

Paradigmas de Programação

Prof. Maicon R. Zatelli

Haskell - Programação Funcional
Cálculo Lambda

Universidade Federal de Santa Catarina
Florianópolis - Brasil

2018/1

Haskell

Operadores Lambda no Haskell

- \
- ->

Teste os seguintes exemplos no Prelude:

```
((\x -> x * x) 3)
```

```
((\x -> x + 10) 3)
```

```
((\x y -> x + y) 3 5)
```

Quais são os resultados?

Utilizando expressões Lambda em um programa Haskell

```
main = do
    print ((\x y -> x + y) 3 5)
```

Haskell

```
main = do
  print (map (\x -> x + 1) [1,2,3,4])
```

- Aqui utilizamos a função map, a qual recebe como parâmetro uma expressão Lambda que soma 1 a cada elemento da lista.

Haskell

```
main = do
  print (map (\x -> if x == 0 then 'a' else 'b') [0,1,0,1,1])
```

- Aqui criamos uma string formada por a e b, onde 0 será substituído por a e 1 será substituído por b.

Haskell

Implementando o fatorial utilizando expressões Lambda em Haskell

```
import Data.Function

fatorial :: Int -> Int
fatorial = fix (\f n -> if n == 0 then 1 else n * (f (n-1)))

main = do
    print (fatorial 5)
```

Haskell

Entendendo o combinador (ou função) `fix` do Haskell

```
fix :: (a -> a) -> a
fix f = let {x = f x} in x
```

- A função `fix`, assim como o combinador `Y`, permite que uma função possa chamar ela mesma infinitamente.
- Assim, temos algo como $x = f\ x = f\ (f\ x) = f\ (f\ (f\ x)) \dots$
- A palavra reservada `in` faz com que o que vier entre `let` e `in` (que são declarações, atribuições) sejam válidos apenas dentro do que vier depois de `in` (neste caso como expressão). Assim tem-se que: `let {atribuições} in {expressão}` significa que essas atribuições são somente válidas dentro do escopo da expressão.

Haskell

Entendendo a palavra reservada in

```
main = do
  let c = 4
  let a = 5; b = 2 in (print (a + b))
  let d = 7
  print a
  print b
  print c
  print d
```

- Note que haverá um erro em tempo de compilação, dizendo que a e b não pertencem ao escopo nas linhas 5 e 6.
- Comente as linhas 5 e 6 e veja o que acontece.

Haskell - Alguns Links Úteis

- https://wiki.haskell.org/Anonymous_function
- <http://learnyouahaskell.com/higher-order-functions>
- https://en.wikibooks.org/wiki/Haskell/Fix_and_recursion

Ver atividade no Moodle