PRÁCTICA 3

ENTRADA Y SALIDA

Pablo Cordón Hidalgo

ÍNDICE

CÓDIGO FUENTE EJERCICIO

```
--El programa le pide al jugador humano que piense un número
entre 1 y 100 y tratara de acertar
--el número que ha pensado preguntando al jugador. El jugador
responderá encontrado, mayor o
--menor y en función de la respuesta, se realizara una
modificación del número buscado
--calcularemos el nuevo número de la siguiente forma:
--proximo = (x+y) div 2

juego_busqueda_mejorada :: IO () --Nuestra funcion
juego_busqueda_mejorada es una funcion que devuelte el tipo
de dato IO, y no tiene ninguna entrada.
juego_busqueda_mejorada =
    do putStrLn "Piensa un numero entre el 1 y el 100."
--Usando el bloque do definimos la secuencia de acciones
       adivina_numero 1 100 --Entramos por primera vez en la
funcion adivina_numero con el rango que nos dice el
enunciado: entre 1 y 100
       putStrLn "Fin del juego"

adivina_numero :: Int -> Int -> IO () -- A esta función que
usaremos recursivamente se le pasan dos Int (rango del numero
a adivinar) y
--devuelve un tipo IO (salida de la funcion) con return()
adivina_numero x y =
    do putStr ("Piensas en el numero " ++ show proximo ++ "?
[No es mayor? / No es menor? / encontrado] ") --Imprimos por
pantalla el número que creemos que puede ser
       respuesta <- getLine --Realizamos la operacion de
entrada/salida de leer por pantalla con getline y la
almacenamos en la variable respuesto con <-
       case respuesta of
         "mayor"  -> adivina_numero (proximo+1) y
--Recursivamente llamamos a la funcion con un rango mayor
         "menor"  -> adivina_numero x (proximo-1)
```

```
--Recursivamente llamamos a la funcion con un rango menor
        "encontrado" -> return () --Salimos de la funcion
adivina_numero y volvemos a juego_busqueda_mejorada
        _         -> adivina_numero x y --Recursivamente
llamamos a la funcion con el mismo rango
    where
      proximo = (x+y) `div` 2
```

DESCRIPCIÓN EJERCICIO

En este ejercicio realizaremos la implementación de un simple juego en el que el programa deberá adivinar un número que el jugador humano debe pensar dentro de un rango. Mediante indicaciones al programa sobre si el número que estamos pensando es mayor, menor, o correcto, este irá cerrando el cerco hasta que consiga adivinarlo y termine el programa.

para empezar, definimos la función principal: **juego_busqueda_mejorada**

```
juego_busqueda_mejorada :: IO ()
juego_busqueda_mejorada =
    do putStrLn "Piensa un numero entre el 1 y el 100."
       adivina_numero 1 100
       putStrLn "Fin del juego"
```

Esta función desarrolla una acción de entrada/salida (IO()) y devuelve el valor (), es decir, no devuelve nada y simplemente representa strings.

Dentro de esta función se llama a la segunda función **adivina_numero x y** , que es la función que ejecutará recursivamente el juego y al que se le pasan como parámetros los dos números que queremos tener como rango. (En este caso entre 1 y 100)

```
adivina_numero :: Int -> Int -> IO ()
adivina_numero x y =
    do putStr ("Piensas en el numero " ++ show proximo ++ "? [No es
mayor? / No es menor? / encontrado] ")
       respuesta <- getLine
       case respuesta of
         "mayor"  -> adivina_numero (proximo+1) y
         "menor"  -> adivina_numero x (proximo-1)
         "encontrado" -> return ()
         _        -> adivina_numero x y
       where
         proximo = (x+y) `div` 2
```

En esta función calculamos el próximo número que puede estar dentro del rango con la expresión `where`

```
        proximo = (x+y) `div` 2
```

Se pregunta entonces al usuario si el número es mayor, menor o igual a este, y dependiendo de la respuesta, llamaremos recursivamente a la función con unos argumentos u otros.

```
do putStr ("Piensas en el numero " ++ show proximo ++ "? [No es mayor? /
No es menor? / encontrado] ")
      respuesta <- getLine
      case respuesta of
        "mayor"  -> adivina_numero (proximo+1) y
        "menor"  -> adivina_numero x (proximo-1)
        "encontrado" -> return ()
        _        -> adivina_numero x y
```

Finalmente, si el número dado por la aplicación es el que pensamos, al escribir *"encontrado"* se ejecutará la acción **return(),** que no supone ninguna ruptura de flujo y permite volver a la función anterior **(juego_busqueda_mejorada)**, que finalmente acaba con un `putStrLn "Fin del juego"`