Exemplos em Haskell

ERICK GALANI MAZIERO
erick.maziero@ufla.br

Departamento de Ciências da Computação
Universidade Federal da Lavras

Atividade Avaliativa

1. Receber nome de arquivo como argumento
2. Ler arquivo
3. Separar e converter valores
4. Calcular coeficiente da correlação de Pearson
5. Exibir resultado no console

Verificando os tipos das funções

```
ghci> :t getLine
getLine :: IO String

ghci> :t putStrLn
putStrLn :: String -> IO ()
```

http://learnyouahaskell.com/types-and-typeclasses

Blocos do

```
import System.IO  --módulo para operações de IO

main = do
    putStrLn "Hello, what's your name?"
    name <- getLine
    putStrLn ("Hey " ++ name ++ ", you rock!")
```

http://learnyouahaskell.com/input-and-output

Arquivos e argumentos

```
import System.IO   --módulo para operações de IO
import System.Environment  -- para tratamento de argumentos

main = do
    (nomearq:_) <- getArgs-- lembrem de cabeça:cauda de lista
    contents <- readFile nomearq
    putStrLn contents
```

Split string por quebra de linha

```
import System.IO   --módulo para operações de IO
import System.Environment  -- para tratamento de argumentos

main = do
    (nomearq:_) <- getArgs -- lembrem de cabeça:cauda de lista
    contents <- readFile nomearq
    linesOfFile <- lines contents
    putStrLn unlines linesOfFile
```

http://learnyouahaskell.com/modules#data-list

Split string por espaço

```
import System.IO   --módulo para operações de IO
import System.Environment  -- para tratamento de argumentos

main = do
    (nomearq:_) <- getArgs -- lembrem de cabeça:cauda de lista
    contents <- readFile nomearquivo
    linesOfFile <- words contents
    putStrLn unwords linesOfFile
```

http://learnyouahaskell.com/modules#data-list

Considerando funções implementadas

- Veja a função `words` na documentação:
  - https://www.haskell.org/onlinereport/standard-prelude.html

- A função `words` faz o split por espaço, posso alterá-la para quebrar por vírgula?? Como?

Implementação da função `words`

```
words     :: String -> [String]
words s   =  case dropWhile Char.isSpace s of
                  "" -> []
                  s' -> w : words s''
                  where (w, s'') = break Char.isSpace s'
```

Alterando implementação da função words

```
wordsByComma :: (Char -> Bool) -> String -> [String]
wordsByComma d s = case dropWhile d s of
                        "" -> []
                        s' -> w : wordsByComma d s''
                        where (w, s'') = break d s’
```

```
splittedValues = wordsByComma (==',') ”valor,valor,valor,valor”
```

Conversão entre tipos

- String para Float

```
toString valor = show valor
```

- Float para String

```
toFloat valor = read valor :: Float
```

List Comprehension

```
toFloat valor = read valor :: Float

listaStr = [”1.3” ”5.6” ”8”]

listaFloat = [toFloat elem | elem <- listaStr]

print $ listaFloat
```

http://learnyouahaskell.com/starting-out#im-a-list-comprehension

Funções auxiliares

```
lista1 = [1, 3, 5, 6]
lista2 = [3, 4, 6, 7]
sumLists lst1 lst2 = zipWith (+) lst1 lst2
sumList lst = sum lst
lista3 = sumLists lista1 lista2
print $ lista3
[4,7,11,13]
print $ sumList lista3
35
```

http://learnyouahaskell.com/modules

Funções auxiliares

```
lista1 = [1, 3, 5, 6]
g v x = v * x
f v lst = map (g v) lst
lista2 = f 2 lista1
print $ lista2

[2,6,10,12]
```

http://learnyouahaskell.com/higher-order-functions#maps-and-filters

Referência Bibliográfica

http://learnyouahaskell.com/input-and-output