0 [1,2,3]	6
foldr (++) [] ["hola"," que"," tal"]	“hola que tal”

Les tres funcions són molt útils per ajudar-nos a implementar-ne de noves.

El següent exemple és una funció que aprofita les tres estudiades per tal de, donada una llista de nombres enters positius i negatius, retornar el valor absolut més gran:

```
abs_maxi::[Int]->Int
abs_maxi lista=foldr (max) (-1000) (map (abs) lista)
```

Funcions parcialment aplicades

Una funció qualsevol pot utilitzar-se, com ja hem vist, per ajudar a construir una altra. Quan es realitza aquest aprofitament de funcions, podem estalviar-nos d'especificar alguns paràmetres, com als següent exemples:

multiplica x y = x*y

Aplicació parcial de la funció multiplica	Explicació
multiplicaPer2 = multiplica 2	El paràmetre enter que rebrà la funció "multiplicaPer2" no cal escriure'l i per defecte s'entén que aquest es passarà a la funció multiplica com a l'operand que li falta
concatenarLlista = foldr (++) []	El paràmetre de tipus llista que rebrà la funció "concatenarLlista" no cal escriure'l i per defecte s'entén que aquesta es passarà a la funció "foldr" com a l'operant que li falta
sumList = foldr (+) 0	El paràmetre de tipus llista que rebrà la funció "sumList" no cal escriure'l i per defecte s'entén que aquesta es passarà a la funció "foldr" com a l'operant que li falta