Projecto Top 10: entrada y salida (IO) y estructura de datos Map (diccionario)

Referencia útil

- capítulo "Entrada y Salida" (http://aprendehaskell.es/content/EntradaSalida.html) de Aprende Haskell (hasta antes de la sección Aleatoriedad).
- capítulo "Módulos"(http://aprendehaskell.es/content/Modulos.html), en particular `Data.List` y `Data.Map`
- el libro Aprende Haskell es desactualizado, en realidad hay que importar `Data.Map.Strict` en lugar de `Data.Map`
- el buscador Hoogle: https://www.haskell.org/hoogle/

Projecto Top 10

1. Hacer un programa que lee un archivo de texto, y lo muestra con los carácteres en mayúscula (usando la función `toUpper` del módulo `Data.Char`.
2. Vamos a hacer un programa que hace un poco más de procesamiento de datos.

   La idea es hacer un programa que lee un archivo de texto y muestre las 10 palabras más frecuentes.

   Como archivos de texto pueden bajar libros desde proyecto Gutenberg (https://www.gutenberg.org/) en formato `.txt`. Inclusive tiene libros grandes como este: https://www.gutenberg.org/ebooks/2600

   Se puede dividir el trabajo en 3 partes:

   1. leer el contenido del archivo como una lista de palabras
   2. contar cuantas veces aparecen cada palabra
   3. ordenar las palabras de más frecuente a menos frecuente, y mostrar las 10 más frecuentes

Hay varias formas de hacer la parte 2. (y esto influye la parte 3.).

Vamos a usar la estructura de datos `Map` de `Data.Map.Strict`, porque es una estructura optimizada que puede soportar muchos datos (pensar en abrir un archivo con millones de palabras distintas). Más precisamente, la idea es guardar las palabras y su número de ocurrencias en un diccionario de tipo `Map String Int`.

Pasos:

a. importar los módulos `Data.Map.Strict` y `Data.List` dándoles un prefijo para cada uno para evitar colisiones de nombres (ver "import qualified" en Aprende Haskell). Definir una función `main` que lee el contenido de un archivo (de nombre indicado en duro) y define las palabras como una lista de Strings.
b. definir una función `agregar :: Map String Int -> String -> Map String Int` que agrega una palabra a un diccionario (posiblemente el diccionario ya tiene la palabra!). Fijarse (en Hoogle) e, la documentación de Data.Map.Strict qué función(es) les puede(n) ayudar para definir `agregar`.
c. Modificar el `main` para agregar a un diccionario vacío (buscar en `Data.Map.Strict`) todas las palabras del archivo (¿qué función de orden superior nos permite hacer eso?).
d. definir la función `top10 :: Map String Int -> [(Int, String)]` tal que `top 10 m` sean las 10 palabras más frecuentes del diccionario `m`. Definir `top10` requiere (entre otras cosas) ordenar una lista. Fijarse en el módulo Data.List qué función(es) puede(n) ser útil(es) para eso.
e. Mostrar el top 10 de las palabras del archivo en la salida estandard.

Después de terminar cada paso se puede mostrar en la salida estandard lo que llegaron a definir, usando (posiblemente) la función `show` en combinación con `putStrLn`.

Probar con archivos *grandes* (varios megas).

Pueden mejorar el programa implementando su propia función que corta una `String` en lista de palabras, tomando en cuenta la puntuación, para mejorar la precisión del programa.