Programação Funcional - Tipos

```
universidade = "Universidade Federal de Alfenas"
professor = "Romário da Silva Borges"
```

Aprenderemos nesta aula...

- Compreender o uso de Tipos em Haskell;

Tipos

As linguagens funcionais modernas apresentam um sistema de tipos robusto que permite ao compilador **verificar se os operandos usados nas operações são consistentes**.

consistência = maior previsibilidade e segurança!

Características

- **Fortemente tipada:** O compilador impede operações entre tipos incompatíveis. (mais segurança e previsibilidade)

- **Tipagem Estática:** Todos os tipos são verificados em tempo de compilação. (erros detectados precocemente)

- **Inferência de Tipos:** O compilador deduz tipos sem necessidade de anotação explícita. (não é necessário declarar o tipo o tempo todo)

Características

- **Polimorfismo paramétrico:** Funções genéricas funcionam com múltiplos tipos. (reutilização, generalização e evita duplicação). Ex: length :: [a] -> Int
- **Classes de Tipos**: Um sistema que define conjuntos de operações que certos tipos devem implementar. (Permite criar tipos próprios).
- **Imutabilidade:** Os valores não mudam, toda atribuição cria um novo valor. (evita efeitos colaterais, facilita paralelismo sem conflitos de estados)

Inferência de Tipos

Exemplo do uso de Inferência de Tipo:

calculaDoisValores a b = a + b

Devido ao operador de soma (+), o compilador infere o uso mais geral e, portanto, restringe seus tipos a qualquer valor que pertença à classe **Num** (Int, Float, Double, Integer...)

Tipos

Por quê declarar os tipos, se o próprio compilador infere?
Mesmo que o compilador consiga inferir, declarar explicitamente o tipo traz benefícios de:

- **Clareza,** documentando o que a função fez;
- **Segurança,** evitando inferências inesperadas;
- **Controle,** restringindo a generalização;
- **Boas práticas,** torna o código mais previsível;

Assinatura de tipo de função

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

```
somaCincoValores :: Int -> Int -> Int -> Int -> Int -> Int
somaCincoValores a b c d e = a + b + c + d + e
```

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

Consulta do tipo de uma expressão em GHCI

No GHCi, o comando :t calcula o tipo de uma expressão, sem avaliar a expressão.

```
:t
'a' :: Char

:t 71
71 :: Num a => a
```

Considerações finais

O sistema de tipos em Haskell não é apenas um mecanismo de segurança, mas parte essencial da filosofia funcional da linguagem. Os tipos **descrevem o comportamento das funções** e **valores**, garantindo que a programação funcional possa ser feita de forma **segura**, **pura** e **previsível**.

Hora de praticar!

Escreva o tipo de cada função abaixo

- buscarLetrasCaixaAlta lista = [x | x <- lista, elem x ['A'..'Z']]
- circunferencia r = 2 * pi * r
- fatorial n = product [1..n]

Teste os seguintes valores: True, “Haskell”, ‘a’, 1 == 1, (False, “muito bom”, 10) com o comando :t

Referências

- Malaquias, J. R. (2018). *Programação Funcional em Haskell.* Universidade Federal de Ouro Preto, Departamento de Computação.
- DU BOIS, André Rauber. *Programação Funcional com a Linguagem Haskell.* Porto Alegre: Bookman, 2015.

Programação Funcional