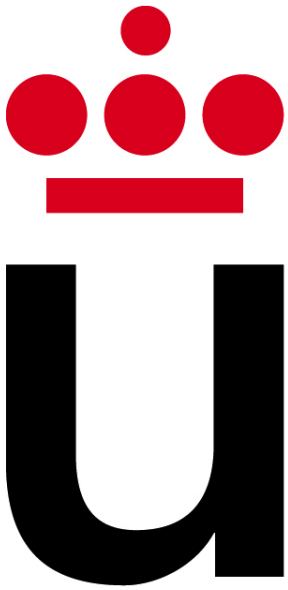
Paradigmas de Programación

Práctica 1

Haskell

GII+GIS: Luis Guillermo León Gámez



Móstoles, 20 de noviembre de 2015

Índice

1. Introducción

En esta práctica se pide hacer un programa en Haskell que sea capaz de procesar una colección de documentos científicos con el fin de conocer diferente información relacionada con ellos.
Los diferentes artículos que se proporcionan se corresponden con artículos científicos escritos en inglés y relacionados con la investigación sobre Enfermedades Raras. En todos los artículos aparece el nombre de una enfermedad rara acompañada de su acrónimo al menos una vez.

2. Módulos

2.1. Main.hs

Este módulo se utiliza esclusivamente para mostrar los distintos menús del programa. Como al principio se optó por un diseño poco eficiente (en algunas opciones), el primer menú, "**Menú de modos**", muestra dos opciones:

1. Utilizar dinámicamente los documentos.
2. Cargar los documentos antes de usarlos.

- (exit)

La primera opción no tarda nada en mostrar el "**Menú de ejercicios**", y los ejercicios 1, 2 y 8 los realiza de manera casi inmediata. Sin embargo, como el resto de ejercicios conlleva una búsqueda de acrónimos, tarda un rato en ejecutarse cada uno. Por este motivo decidí crear la segunda opción, que carga los documentos (junto con la búsqueda de acrónimos por documento) para no tener que volver a cargarlos en un futuro. La tercera opción está en ambos menús, y te permite salir del programa escribiendo "*exit*". El segundo menú muestran los ejercicios pedidos y 2 opciones extra (además de la opción "*exit*" y la opción "*mode*", que permite volver al "**Menú de modos**") que se explican más adelante.

2.2. Utils.hs

Este módulo contiene los métodos correspondientes a cada opción del "**Menú de ejercicios**" de la primera implementación (es decir, del modo que utiliza dinámicamente los documentos).

2.3. Utils2.hs

Este módulo contiene los métodos correspondientes a cada opción del "**Menú de ejercicios**" de la segunda implementación (es decir, del modo que carga los documentos antes de usarlos).

2.4. Papers.hs

Este módulo solo exporta la función *getPapers*, que se encarga de devolver un array de String compuesto por los path de todos los documentos.

2.5. Documento.hs

Este módulo posee todos los métodos relacionados con la lectura de los documentos y tratamiento de dichos documentos. Posee los siguientes datos y funciones relevantes:

2.5.1. Data Document

- path :: String
- journal :: String
- ident :: Int
- year :: Int
- title :: Title
- abstract :: String
- sections :: [[String]]*
- acronyms :: [Acronym]

*sections:: [[Title 1 , Section 1] , [Title 2 , Section 2] , ... , [Title n , Section 2]]

2.5.2. Functions get

Estas funciones permiten acceder a los campos del tipo de dato "*Document*", o pedir una instancia de dicho tipo de dato a partir de un path (o lista de paths).

2.5.3. Functions readDocument

Esta función está definida varias veces. La primera (*readDocument*) lee el documento de manera completa (realizando la búsqueda de acrónimos). Como para la implementación antigua era muy ineficiente, se creó un método "*readDocument*_n" por cada ejercicio, leyendo solo la información necesaria en cada una.

2.5.4. Functions show

Estás funciones muestran por pantalla la información del tipo de dato "*Document*" de distintas formas (todas necesarias para los distintos apartados).

2.5.5. Functions others

Estas funciones se encargan de filtrar, ordenar, y en resumen, realizar distintas operaciones con el tipo de datos "*Document*", necesarias para distintos ejercicios.

2.6. Acronimo.hs

Este módulo posee todos los métodos relacionados con la lectura de acrónimos y el tratamiento de dichos acrónimos. Posee los siguientes datos y funciones relevantes:

2.6.1. Data Acronym

- acr :: String
- exp :: String
- pos :: Int

(acr: acrónimo, exp: forma expandida, pos: posición en el texto)

2.6.2. Functions get

Estas funciones permiten acceder a los campos del tipo de dato "*Acronym*".

2.6.3. Functions readAcronyms

Estos métodos se encargan de realizar la búsqueda de acrónimos en el texto. Se ha optado por seguir un método de búsqueda de derecha a izquierda. Existen 2 funciones para ello, *quickSearch* busca los acrónimos sin la forma expandida, y *searchAcronyms* busca los acrónimos con sus respectivas formas expandidas. Los pasos que he utilizado son los siguientes:

1. Se invierte el texto completo.
2. Se lee carácter a carácter el texto.
3. Si un carácter es mayúscula, y su siguiente también lo es, se considera un posible acrónimo.
4. Se llama a un método que busca la posición final del acrónimo en el texto.
5. Si es escesivamente largo (tope de 5 caractéres) se ignora y se sigue buscando.
6. Se intenta buscar la forma expandida si el carácter previo al acrónimo y su siguiente son paréntesis (en caso contrario, se devuelve como un acrónimo sin forma expandida).
7. Se busca la forma expandida hacia delante (al estar el texto invertido, la búsqueda es inversa).
8. Si se encuentra una letra que coincide con con la primera letra del acrónimo, se "tacha" esta y se sigue buscando la siguiente letra que coincida con la segunda letra del acrónimo, y así hasta encontrar la úlima.
9. Como todos los acrónimos comienzan siempre con la misma letra que su respectiva forma expandida, la letra que coincida con la última letra del acrónimo no puede venir seguida de otra letra, ergo si viene seguida de otra letra, se vuelve a buscar la última letra del acrónimo desde dicho punto hasta que se cumpla esta condición.

10. Si la forma expandida es escesivamente larga (tope 50 caractéres) se descarta como una forma expandida aceptable y se devuelve un acrónimo sin forma expandida.

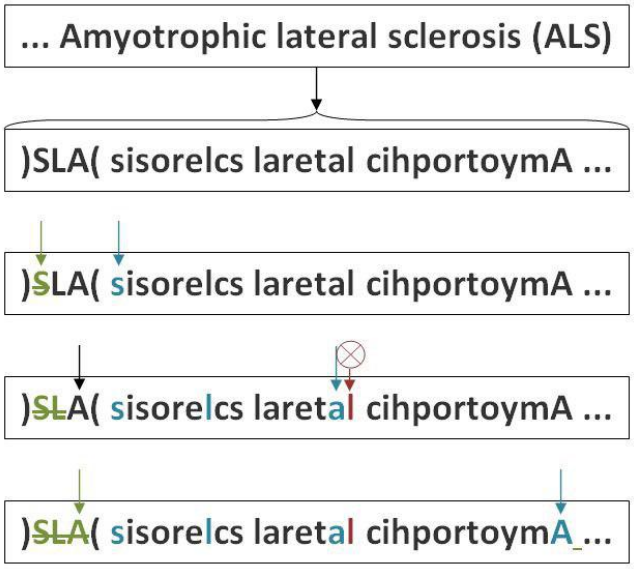


Figura 1: *Croquis - Lectura forma expandida*

2.6.4. Functions show

Estás funciones muestran por pantalla la información del tipo de dato "*Acronym*".

2.6.5. Functions others

Estas funciones se encargan de eliminar acrónimos repetidos, o contar acrónimos duplicados.

2.7. Aux.hs

Este módulo fue creado para facilitar la realización del ejercicio 9, por lo que no se corresponde con ningún tipo de dato del documento o elemento pedido en el enunciado. El objetivo era tener una estructura donde poder comparar los acrónimos más utilizados en los documentos, y poder agrupar según esto los documentos en temáticas.

2.7.1. Data Extra

- doc :: Document
- aux :: [Auxiliar]

2.7.2. Data Auxiliar

- acronym :: Acronym
- n :: Int*

*n: indica el número de veces que aparece el acrónimo en el documento.

2.7.3. Functions get

Estas funciones permiten acceder a los campos de los tipos de dato "*Extra*" y "*Auxiliar*".

2.7.4. Functions read

Estas funciones guardan en los tipos de datos mencionados la información sacada de los documentos leidos. La manera en la que lee los datos "*Document*" es la siguiente:

1. Recibe una lista de documentos, guarda en Extra(doc) el documento.
2. En Extra(aux) se guarda la lista de acrónimos del documento, ordenada según el número de veces que se repiten dichos acrónimos (eliminando los repetidos).
3. Se simplifican las listas de acrónimos escogiendo los más repetidos.

De este conjunto de funciones toma especial interés la función "*simplify*", que se encarga de reducir el número de acrónimos y quedarse con los que más aparecen. Esta función se basa en que se puede establecer una relación entre la variedad de acrónimos que aparecen en un texto, y su frecuencia, que ayuda a ver cuántos acrónimos son significativamente importantes. Según he podido comprobar de los documentos dados, esta relación sería:

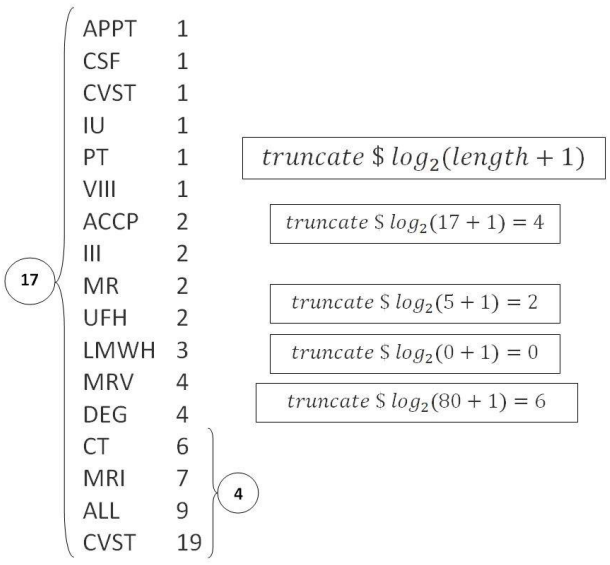


Figura 2: *Ecuación - Relación variedad-frecuencia*

2.7.5. Functions clustering

Estas funciones agrupan los documentos que traten sobre una misma enfermedad/temática. El mecanismo consiste en recorrer la lista [*Extra*] leída previamente comparando el acrónimo más usado de cada documento y agrupándolos por dicho acrónimo. En este caso basta con leer solo el acrónimo más usado de cada documento para conocer el tema del artículo. Si la base de datos fuera más grande y se precisase un agrupación más exigente, se podrían comparar más de los acrónimos más usados en cada documento.

2.7.6. Functions show

Estas funciones se encargan de mostrar por pantalla la información de los tipos de dato "*Extra*" y "*Auxiliar*", y de mostrar por pantalla los clusters generados en la sección de clustering.

2.8. Extras.hs

Este archivo solo contiene 3 métodos genéricos que se usan desde casi todos los módulos:

- **prompt**: utilizado para poder mostrar el prompt en el archivo compilado.
- **clearUp**: utilizado para limpiar la pantalla y que sea más vistoso.
- **printPoints**: utilizado para imprimir puntitos de carga.

3. Ejercicios

En esta parte se comenta la resolución de los distintos ejercicios (ubicados tanto en el archivo Utils.hs como Utils2.hs). Tras haber implementado el resto de módulos, basta con llamar a las respectivas operaciones implementadas en el resto de módulos.

3.1. Ejercicio 1

Mostrar los títulos de los artículos ordenados alfabéticamente y publicados en un año dado.

```
putStrLn $ showAllDocumentsTitles $ orderByTitle docs
```

3.2. Ejercicio 2

Mostrar el listado de revistas en las que se han publicado los artículos de toda la colección.

```
putStrLn $ showPointed $ nub $ getJournals docs
```

3.3. Ejercicio 3

Dado un acrónimo, buscarlo en los diferentes artículos y mostrar los títulos de aquellos que contengan el acrónimo.

```
putStrLn $ showAllDocumentsTitles docs
```

3.4. Ejercicio 4

Dado el nombre de una revista y un acrónimo, mostrar los títulos de los artículos publicados en dicha revista que contengan el acrónimo.

```
putStrLn $ showAllDocumentsTitles $ filterByAcronym acronym $ ...
... filterByJournal journal (removeDuplicatedAcronyms docs)
```

3.5. Ejercicio 5

Dado un año de publicación, mostrar para cada artículo publicado en ese año el listado de acrónimos que contiene acompañados de sus formas expandidas.

```
putStrLn $ showAllDocumentsTitlesAndAcronyms $ ...
... filterByYear (read year::Int) (removeDuplicatedAcronyms docs)
```

3.6. Ejercicio 6

Dado un identificador de artículo, mostrar un listado de los acrónimos que contiene, acompañado del número de veces que aparece cada acrónimo en el artículo.

```
putStrLn $ showAllDocumentsTitlesAndAcronyms2 $ filterById (read id::Int) docs
```

3.7. Ejercicio 7

Mostrar los títulos e identificador de todos aquellos artículos que no contengan ningún acrónimo.

```
putStrLn $ showAllDocumentsTitlesAndIds $ filterByNoAcronyms docs
```

3.8. Ejercicio 8

Dado el nombre de una revista, mostrar toda la información de los artículos publicados en dicha revista.

```
putStrLn $ showAllDocuments docs
```

3.9. Ejercicio 9

Tratar de agrupar los artículos que se refieran a la misma enfermedad o a la misma temática.

```
clustering $ sortByAuxLength $ readExtras docs
```

4. Extensiones

Como se ha mencionado, existen 2 opciones más en el menú principal:

- **(all)**: Muestra uno a uno la información de los artículos (path, id, title, year, section_number, section_titles, acronyms). Se puede avanzar escribiendo "n" (o pulsando enter), retroceder escribiendo "p", ir a un determinado documento escribiendo el número, y salir escribiendo "stop".
- **(top)**: Muestra la lista de documentos con sus acrónimos más usados.

5. Comentarios

He decidido mantener ambas implementaciones (tanto la de operar dinámicamente con los documentos como la de cargar previamente los documentos) porque para probar ciertos ejercicios, uno es más rápido que el otro, y en otras ocasiones al revés.

Para ejecutar el código basta con abrir la terminal en la carpeta donde se encuentra el archivo *Practica1* y ejecutarlo.

```
./Practica1
```

El código de compilación es el siguiente:

```
ghc -odir Program/O_files/ -hidir Program/HI_files/ ...
... -o Program/Practica1 --make Main.hs
```

Si se quiere ejecutar con ghci:

```
ghc Main.hs
```

Tras iniciarse ghci, ejecutar "main".