Currying Research

- Santiago Caballero Manzaneda

Una función "**curried**" toma sus argumentos de manera parcial. Es decir, en lugar de recibir todos sus argumentos a la vez, acepta un solo argumento a la vez y devuelve una nueva función que espera el siguiente argumento. Esta técnica lleva el nombre del lógico y matemático Haskell Curry. En Haskell, todas las funciones son "curried" de manera predeterminada.

Una función curry es una que retorna progresivamente una función más específica por cada uno de los argumentos dados hasta que ya no sean necesarios más parámetros. Una función parcialmente aplicada, por otra parte, es una función que es "parcialmente" ejecutada y está lista para su inmediata ejecución una vez dado el resto de los parámetros esperados.

- **Examples**:

```
-- Example 1
getSqrtOfAnInt :: Integer -> Integer
getSqrtOfAnInt = floor . sqrt  . fromInteger

-- Example 2
setPrintMethod ::(String -> IO ()) -> Int -> Int -> IO ()
setPrintMethod printMethod x y = printMethod "Printing the value

printAddition ::  Int -> Int -> IO ()
printAddition = setPrintMethod putStrLn

-- Example 3
getShapeArea :: String -> (Int -> Int -> Int)
getShapeArea "rectangle" x y = x * y
getShapeArea "triangle" x y = div (x * y) 2
getShapeArea _ _ _ = 0

-- Example 4
concatenate :: [a] -> [a] -> [a]
concatenate xs ys = xs ++ ys

-- Example 5
filterList :: (a -> Bool) -> [a] -> [a]
filterList = filter
```

En pocas palabras, *currying* es una técnica que traduce la evaluación de una función que toma múltiples argumentos en una evaluación de una secuencia de funciones, cada una de funciones de la secuencia espera un único argumento. *Currying* está relacionado con el concepto de aplicacion parcial, pero no es lo mismo.