TECNOLÓGICO NACIONAL DE MÉXICO
Campus Colima

Departamento de Sistemas y Computación

INGENIERÍA EN SISTEMAS COMPUTACIONALES

“Programación Lógica y Funcional”

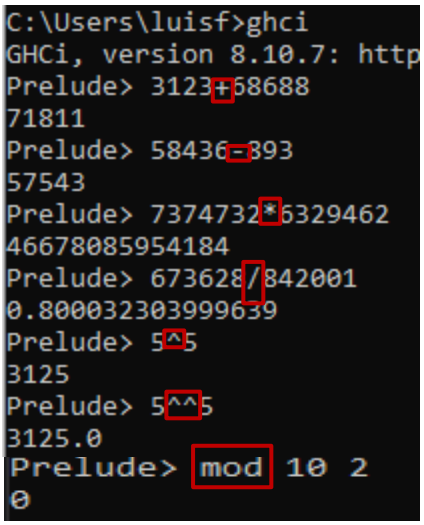
A2.4 Primeros pasos en Haskell

Que presenta:

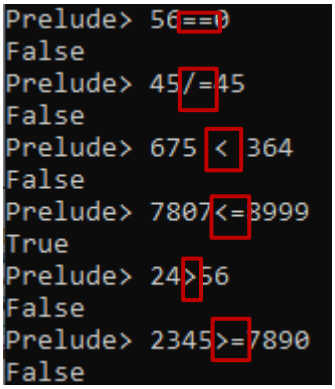
Barragán Flores Luis Francisco

Villa de Álvarez, Colima, México a 22 de marzo de 2022

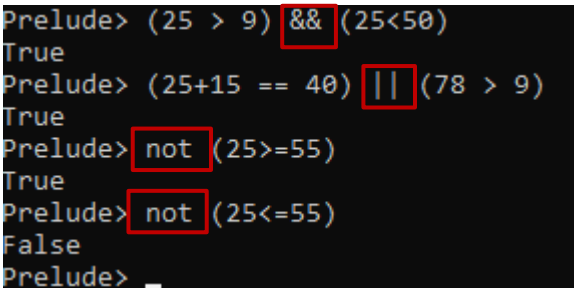
1. Probar los operadores para cada uno de los siguientes tipos:
    1. Aritméticos: *, -, +, /, ^, mod



    2. Relacionales: ==, /=, <, <=, >, >=

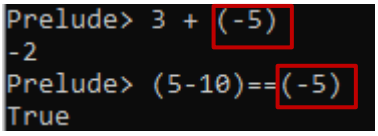


    3. Lógicos: &&, ||, not



2. ¿Qué debemos hacer cuando se requiere realizar una operación con un número negativo?

Lo que debemos de hacer es cerrar entre paréntesis el número o los números que se tengan.



3. Identificar y documentar las funciones básicas predefinidas para Haskell
   - **x + y** es la suma de x e y.
   - **x - y** es la resta de x e y.
   - **x / y** es el cociente de x entre y.
   - **x^y** es x elevado a y.
   - **x == y** se verifica si x es igual a y.
   - **x /= y** se verifica si x es distinto de y.
   - **x < y** se verifica si x es menor que y.
   - **x <= y** se verifica si x es menor o igual que y.
   - **x > y** se verifica si x es mayor que y.
   - **x >= y** se verifica si x es mayor o igual que y.
   - **x && y** es la conjunción de x e y.
   - **x || y** es la disyunción de x e y.
   - **x:ys** es la lista obtenida añadiendo x al principio de ys.
   - **xs ++ ys** es la concatenación de xs e ys.
   - **xs !! n** es el elemento n–ésimo de xs.
   - **abs** x es el valor absoluto de x.
   - **and** xs es la conjunción de la lista de booleanos xs.
   - **drop** n xs borra los n primeros elementos de xs.
   - **elem** x ys se verifica si x pertenece a ys.
   - **even** x se verifica si x es par.
   - **head** xs es el primer elemento de la lista xs.
   - **init** xs es la lista obtenida eliminando el último elemento de xs.
   - **last** xs es el último elemento de la lista xs.
   - **length** xs es el número de elementos de la lista xs.
   - **map** f xs es la lista obtenida aplicado f a cada elemento de xs.
   - **maximum** xs es el máximo elemento de la lista xs.
   - **minimum** xs es el mínimo elemento de la lista xs.
   - **mod** x y es el resto de x entre y.
   - **repeat** x es la lista infinita [x, x, x, ...].
   - **tail** xs es la lista obtenida eliminando el primer elemento de xs.
   - **take** n xs es la lista de los n primeros elementos de xs

4. Realizar las siguientes acciones para listas:
    1. Generar dos listas de números

```
C:\Users\luisf>ghci
GHCi, version 8.10.7: https://www.haskell.org/ghc/  :? for help
Prelude> let lista1 = [1,2,3,4,5]
Prelude> let lista2 = [6,7,8,9,0]
Prelude> lista1
[1,2,3,4,5]
Prelude> lista2
[6,7,8,9,0]
Prelude>
```

2. Generar dos listas de caracteres

```
Prelude> let lista1 = ['A','B','C']
Prelude> lista1
"ABC"
Prelude> let lista2 = ['a','b','c']
Prelude> lista2
"abc"
```

3. Concatenar dos listas

```
Prelude> lista1++lista2
"ABCabc"
Prelude> "Hola " ++ "Mundo"
"Hola Mundo"
```

4. Aplicar el operador **:** en una lista, ¿qué acción realiza?

Agrega el valor que indiquemos a la lista en la posición número cero de esta.

```
Prelude> 0: [1..10]
[0,1,2,3,4,5,6,7,8,9,10]
```

5. Describir que representa cada una de las siguientes definiciones: [], [[]] y [[],[],[]]

   [ ] = Representa una lista vacía.

```
Prelude> let lista = []
Prelude> lista
[]
```

   [[ ]] = Representa una lista que contiene un elemento (una lista vacía).

```
Prelude> let lista = [[]]
Prelude> lista
[[]]
```

[[ ] , [ ] , [ ]] = Representa una lista que contiene tres elementos

```
Prelude> let lista = [[],[],[]]
Prelude> lista
[[],[],[]]
```

6. Para qué se utiliza el operador **!!** en una lista

   Obtiene el elemento de la lista.

```
Prelude> [0..100]!!5
5
```

7. Definir una lista de listas y aplicar los operadores **:** y **!!**

```
Prelude> let lista = [1..50]
Prelude> lista
[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43
,44,45,46,47,48,49,50]
Prelude> 0: lista
[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,
43,44,45,46,47,48,49,50]
Prelude> lista !!5
6
```

8. Aplicar las siguientes funciones básicas de listas a una lista: **head, tail, last, init, length, null, reverse, take, drop, maximum, minimum, sum, product, elem**

```
Prelude> let lista = ['a','b','c','d']
Prelude> lista
"abcd"
Prelude> head lista
'a'
Prelude> tail lista
"bcd"
Prelude> last lista
'd'
Prelude> init lista
"abc"
Prelude> lenght lista

<interactive>:38:1: error:
    * Variable not in scope: lenght :: [Char] -> t
    * Perhaps you meant `length' (imported from Prelude)
Prelude> length lista
4
Prelude> null lista
False
Prelude> reverse lista
"dcba"
```

```
Prelude> take 3 lista
"abc"
Prelude> lista = [1..100]
Prelude> lista
[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43
,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83
,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99,100]
Prelude> take 5 lista
[1,2,3,4,5]
Prelude> drop 10 lista
[11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50
,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90
,91,92,93,94,95,96,97,98,99,100]
Prelude> maximun lista
```

```
Prelude> maximum [0,1111111,34,10,1,99999999]
99999999
Prelude> minimum [0,1111111,34,10,1,99999999]
0
Prelude> sum lista
5050
Prelude> product [1..10]
3628800
```

```
Prelude> 10 `elem` [1,5,7,0,10,99]
True
Prelude> 2 `elem` [1,5,7,9,10,100]
False
```

5. Definir tres listas diferentes utilizando Texas rangos.

```
Prelude> [1..100]
[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,4
,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,8
,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99,100]
```

```
Prelude> ['a'..'z']
"abcdefghijklmnopqrstuvwxyz"
```

```
Prelude> ['A'..'Z']
"ABCDEFGHIJKLMNOPQRSTUVWXYZ"
```

6. Generar una lista utilizando la función **cycle** y otra lista utilizando la función **repeat (replicate)**

```
Prelude> take 15 (cycle "Luis")
"LuisLuisLuisLui"
Prelude> take 15 (cycle [1..10])
[1,2,3,4,5,6,7,8,9,10,1,2,3,4,5]
```

```
Prelude> take 15 (repeat "Luis")
["Luis","Luis","Luis","Luis","Luis","Luis","Luis","Luis","Luis","Luis","Luis","Luis","Luis","Luis","Luis"]
Prelude> take 15 (repeat [1..5])
[[1,2,3,4,5],[1,2,3,4,5],[1,2,3,4,5],[1,2,3,4,5],[1,2,3,4,5],[1,2,3,4,5],[1,2,3,4,5],[1,2,3,4,5],[1,2,3,4,5],[1,2,3,4,5]
,[1,2,3,4,5],[1,2,3,4,5],[1,2,3,4,5],[1,2,3,4,5],[1,2,3,4,5]]
```

7. Definir cinco listas diferentes utilizando listas intensionales.

1.-

```
Prelude> [(x,y) | x <- [1,2], y <- [3,4,5]]
[(1,3),(1,4),(1,5),(2,3),(2,4),(2,5)]
```

2.-

```
Prelude> [(x,y) | x <- ['a','b'], y <- ['c','d','e']]
[('a','c'),('a','d'),('a','e'),('b','c'),('b','d'),('b','e')]
```

3.-

```
Prelude> [(x) | x <- [1..10], x>=5]
[5,6,7,8,9,10]
```

4.-

```
Prelude> [(x,y) | x <- ['a'..'d'], y <- ['e'..'h']]
[('a','e'),('a','f'),('a','g'),('a','h'),('b','e'),('b','f'),('b','g'),('b','h'),('c','
,('d','e'),('d','f'),('d','g'),('d','h')]
```

5.-

```
Prelude> let lista = [(x) | x <- [1..10], x<5]
Prelude> lista
[1,2,3,4]
```

Conclusión

Tras realizar este trabajo pude ver un poco más acerca del funcionamiento de haskell y pude comprender y entender de mejor manera como trabajar con el para programar, ya que como se muestra, vimos bastantes funciones básicas, como lo es la suma, resta, multiplicación, división y elevar un número al cuadro, hasta algunas de las formas para trabajar con las listas, como lo son el head, tail, last, init, length, null, reverse, take, drop, maximum, minimum, sum, product, elem, estas funciones básicas nos van a ser muy útiles para poder realizar nuestros proyectos en dicho lenguaje.

Referencias

[1] aprendehaskell.es, «Aprende haskell por el bien de todos,» aprendehaskell.es, [En línea]. Available: http://aprendehaskell.es/content/Empezando.html. [Último acceso: 22 03 2022].