UDESC

Introdução à Programação Funcional usando

Aulas 3 e 4

Declaração de Funções e Recursão

Declaração de Função

Identificador de variável: Letra minúscula ou _ seguida por letras ou dígitos ou _ ou ‘

varId pat_1 pat_2 ... pat_n = expr ⟶ Expressão

Sequência de zero ou mais padrões, que podem ser **identificadores de variáveis,** construtores de dados ou constantes.

UDESC

Declaração de Função

```
maior a b = if a > b then a else b

> maior 10 5
```

UDESC

Declaração de Função

```
negacao x = if x == True then False else True
```

```
negacao x = if x then False else True
```

UDESC

Casamento de Padrões
(*Pattern Matching*)

```
negacao True  = False
negacao False = True
```

```
> negacao False
```

UDESC

Combinando Funções

```
maior a b = if a > b then a else b

maior3 a b c = if a > b then (if a > c then a else c)
                        else (if b > c then b else c)
```

UDESC

Combinando Funções

```
maior a b = if a > b then a else b

maior3 a b c = maior (maior a b) c
```

Definição Recursiva

Uma definição recursiva é uma definição que faz referência ao próprio objeto que está sendo definido, a definição deve conter um ou mais casos base que serão alcançados após sucessivas aplicações da definição.

Por exemplo: O fatorial de um inteiro positivo pode ser definido de forma recursiva como:

$0! = 1$

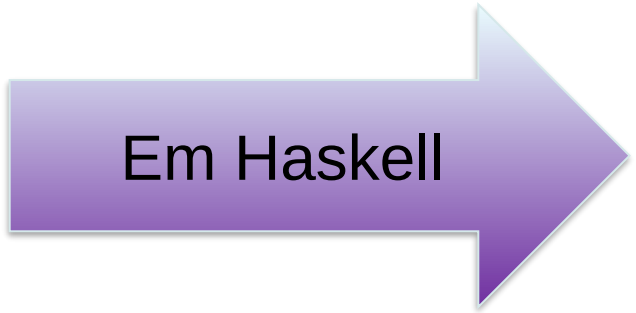
$n! = n\,(n - 1)!$

UDESC

Funções Recursivas

$$0! = 1$$

$$n! = n\,(n - 1)!$$




```
fat 0 = 1
fat n = n * fat (n-1)
```

UDESC

Funções Recursivas

```
fat 0 = 1
fat n = n * fat (n-1)

> fat 3
    3 * fat 2
    3 * 2 * fat 1
    3 * 2 * 1 * fat 0
    3 * 2 * 1 *   1
```

Funções Recursivas

Potência com expoente inteiro:

$$a^0 = 1$$

$$a^e = aa^{e-1}$$

Em Haskell

```
pot a 0 = 1
pot a e = a * pot a (e-1)
```

```
pot _ 0 = 1
pot a e = a * pot a (e-1)
```

Funções Recursivas

```
pot _ 0 = 1
pot a e = a * pot a (e -1)
```

```
> pot 2 3
  2 * pot 2 2
  2 * 2 * pot 2 1
  2 * 2 * 2 * pot 2 0
  2 * 2 * 2 *     1
```

Funções Recursivas

O **Máximo Divisor Comum** de 2 números inteiros positivos, diferentes de 0, pode ser obtido pelo **Algoritmo de Euclides** descrito no livro VII da obra Elementos (≈ 300 A.C.).

a	b	
21	9	21 – 9 = 12
12	9	12 – 9 = 3
3	9	9 – 3 = 6
3	6	6 – 3 = 3
3	3	

$$mdc(a, b) = \begin{cases} a = b, a \\ a > b, mdc(a - b, b) \\ a < b, mdc(a, b - a) \end{cases}$$

Funções Recursivas

$$mdc(a, b) = \begin{cases} a = b, a \\ a > b, mdc(a - b, b) \\ a < b, mdc(a, b - a) \end{cases}$$

```
mdc a b = if a == b then a else (if a > b then mdc (a - b) b else mdc a (b - a))
```

UDESC

Funções Recursivas

$$\text{mdc}(a, b) = \begin{cases} a = b, a \\ a > b, \text{mdc}(a - b, b) \\ a < b, \text{mdc}(a, b - a) \end{cases}$$

```
mdc a b  | a == b = a
         | a > b   = mdc (a - b) b
         | a < b   = mdc a (b - a)
```

Funções Recursivas

$$\text{mdc}(a, b) = \begin{cases} a = b, a \\ a > b, \text{mdc}(a - b, b) \\ a < b, \text{mdc}(a, b - a) \end{cases}$$

```
mdc a b  | a == b      = a
         | a > b       = mdc (a - b) b
         | otherwise = mdc a (b - a)

otherwise = True
```