Paradigmas de Programação

Linguagem Haskell

Profª Andréa Schwertner Charão

DLSC/CT/UFSM

Uso de "let"

- Especifica sub-expressões que serão usadas em uma expressão final
- Como em matemática: "seja x ..."
- Exemplo:

```
myfunc a b =
        let x = 3*a
            y = 6*b
        in  x+y
```

Uso de "let"

- Outro exemplo:

```
cylinder :: Float -> Float -> Float
cylinder r h =
        let sideArea = 2 * pi * r * h
            topArea = pi * r^2
        in  sideArea + 2 * topArea
```

http://learnyouahaskell.com/syntax-in-functions

Uso de "where"

- Cláusulas que têm que ser satisfeitas na equação/função
- Como em matemática: "onde x ..."
- Exemplo:

```
funcwhere :: Int -> Int -> Int

funcwhere x y = a + b
        where a = 3*x
              b = 6*y
```

Uso de "where"

- Outro exemplo (sem where X com where):

```
bmiTell1 :: Float -> Float -> String
bmiTell1 weight height
    | weight / height ^ 2 <= 18.5 = "You're underweight, you emo, you!"
    | weight / height ^ 2 <= 25.0 = "You're supposedly normal. Pffft, I bet you're ugly!"
    | weight / height ^ 2 <= 30.0 = "You're fat! Lose some weight, fatty!"
    | otherwise                   = "You're a whale, congratulations!
```

```
bmiTell2 :: Float -> Float -> String
bmiTell2 weight height
    | bmi <= 18.5 = "You're underweight, you emo, you!"
    | bmi <= 25.0 = "You're supposedly normal. Pffft, I bet you're ugly!"
    | bmi <= 30.0 = "You're fat! Lose some weight, fatty!"
    | otherwise   = "You're a whale, congratulations!"
   where bmi = weight / height ^ 2
```

http://learnyouahaskell.com/syntax-in-functions

Exemplo: validação de CPF

- Cálculo do **primeiro** dígito verificador (DV1)

Dígitos CPF	2	2	2	3	3	3	4	4	4
Multiplicadores	10	9	8	7	6	5	4	3	2
Resultado	20	18	16	21	18	15	16	12	8

Somatório = 20+18+16+21+18+15+164+12+8 = 144

DV1 = 144 % 11 (resto da divisão) = 1

DV1 = if DV1 < 2 then 0 else 11-DV1 = 0

DV1 = 0

Exemplo: validação de CPF

- Cálculo do **segundo** dígito verificador (DV2)

Dígitos CPF	2	2	2	3	3	3	4	4	4	0
Multiplicadores	11	10	9	8	7	6	5	4	3	2
Resultado	22	20	18	24	21	18	20	16	12	0

Somatório = 22+20+18+24+21+18+20+16+12+0 = 171

DV2 = 171 % 11 (resto da divisão) = 6

DV2 = if DV2 < 2 then 0 else 11-DV2 = 5

DV2 = 5

CPF: 222.333.444-05

Exemplo: validação de CPF

- Código em Haskell, usando let

```
isCpfOk :: [Int] -> Bool
isCpfOk cpf =

  let -- calcula primeiro digito
      digitos1 = take 9 cpf
      expr1 = mod (sum (zipWith (*) digitos1 [10,9..])) 11
      dv1 = if expr1 < 2 then 0 else 11-expr1

      -- calcula segundo digito
      digitos2 = digitos1 ++ [dv1]
      expr2 = mod (sum (zipWith (*) digitos2 [11,10..])) 11
      dv2 = if expr2 < 2 then 0 else 11-expr2

   in dv1 == cpf !! 9 && dv2 == cpf !! 10
```