Haskell: Entrada e Saída

UFRN, 2018

Sistema de E/S Haskell

- Sistema de E/S puramente funcional, mas com a expressividade de linguagens de programação convencionais.
- Haskell integra operações de E/S ao contexto funcional via monads.

Monads

- Um monad pode ser visto como um tipo de dado abstrato com valores abstratos que são ações que modificam estados.
- Ações:
  - Lêem e modificam estados do sistema, como a escrita de arquivos, leitura de entradas, etc.
  - Devem ser ordenadas de forma bem definida para a execução do programa.
  - Retornam valores, mesmo que nulos.

Funções de Saída

- Funções de escrita na saída padrão, ex:

```
putChar   :: Char -> IO ()
putStr    :: String -> IO ()
putStrLn  :: String -> IO ()
print     :: Show a => a -> IO ()
```

- Exemplo de uso:

```
main = print ([(n, 2^n) | n <- [0..19]])
```

Funções de Entrada

- Funções de leitura na saída padrão, ex:

```
getChar      :: IO Char
   getLine      :: IO String
   getContents :: IO String
   interact     :: (String -> String) -> IO ()
   readIO       :: Read a => String -> IO a
   readLn       :: Read a => IO a
```

- Exemplo de uso:

```
-- import Data.Char
main = interact (filter isAscii)
```

Arquivos

- Funções para arquivos texto, ex:

```
type FilePath =  String
   writeFile     ::  FilePath -> String  -> IO ()
   appendFile ::  FilePath -> String  -> IO ()
   readFile      ::  FilePath -> IO String
```

- Exemplo de uso:

```
main = appendFile "squares"
                          (show [(x,x*x) | x <- [0,0.1..2]])
```

Seqüenciamento

```
01  main :: IO ()
02  main =  do c <- getChar
                putChar c
```

```
01  getLine :: IO String
02  getLine = do c <- getChar
03                if c == '\n' then return ""
04                else do s <- getLine
05                        return (c:s)
```

Exemplo 1 – String

```
01 import IO
02 main = do hdl <- openFile "filename.txt“ WriteMode
03              hPutStr hdl "Hello, world!“
04              hClose hdl
05          hdl <- openFile "filename.txt" ReadMode
06          x <- hGetContents hdl
07    putStr x
08    hClose hdl
```

Exemplo 2 – ByteString

```
01  import qualified Data.ByteString.Char8 as B
02  imprimir :: [B.ByteString] -> IO ()
03  imprimir [] = putStr ""
04  imprimir (a:x) = do putStr (B.unpack a)
05                      imprimir x
06  main = do x <- B.readFile "filename.txt"
07            let y = B.splitWith (==(' ')) x in imprimir (clean y)
08  clean :: [B.ByteString] -> [B.ByteString]
09  clean [] = []
10  clean (a:x) = if a == (B.pack "") then clean x
11                else a : clean x
```