DCC909 – Programação Funcional

AULA 03

Carlos Bruno Oliveira Lopes

Engenheiro de Computação

Mestre em Ciência da Computação

HASKELL

Tipo

- é uma coleção de valores relacionados;
- Eles classificam os valores de acordo com as suas características;

tipo	características	exemplos de valores
`Int`	– inteiros de precisão fixa – limitado (tem um valor mínimo e um valor máximo) – faixa de valores determinada pelo tamanho da palavra da plataforma	`876` `2012`
`Integer`	– inteiros de precisão arbitrária – ilimitado (qualquer número inteiro pode ser representado desde que haja memória suficiente) – menos eficiente que `Int`	`10` `754738748784003045452334209238`
`Float`	– aproximação de números reais em ponto flutuante – precisão simples	`4.56` `0.201E10`
`Double`	– aproximação de números reais em ponto flutuante – precisão dupla	`78643` `987.3201E-60`

`Rational`	– números racionais
	– precisão arbitrária
	– representados como uma razão de dois valores do tipo `Integer`
	– os valores podem ser construídos usando o operador `%` do módulo `Data.Ratio` (precedência 7 e associatividade à esquerda)
	`import Data.Ratio`
		`3 % 4`
		`8 % 2`
		`5 % (-10)`

`Bool`	– valores lógicos	`False` `True`

`Char`	– enumeração cujos valores representam caracteres unicode
	– estende o conjunto de caracteres ISO 8859-1 (latin-1), que é uma extensão do conjunto de caracteres ASCII
		`'B'`
		`'!'`
		`'\n'` nova linha
		`'\LF'` nova linha
		`'\^J'` nova linha
		`'\10'` nova linha
		`'\''` aspas simples
		`'\\'` barra invertida

`String`	– sequências de caracteres	`"Brasil"`
		`""`
		`"bom\ndia"`
		`"altura:\10\&199.4"`
		`"primeiro/ /segundo"`

HASKELL

- Tipos função
  - Uma função possui um tipo que caracterizado pelos tipos de argumentos e pelo tipo do resultado da função;
  - Um tipo função é definido em haskell usando o operador de tipo ->

$$t_1 \ -> \ \ldots \ -> t_n$$

onde

- $t_1, \ldots, t_{n-1}$ são os tipos dos argumentos
- t_n é o tipo do resultado.

Ex.:

Bool -> Bool

tipo das funções com um argumento do tipo Bool, e resultado do tipo Bool.

Int -> Double -> Double -> Bool

tipo das funções com três argumentos, sendo o primeiro do tipo Int e os demais do tipo Double, e o resultado do tipo Bool

HASKELL

Checagem de tipo

- A aplicação de uma função a um ou mais argumentos de tipo inadequado constitui um erro de tipo.
  - Ex.:

```
Prelude> not 'A'

<interactive>:6:5:
    Couldn't match expected type 'Bool' with actual type 'Char'
    In the first argument of 'not', namely 'A'
    In the expression: not 'A'
    In an equation for 'it': it = not 'A'
```

- Haskell é uma linguagem fortemente tipada, com um sistema de tipos muito avançado.

HASKELL

Assinatura de tipo em definições

- Ao fazer uma definição de variável ou função, o seu tipo pode ser definido usando uma assinatura de tipo imediatamente antes da equação;
- A anotação consiste em escrever o nome e o tipo separados pelo símbolo ::
  - Ex.:

```
media2 :: Double -> Double -> Double
media2 x y = (x + y)/2
```

```
notaFinal :: Double
notaFinal = media2 4.5 7.2
```

```
discriminante :: Double -> Double -> Double -> Double
discriminante a b c = b^2 - 4*a*c
```

HASKELL

Consulta do tipo de uma expressão no GHCi

- No GHCi, o comando :type (ou de forma abreviada :t) calcula o tipo de uma expressão, sem avaliar a expressão.
  - Ex.:

```
Prelude> not False
True

Prelude> :type not False
not False :: Bool

Prelude> :type 'Z'
'Z' :: Char
```

```
Prelude> :t 2*(5 - 8) <= 6 + 1
2*(5 - 8) <= 6 + 1 :: Bool

Prelude> :type not
not :: Bool -> Bool

Prelude> :t 69          -- 69 é de qualquer tipo a onde a é um tipo numérico
69 :: Num a => a
```

Questão: Força gravitacional

HASKELL – exercícios

A lei da gravitação universal, proposta por Newton a partir das observações de Kepler sobre os movimentos dos corpos celestes, diz que:

> Dois corpos quaisquer se atraem com uma força diretamente proporcional ao produto de suas massas e inversamente proporcional ao quadrado da distância entre eles.

Essa lei é formalizada pela seguinte equação:

$$F = G\frac{m_1 m_2}{d^2}$$

onde:

- F é força de atração em Newtons (N),
- G é a constante de gravitação universal ($6.67 \times 10^{-11}\,\mathrm{N\,m^2/kg^2}$),
- m_1 e m_2 são as massas dos corpos envolvidos, em quilos (kg), e
- d é a distância entre os corpos em metros (m).

a) Defina uma variável para denotar a constante de gravitação universal.

b) Defina uma função que recebe as massas dos dois corpos e a distância entre eles, e resulta na força de atração entre esses dois corpos. Use a variável definida em *a*).

c) Teste suas definições no ambiente interativo calculando a força de atração entre a terra e a lua sabendo que a massa da terra é 6×10^{24} kg, a massa da lua é 1×10^{23} kg, e a distância entre eles é 4×10^5 km.

Use anotações de tipo apropriadas para os nomes sendo definidos.
Exemplos:

```
forcaGravidade 6e24 1e23 (4e5 * 1000) ⇝ 2.5012499999999997e20
```

HASKELL - exercício

Questão: Salário Líquido

Defina uma função que recebe o salário base de um funcionário e resulta no salário líquido a receber, sabendo-se que o funcionário tem gratificação de 10% sobre o salário base e paga imposto de 7% sobre o salário base.

Use uma anotação de tipo para a função.

Exemplos:

```
salario 1000 ⇝ 1030.0
salario 850  ⇝ 875.5
```

HASKELL

Expressão condicional

– Uma expressão condicional tem forma

$$\texttt{if condição then } exp_1 \texttt{ else } exp_2$$

onde **condição** é uma expressão booleana (predicado) e **exp_1** (consequência) e **exp_2** (alternativa) são expressões de um mesmo tipo.

- O valor da expressão condicional é o valor de **exp_1** se a condição é verdadeira,
- ou o valor de **exp_2** se a condição é falsa.

• Ex.:

```
if True then 1 else 2          ⇝ 1
if False then 1 else 2         ⇝ 2
if 2>1 then "OK" else "FAIL"   ⇝ "OK"
if even 5 then 3+2 else 3-2    ⇝ 1
```

HASKELL

Expressão condicional

– A expressão condicional é uma expressão, e portanto sempre possui um valor.

• Dessa forma, ela pode ser usada dentro de outra expressão.

Ex.:

```
5 * (if True then 10 else 20)          ⇝ 50
5 * if True then 10 else 20            ⇝ 50
length (if 2<=1 then "OK" else "FAIL") ⇝ 4
```

```
(if even 2 then 10 else 20) + 1 ⇝ 11
if even 2 then 10 else 20 + 1   ⇝ 10
```

OBS.: A cláusula **else** de uma expressão condicional não é opicional

```
if True then 10 ⇝ ERRO DE SINTAXE
```

HASKELL

Expressão condicional

– Regra de inferência de tipo

$$\frac{test :: \texttt{Bool} \quad e_1 :: a \quad e_2 :: a}{\texttt{if } test \texttt{ then } e_1 \texttt{ else } e_2 :: a}$$

• Consequência e a alternativa devem ser do mesmo tipo, que significa que é o tipo do resultado.

Ex.:

```
Prelude> :type if 4>5 then 'S' else 'N'
if 4>5 then 'S' else 'N' :: Char

Prelude> :type if odd 8 then "pedro" else "mariana"
if odd 8 then "pedro" else "mariana" :: [Char]

Prelude> :type if mod 17 2 == 0 then length "banana" else 0
if mod 17 2 == 0 then length "banana" else 0 :: Int
```

```
Prelude> if fromEnum 'A' then "ok" else "bad"
<interactive>:2:4:
    Couldn't match expected type 'Bool' with actual type 'Int'
    In the return type of a call of 'fromEnum'
    In the expression: fromEnum 'A'
    In the expression: if fromEnum 'A' then "ok" else "bad"

Prelude> if mod 17 2 /= 0 then not True else 'H'
<interactive>:7:37:
    Couldn't match expected type 'Bool' with actual type 'Char'
    In the expression: 'H'
    In the expression: if mod 17 2 /= 0 then not True else 'H'
    In an equation for 'it':
        it = if mod 17 2 /= 0 then not True else 'H'
```

HASKELL – exercício

Determine o valor e o tipo das expressões seguintes caso a expressão esteja correta. Se a expressão estiver incorreta, indique qual é o problema encontrado.

a) if sqrt (abs (10 - 35) * 100) < 5 then "aceito" else "negado"

b) if pred 'B' then 10 else 20

c) if odd 1 then sqrt 9 else pred 'B'

d) 4 * if 'B' < 'A' then 2 + 3 else 2 - 3

e) signum (if 'B' < 'A' then 2 + 3 else 2) - 3

HASKELL

Definição de função com expressão condicional

Ex.:

- Função para calcular o valor absoluto de um número inteiro pode ser definida como segue:

```
valorAbsoluto :: Int -> Int
valorAbsoluto n = if n >= 0 then n else -n
```

- Função para determinar o sinal de um número inteiro:

```
sinal :: Int -> Int
sinal n = if n < 0
            then -1
            else if n == 0
                   then 0
                   else 1
```

HASKELL – exercício

Defina uma função **max3** que recebe três valores inteiros e resulta no maior deles. Use expressões condicionais aninhadas.

- Faça uma anotação de tipo para a função em seu código.
- Teste sua função no ambiente interativo.

HASKELL

Equações com guardas

- Funções podem ser definidas por meio de equações com guardas, onde uma sequência de expressões lógicas, chamadas guardas, é usada para escolher entre vários possíveis resultados.
- "Espécie de escolha-caso"
- Uma equação com guarda é formada por uma sequência de cláusulas escritas logo após a lista de argumentos.
  - Cada **cláusula** é introduzida por uma barra vertical (|) e consiste em uma condição, chamada guarda, e uma expressão (resultado), separados por =.

$$
\begin{aligned}
f\ arg_1\ \ldots\ arg_n \\
&\mid guarda_1 = exp_1 \\
&\vdots \\
&\mid guarda_m = exp_m
\end{aligned}
$$

Obs.:
- cada guarda deve ser uma ***expressão lógica***, e
- os resultados devem ser todos do ***mesmo tipo***.

HASKELL

$$
\begin{aligned}
f\ arg_1\ \ldots\ arg_n \\
&\mid guarda_1 = exp_1 \\
&\vdots \\
&\mid guarda_m = exp_m
\end{aligned}
$$

Obs.:
- cada guarda deve ser uma ***expressão lógica***, e
- os resultados devem ser todos do ***mesmo tipo***.

Equações com guardas

– Quando a função é aplicada, as guardas são verificadas na sequência em que foram escritas.

Ex.:

- Função para calcular o valor absoluto de um número:

```
vabs :: Integer -> Integer
vabs n | n >= 0 = n
       | n < 0  = - n
```

```
vabs 89
  ?? 89 >= 0
  ?? ⇝ True
⇝ 89
```

  - » Os guardas são:
    - n >= 0
    - n < 0
  - » As expressões associadas são respectivamente:
    - n
    - -n

HASKELL

$$
\begin{aligned}
f\ arg_1\ \ldots\ arg_n \\
&\mid guarda_1 = exp_1 \\
&\vdots \\
&\mid guarda_m = exp_m
\end{aligned}
$$

Obs.:
- cada guarda deve ser uma ***expressão lógica***, e
- os resultados devem ser todos do ***mesmo tipo***.

Equações com guardas

Ex.:

- Função para calcular o valor absoluto de um número:

```
vabs :: Integer -> Integer
vabs n | n >= 0 = n
       | n < 0  = - n
```

- Definição de **vabs** o teste n < 0 pode ser substituído pela constante **True**, pois ele somente será usado se o primeiro teste n >= 0 for falso, e se isto acontecer, com certeza n < 0 é verdadeiro:

```
vabs n | n >= 0     = n
       | otherwise = -n
```

- A condição True pode também ser escrita como **otherwise**:

```
vabs n | n >= 0 = n
       | True   = -n
```

  - Otherwise é uma condição que captura todas as outras situações que ainda não foram consideradas.
  - Ela é definida no prelúdio simplesmente como o valor verdadeiro:

```
otherwise :: Bool
otherwise  = True
```

HASKELL

$$
\begin{aligned}
f\ arg_1\ \ldots\ arg_n \\
&\mid guarda_1 = exp_1 \\
&\vdots \\
&\mid guarda_m = exp_m
\end{aligned}
$$

Obs.:
- cada guarda deve ser uma ***expressão lógica***, e
- os resultados devem ser todos do ***mesmo tipo***.

Equações com guardas

Ex.:

- Função que determinar o sinal de um número inteiro:

```
sinal :: Int -> Int
sinal n | n < 0     = -1
        | n == 0    = 0
        | otherwise = 1
```

- Função para análise do índice de massa corporal:

```
analisaIMC :: Float -> String
analisaIMC imc
  | imc <= 18.5 = "Você está abaixo do peso, seu emo!"
  | imc <= 25.0 = "Você parece normal. Deve ser feio!"
  | imc <= 30.0 = "Você está gordo! Perca algum peso!"
  | otherwise   = "Você está uma baleia. Parabéns!"
```

HASKELL

$$
\begin{aligned}
f\ arg_1\ \ldots\ arg_n \\
&\mid guarda_1 = exp_1 \\
&\vdots \\
&\mid guarda_m = exp_m
\end{aligned}
$$

Obs.:
- cada guarda deve ser uma ***expressão lógica***, e
- os resultados devem ser todos do ***mesmo tipo***.

Equações com guardas

– Se todas as guardas foram testada, ou seja, todas as opções foram falsas, ocorre um erro em tempo de execução.

- O erro ocorre porque o programa tentará executar a próxima opção e a mesma não existirá e um erro será reportado.

Ex.:

```
funcaoTeste :: Int -> Int -> Int
funcaoTeste x y | x > y = 1
                | x < y = -1
```

```
funcaoTeste 2 3 ⇝ -1
funcaoTeste 3 2 ⇝ 1
funcaoTeste 2 2 ⇝ ERRO
```

HASKELL – exercício

Defina uma função **min3** que recebe três valores inteiros e resulta no menor valor deles. Utilize equações com guardas.

- Faça uma anotação de tipo para a função em seu código.
- Teste sua função no ambiente interativo.

HASKELL – exercícios

Redefina a função a seguir usando guardas no lugar de expressões condicionais.

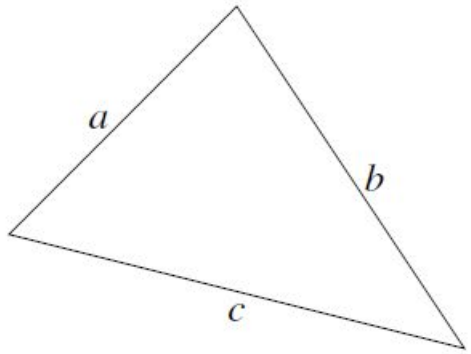
```
describeLetter :: Char -> String
describeLetter c =
  if c >= 'a' && c <= 'z'
    then "Lower case"
    else if c >= 'A' && c <= 'Z'
           then "Upper case"
           else "Not an ASCII letter"
```

HASKELL

Definições locais e guardas

- Uma equação pode ter definições locais que são introduzidas na cláusula **where**;
  - O escopo dos nomes definidos localmente restringe-se à equações definidas dentro da cláusula **where**.

HASKELL



Definições locais e guardas

Ex.:

- Função que calcula a área de um triângulo somente quando as medidas dos lados do triângulo são válidas. Isto acontece somente quando cada medida é positiva e menor do que a soma das medidas dos outros dois lados. Caso as medidas não sejam válidas o resultado da função é zero.
- Utilizando a fórmula de Hieron $A = \sqrt{s(s-a)(s-b)(s-c)}$ e $s = \frac{a+b+c}{2}$ para o cálculo da área A, sendo a, b e c as medidas dos lados, e s o semiperímetro do triângulo.

```
areaTriangulo a b c
  | medidasValidas = sqrt (s * (s-a) * (s-b) * (s-c))
  | otherwise      = 0
  where
    medidasValidas = a > 0 && b > 0 && c > 0 &&
                     a < b + c &&
                     b < a + c &&
                     c < a + b
    s = (a + b + c)/2
```

HASKELL

Definições locais e guardas

Ex.:

– Função para análise do índice de massa corporal:

```haskell
analisaIMC peso altura
  | imc <= 18.5 = "Você está abaixo do peso, seu emo!"
  | imc <= 25.0 = "Você parece normal. Deve ser feio!"
  | imc <= 30.0 = "Você está gordo! Perca algum peso!"
  | otherwise   = "Você está uma baleia. Parabéns!"
  where
    imc = peso / altura^2
```

```haskell
analisaIMC peso altura
  | imc <= magro  = "Você está abaixo do peso, seu emo!"
  | imc <= normal = "Você parece normal. Deve ser feio!"
  | imc <= gordo  = "Você está gordo! Perca algum peso!"
  | otherwise     = "Você está uma baleia. Parabéns!"
  where
    imc    = peso / altura^2
    magro  = 18.5
    normal = 25.0
    gordo  = 30.0
```

HASKELL – exercício

Questão: Número de raízes reais da equação do segundo grau

Defina uma função chamada `numRaizes` que recebe os três coeficientes de uma equação do segundo grau

$$ax^2 + bx + c = 0$$

e calcula a quantidade de raízes reais distintas da equação. Assuma que a equação é não degenerada (isto é, o coeficiente do termo de grau dois não é zero).

Use uma definição local para calcular o discriminante da equação.

$$\Delta = b^2 - 4ac$$

Se Δ for positivo a equação tem duas reais reais e distintas, se for nulo, a equação tem uma raiz real, e se for negativo, a equação não tem raízes reais.

Especifique o tipo da função.

HASKELL – Exercícios

1. Defina uma funções para o calcula a média dos alunos dados três notas como entrada. A função deve verificar quais são as duas maiores notas das três e calcular média usando essas notas.
2. Defina uma função que calcula a área a hipotenusa de um triângulo somente quando as medidas dos lados do triângulo são válidas. Isto acontece somente quando cada medida é positiva e menor do que a soma das medidas dos outros dois lados. Caso as medidas não sejam válidas o resultado da função é zero.
3. Defina uma função que verifique se um triângulo é equilátero, ou seja, quando os seus três lados são congruentes (isto é, iguais).
4. Defina uma função que calcula o salário líquido de uma pessoa dado como entrada o salário bruto. Os desconto sobre o contribuinte são realizados de acordo com as seguintes regras:
   - Salário até R$1500,00 estão isentos de imposto (ou seja, não pagam);
   - Salário até R$3000,00 será taxado com 15% de imposto;
   - Salário acima de R$3000,00 será taxado com 27,5% de imposto.