Expressões e Funções

Linguagem Haskell

Maria Adriana Vidigal de Lima

Faculdade de Computação - UFU

Agosto - 2009

1. Expressões e Funções
   - Linguagem Haskell

Funções

A definição de uma função deve obedecer à sintaxe seguinte:

```
nome_função :: tipo_arg1 -> ... -> tipo_argN -> tipo_saída
nome_função arg1 ... argN = <expressão_resultado>
```

Sendo `tipo1,...,tipoN` os tipos dos parâmetros de entrada da função.

Considere uma função que retorna o menor entre dois números:

menor 5 4 = 4

menor 0 0 = 0

menor 3 7 = 3

Exemplo

A declaração dos tipos dos valores de entrada e saída da função é definida por:

```
menor :: Int -> Int -> Int
```

Estratégia para a definição do problema: uso de uma *expressão de seleção*

menor x y =
se $x \leq y$ *então o resultado da expressão é x*
senão o resultado da expressão é y

Expressão de Seleção

Sintaxe de uma expressão de seleção bidirecional:

```
if <condição> then
                 <resultado1>
               else
                 <resultado2>
```

Função Menor:

```
menor :: Int -> Int -> Int
menor x y = if x <= y then x
                      else y
```

Expressão de Seleção

Sintaxe de uma expressão de seleção multidirecional (estilo *guardas*):

```
nome_função par1 ... parN
    | <condição1> = <resultado1>
    | ...
    | <condiçãoN> = <resultadoN>
    | otherwise = <resultado>
```

As *guardas* permitem estabelecer uma distinção entre casos diferentes da definição de uma função.

Outro Exemplo

Função para retornar o maior entre três números:

```
maxTres :: Int -> Int -> Int -> Int
maxTres x y z
    | x >= y && x >= z = x
    | y >= z           = y
    | otherwise        = z
```

Avaliação de uma aplicação da função `maxTres`

```
> maxTres 4 3 2
  ?? 4 >= 3 && 4 >= 2
  ??   True && True
  ??   True
  4
```

Outro Exemplo

Avaliação de uma aplicação da função `maxTres`

```
> maxTres 6 (4+3) 5
  ?? 6 >= (4+3) && 6 >= 5
  ?? 6 >= 7 && 6 >= 5
  ?? False && True
  ?? False
  ?? 7 >= 5
  ?? True
  7
```

Neste exemplo avaliamos inicialmente a primeira condição, $6 \geq (4 + 3)$ && $6 \geq 5$ que resultou em *False*; e em seguida avaliamos $7 \geq 5$ que é verdadeira. Assim, o resultado é 7.

Funções do módulo Prelude

even - verifica se um valor dado é par

odd - verifica se um valor dado é impar

rem - resto da divisão inteira

mod - resto da divisão inteira

ceiling - arredondamento para cima

floor - arredondamento para baixo

round - arredondamento para cima e para baixo

truncate - parte inteira do número

Funções

```
Main> floor (6.5)
6

Main> floor (6.5) + 5.5
ERROR - Unresolved overloading
*** Type       : (Fractional a, Integral a) => a
*** Expression : floor 6.5 + 5.5

Main> fromIntegral(floor (6.6)) + 5.5
11.5
```

A função `fromIntegral` converte um inteiro em real.

Funções do módulo Prelude

sin - seno de ângulo em radianos

cos - coseno

tan - tangente

asin - arco seno

acos - arco coseno

atan - arco tangente

abs - valor absoluto

sqrt - raiz quadrada, valor positivo apenas

exp - exponencial base e

Funções do módulo Prelude

log - logaritmo natural (base e)

logBase - logaritmo na base dada

min - menor de 2 objetos dados

max - maior de 2 objetos dados

gcd - máximo divisor comum

lcm - mínimo múltiplo comum