Entrada y Salida

Curso 2004/05, Fecha:9/12/2004

Enunciado 1 (compara2) *En Haskell, las funciones de Entrada/Salida se marcan mediante un tipo especial* ***IO****. El tipo* ***IO*** *a indica una acción de Entrada/Salida (con efectos laterales) que devuelve un valor de tipo a*

Este tipo incluye dos operaciones básicas:

- ***return*** :: *a* →***IO*** *a*
- *(>>=)* :: ***IO*** *a* →*(a* →***IO*** *b)* →***IO*** *b*

Existen varias funciones predefinidas para leer/escribir valores, por ejemplo:

- ***getChar*** ::***IO Char*** *lee un caracter*
- ***putChar*** ::***Char*** →***IO*** *()* *imprime un caracter*
- ***getLine*** ::***IO String*** *lee un línea*
- ***putStr*** :: ***String*** →***IO*** *()* *imprime una línea*

A continuación se presenta una función que pide 2 líneas al usuario e indica si son iguales

```
> compara :: IO ()
> compara = getLine >>= (\s1 →
>           getLine >>= (\s2 →
>           if s1 == s2 then putStr "Son iguales"
>                       else putStr "Son diferentes"))
```

La notación ***do*** *permite simplificar funciones como la anterior, que podrían re-escribirse como:*

```
> compara2 :: IO ()
> compara2 = do
>             s1 ← getLine
>             s2 ← getLine
>             if s1 == s2 then putStr "Son iguales"
>                         else putStr "Son diferentes"
```

- *Modificar la función anterior para que muestre un mensaje por pantalla pidiendo las cadenas*

- *Modificar la función anterior para que solicite 3 cadenas en lugar de 2*

Enunciado 2 (getInt) *La siguiente función getInt toma como parámetro un mensaje que muestra por pantalla y devuelve el valor numérico que introduce el usuario*

```
> getInt :: String → IO Int
> getInt msg = do
>                putStrLn msg
>                cs ← getLine
>                return (read cs)
```

- *Escribir una función que pida un número e indique si es par*
- *Escribir una función que pida 2 números e indique si el primero es mayor que el segundo*

Enunciado 3 (bucle) *Construir una función que solicite números de forma repetida hasta que se introduzca el número cero.*

Modificar la función anterior para que al finalizar devuelva la suma de los números introducidos

Enunciado 4 (aleas) *La generación de números aleatorios requiere efectos laterales. Haskell proporciona la librería* ***Random*** *que incluye varias funciones para generación de números aleatorios. Las más habituales podrían ser:*

- ***randomIO*** *::**IO** a devuelve un valor aleatorio de tipo a (en realidad, a debe pertenecer a la clase de tipos* ***Random****)*
- ***randomRIO*** *::(a,a) →**IO** a devuelve un valor aleatorio dentro de un intervalo de valores.*

A modo de ejemplo, el siguiente programa solicita un número n y escribe n números aleatorios entre 0 y 6

```
> aleas :: IO ()
> aleas = do
>          n ← getInt "Numeros? "
>          ponAleas n

> ponAleas :: Int → IO ()
> ponAleas n
>   | n == 0 = return ()
>   | n >  0 = do
>                a ← randomRIO (0,100)
>                putStr (" " ++ show (a :: Int))
>                ponAleas (n - 1)
```

- *Modificar el programa anterior para que además de imprimir los números generados, los devuelva en una lista*
- *Modificar el programa anterior para que calcule la media de los números aleatorios generados*

Enunciado 5 (histograma(Opcional)) *Modificar el programa anterior para que calcule un la frecuencia de cada uno de los valores entre 0 y 100*

Representar dichas frecuencias como un histograma en formato SVG

Enunciado 6 (juego) *Construir un juego en el que se genere un número aleatorio entre 0 y 100, y luego se vaya preguntando al usuario números hasta que el usuario acierte el número generado. En cada iteración, indicar si el número a acertar es mayor o menor que el número introducido.*

Modificar el programa anterior para que al acertar indique cuántos intentos se han realizado

Enunciado 7 (cuentaCar) *Existen varias funciones predefinidas que permiten leer y escribir contenido en un fichero. Por ejemplo:*

- ***readFile*** :: ***String*** →***IO String****: se le pasa como argumento el nombre de un fichero y devuelve su contenido*
- ***writeFile*** :: ***String*** →***String*** →***IO*** *(): toma como argumento el nombre de un fichero y una cadena de caracteres y como efecto lateral escribe en dicho fichero la cadena de caracteres*

Escribir un programa Haskell que solicite el nombre de un fichero e indique cuántos caracteres contiene

Enunciado 8 (numera) *La función predefinida* ***lines*** :: ***String*** →*[****String****] toma como argumento una cadena de caracteres y devuelve una lista con las líneas que contiene, es decir, fragmentos separados por el carácter \n*

Escribir una función que lea un fichero y escriba su contenido numerando cada línea. Por ejemplo, si el contenido fuese:

```
holahola
adiosadios
pepepepe
```

La salida sería:

```
1. holahola
2. adiosadios
3. pepepepe
```

Enunciado 9 (histoFichero(Opcional)) *Escribir una función que calcule la frecuencia de cada letra de un fichero.*

Modificar el programa anterior para que genere un fichero SVG con el histograma

Enunciado 10 (catch(Opcional)) *La función* ***catch*** :: ***IO*** *a* →*(****IOError*** →***IO*** *a)* →***IO*** *a funciona de la siguiente forma:* ***catch*** *ac f intenta ejecutar la acción ac. Si no hay errores devuelve lo que devuelva dicha acción. Si se producen errores, entonces llama a la función f pasándole como argumento el error producido.*

Por ejemplo, la siguiente función solicita el nombre de un fichero para contar el número de caracteres y, si se produce un error, lo comunica y vuelve a solicitar el fichero.

```
> cuentaSeguro =
>   catch cuentaCar
>         (\e → do
>                putStrLn ("Hubo un error: " ++ show e)
>                cuentaSeguro)
```

Modificar el juego para que el programa no finalice si el usuario introduce valores no numéricos, sino que continúe pidiendo valores hasta que sean números

Enunciado 11 (Figuras Aleatorias(Opcional)) *Generar quadtrees y octrees con figuras aleatorias. Pueden emplearse varios procedimientos: tomar un octree dado y convertir cada elemento básico en un color aleatorio, decidir si se realizan subdivisiones a partir de un valor aleatorio, etc.*