Baltazar Coyotl Miguel Angel

Práctica número 3

Funciones (Tarea 3)

1° y 3° opción

3 de noviembre del 2021

Grupo 3CM2

Materia: Paradigmas de Programación

Indice.-

Introducción.

Al programar con Haskell es diferente que con C ya que es un lenguaje de programación declarativo donde el programador especifica lo que quiere hacer, en lugar de lidiar con el estado de los objetos. Es decir, las funciones estarían en un primer lugar y nos centraremos en expresiones que pueden ser asignadas a cualquier variable.

En la práctica número 3 se pide realizar el respectivo código en Haskell por lo cual es importante que este lenguaje es puramente funcional y fue creado por Haskell Brooks Curry por ello su nombre. Los programas escritos en Haskell se representan siempre como funciones matemáticas, pero estas funciones nunca tienen efectos secundarios ni derivados. De este modo, cada función utilizada siempre devuelve el mismo resultado con la misma entrada, y el estado del programa nunca cambia. Por esto, el valor de una expresión o el resultado de una función dependen exclusivamente de los parámetros de entrada en el momento. En Haskell no pueden hacerse construcciones de lenguaje imperativo para programar una secuencia de declaraciones.

Desarrollo

En la practica 3 se nos pide realizar dos funciones las cuales en este caso van a ser la primera que es sobre eliminar la primera ocurrencia del átomo a en una lista y la tercera para calcular el producto escalar de dos listas.

Primera función

En este caso se llamará rember nos pide eliminar la primera ocurrencia del átomo a en una lista:

rember 3 [1,2,3,4,5,6] = [1,2,4,5,6]

Por lo cual se define la función y se establece los parámetros que se le dará y el valor de retorno que tendrá

```
rember ::Eq a => a -> [a] -> [a]
```

Consecuentemente se establece un caso base con el cual la función dejara su recursividad

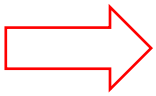
```
rember _ [] = []
```

Finalmente se completa la función.

```
rember a (y:ys) = if
                    y == a then ys
                  else
                    y : rember a ys
```

Para que al compilarlo nos muestre lo siguiente:

```
putStrLn "Primer ejercicio: "
putStrLn "3 [1,2,3,4,5,6] "
putStrLn "The answer is: "
print (rember 3 [1,2,3,4,5,6])
```



```
Primer ejercicio:
3 [1,2,3,4,5,6]
The answer is:
[1,2,4,5,6]
```

Tercer función.

Para la siguiente funcion llamada **dotProduct** se define los parametros y el tipo de retorno

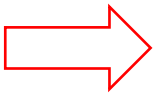
```
dotProduct :: Num b => [b] -> [b] -> b
```

Finalmente se ocupan condicionales para que la funcion funcione como se pide, esto se puede ver a continuacion:

```
dotProduct x y = foldl1 (+) (zipWith (*) x y)
```

Al ejecutar el codigo se tiene lo siguiente:

```
putStrLn "Third exercise "
putStrLn "[1,2,3,4] [5,6,7,8]"
putStrLn "The answer is: "
print (dotProduct [1,2,3,4] [5,6,7,8])
```



```
Third exercise
[1,2,3,4] [5,6,7,8]
The answer is:
70
```

Conclusión

Finalmente se logró identificar y aplicar lo aprendido sobre Haskell, utilizándolo para realizar ciertas funciones que son requeridas, así como se hizo con la primera función en la cual se pide eliminar la primera ocurrencia del átomo a en una lista, esto se logro con recursividad, la cual es una parte muy importante que se ocupa en su matoria en Haskell.

Bibliografía

(2019). Obtenido de EcuRed:

https://www.ecured.cu/LISP

Digital Guide. (9 de octubre de 2020). Obtenido de

https://www.ionos.mx/digitalguide/paginas-web/desarrollo-web/que-es-haskell/