HASKELL SUDOKU

RAÚL MUÑOZ DÁVILA, JUAN LOPEZ GONZALEZ

Funcionalidad del programa

El programa generado en Haskell permite al usuario elegir la dificultad para crear sudokus con diferente número de huecos aleatoriamente generados a partir de una semilla. Además, el usuario puede jugar con el sudoku generado y finalizar el juego cuando lo complete. Si el sudoku se completa correctamente, se imprimirá un mensaje indicándolo, de lo contrario se indicará que ha habido errores.

La generación del sudoku comienza rellenando los tres cuadrados 3x3 de la diagonal principal de manera aleatoria. A continuación, se resuelve el sudoku para posteriormente crear los huecos que el usuario debe rellenar.

El usuario debe seleccionar la dificultad deseada y, a través del teclado, elegir casillas vacías y números del 1 al 9 para completarlas. Si el usuario intenta elegir una casilla ya inicialmente rellenada por el programa, se le pedirá que seleccione otra casilla. Una vez que el usuario ha completado todas las casillas, el programa imprimirá por pantalla si la solución es válida y le permitirá jugar de nuevo en caso de que quiera.

Principales funciones

Las funciones que caben destacar del código son:

GetSeed

```
-- Obtiene la hora actual del sistema como un número entero redondeado a
milisegundos para la generacion de una semilla aleatoria
--POST: semilla generada a partir de la hora actual del sistema
getSeed :: IO Int
getSeed = fmap (round . (* 1000) . toRational . utctDayTime)
getCurrentTime
```

La función **getSeed** es una función que devuelve una semilla (seed) para generar números aleatorios. Utiliza la función **getCurrentTime** de la librería **Data.Time.Clock** para obtener la hora actual del sistema y luego aplica una serie de transformaciones para convertirla en un número entero que puede ser utilizado como semilla.

GenSudoku

```
genSudoku :: Int -> [[Int]] -> [[Int]]
genSudoku semilla sudoku = do
  let sdk1 = llenarCuadrados semilla sudoku 0 0 0
  let sdk2 = llenarCuadrados (semilla+1) sdk1 3 3 3
  llenarCuadrados (semilla+2) sdk2 6 6 6
```

La función **genSudoku** toma una semilla (un número entero) y una cuadrícula de sudoku vacía (representada por una lista de listas de enteros). Luego, rellena los 3 cuadrados 3x3 de la diagonal principal utilizando la semilla proporcionada para la aleatoriedad.

Para ello, utiliza la función **llenarCuadrados** tres veces, con diferentes parámetros. La función **llenarCuadrados** es responsable de llenar una cuadrícula de sudoku parcialmente completa, y devuelve una cuadrícula de sudoku completamente completa.

La primera vez que se llama a **llenarCuadrados**, utiliza la semilla proporcionada y comienza a llenar el primer cuadrado 3x3 en la parte superior izquierda del sudoku. La segunda vez que se llama a **llenarCuadrados**, utiliza la semilla incrementada en 1 y comienza a llenar el segundo cuadrado 3x3 en el centro del sudoku. La tercera y última vez que se llama a **llenarCuadrados**, utiliza la semilla incrementada en 2 y comienza a llenar el último cuadrado 3x3 en la parte inferior derecha del sudoku.

Al final de la última llamada a **llenarCuadrados**, se devuelve el sudoku resultante.

LlenarCuadrados

```
--PRE: semilla, matriz sudoku, x: limite (valor de p o q inicial) ,
(p,q)= esquina de arriba a la izquierda de un cuadrado 3x3
--POST: Sudoku con los cuadrados 3x3 de la diagonal principal completos
llenarCuadrados :: Int -> [[Int]] -> Int -> Int -> Int -> [[Int]]
llenarCuadrados semilla sudoku x p q
  | p == x + 3 = sudoku
  | q == x + 3 = llenarCuadrados semilla sudoku x (p+1) x
  | otherwise = llenarCuadrados semilla (setElem (p,q) (genNuevoNumero
semilla (p,q) sudoku) sudoku) x p (q+1)
```

La función **llenarCuadrados** se encarga de llenar los cuadrados 3x3 de la diagonal principal de un sudoku, usando la técnica de backtracking. Recibe como parámetros la semilla para generar números aleatorios, una matriz que representa el sudoku, las coordenadas actuales p y q, y el límite x para no salirnos del cuadrado.

La función recorre las filas y columnas de los cuadrados 3x3 de la diagonal principal, y para cada posición, genera un nuevo número aleatorio usando la función **genNuevoNumero**. Luego, utiliza la función **setElem** para actualizar la matriz del sudoku con el nuevo número generado.

La función utiliza recursión para recorrer todas las posiciones de los cuadrados 3x3 de la diagonal principal, y retorna el sudoku completo con los cuadrados 3x3 llenos.

SolveSudoku

```
--PRE: Un sudoku en formato String sin resolver
--POST: El sudoku resuelto mediante backtracking en formato String
solveSudoku :: String -> String
solveSudoku s = head (f [] s)
  where
    f x s@(h:y) =
      let
        (r, c) = divMod (length x) 9
        m # n = m `div` 3 == n `div` 3
        e = [0..8]
      in [ a | z <- ['1'..'9'],
               h == z || h == '0' && notElem z ([(x ++ s) !! (i * 9 + j)
|
                                                 i <- e, j <- e, i == r
|| j == c || i # r && j # c]),
               a <- f (x ++ [z]) y ]
    f x [] = [x]
```

La función **solveSudoku** utiliza un algoritmo de backtracking para resolver un sudoku en formato de cadena de texto.

Primero, la función define una lista vacía **f [] s** que contiene todas las soluciones posibles del sudoku. Luego, en la primera llamada recursiva de la función **f**, se extrae el primer carácter de la cadena de entrada **s**, que corresponde a la primera casilla del sudoku.

Se divide la longitud actual de la lista **x** por 9 para obtener las coordenadas de la casilla actual en el formato de matriz 9x9. Luego, se utiliza la notación **m # n** para verificar si dos casillas se encuentran en el mismo cuadrante del sudoku.

A continuación, se define una lista **e** con los valores **[0, 1, ..., 8]** y se crea una lista de los posibles valores de la casilla actual. Si la casilla actual ya tiene un valor, solo se incluye ese valor en la lista. De lo contrario, se consideran todos los valores del 1 al 9 que no se encuentran en la misma fila, columna o cuadrante que la casilla actual.

Por último, se hace una llamada recursiva a la función **f** con todas las combinaciones posibles de los valores de la casilla actual y los valores ya asignados en la lista **x**. Si se encuentra una solución, se agrega a la lista de soluciones **f**.

Una vez que se han considerado todas las casillas del sudoku, la función devuelve la primera solución encontrada, que se encuentra en la posición 0 de la lista **f**.

GetNoValidos

```
--PRE: posicion y sudoku
--POST: lista de los elementos que no se pueden escoger (no validos) para
dicha posicion
getNoValidos :: (Int,Int) -> [[Int]]-> [Int]
getNoValidos (x,y) sudoku = getFila (x,y) sudoku ++ getColumna (x,y)
sudoku ++ getCuadrado (x,y) sudoku

--PRE: posicion y sudoku
--POST: lista de los elementos que no se pueden escoger de su columna (no
validos) para dicha posicion
getColumna :: (Int,Int) -> [[Int]] -> [Int]
getColumna (x,y) = getColumnaAux (x,y) ([0..x-1]++[x+1..8])

--Funcion auxiliar para getColumna
getColumnaAux :: (Int,Int) -> [Int] -> [[Int]] -> [Int]
getColumnaAux (x,y) l sudoku = [sudoku !! x !! y | x <- l]

--PRE: posicion y sudoku
--POST: lista de los elementos que no se pueden escoger de su fila (no
validos) para dicha posicion
getFila :: (Int,Int) -> [[Int]] -> [Int]
getFila (x,y) = getFilaAux (x,y) ([0..y-1]++[y+1..8])

--Funcion auxiliar para getFila
getFilaAux :: (Int,Int) -> [Int] -> [[Int]] -> [Int]
getFilaAux (x,y) l sudoku = [sudoku !! x !! y | y <- l]
```

(Por simplicidad se omite el código de getCuadrado)

La función **getNoValidos** toma una posición **(x,y)** en un tablero de sudoku representado como una lista de listas de enteros, y devuelve una lista de enteros que no son válidos para esa posición en particular. Los números no válidos son aquellos que ya están presentes en la fila, la columna o el cuadrado al que pertenece la posición.

Para obtener los números no válidos para una posición dada, la función llama a otras tres funciones auxiliares: **getFila**, **getColumna** y **getCuadrado. getFila** devuelve los números presentes en la fila de la posición dada, **getColumna** devuelve los números presentes en la columna de la posición dada y **getCuadrado** devuelve los números presentes en el cuadrado al que pertenece la posición dada. La función **getNoValidos** luego concatena las tres listas resultantes y devuelve la lista final de números no válidos para la posición dada.

huecosRandom

```
--PRE: un entero n y una semilla
--POST: devuelve una lista de n pares diferentes de enteros (x,y)
huecosRandom :: Int -> Int -> [(Int, Int)]
huecosRandom n semilla = take n $ shuffle' coords (length coords)
(mkStdGen semilla)
  where
    coords = [(x, y) | x <- [1..9], y <- [1..9]]
```

La función **huecosRandom** toma dos argumentos: un entero **n** y una semilla **semilla**. Devuelve una lista de **n** pares de enteros que representan las coordenadas de las celdas de un sudoku que se deben vaciar para crear un puzzle.

La lista se crea a partir de una lista de todas las posibles coordenadas en el sudoku, es decir, todas las combinaciones posibles de dos números entre 1 y 9. Luego, se utiliza la función **shuffle'** de la librería System.Random.Shuffle para mezclar aleatoriamente esta lista de coordenadas. Finalmente, se toman los primeros **n** elementos de la lista mezclada para obtener la lista final de pares de coordenadas.

Comprobar

```
--PRE: Matriz sudoku
--POST: Comprueba que ninguna casilla del sudoku entre en conflicto con
otra
comprobar :: [[Int]] -> Bool
comprobar sudoku = comprobarAux sudoku (0,0) True

-- Funcion auxiliar para comprobar
comprobarAux :: [[Int]] -> (Int,Int) -> Bool -> Bool
comprobarAux sudoku (x,y) b
  | not b = False
  | x == 9 = b
  | y == 9 = comprobarAux sudoku (x+1,0) b
  | otherwise = comprobarAux sudoku (x,y+1) (b && e /= 0 && e `notElem`
getNoValidos (x,y) sudoku)
    where e = sudoku !! x !! y
```

La función **comprobar** verifica que un sudoku dado sea válido, es decir, que ninguna casilla esté en conflicto con otra. La función utiliza la función auxiliar **comprobarAux** para recorrer la matriz del sudoku. Para cada casilla, comprueba si el número en esa casilla es distinto de 0 (ya que 0 significa que la casilla está vacía) y que no está presente en la lista de valores inválidos para esa casilla (que se obtiene mediante la función **getNoValidos**). Si se encuentra alguna casilla que no cumple estas condiciones, se devuelve **False**, indicando que el sudoku no es válido. En caso contrario, se devuelve **True**.

volverAjugar

```haskell
--Funcion que permite jugar varias veces al sudoku
volverAjugar :: IO ()
volverAjugar = do
  putStrLn "¿Quieres volver a jugar? (s/n)"
  respuesta <- getLine
  if respuesta == "s"
    then do
      let sudoku = [[0,0,0,0,0,0,0,0,0],
                    [0,0,0,0,0,0,0,0,0],
                    [0,0,0,0,0,0,0,0,0],
                    [0,0,0,0,0,0,0,0,0],
                    [0,0,0,0,0,0,0,0,0],
                    [0,0,0,0,0,0,0,0,0],
                    [0,0,0,0,0,0,0,0,0],
                    [0,0,0,0,0,0,0,0,0],
                    [0,0,0,0,0,0,0,0,0]]
      jugar sudoku
      volverAjugar
    else
      putStrLn "¿Hasta pronto!"
```

Esta función permite volver a jugar al sudoku una vez finalizado el primer juego llamándose a sí misma de forma recursiva.

Explicación del código

Inicialmente se parte de un sudoku vacío (todo 0s), y se llama a la función genSudoku con la semilla obtenida por getSeed. Esta, a su vez, utiliza llenarCuadrados para completar los 3 cuadrados 3x3 de la diagonal principal del sudoku (utiliza internamente getNoValidos en la función getNuevoNumero para este proceso).

Posteriormente se utiliza solveSudoku resolver el sudoku y obtener una matriz válida, sobre la que se crean huecos (que el jugador deberá completar y cuya cantidad dependerá de la dificultad seleccionada) mediante la función huecosRandom.

Al finalizar el juego, se utilizará la función comprobar para ver si la solución creada por el usuario es válida y si el jugador lo desea, podrá volver a jugar mediante la función volverAjugar.

Uso y pruebas

1. **Elección de dificultad y Generación del sudoku inicial**

```
*Sudoku_gen> :main
Bienvenido al Sudoku
La solución que has proporcionado es:
   1   2   3   4   5   6   7   8   9
 +-----------+-----------+-----------+
Introduzca la dificultad:
1) Facil
2) Medio
3) Dificil
1
   1   2   3   4   5   6   7   8   9
 +-----------+-----------+-----------+
1| 6 | 4 | 0 | 1 | 0 | 3 | 0 | 0 | 7 |
2| 7 | 2 | 1 | 0 | 0 | 0 | 0 | 3 | 0 |
3| 0 | 3 | 9 | 0 | 6 | 0 | 0 | 0 | 0 |
 +-----------+-----------+-----------+
4| 0 | 5 | 0 | 7 | 4 | 9 | 0 | 0 | 8 |
5| 4 | 0 | 0 | 5 | 1 | 2 | 9 | 0 | 0 |
6| 0 | 7 | 2 | 6 | 0 | 8 | 5 | 0 | 0 |
 +-----------+-----------+-----------+
7| 0 | 0 | 0 | 0 | 0 | 0 | 6 | 4 | 0 |
8| 0 | 0 | 4 | 0 | 0 | 6 | 3 | 0 | 2 |
9| 3 | 9 | 6 | 2 | 8 | 4 | 0 | 5 | 1 |
 +-----------+-----------+-----------+
   1   2   3   4   5   6   7   8   9
 +-----------+-----------+-----------+
```

2. Introducción de elementos en el sudoku

```
   1   2   3   4   5   6   7   8   9
 +-----------+-----------+-----------+
1| 6 | 4 | 0 | 1 | 0 | 3 | 0 | 0 | 7 |
2| 7 | 2 | 1 | 0 | 0 | 0 | 0 | 3 | 0 |
3| 0 | 3 | 9 | 0 | 6 | 0 | 0 | 0 | 0 |
 +-----------+-----------+-----------+
4| 0 | 5 | 0 | 7 | 4 | 9 | 0 | 0 | 8 |
5| 4 | 0 | 0 | 5 | 1 | 2 | 9 | 0 | 0 |
6| 0 | 7 | 2 | 6 | 0 | 8 | 5 | 0 | 0 |
 +-----------+-----------+-----------+
7| 0 | 0 | 0 | 0 | 0 | 0 | 6 | 4 | 0 |
8| 0 | 0 | 4 | 0 | 0 | 6 | 3 | 0 | 2 |
9| 3 | 9 | 6 | 2 | 8 | 4 | 0 | 5 | 1 |
 +-----------+-----------+-----------+
   1   2   3   4   5   6   7   8   9
 +-----------+-----------+-----------+
Introduzca la fila en la que desea escribir un numero (entre 1 y 9):
1
Introduce la coluna en la que desea escribir un numero (entre 1 y 9):
3
Introduce el un numero entre 1 y 9 :
3
1| 6 | 4 | 3 | 1 | 0 | 3 | 0 | 0 | 7 |
2| 7 | 2 | 1 | 0 | 0 | 0 | 0 | 3 | 0 |
3| 0 | 3 | 9 | 0 | 6 | 0 | 0 | 0 | 0 |
 +-----------+-----------+-----------+
4| 0 | 5 | 0 | 7 | 4 | 9 | 0 | 0 | 8 |
5| 4 | 0 | 0 | 5 | 1 | 2 | 9 | 0 | 0 |
6| 0 | 7 | 2 | 6 | 0 | 8 | 5 | 0 | 0 |
 +-----------+-----------+-----------+
7| 0 | 0 | 0 | 0 | 0 | 0 | 6 | 4 | 0 |
8| 0 | 0 | 4 | 0 | 0 | 6 | 3 | 0 | 2 |
9| 3 | 9 | 6 | 2 | 8 | 4 | 0 | 5 | 1 |
 +-----------+-----------+-----------+
   1   2   3   4   5   6   7   8   9
 +-----------+-----------+-----------+
```

3. Introducción de elementos no válidos

```
-
1| 6 | 4 | 3 | 1 | 0 | 3 | 0 | 0 | 7 |
2| 7 | 2 | 1 | 0 | 0 | 0 | 0 | 3 | 0 |
3| 0 | 3 | 9 | 0 | 6 | 0 | 0 | 0 | 0 |
 +-----------+-----------+-----------+
4| 0 | 5 | 0 | 7 | 4 | 9 | 0 | 0 | 8 |
5| 4 | 0 | 0 | 5 | 1 | 2 | 9 | 0 | 0 |
6| 0 | 7 | 2 | 6 | 0 | 8 | 5 | 0 | 0 |
 +-----------+-----------+-----------+
7| 0 | 0 | 0 | 0 | 0 | 0 | 6 | 4 | 0 |
8| 0 | 0 | 4 | 0 | 0 | 6 | 3 | 0 | 2 |
9| 3 | 9 | 6 | 2 | 8 | 4 | 0 | 5 | 1 |
 +-----------+-----------+-----------+
   1   2   3   4   5   6   7   8   9
 +-----------+-----------+-----------+
Introduzca la fila en la que desea escribir un numero (entre 1 y 9):
15
Introduce la coluna en la que desea escribir un numero (entre 1 y 9):
5
Introduce el un numero entre 1 y 9 :
5
La posicion introducida no es modificable, introduzca otra posicion :
Introduzca la fila en la que desea escribir un numero (entre 1 y 9):
1
Introduce la coluna en la que desea escribir un numero (entre 1 y 9):
2
Introduce el un numero entre 1 y 9 :
3
La posicion introducida no es modificable, introduzca otra posicion :
Introduzca la fila en la que desea escribir un numero (entre 1 y 9):
```

4. Fin de juego con solución aportada incorrecta

```
4| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 7 | 2 |
5| 1 | 7 | 3 | 2 | 5 | 6 | 7 | 8 | 9 |
6| 2 | 2 | 3 | 3 | 7 | 6 | 7 | 6 | 9 |
 +-----------+-----------+-----------+
7| 9 | 1 | 3 | 8 | 5 | 6 | 4 | 5 | 3 |
8| 3 | 8 | 3 | 9 | 4 | 7 | 1 | 2 | 9 |
9| 6 | 2 | 4 | 4 | 5 | 5 | 7 | 8 | 0 |
 +-----------+-----------+-----------+
Introduzca la fila en la que desea escribir un numero (entre 1 y 9):
9
Introduce la coluna en la que desea escribir un numero (entre 1 y 9):
9
Introduce el un numero entre 1 y 9 :
9
1| 4 | 6 | 3 | 5 | 5 | 3 | 8 | 8 | 7 |
2| 8 | 2 | 3 | 4 | 1 | 6 | 6 | 3 | 9 |
3| 1 | 3 | 9 | 4 | 8 | 6 | 2 | 8 | 9 |
 +-----------+-----------+-----------+
4| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 7 | 2 |
5| 1 | 7 | 3 | 2 | 5 | 6 | 7 | 8 | 9 |
6| 2 | 2 | 3 | 3 | 7 | 6 | 7 | 6 | 9 |
 +-----------+-----------+-----------+
7| 9 | 1 | 3 | 8 | 5 | 6 | 4 | 5 | 3 |
8| 3 | 8 | 3 | 9 | 4 | 7 | 1 | 2 | 9 |
9| 6 | 2 | 4 | 4 | 5 | 5 | 7 | 8 | 9 |
 +-----------+-----------+-----------+
La solución es:
   1   2   3   4   5   6   7   8   9
 +-----------+-----------+-----------+
1| 4 | 6 | 1 | 5 | 2 | 3 | 8 | 9 | 7 |
2| 8 | 5 | 2 | 7 | 1 | 9 | 6 | 3 | 4 |
3| 7 | 3 | 9 | 4 | 8 | 6 | 2 | 1 | 5 |
 +-----------+-----------+-----------+
4| 1 | 4 | 3 | 6 | 5 | 8 | 9 | 7 | 2 |
5| 5 | 7 | 6 | 2 | 9 | 1 | 3 | 4 | 8 |
6| 2 | 9 | 8 | 3 | 7 | 4 | 5 | 6 | 1 |
 +-----------+-----------+-----------+
7| 9 | 1 | 7 | 8 | 6 | 2 | 4 | 5 | 3 |
8| 3 | 8 | 5 | 9 | 4 | 7 | 1 | 2 | 6 |
9| 6 | 2 | 4 | 1 | 3 | 5 | 7 | 8 | 9 |
 +-----------+-----------+-----------+
Sudoku resuelto incorrectamente
FIN DEL JUEGO
```

5. Fin de juego con solución correcta

```
Introduzca la fila en la que desea escribir un numero (entre 1 y 9):
9
Introduce la coluna en la que desea escribir un numero (entre 1 y 9):
9
Introduce el un numero entre 1 y 9 :
3
1| 7 | 1 | 4 | 2 | 3 | 5 | 8 | 6 | 9 |
2| 2 | 6 | 9 | 4 | 1 | 8 | 3 | 7 | 5 |
3| 8 | 3 | 5 | 7 | 9 | 6 | 1 | 2 | 4 |
 +-----------+-----------+-----------+
4| 3 | 2 | 8 | 1 | 4 | 7 | 5 | 9 | 6 |
5| 1 | 4 | 6 | 5 | 8 | 9 | 2 | 3 | 7 |
6| 5 | 9 | 7 | 3 | 6 | 2 | 4 | 8 | 1 |
 +-----------+-----------+-----------+
7| 4 | 7 | 1 | 9 | 2 | 3 | 6 | 5 | 8 |
8| 6 | 5 | 3 | 8 | 7 | 1 | 9 | 4 | 2 |
9| 9 | 8 | 2 | 6 | 5 | 4 | 7 | 1 | 3 |
 +-----------+-----------+-----------+
La solución es:
   1   2   3   4   5   6   7   8   9
 +-----------+-----------+-----------+
1| 7 | 1 | 4 | 2 | 3 | 5 | 8 | 6 | 9 |
2| 2 | 6 | 9 | 4 | 1 | 8 | 3 | 7 | 5 |
3| 8 | 3 | 5 | 7 | 9 | 6 | 1 | 2 | 4 |
 +-----------+-----------+-----------+
4| 3 | 2 | 8 | 1 | 4 | 7 | 5 | 9 | 6 |
5| 1 | 9 | 6 | 5 | 8 | 2 | 4 | 3 | 7 |
6| 5 | 4 | 7 | 3 | 6 | 9 | 2 | 8 | 1 |
 +-----------+-----------+-----------+
7| 4 | 7 | 1 | 9 | 2 | 3 | 6 | 5 | 8 |
8| 6 | 5 | 3 | 8 | 7 | 1 | 9 | 4 | 2 |
9| 9 | 8 | 2 | 6 | 5 | 4 | 7 | 1 | 3 |
 +-----------+-----------+-----------+
Sudoku resuelto
FIN DEL JUEGO
```

6. **Volver a jugar una vez terminado el sudoku**

```
Sudoku resuelto incorrectamente
La solución es:
   1   2   3   4   5   6   7   8   9
 +-----------+-----------+-----------+
1| 3 | 4 | 5 | 1 | 2 | 6 | 7 | 8 | 9 |
2| 2 | 6 | 7 | 3 | 9 | 8 | 4 | 1 | 5 |
3| 9 | 8 | 1 | 7 | 4 | 5 | 3 | 2 | 6 |
 +-----------+-----------+-----------+
4| 1 | 7 | 2 | 5 | 6 | 3 | 8 | 9 | 4 |
5| 4 | 5 | 3 | 8 | 7 | 9 | 1 | 6 | 2 |
6| 6 | 9 | 8 | 4 | 1 | 2 | 5 | 3 | 7 |
 +-----------+-----------+-----------+
7| 8 | 3 | 9 | 6 | 5 | 4 | 2 | 7 | 1 |
8| 7 | 2 | 4 | 9 | 8 | 1 | 6 | 5 | 3 |
9| 5 | 1 | 6 | 2 | 3 | 7 | 9 | 4 | 8 |
 +-----------+-----------+-----------+
FIN DEL JUEGO
¿Quieres volver a jugar? (s/n)
s
Bienvenido al Sudoku
   1   2   3   4   5   6   7   8   9
 +-----------+-----------+-----------+
Introduzca la dificultad:
1) Facil
2) Medio
3) Dificil
1
   1   2   3   4   5   6   7   8   9
 +-----------+-----------+-----------+
1| 0 | 0 | 0 | 3 | 0 | 0 | 0 | 7 | 9 |
2| 7 | 0 | 9 | 0 | 0 | 1 | 0 | 2 | 0 |
3| 0 | 5 | 0 | 7 | 0 | 9 | 8 | 0 | 1 |
 +-----------+-----------+-----------+
4| 2 | 0 | 5 | 0 | 7 | 0 | 1 | 0 | 4 |
5| 0 | 7 | 0 | 0 | 0 | 4 | 9 | 0 | 8 |
6| 0 | 9 | 0 | 6 | 1 | 0 | 7 | 0 | 2 |
 +-----------+-----------+-----------+
7| 9 | 4 | 0 | 1 | 3 | 6 | 0 | 8 | 0 |
8| 0 | 0 | 0 | 5 | 9 | 2 | 4 | 0 | 7 |
9| 5 | 0 | 0 | 0 | 4 | 0 | 3 | 9 | 0 |
 +-----------+-----------+-----------+
   1   2   3   4   5   6   7   8   9
 +-----------+-----------+-----------+
Introduzca la fila en la que desea escribir un numero (entre 1 y 9):
```

Limitaciones

La anterior limitación de no poder volver a jugar tras finalizar el juego ha sido corregida mediante la función volverAjugar.