Haskell: Padrões Simples e Definição de Operadores

UFRN, 2018

Operadores

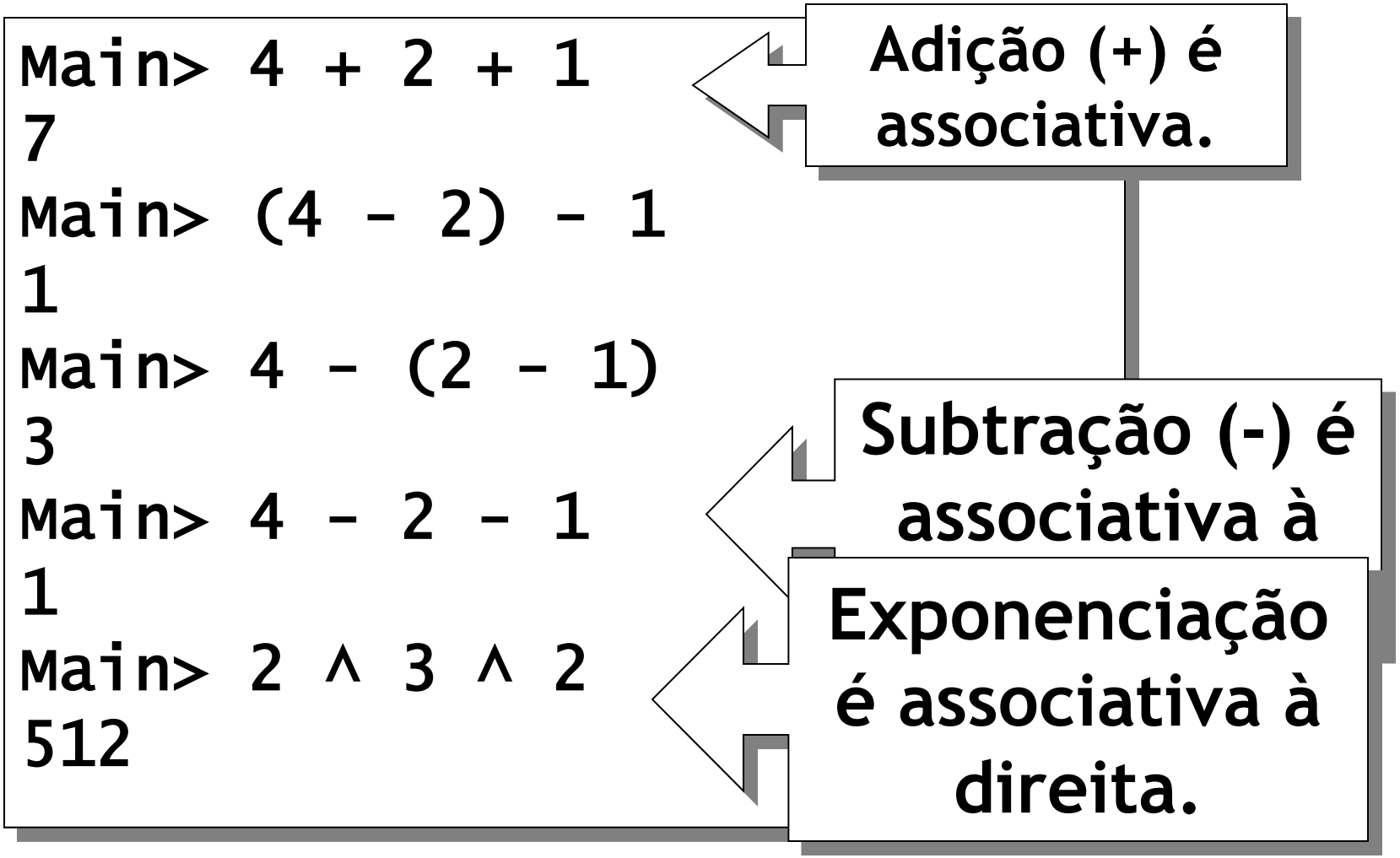
- A linguagem Haskell contém vários operadores, como: +, -, *, /, `div`, `mod`, ^, etc.
- Operadores são infixos, o que significa que são escritos entre os seus argumentos.
- Parênteses podem ser usados em qualquer operação: (((4+8)*3)+2)
- Propriedades que auxiliam na eliminação de parênteses: associatividade e prioridade de operadores.

Associatividade

- **Uma operação op é associativa se**

  (*x* op *y*) op *z* = *x* op (*y* op *z*)
- **Operações associativas dispensam parênteses.**
- **Uma operação não associativa é classificada como:**
  - **associativa à esquerda; ou**
  - **associativa à direita.**

Associatividade



Precedência de Operadores

```
Main> 2 + 3 * 4
14
Main> 2 ^ 3 * 4
32
Main> fat (-1)+1
2
```

* tem precedência 7, + tem prioridade 6.

Operador ^ tem precedência 8.

Funções têm precedência máxima.

```
1  fat :: Int -> Int
2  fat n
3    | n <= 0    = 1
4    | otherwise = n * fat (n-1)
```

Definindo Operadores

- Haskell permite a definição de novos operadores.
- Os nomes dos operadores são formados por: `! # $ % & * + . / < = > ? @ \ ^ | : - ~`
- Restrições:
  - nomes não podem começar por ":"
  - "-" e "~" só podem ser o primeiro símbolo
  - símbolos (e combinações) reservados:

    `:: => = @ \ | ^ <- ->`
- Mudar associatividade ou precedência:

  `infix  infixl  infixr`

Definindo Operadores

```
1  infixl 7 &&&
2
3  (&&&) :: Int -> Int -> Int
4  a &&& b
5    | a > b          = a
6    | otherwise      = b
```

```
Main> 10 &&& 20
20
```

Definições com Padrões

- O padrão "_" casa com qualquer argumento.
  - É usado quando o valor do argumento não é necessário no lado direito da equação.

```
1 isZero :: Int -> Bool
2 isZero 0   = True
3 isZero _   = False
```

Definições com Padrões

- Padrões e casamento de padrões:

```
1  fib :: Int -> Int
2
3  fib 0 = 0
4  fib 1 = 1
5  fib n
6    | n > 1 = fib(n-1) + fib(n-2)
7    | otherwise = 0
```

Programamando com Booleanos

- Os valores booleanos são representados pelas constantes `False` e `True`, do tipo `Bool`.
- Operadores:

  `&&` ("e" lógico)

  `||` ("ou" lógico)

  `not` (negação)

Programando com Booleanos

```
1 meuNot :: Bool -> Bool
2 meuNot True = False
3 meuNot False = True
```

```
4 exOr False x = x
5 exOr True x = not x
```

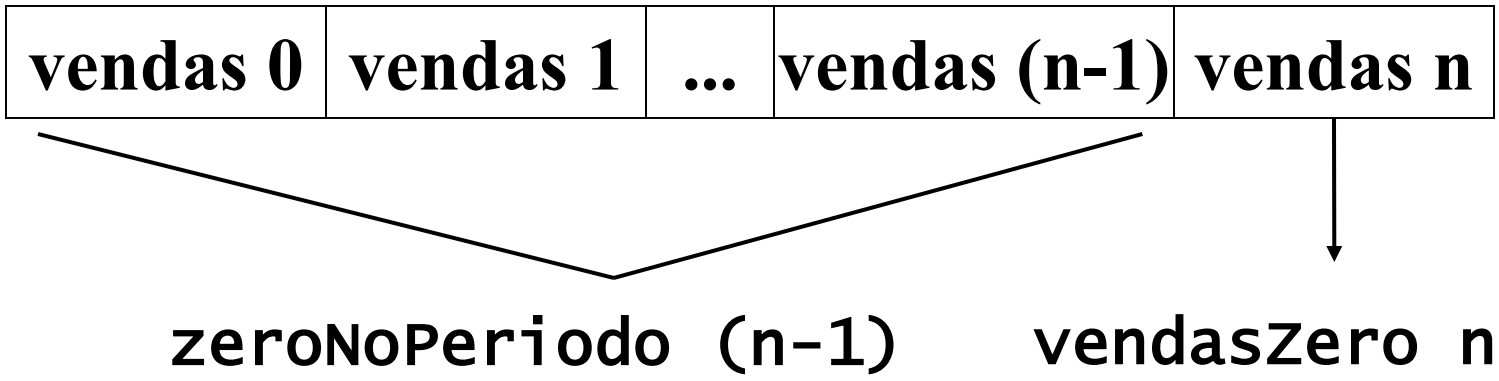
Programando com Booleanos

- Construir a função `zeroNoPeriodo` que, dado o número n de um dia, retorne:
  - True, se há algum dia do período 0,...,n em que não houve venda;
  - False, caso contrário.

```
1 zeroNoPeriodo :: Int -> Bool
2 zeroNoPeriodo 0 = vendasZero 0
3 zeroNoPeriodo n = ???????????
```

Programando com Booleanos

```
1 zeroNoPeriodo :: Int -> Bool
2 zeroNoPeriodo 0 = vendasZero 0
3 zeroNoPeriodo n = zeroNoPeriodo (n-1)
                    || vendasZero n
```



Caracteres e Strings

- **Char : tipo de Haskell associado a caracteres.**
  - **Caracteres individuais são inseridos em aspas simples. Exemplos: ´d´, ´3´.**
- **Caracteres especiais:**

´\t´	tab
´\n´	new line
´\\´	barra invertida
´\´´	aspa simples
´\"´	aspa dupla
´\97´	caractere ´a´

Caracteres e Strings

- Codificação padrão ASCII
- Funções chr e ord (: load Data.Char).

```
1 desl = ord ´A´ - ord ´a´
2
3 maiuscula :: Char -> Char
4 maiuscula ch = chr (ord ch + desl)
```

```
5 eDigito :: Char -> Bool
6 eDigito ch = (´0´<= ch) && (ch <=´9´)
```

Caracteres e Strings

- Strings de caracteres pertencem ao tipo `String`.
- São inseridas entre aspas duplas. Exemplos:

  "abcdef"

  "uma linha\noutra linha"

  ""

  "O caractere \´a\´ : \97"
- Strings são concatenadas usando o operador ++.

Caracteres e Strings

- Haskell permite que novos tipos sejam criados, usando a palavra-chave `type`.
- O tipo `String` é, na realidade, uma lista de caracteres:

```
type String = [Char]
```

- As operações que manipulam listas podem ser usadas também para strings.

Números de Ponto Flutuante

- Números de ponto flutuante são representados, em Haskell, pelos tipos `Float` e `Double`. Exemplos:

  ```
  3.141592, -1.2345
  ```

- Haskell usa também notação científica:

  `9.87654e02` = $9.87654 * 10^{2}$ = 987.654

  `31415.92e-4` = $31415.92 * 10^{-4}$ = 3.141592

- Haskell oferece uma série de funções que atuam sobre números de ponto flutuante.

Números de Ponto Flutuante

```
Main> sin (pi/2)
1.0
Main> 1.1 ^ 10
2.59374
Main> 1.1 ** 10.0
2.59374
Main> sqrt 2
1.41421
```

```
Main> round 2.49
2
Main> ceiling 2.49
3
Main> floor 2.49
2
Main> log 2
0.693147
main> logBase 2 16
4.0
```