INSTITUTO TECNOLOGICO DEL SUR DE GUANAJUATO

Programación lógica y funcional

Ejercicio: Kata FizzBuzz

Realizado por:

- Lizzeth Magaña Domínguez

Contenido

Funciones del programa

Función principal: fizzBuzz

La primera función que se realizo fue la de FizzBuzz, está será la función principal del programa. Esta función recibe un valor entero y de salida tendrá un String.

```
fizzBuzz :: Int -> String
fizzBuzz n
    | n `mod` 15 == 0 = "FizzBuzz!"
    | n `mod` 3 == 0 = "Fizz!"
    | n `mod` 5 == 0 = "Buzz!"
    | otherwise = numberToWords n
```

Como se puede ver en la imagen que se anexa, podemos ver como se emplea el uso del modulo para saber si un número leído cumple con alguna de las siguientes condiciones

- Es múltiplo de 3 y de 15: Regresa la palabra FizzBuzz
- Es múltiplo de 3: Regresa la palabra Fizz
- Es múltiplo de 5: Regresa la palabra Buzz
- No cumple con ninguna de las condiciones anteriores: Hace llamado a la siguiente función.

Función numberToWords

Esta función recibe un valor entero y regresa un String. Lo que hace es convertir un número entero en palabras. Esto lo realiza en base a lo siguiente:

- Si el número es igual a 0 se pone la palabra "Zero"
- Si el número es menor que 20 va a mandar llamar la función numIniciales donde se buscará su representación en una lista de palabras
- Si el número esta entre 20 y 99, divide el número en decenas y unidades. Luego, busca la representación de las decenas en la lista decenas y agrega la representación en palabras de las unidades de la función rest.
- Si el número está entre 100 y 999, divide el número en centenas y el resto de unidades. Luego busca la representación de las centenas en la lista numIniciales, agrega la palabra "Hundred" y llama a la función rest.
- Si el número está fuera de estos rangos, devuelve "Número fuera de rango".

```
numberToWords :: Int -> String
numberToWords n
    | n == 0 = "Zero"
    | n < 20 = numIniciales !! n
    | n < 100 = decenas !! (n `div` 10) ++ rest (n `mod` 10)
    | n < 1000 = numIniciales !! (n `div` 100) ++ " Hundred" ++ rest (n `mod` 100)
    | otherwise = "Numero fuera de rango"
```

Función rest

Esta función toma un número entero como entrada y devuelve un string. Esta función trabaja de la siguiente manera:

- Si el número es menor que 20, busca la representación en palabras en la lista numIniciales.
- De lo contrario, divide el número en decenas y unidades. Luego, busca la representación en palabras de las decenas en la lista de decenas y agrega un espacio seguido de la representación en palabras de las unidades desde la lista numIniciales.

```
rest :: Int -> String
rest n
    | n < 20 = numIniciales !! n
    | otherwise = decenas !! (n `div` 10) ++ " " ++ numIniciales !! (n `mod` 10)
```

Función numIniciales

Esta función contiene las representaciones de los números del 1 al 19 en palabras. Estos valores serán útiles para convertir los números en palabras con la función numerToWords

```
numIniciales :: [String]
numIniciales = ["", "One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine", "Ten",
                "Eleven", "Twelve", "Thirteen", "Fourteen", "Fifteen", "Sixteen", "Seventeen", "Eighteen", "Nineteen"]
```

Función decenas

Esta función contiene las representaciones de las decenas desde el 20 al 90 en palabras. Estos valores se utilizarán para construir las representaciones de los números entre 20 y 99 en la función numberToWords.

```
decenas :: [String]
decenas = ["", "", "Twenty ", "Thirty ", "Forty ", "Fifty ", "Sixty ", "Seventy ", "Eighty ", "Ninety "]
```

Evidencias de su funcionamiento

Primero debemos cargar el archivo.

```
ghci> :load FizzBuzz.hs
[1 of 1] Compiling FizzBuzz         ( FizzBuzz.hs, interpreted )
Ok, one module loaded.
ghci>
```

Ahora si a hacer algunos ejemplos haciendo uso de la función fizzBuzz

```
ghci> fizzBuzz 34
"Thirty Four"
```

```
ghci> fizzBuzz 50
"Buzz!"
```

```
ghci> fizzBuzz 39
"Fizz!"
```

```
ghci> fizzBuzz 60
"FizzBuzz!"
```

Código completo

```
-- Realizado por Lizzeth Magaña Domínguez

module FizzBuzz where

-- | Función que recibe un número entero y devuelve la cadena correspondiente

fizzBuzz :: Int -> String

fizzBuzz n

    | n `mod` 15 == 0 = "FizzBuzz!"

    | n `mod` 3 == 0 = "Fizz!"

    | n `mod` 5 == 0 = "Buzz!"

    | otherwise = numberToWords n

-- | Función para convertir números a palabras

numberToWords :: Int -> String

numberToWords n

    | n == 0 = "Zero"

    | n < 20 = numIniciales !! n

    | n < 100 = decenas !! (n `div` 10) ++ rest (n `mod` 10)

    | n < 1000 = numIniciales !! (n `div` 100) ++ " Hundred" ++ rest (n `mod` 100)

    | otherwise = "Numero fuera de rango"

-- | Función para convertir los numeros restantes a palabras

rest :: Int -> String

rest n

    | n < 20 = numIniciales !! n

    | otherwise = decenas !! (n `div` 10) ++ " " ++ numIniciales !! (n `mod` 10)

-- | Listas con las palabras para los números pequeños

numIniciales :: [String]

numIniciales = ["", "One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine", "Ten",

            "Eleven", "Twelve", "Thirteen", "Fourteen", "Fifteen", "Sixteen", "Seventeen", "Eighteen", "Nineteen"]

-- | Listas con las palabras para las decenas

decenas :: [String]

decenas = ["", "", "Twenty ", "Thirty ", "Forty ", "Fifty ", "Sixty ", "Seventy ", "Eighty ", "Ninety "]
```