Paradigmes i Llenguatges de Programació
Pràctica de Haskell
[Verificador d'escacs]

Robert Ripoll
David Suárez

May 2019

Be aware, she can move in any direction.

Contents

1 Introducció i objectius

Partint de l'enunciat:
Es demana que feu un programa interactiu amb Haskell on es demani a l'usuari el nom del fitxer on hi hagi la partida en notació algebraica estesa per a analitzar i el programa vagi representant textualment la situació de la partida, tirada a tirada, fins al final [...].

Hem desenvolupat un petit programa en Haskell que ens permet llegir fitxers de partides o conjunt de jugades d'escacs, tot seguint la notació algebraica especificada al document *escacs.pdf* de l'enunciat de la pràctica.

Si bé tot els recursos que hem fet servir, formen part del propi GHCI, hem hagut d'explicitar les següents exportacions de Data:

```
import Data.Char
import Data.Maybe
```

La sortida del nostre programa, és una representació ASCII d'un tauler d'escacs. D'altra banda, totes les partides han de partir de l'escenari inicial d'un tauler d'escacs, tal com representem a continuacio:

```
    A   B   C   D   E   F   G   H
  #################################
  #   #   #   #   #   #   #   #   #
8- # t # c # a # d # r # a # c # t #
  #   #   #   #   #   #   #   #   #
  #################################
  #   #   #   #   #   #   #   #   #
7- # p # p # p # p # p # p # p # p #
  #   #   #   #   #   #   #   #   #
  #################################
  #   #   #   #   #   #   #   #   #
6- #   #   #   #   #   #   #   #   #
  #   #   #   #   #   #   #   #   #
  #################################
  #   #   #   #   #   #   #   #   #
5- #   #   #   #   #   #   #   #   #
  #   #   #   #   #   #   #   #   #
  #################################
  #   #   #   #   #   #   #   #   #
4- #   #   #   #   #   #   #   #   #
  #   #   #   #   #   #   #   #   #
  #################################
  #   #   #   #   #   #   #   #   #
3- #   #   #   #   #   #   #   #   #
  #   #   #   #   #   #   #   #   #
  #################################
  #   #   #   #   #   #   #   #   #
2- # P # P # P # P # P # P # P # P #
  #   #   #   #   #   #   #   #   #
  #################################
  #   #   #   #   #   #   #   #   #
1- # T # C # A # D # R # A # C # T #
  #   #   #   #   #   #   #   #   #
  #################################
```

Representació ASCII d'un tauler d'escacs

2 Exemples d'execució

2.1 Exemple: Pastor

En aquest exemple, utilitzarem el fitxer d'entrada donat pel professor *pastor.txt* on es realitza la coneguda jugada, per part de les peces Blanques. La partida no conté cap error, i pertant, acaba correctament:

```
llegirPartida "pastor.txt"
```

```
Ronda 1. Blanques: Jugada 'P' 'e'2 'e'4 Negres: Jugada 'p' 'e'7 'e'5
Ronda 2. Blanques: Jugada 'A' 'f'1 'c'4 Negres: Jugada 'c' 'b'8 'c'6
Ronda 3. Blanques: Jugada 'D' 'd'1 'h'5 Negres: Jugada 'c' 'g'8 'f'6
Ronda 4. Blanques: EscacMat 'D' 'h'5 'f'7 Negres: No juguen
```

```
   A    B    C    D    E    F    G    H
 #################################################
 #    #    #    #    #    #    #    #    #
8- #  t #    #  a #  d #  r #  a #    #  t #
 #    #    #    #    #    #    #    #    #
 #################################################
 #    #    #    #    #    #    #    #    #
7- #  p #  p #  p #  p #    #  D #  p #  p #
 #    #    #    #    #    #    #    #    #
 #################################################
 #    #    #    #    #    #    #    #    #
6- #    #    #  c #    #    #  c #    #    #
 #    #    #    #    #    #    #    #    #
 #################################################
 #    #    #    #    #    #    #    #    #
5- #    #    #    #    #  p #    #    #    #
 #    #    #    #    #    #    #    #    #
 #################################################
 #    #    #    #    #    #    #    #    #
4- #    #    #  A #    #  P #    #    #    #
 #    #    #    #    #    #    #    #    #
 #################################################
 #    #    #    #    #    #    #    #    #
3- #    #    #    #    #    #    #    #    #
 #    #    #    #    #    #    #    #    #
 #################################################
 #    #    #    #    #    #    #    #    #
2- #  P #  P #  P #  P #    #  P #  P #  P #
 #    #    #    #    #    #    #    #    #
 #################################################
 #    #    #    #    #    #    #    #    #
1- #  T #  C #  A #    #  R #    #  C #  T #
 #    #    #    #    #    #    #    #    #
 #################################################
```

Configuració final en *pastor.txt*

2.2 Exemple: Pastor amb error

En aquest exemple, fem servir un fitxer *pastor_error.txt* on es realitza la coneguda jugada, per part de les peces Blanques. No obstant, hem determinat un error en la descripció de les jugades, i és que el darrer moviment de les blanques, malgrat ser Escac Mat, s'indica només l'escac. És a dir Dh5xf7+ en lloc de Dh5xf7++ :

```
llegirPartida "pastor_error.txt"
```

```
Ronda 1. Blanques: Jugada 'P' 'e'2 'e'4 Negres: Jugada 'p' 'e'7 'e'5
Ronda 2. Blanques: Jugada 'A' 'f'1 'c'4 Negres: Jugada 'c' 'b'8 'c'6
Ronda 3. Blanques: Jugada 'D' 'd'1 'h'5 Negres: Jugada 'c' 'g'8 'f'6
Ronda 4. Blanques: Escac 'D' 'h'5 'f'7 Negres: No juguen
INVALID: Ronda 4. Jugador amb peces Blanc: S'ha indicat ESCAC, i és MAT
```

2.3 Exemple: Prova amb error

En aquest exemple, fem servir un fitxer *prova.txt* on es realitza senzillament un moviment impossible de la Dama Blanca, partint de la disposició inical del tauler, la volem situar a H5, pertant Dd1h5.

```
llegirPartida "prova.txt"
```

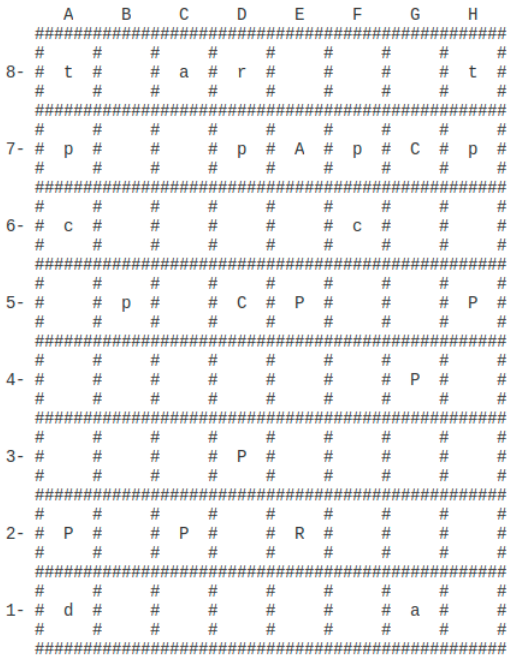
```
Ronda 1. Blanques: Jugada 'D' 'd'1 'h'5 Negres: No juguen
INVALID: Ronda 1. Jugador amb peces Blanc: Hi ha una peça pel mig entre la posició origen i la posició destí la jugada.
```

2.4 Exemple: Partida inmortal

En aquest exemple, fem servir el fitxer *partida_inmortal.txt* donat amb la pràctica, la partida acaba amb la victoria per part de les peces Blanques. És una partida amb 23 rondes que transcorren amb normalitat:

```
llegirPartida "partida_inmortal.txt"
```

```
Ronda 1. Blanques: Jugada 'P' 'e'2 'e'4 Negres: Jugada 'p' 'e'7 'e'5
Ronda 2. Blanques: Jugada 'P' 'f'2 'f'4 Negres: Captura 'p' 'e'5 'f'4
Ronda 3. Blanques: Jugada 'A' 'f'1 'c'4 Negres: Escac 'd' 'd'8 'h'4
Ronda 4. Blanques: Jugada 'R' 'e'1 'f'1 Negres: Jugada 'p' 'b'7 'b'5
Ronda 5. Blanques: Captura 'A' 'c'4 'b'5 Negres: Jugada 'c' 'g'8 'f'6
Ronda 6. Blanques: Jugada 'C' 'g'1 'f'3 Negres: Jugada 'd' 'h'4 'h'6
Ronda 7. Blanques: Jugada 'P' 'd'2 'd'3 Negres: Jugada 'c' 'f'6 'h'5
Ronda 8. Blanques: Jugada 'C' 'f'3 'h'4 Negres: Jugada 'd' 'h'6 'g'5
Ronda 9. Blanques: Jugada 'C' 'h'4 'f'5 Negres: Jugada 'p' 'c'7 'c'6
Ronda 10. Blanques: Jugada 'P' 'g'2 'g'4 Negres: Jugada 'c' 'h'5 'f'6
Ronda 11. Blanques: Jugada 'T' 'h'1 'g'1 Negres: Captura 'p' 'c'6 'b'5
Ronda 12. Blanques: Jugada 'P' 'h'2 'h'4 Negres: Jugada 'd' 'g'5 'g'6
Ronda 13. Blanques: Jugada 'P' 'h'4 'h'5 Negres: Jugada 'd' 'g'6 'g'5
Ronda 14. Blanques: Jugada 'D' 'd'1 'f'3 Negres: Jugada 'c' 'f'6 'g'8
Ronda 15. Blanques: Captura 'A' 'c'1 'f'4 Negres: Jugada 'd' 'g'5 'f'6
Ronda 16. Blanques: Jugada 'C' 'b'1 'c'3 Negres: Jugada 'a' 'f'8 'c'5
Ronda 17. Blanques: Jugada 'C' 'c'3 'd'5 Negres: Captura 'd' 'f'6 'b'2
Ronda 18. Blanques: Jugada 'A' 'f'4 'd'6 Negres: Captura 'a' 'c'5 'g'1
Ronda 19. Blanques: Jugada 'P' 'e'4 'e'5 Negres: Escac 'd' 'b'2 'a'1
Ronda 20. Blanques: Jugada 'R' 'f'1 'e'2 Negres: Jugada 'c' 'b'8 'a'6
Ronda 21. Blanques: Escac 'C' 'f'5 'g'7 Negres: Jugada 'r' 'e'8 'd'8
Ronda 22. Blanques: Escac 'D' 'f'3 'f'6 Negres: Captura 'c' 'g'8 'f'6
Ronda 23. Blanques: EscacMat 'A' 'd'6 'e'7 Negres: No juguen
```



Configuració final en *partida_inmortal.txt*

2.5 Exemple: Rei destapat amb error

En aquest exemple, fem servir el fitxer *rei_destapat.txt* on ens trobem en la situació on les peces Blanques intenten realitzar un moviment amb el Peó situat a G3, que destaparien el rei i el posarien en Escac:

```
llegirPartida "rei_destapat.txt"
```

```
Ronda 1. Blanques: Jugada 'P' 'e'2 'e'4 Negres: Jugada 'p' 'e'7 'e'5
Ronda 2. Blanques: Jugada 'P' 'f'2 'f'4 Negres: Captura 'p' 'e'5 'f'4
Ronda 3. Blanques: Jugada 'A' 'f'1 'c'4 Negres: Escac 'd' 'd'8 'h'4
Ronda 4. Blanques: Jugada 'P' 'g'2 'g'3 Negres: Jugada 'p' 'b'7 'b'5
Ronda 5. Blanques: Jugada 'P' 'g'3 'g'4 Negres: Captura 'd' 'h'4 'g'4
INVALID: Ronda 5. Jugador amb peces Blanc: La situació actual és d'escac, i la jugada segueix en escac.
```

2.6 Exemple: Escac eliminat amb error

En aquest exemple, fem servir el fitxer *escac_eliminat.txt* on ens trobem amb un moviment invàlid del Peó Negre que es troba a F4, quan s'indica a la partida que realitza una **captura**. Imaginem que la partida està mal representada, i es volía fer la jugada Pf4xg3. No obstant, sent fidels al fitxer, tenim la següent partida:

```
llegirPartida "escac_eliminat.txt"
```

```
Ronda 1. Blanques: Jugada 'P' 'e'2 'e'4 Negres: Jugada 'p' 'e'7 'e'5
Ronda 2. Blanques: Jugada 'P' 'f'2 'f'4 Negres: Captura 'p' 'e'5 'f'4
Ronda 3. Blanques: Jugada 'A' 'f'1 'c'4 Negres: Escac 'd' 'd'8 'h'4
Ronda 4. Blanques: Jugada 'P' 'g'2 'g'3 Negres: Escac 'd' 'h'4 'g'3
Ronda 5. Blanques: Captura 'P' 'h'2 'g'3 Negres: Captura 'p' 'f'4 'f'3
INVALID: Ronda 5. Jugador amb peces Negre: S'ha indicat CAPTURA,
i no ho és.
```

2.7 Exemple: Escac no resolt amb error

En aquest exemple, fem servir el fitxer *escac_no_resolt.txt* on ens trobem amb el Rei Negre que es troba en Escac i el moviment que preten fer per ensortir-se'n, és invàlid pel multiples raons. En primer lloc, continuaria estant en Escac, i daltra banda també hi ha un Peó Negre a F2.

```
llegirPartida "error_escac_no_resolt.txt"
```

```
Ronda 1. Blanques: Jugada 'P' 'e'2 'e'4 Negres: Jugada 'p' 'e'7 'e'5
Ronda 2. Blanques: Jugada 'P' 'f'2 'f'4 Negres: Captura 'p' 'e'5 'f'4
Ronda 3. Blanques: Jugada 'A' 'f'1 'c'4 Negres: Escac 'd' 'd'8 'h'4
Ronda 4. Blanques: Jugada 'R' 'e'1 'f'2 Negres: Jugada 'p' 'b'7 'b'5
INVALID: Ronda 4. Jugador amb peces Blanc: La situació actual és
d'escac, i la jugada segueix en escac.
```

2.8 Exemple: Enroc llarg

En aquest exemple, fem servir el fitxer *enroc_llarg.txt* on fem una partida simètrica pel que fa als moviments dels dos colors, possiblititant que tots dos facin l'enroc llarg.

```
llegirPartida "enroc_llarg.txt"
```

```
Ronda 1. Blanques: Jugada 'P' 'd'2 'd'4 Negres: Jugada 'p' 'd'7 'd'5
Ronda 2. Blanques: Jugada 'A' 'c'1 'e'3 Negres: Jugada 'a' 'c'8 'e'6
Ronda 3. Blanques: Jugada 'C' 'b'1 'a'3 Negres: Jugada 'c' 'b'8 'a'6
Ronda 4. Blanques: Jugada 'D' 'd'1 'd'2 Negres: Jugada 'd' 'd'8 'd'7
Ronda 5. Blanques: EnrocLlarg Negres: EnrocLlarg
```

```
   A   B   C   D   E   F   G   H
 ###############################################
 #   #   #   #   #   #   #   #   #
8# # # r # t # # a # c # t #
 #   #   #   #   #   #   #   #   #
 ###############################################
 #   #   #   #   #   #   #   #   #
7# p # p # p # d # p # p # p # p #
 #   #   #   #   #   #   #   #   #
 ###############################################
 #   #   #   #   #   #   #   #   #
6# c # # # # a # # # #
 #   #   #   #   #   #   #   #   #
 ###############################################
 #   #   #   #   #   #   #   #   #
5# # # # p # # # # #
 #   #   #   #   #   #   #   #   #
 ###############################################
 #   #   #   #   #   #   #   #   #
4# # # # P # # # # #
 #   #   #   #   #   #   #   #   #
 ###############################################
 #   #   #   #   #   #   #   #   #
3# C # # # A # # # #
 #   #   #   #   #   #   #   #   #
 ###############################################
 #   #   #   #   #   #   #   #   #
2# P # P # P # D # P # P # P # P #
 #   #   #   #   #   #   #   #   #
 ###############################################
 #   #   #   #   #   #   #   #   #
1# # # R # T # # A # C # T #
 #   #   #   #   #   #   #   #   #
 ###############################################
```

Configuració final en *enroc_llarg.txt*

3 Codi font i funcions

3.1 Tipus de dades

- data Color = Blanc | Negre deriving (Eq, Show)
- data TipusPeca = Rei | Reina | Torre | Alfil | Cavall | Peo deriving Eq
- data Peca = Pec TipusPeca Color deriving Eq
- data Tauler = Tau [(Posicio, Peca)]
- data Posicio = Char :/ Int deriving Eq
- data JugadaGenerica = Jugada Peca Posicio Posicio | Captura Peca Posicio Posicio | EnrocCurt Peca Posicio Posicio | EnrocLlarg Peca Posicio Posicio | Escac Peca Posicio Posicio | EscacMat Peca Posicio Posicio deriving (Eq, Show)
- data Jugada = Jug Peca Posicio Posicio deriving Eq
- type Casella = (Posicio, Peca)
- data Partida = Par Tauler Color (Actualment no utilitzada)

3.2 Moviments

posicioUp :: Posicio -> Posicio

Retorna el desplaçament d'una posició més adalt respecte la posició passada per paràmetre.

posicioDown :: Posicio -> Posicio

Retorna el desplaçament d'una posició més adalt respecte la posició passada per paràmetre.

posicioRight :: Posicio -> Posicio

Retorna el desplaçament d'una posició més a la dreta respecte la posició passada per paràmetre.

posicioLeft :: Posicio -> Posicio

Retorna el desplaçament d'una posició més a l'esquerra respecte la posició passada per paràmetre.

posicioDiagSupEsq :: Posicio -> Posicio

Retorna el desplaçament d'una posició més a la diagonal superior esquerra respecte la posició passada per paràmetre.

posicioDiagSupDreta :: Posicio -> Posicio

Retorna el desplaçament d'una posició més a la diagonal superior dreta respecte la posició passada per paràmetre.

posicioDiagInfEsq :: Posicio -> Posicio

Retorna el desplaçament d'una posició més a la diagonal inferior esquerra respecte la posició passada per paràmetre.

posicioDiagInfDreta :: Posicio -> Posicio

Retorna el desplaçament d'una posició més a la diagonal inferior dreta respecte la posició passada per paràmetre.

3.3 Mètodes rellevants

trobarPeca :: Tauler -> Posicio -> Maybe Peca

Retorna la peça ("Just Peça") del tauler passat per paràmetre que es troba a la posició passada per paràmetre. Si la posició no existeix retorna "Nothing".

trobarPeces :: Tauler -> [Posicio] -> [Maybe Peca]

Retorna un conjunt de peces que tenen la posició passada per paràmetre en una llista, d'acord amb el tauler passat per paràmetre. Si una posició de les passades a la llista no es troba, s'afegirà un "Nothing" a la llista a retornar; en cas que es trobi, s'afegirà un "Just Peca".

aplicarFunc :: (Posicio -> Posicio) -> Posicio -> [Posicio]

Aplica una funció passada per paràmetre a una posició passada per paràmetre, i retorna, en un llistat, el resultat d'aplicar la funció a la posició, de tornar a aplicar la funció al previ resultat i així successivament fins trobar una posició no vàlida.

generarMoviments :: Peca -> Posicio -> [Posicio]

Genera les posicions on pot anar una peça passada per paràmetre trobant-se en una posició passada per paràmetre, d'acord amb les regles del joc i com es pot moure la peça.

generarPosicions :: Posicio -> Posicio -> (Posicio -> Posicio) -> [Posicio]

Retorna les posicions dins d'un interval passat per paràmetre (posA, posB) aplicant una funció passada per paràmetre fins trobar una posició no vàlida o

que sigui la posició B. Dins d'aquesta llista de posicions s'exclouen tant posA com posB.

posicionsEntre :: Posicio -> Posicio -> [Posicio]

Retorna les posicions que es troben dins de l'interval passat per paràmetre (posA, posB). Dins d'aquesta llista de posicions s'exclouen tant posA com posB.

alguEntre :: Tauler -> Posicio -> Posicio -> Bool

Retorna cert si hi ha alguna peça entre dues posicions passades per paràmetre (excloent les dues posicions, es clar); fals altrament.

aplicarJugada :: Tauler -> Jugada -> Bool -> [(Posicio, Peca)]

Retorna el resultat d'aplicar una jugada passada per paràmetre sobre el tauler passat per paràmetre. El booleà passat per paràmetre indica si s'ha trobat la casella de la posició destí de la jugada o no.

fesJugada :: Tauler -> Jugada -> Tauler

Retorna un nou tauler resultant d'aplicar la jugada passada per paràmetre, partint del tauler (estat actual del joc) passat per paràmetre.

trobarRei :: Tauler -> Color -> Posicio

Retorna la posició del rei del bàndol del color passat per paràmetre d'acord amb el tauler passat per paràmetre.

jugadesColor :: Tauler -> Color -> [Jugada]

A partir de l'estat actual de la partida (el tauler passat per paràmetre) i un bàndol (color passat per paràmetre), retorna les possibles jugades vàlides i legals que pot fer.

movimentsColor :: Tauler -> Color -> [Posicio]

Retorna un llistat amb els moviments que poden fer les peces del bàndol del color passat per paràmetre d'acord amb el tauler passat per paràmetre.

escac :: Tauler -> Color -> Bool

Retorna cert si el bàndol del color passat per paràmetre està en escac a l'estat actual de la partida (el tauler passat per paràmetre); retorna fals altrament.

jugadaValida :: Tauler -> Jugada -> Int

Si la jugada és vàlida, retornarà 0 o 1. Això es determinarà a partir de la jugada que es vol fer i l'estat del tauler, que són elements que es passen per paràmetre. Retornarà 0 quan la jugada passada sigui vàlida i no es capturi a cap peça contrincant. Si la jugada és vàlida però es captura una peça contrincant, es retornarà 1. Si la jugada no és vàlida, retorna:

- -1 -> "La casella origen de la jugada esta buida"
- -2 -> "La peça de la casella origen no coincideix amb la peça de la jugada"
- -3 -> "El moviment de la jugada no és vàlid d'acord amb els moviments que pot fer la peça"
- -4 -> "Hi ha una peça pel mig entre la posició origen i la posició destí la jugada"
- -5 -> "La posició destí de la jugada està ocupada per una peça del mateix jugador que fa la jugada"
- -6 -> "La situació actual és d'escac, i la jugada segueix en escac"

escacMat :: Tauler -> Color -> Bool

Retorna cert si el bàndol amb el color passat per paràmetre està en escac mat; retorna fals altrament.

llegirLinia :: String -> (String, JugadaGenerica, Maybe JugadaGenerica)

Llegeix una linia del tipus "1. Pe2e4 Pe7e5" i ho parseja en forma de Tupla, tenint en compte si son 2 o 3 paràmetres.

llegirJugada :: String -> Color -> JugadaGenerica

Donat un String com per exemple "Pe7e5" o "Dh5xf7++" o "0-0" retorna el tipus de JugadaGenerica concret que és.

tractaUnaJugada :: String -> Tauler -> Color -> JugadaGenerica -> Tauler

Des d'aquesta funció es controlen els errors d'invalidesa d'una determinada jugada. Tanmateix és la que a partir de subtipus de JugadaGenerica donat, i el tauler, s'efectuaran les comprovacions i les crides a fesJugada.

evalua :: Tauler -> (String, JugadaGenerica, Maybe JugadaGenerica) -> Tauler

Donat un Tauler i una tupla de tres elements, comprovem si es tracta d'una jugada de blanques unicament o bé de ambdós colors intervenint. Retorna el tauler havent aplicat les jugades.

iteraRondes :: Tauler -> [(String, JugadaGenerica, Maybe JugadaGenerica)] -> IO()

En aquesta funció es mostren els missatges de informació sobre les diferents Rondes, amb el respectiu tauler, etc. No només mostra, sinó que recursivament crida a la funció evalua mentre mostra missatges.

llegirPartida :: String -> IO()

Exemple d'ús: llegirPartida "pastor.txt" Interpreta la partida i la tradueix a Jugades, s'evaluen a "evalua"

4 Resultat i valoracions

El resultat és un petit programa que contempla la majoría de casuístiques del joc. No obstant, quedaria pendent vàries de les millores proposades com pot ser detectar taules, etc.

Pel que fa al paradigma de programació funcional, i concretament al Haskell la comprensió de funcions és més fàcil respecte la programació declarativa i imperativa ja que les funcions estan escrites a un molt alt nivell.
El fet que Haskell sigui pur i garanteixi transparència referencial ens facilita el testing, ja que si una funció retorna un valor equivocat només haurem de traçar el fil i seguir els valors que són erronis, ignorant la resta de l'aplicació. Comparant aquest tipus de programació (la funcional) amb la programació tradicional imperativa, amb la programació funcional estàs descrivint el que vols en comptes de com ho vols.

References

[1] **Pàgina de consulta**
`http://zvon.org/other/haskell/Outputglobal/index.html`

[2] **Pàgina de consulta**
`https://wiki.haskell.org/Haskell`