

Funcional 2:

Aplicación Parcial y Composición

Aplicación Parcial

Nota previa Hay aplicación parcial cuando a una función la evaluamos con menor cantidad de argumentos con la que está definida.

Veamos un Ejemplo:

La función `(*)` recibe dos enteros como argumento y devuelve el resultado de la multiplicación.

```
(*) :: Int -> Int -> Int
```

Si queremos aplicar la función `(*)` con un solo argumento nos quedaría:

`(2*)` Esto nos devuelve una función que es `(Int->Int)`, recibe un entero y devuelve otro entero.

Si esta última función la aplico sobre un entero, nos devuelve otro entero como resultado Ej:

```
Main> (2*) 9 18
```

Ejercicios

1. Definir una función **siguiente**, que al invocarla con un número cualquiera me devuelve el resultado de sumar a ese número el 1.

```
Main> siguiente 3  
4
```

2. Definir la función **mitad** que al invocarla con un número cualquiera me devuelve la mitad de dicho número, ej:

```
Main> mitad 5  
2.5
```

3. Definir una función **inversa**, que invocando a la función con un número cualquiera me devuelva su inversa.

```
Main> inversa 4  
0.25  
Main> inversa 0.5  
2.0
```

4. Definir una función **triple**, que invocando a la función con un número cualquiera me devuelva el triple del mismo.

```
Main> triple 5  
15
```

5. Definir una función **esNumeroPositivo**, que invocando a la función con un número cualquiera me devuelva `true` si el número es positivo y `false` en caso contrario.

```
Main> esNumeroPositivo (-5)
```

```
False
Main> esNumeroPositivo 0.99
True
```

Composición

Nota previa

Se base en el concepto matemático de composición de funciones.

$$g(f(x)) = (g \circ f) x$$

La imagen de $f(x)$ tiene que coincidir con el dominio de $g(x)$

Acá la notación es $(g . f) x$. Veamos un Ejemplo:

```
g n = even n
f n = 3 + n
```

```
Main> (g . f) 4
False
Main> (g . f) 3
True
```

Ejercicios

6. Resolver la función del ejercicio 2 de la guía anterior **esMultiploDe/2**, utilizando aplicación parcial y composición.
7. Resolver la función del ejercicio 5 de la guía anterior **esBisiesto/1**, utilizando aplicación parcial y composición.

8. Resolver la función **inversaRaizCuadrada/1**, que da un número n devolver la inversa su raíz cuadrada.

```
Main> inversaRaizCuadrada 4
0.5
```

Nota: Resolverlo utilizando la función inversa Ej. 2.3, sqrt y composición.

9. Definir una función **incrementMCuadradoN**, que invocándola con 2 números m y n , incrementa un valor m al cuadrado de n por Ej:

```
Main> incrementMCuadradoN 3 2
11
```

Incrementa 2 al cuadrado de 3, da como resultado 11. Nota: Resolverlo utilizando aplicación parcial y composición.

10. Definir una función **esResultadoPar/2**, que invocándola con número n y otro m , devuelve true si el resultado de elevar n a m es par.

```
Main> esResultadoPar 2 5
```

True

Main> esResultadoPar 3 2

False

Nota Obvia: Resolverlo utilizando aplicación parcial y composición.

11. asdasdsa