Paradigmas de Programação

Prof. Maicon R. Zatelli

Haskell - Programação Funcional
Introdução

Universidade Federal de Santa Catarina
Florianópolis - Brasil

Haskell - Introdução

HASKELL (nome em homenagem a Haskell Curry) - 1990

Funcional (diretamente derivada da ML)

Criada pela Universidade de Glasgow.

Linguagem puramente funcional e baseada em calculo lambda tipado.

`https://www.haskell.org/`

`https://www.tutorialspoint.com/compile_haskell_online.php`

Haskell - Compilando e Executando

Faça o download do ghc e instale em seu computador:

- https://www.haskell.org/ghc/

Para compilar um único arquivo use o seguinte comando:

- ghc −o nomeExecutavel arquivo.hs

Para compilar um arquivo e seus respectivos módulos use o seguinte comando:

- ghc −−make arquivo.hs −v

Para executar, basta executar o arquivo resultante da compilação. Ex: ./nomeExecutavel

Haskell - Prelude

É o cliente Haskell.

Pode ser aberto por meio do comando `ghci`

Por meio do Prelude, pode-se:

- :t expressão/função –Descobrir tipo
- :cd pasta –Navegar em pastas
- :l arquivo.hs –Carregar um arquivo .hs
- :h – Ver o ajuda

Haskell - A Linguagem

Considerados brancos: espaços, <cr>, <lf>, tabulações, comentários.

Comentário de linha: $--$
Comentário de multilinha: $\{- \; -\}$

Identificadores reservados

case | class | data | default | deriving | do | else | if | import | in | infix | infixl | infixr | instance | let | module | newtype | of | then | type | where

Operadores

Aritméticos

- +, -, *, /, ‘mod‘, ‘div‘
- ** (exponenciação com ponto flutuante)
- ^(exponenciação inteira positiva)
- ^^(exponenciação inteira)

Lógicos

- ||, &&, not

Relacionais

- ==, /=, >, >=, <=, >

Haskell - A Linguagem

Construtores de listas

- :
- [ ]
- ..

Construtores Lambda

- \
- ->

Concatenação

- ++

O caracter **Sublinhado** _

- É considerado caracter minúsculo se associado a um identificador;
- É considerado "coringa" se usado isolado.

Identificadores iniciados com _ não são verificados com compiladores, portanto não podem ser acessados.

Exemplo:

```
select_segundo (_x, y, _) = y
```

Esta função recebe uma tripla e retorna o segundo elemento. Note que o primeiro e o terceiro não importam, portanto podemos escrever com _ no início do identificador ou mesmo _ sozinho.

Entrada e saída de dados

```
Prelude> :t getLine
getLine :: IO String --Lê da IO uma String

Prelude> :t getChar
getChar :: IO Char --Lê da IO um Char

Prelude> :t putStrLn
putStrLn :: String -> IO () --Recebe uma String e retorna uma ação de IO

Prelude> :t print
print :: Show a => a -> IO () --Aplica show e retorna uma ação de IO
```

Entrada e saída de dados

```
main = putStrLn "Olá, mundo!"
```

```
main = do
    putStrLn "Informe sua idade: "
    idade <- getLine
    putStrLn ("Sua idade eh " ++ idade)
```

- **Note**: identação é importante no Haskell, assim como no Python!
- Leia mais: `https://en.wikibooks.org/wiki/Haskell/Indentation`

Entrada e saída

```
main = do
    print 5.75
    print 6
    print True
    print "Floripa"
    print [1,2,3]
    print ["a", "b", "c"]
```

- "Stringifica" o valor antes, por meio da função `show` e depois imprime. Basicamente, chama a função `putStrLn` chamando antes a função `show`

Entrada e saída

```
main = do
    putStrLn (show 5.75)
    putStrLn (show 6)
    putStrLn (show True)
    putStrLn (show "Floripa")
    putStrLn (show [1,2,3])
    putStrLn (show ["a", "b", "c"])
```

Haskell - A Linguagem

Para vincular valores a um nome, use <- para ações de IO e **let** para outras expressões.

```
--Recebe um parâmetro Float e retorna String
avaliacao :: Float -> String
avaliacao x =
    if x < 6.0 then
        "Reprovado"
    else
        "Aprovado"

main = do
    putStrLn "Informe sua nota: "
    notaString <- getLine
    --Converte notaString para Float
    let nota = (read notaString :: Float)
    --Atribui o resultado da chamada da função avaliacao para resultado
    let resultado = (avaliacao nota)
    putStrLn ("Com " ++ notaString ++ " voce esta " ++ resultado)
    --Chama diretamente a função avaliacao
    putStrLn ("Com " ++ notaString ++ " voce esta " ++ (avaliacao nota))
```

Calculando a área de um retângulo

```
--Recebe dois Float e retorna um Float
areaRetangulo :: Float -> Float -> Float
areaRetangulo base altura = base * altura

main = do
    baseString <- getLine
    alturaString <- getLine
    let base = (read baseString :: Float)
    let altura = (read alturaString :: Float)
    print (areaRetangulo base altura)
```

Calculando o fatorial de *n*

```
-- Recebe um Int e retorna um Int
fatorial :: Int -> Int
fatorial 0 = 1
fatorial n = n * fatorial (n-1)

main = do
    nString <- getLine
    let n = (read nString :: Int)
    print (fatorial n)
```

Implementando o operador lógico E

```
-- Recebe dois parâmetros Bool e retorna um Bool
funcaoE :: Bool -> Bool -> Bool
funcaoE False _ = False
funcaoE _ False = False
funcaoE True True = True

main = print (funcaoE True False)
```

Haskell - A Linguagem

Lendo um caracter e um número e imprimindo

```
main = do
    op <- getChar
    --Ignora espaços e outros caracteres depois do caracter lido
    _ <- getLine
    strX <- getLine
    let x = (read strX :: Float)
    print op
    print x
```

Haskell - A Linguagem

Guarda

```
funcao :: Int -> Int
funcao x | (x == 0) = 0
         | (x == 1) = 1
         | otherwise = 10

main = print (funcao 2)
```

Haskell - Alguns Links Úteis

- `http://learnyouahaskell.com/chapters`
- `http://haskell.tailorfontela.com.br/chapters`
- `https://wiki.haskell.org/Haskell_in_5_steps`

Haskell

Ver atividade no Moodle