Estructuras Discretas

Introducción a Haskell

Universidad Nacional Autónoma de México
Facultad de Ciencias

Acerca de Haskell

- Programación puramente funcional.
- Transparencia referencial.
- Evaluación perezosa.
- Tipado estático.

Tipos de datos.

- Números enteros (Int).
- Números con punto flotante (Float).
- Booleanos (Bool).
- Caracteres (Char).
- Listas
- Tuplas.
- Cadenas de texto (String / [Char]).

Operaciones aritméticas.

```
ghci> succ 0
1
ghci> pred 1
0
ghci> 10 + 4
14
ghci> 10 - 4
6
ghci> 10 * 4
40
ghci> 10 / 4
2.5
ghci> div 10 4
2
ghci> mod 10 2
0
ghci> 10 ** 2
100.0
```

Operaciones Booleanas.

```
ghci> True
True
ghci> False
False
ghci> True || False
True
ghci> True && False
False
ghci> not False
True
ghci>
```

```
ghci> 10 == 10
True
ghci> 10 /= 10
False
ghci> "hola" == "hola"
True
ghci> "hola" /= "hola"
False
ghci> 10 > 0
True
ghci> 10 < 0
False
ghci> 10 >= 10
True
ghci> 10 <= 10
True
```

Listas

En Haskell una lista puede ser:

- Una lista vacía representada por dos corchetes, [ ]
- Una lista compuesta por una cabeza *x* seguida de otra lista *xs*

```
ghci> []
[]
ghci> 2:[]
[2]
ghci> 'H':"ola"
"Hola"
ghci> 0:[1, 2, 3]
[0,1,2,3]
ghci> 1:2:3:4:[]
[1,2,3,4]
```

Construcción de listas.

Para construir una lista se puede utilizar el operador ‘:’ o se puede construir tal cual la lista, por ejemplo:

```
ghci> 1:2:3:4:[]
[1,2,3,4]
ghci> [1, 2, 3, 4]
[1,2,3,4]
```

Rangos

```
ghci> [1..20]
[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]
ghci> ['a'..'z']
"abcdefghijklmnopqrstuvwxyz"
ghci> [2, 4..20]
[2,4,6,8,10,12,14,16,18,20]
```

En Haskell se cuenta con el operador ‘..’ el cual se utiliza cuando se quiere crear una lista de un tamaño en específico, sin embargo, esta tiene que ser de elementos que puedan ser enumerados.

Listas por comprensión

Otra forma de definir listas en Haskell es utilizando una notación muy parecida a la notación con la cual se definen conjuntos en álgebra. Por ejemplo, si se tiene la siguiente definición:

$S = \{2x \mid x \in N, x \leq 10\}$

Esta se puede traducir a Haskell de la siguiente manera:

```
ghci> [2 * x | x <- [1..10]]
[2,4,6,8,10,12,14,16,18,20]
```

Funciones

Una función en Haskell está conformada por dos partes:

- La firma de la función, donde se define su nombre y el tipo de sus parámetros y retorno.
- La definición de la función.

```haskell
suma :: Int -> Int -> Int      -- Firma de la función
suma n m = n + m               -- Definición de la función
```

Condicionales

Haskell cuenta con dos principales formas de condicionales para controlar el flujo de un programa:

- La sentencia *if*.
- Y las llamadas *guardas*, las cuales se utilizan si se tiene más de una condición a verificar

```haskell
lucky :: Int → String
lucky n = if n == 7 then "¡El siete de la suerte!"
                    else "!No es tu dia de suerte!"
```

```haskell
lucky :: Int → String
lucky n
  | n == 7 = "¡El siete de la suerte!"
  | otherwise = "!No es tu dia de suerte!"
```

Caza de patrones (*Pattern matching*)

La caza de patrones se utiliza de una forma similar a los condicionales. Con esta técnica se puede verificar que un parámetro de una función cumpla con ciertas características para poder realizar ciertas acciones, también mediante este se pueden desestructurar ciertos tipos de datos.

```
1  lucky :: Int → String
2  lucky 7 = "¡El siete de la suerte!"
3  lucky n = "!No es tu dia de suerte!"
```