INSTITUTO TECNOLÓGICO SUPERIOR DEL SUR DE GUANAJUATO

TAREA:

EVALUACIÓN: KATA FIZZBUZZ

Materia:

Programación Lógica y Funcional.

Semestre:
6°A

Ing. Sistemas Computacionales

Elaborada por:

Paola Montserrat Ruiz Carmen, S22120146
America Citlalli Lopez Lemus, S22120161

Profesor:

Gustavo Ivan Vega Olvera.

Uriangato, Gto. a 13 de enero del 2025

IMPLEMENTACIÓN DE FIZZBUZZ EN HASKELL

OBJETIVO

Desarrollar un programa en **Haskell** que determine si un número es primo y, en caso contrario, lo convierta a palabras en español. Se busca aplicar el manejo de estructuras condicionales y listas en Haskell.

Descripción del Programa

El programa implementa la función fizzbuzz, que recibe un número entero y evalúa lo siguiente:

- Si el número está fuera del rango permitido (0 a 1,000,000), devuelve un mensaje de error.
- ¡Si el número es primo, devuelve “FizzBuzz!".
- Si el número es 0, devuelve "cero".
- Si el número no es primo, lo convierte a palabras en español.

El programa usa varias funciones auxiliares para realizar la conversión de números a texto.

Explicación de las Funciones

fizzbuzz :: Int -> String

Esta función principal recibe un número entero y determina su salida según las reglas mencionadas.

- Si el número está fuera del rango permitido, devuelve un mensaje de error.
- Si es primo, devuelve "FizzBuzz!".
- Si es 0, devuelve "cero".
- En cualquier otro caso, llama a la función numero para obtener su representación en palabras.

esPrimo :: Int -> Bool

Determina si un número es primo.

- Si el número es menor que 2, devuelve False.
- Usa una comprensión de listas para verificar si el número es divisible por algún valor entre 2 y su raíz cuadrada.

- Devuelve True si no es divisible por ningún número dentro de ese rango.

numero :: Int -> String

Convierte un número en su representación en palabras en español.

- Para números menores a 30, usa la función unicos.
- Para números menores a 100, usa decenas y maneja casos especiales con "y".
- Para centenas, usa centenas.
- Para miles, descompone el número en miles y unidades.
- Maneja de forma especial el caso de "un millón".

unicos :: Int -> String

Convierte los números del 1 al 29 a palabras.

- Usa una lista de palabras predefinida y accede al índice correspondiente.

decenas :: Int -> String

Convierte las decenas (30, 40, 50, etc.) en palabras.

- Usa una lista de palabras y accede al índice correspondiente.

centenas :: Int -> String

Convierte las centenas (100, 200, 300, etc.) en palabras.

- Usa una lista predefinida para devolver la cadena correcta.

Código Fuente:

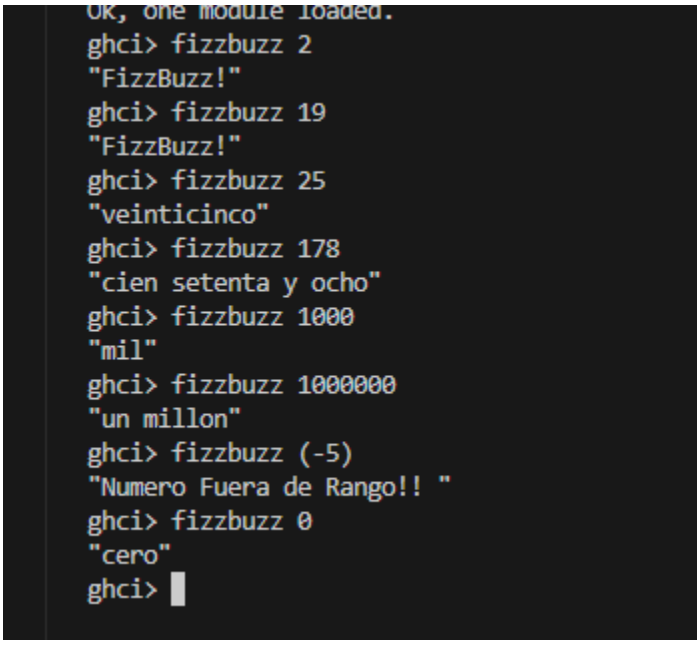
```
module FizzBuzz where

fizzbuzz :: Int -> String

fizzbuzz n

   | n < 0 || n > 1000000 = "Numero Fuera de Rango!! "

   | esPrimo n = "FizzBuzz!"

   | n == 0 = "cero"

   | otherwise = numero n


esPrimo :: Int -> Bool
```

```
esPrimo n
    | n < 2 = False
    | otherwise = all (\x -> n `mod` x /= 0) [2 .. floor (sqrt (fromIntegral n))]

numero :: Int -> String
numero n
    | n < 30 = unicos n
    | n < 100 && n `mod` 10 == 0 = decenas (n `div` 10)
    | n < 100 = decenas (n `div` 10) ++ " y " ++ unicos (n `mod` 10)
    | n < 1000 && n `mod` 100 == 0 = centenas (n `div` 100)
    | n < 1000 = centenas (n `div` 100) ++ " " ++ numero (n `mod` 100)
    | n == 1000 = "mil"
    | n < 1000000 && n `mod` 1000 == 0 = numero (n `div` 1000) ++ " mil"
    | n < 1000000 = numero (n `div` 1000) ++ " mil " ++ numero (n `mod` 1000)
    | otherwise = "un millon"

unicos :: Int -> String
unicos n
    | n > 0 && n < 30 =
    let answers = words ("uno dos tres cuatro cinco seis siete ocho nueve diez " ++
                        "once doce trece catorce quince dieciséis diecisiete dieciocho
diecinueve veinte " ++
                        "veintiuno veintidos veintitres veinticuatro veinticinco veinticeis
veinticiete veintiocho veintinueve")
    in answers !! (n-1)

decenas :: Int -> String
decenas n
    | n > 1 && n <= 9 =
```

```
        answers!!(n-3)
        where
            answers = words "treinta cuarenta cincuenta sesenta setenta ochenta noventa"

centenas :: Int -> String
centenas n
    | n == 1 = "cien"
    | n > 1 && n <= 9 =
        answers !! (n - 1)
        where
            answers = words "cien doscientos trescientos cuatrocientos quinientos seiscientos setecientos ochocientos novecientos"
```

Casos de Prueba

ENTRADA	SALIDA ESPERADA	EXPLICACIÓN
2	"FizzBuzz!"	2 es primo
19	"FizzBuzz!"	19 es primo
25	"veinticinco"	No es primo, convierte a texto
178	"ciento setenta y ocho"	No es primo
1000	"mil"	Caso especial
1000000	"un millon"	Caso especial
-5	"Numero Fuera de Rango!!"	Fuera del rango
0	"cero"	Caso especial



Conclusión

El desarrollo de este programa en Haskell nos permitió reforzar conceptos clave de la programación funcional, como la recursividad, el uso de funciones puras y la composición de funciones. La implementación de la verificación de números primos nos hizo reflexionar sobre la eficiencia algorítmica, especialmente al trabajar con valores grandes. Además, la conversión de números a palabras en español presentó desafíos interesantes, como el manejo adecuado de excepciones gramaticales, lo que nos llevó a estructurar mejor el código y modularizar cada parte para mejorar su claridad y mantenimiento.