Tipos de Dados

João Marcelo Uchôa de Alencar

Universidade Federal do Ceará - Quixadá

25 de Abril de 2017

Tipos de Dados

Tipos Básicos

Tipos Sinônimos

Tipos Estruturados

Tipos Estruturados

Operadores

Exercícios Propostos

Tipos de Dados

- Haskell é fortemente tipada: toda função, variável, constante, etc., tem um só tipo de dado, e é sempre possível determinar seu tipo.
- A principal vantagem é a correção de programas.
- Os compiladores e interpretadores Haskell possuem um sistema de dedução automática para funções que não tem o tipo definido pelo programador.
- Essa dedução só é possível devido a *hierarquia de tipos*. Mas primeiro vamos revisar os tipos *primários* ou *básicos*.

Tipos Básicos

Tipo	**Descrição**	Exemplos
Bool	Booleano	True
Int	Inteiro Limitado	2004
Integer	Inteiro Ilimitado	28904374642004
Float	Real	21.234
Char	Caractere	'h'
String	Cadeia de Caracteres	"Haskell"
Void	Valor Indefinido	_

Tipos Básicos

$nome\_da\_funcao :: Tipo_{arg1} -> Tipo_{arg2} -> Tipo_{arg3} ... -> Tipo_{arg_{saida}}$

- Nome da função, seguido do operador de instância de tipos.
- n tipos como argumentos.
- Tipo final representa a saída.

Booleanos

```
&& :: Bool -> Bool -> Bool
|| :: Bool -> Bool -> Bool
not :: Bool -> Bool
```

- O operador *not* tem prioridade maior que &&, que tem maior prioridade que ||.
- As operações lógicas de mesma prioridade sempre são executadas da esquerda para a direita, exceto quando parênteses são usados.
- Para os operadores $==, /=, <, >, <=, >=$, o True se comporta como 1 e o False como 0.

Inteiros

- O tipo Int tem um limitação de valor equivalente a 2147483647. Para evitar o limite, o tipo usado deve ser Integer.
- Para não haver confusão com a operação de subtração, os números negativos são escritos entre parênteses.

Inteiros

Operador	Descrição	Exemplos
+	Soma	$1 + 2 = 3$
-	Subtração	$5 - 1 = 4$
*	Multiplicação	$2 * 3 = 6$
^	Exponenciação com expoente inteiro	$6 \hat{} 3 = 216$
div	Parte inteira da divisão	div 20 3 = 6
mod	Resto da divisão inteira	mod 20 3 = 2
abs	Valor absoluto	abs (-987) = 987
negate	Inverte o sinal	negate 31 = -31

- O - pode ser tanto unário quanto binário.
- É um operador sobrecarregado.

Divisão entre Inteiros

```
div_1 a b = a / b
div_2 a b = (fromIntegral a) / b
div_3 :: Float -> Float -> Float
div_3 a b = a / b
```

- A primeira opção não tem definição de tipo. Haskell deduz um tipo que permita a divisão.
- fromIntegral faz conversão de inteiros para Fractional.
- A terceira opção declara os tipos como Float. A conversão é direta.

Precedência ou Prioridade para Inteiros

- Avaliação da esquerda para a direita em operadores com mesma prioridade.
- Ordem de prioridade: negação unária - , ^, *, mod, + e - binário.

Operadores Relacionais

Operador	Descrição	Exemplos
$>$	Maior do que	$3 > 2 = \text{True}$
$>=$	Maior e igual a	$3 >= 2 = \text{True}$
$<$	Menor do que	$3 < 2 = \text{False}$
$<=$	Menor e igual a	$3 <= 2 = \text{False}$
$==$	Igual a	$3 == 2 = \text{False}$
$/=$	Diferente de	$3 /= 2 = \text{True}$

Caracteres

Funções definidas no módulo Data.Char

```
-- Converte caractere em número na tabela ASCII
ord :: Char -> Int
-- Converte número ASCII em caractere
chr :: Int -> Char
```

Funções sobre Caractere

Função	Descrição	Exemplos
isLower	Verdade se caractere for minúsculo, falso caso contrário.	isLower 'a' = True
isUpper	Verdade se caractere for maiúsculo, falso caso contrário.	isUpper 'a' = False
toLower	Converte para minúsculo.	toLower 'A' = 'a'
toUpper	Converte para maiúsculo.	toUpper 'a' = 'A'
isDigit	Verifica se é um dígito ('0' a '9').	isDigit '1' = True
digitToInt	Transforma de '0' a '9' em inteiro.	digitToInt '1' = 1
intToDigit	Inverso da anterior	intToDigit 1 = '1'

Reais

Precisão simples é o tipo Float, dupla é Double.

Função	Descrição	Exemplos
+, -, *	Igual aos Inteiros	(+) 13.345 23 = 36.345
/	Divisão fracional	6 / 3 = 2.0
^	Base real, expoente inteiro.	(^) 0.987 56 = 0.48...
**	Base e expoente reais.	(**) 0.765 4.56 = 0.29...
==, / =, <, >, <=, >=	Igual aos inteiros.	3.0 == 3.1 = False
abs	Valor absoluto	abs(-5.67) = 5.67
acos,cos,asin,sin,...	Funções trigonométricas e inversas	sin (2 * pi) = 1.22...
exp	Potências na base *e*	exp 3 = 20.0855
fromIntegral	Converte Int em Float	fromIntegral 3 = 3.00
log	Logaritmo na base *e*	log 10 = 2.30259
logBase	Logaritmo em base qualquer	log 2 10 = 3.32193
negate	Inverte sinal	negate 2.3 = (-2.3)
pi	Retorna o valor de pi	pi = 3.14159...
signum	1.0, 0.0 ou -1.0	signum(-3.6) = -1.0
sqrt	Raiz quadrada	sqrt 4 = 2.0
ceiling, floor, round	Arrendodamentos	...

Tipos Sinônimos

- A declaração *type* permite associar um novo identificador a um tipo já existente.
- Ajuda na legibilidade e documentação do programa.

Tipos Estruturados

- Construídos pelo construtor *data*.
- Permitem criar enumerações.

```
data Cor = Verde | Azul | Amarelo
   deriving (Eq, Show)
corBasica :: Cor -> Bool
corBasica c = (c == Verde || c == Azul || c == Amarelo)
```

Tipos Estruturados

```haskell
data Shape = Circle Float Float Float | Rectangle Float Float Float Float
surface :: Shape -> Float
surface (Circle _ _ r) = pi * r ^ 2
surface (Rectangle x1 y1 x2 y2) = (abs $ x2 - x1) * (abs $ y2 - y1)
```

Operadores

- Podemos construir novos operadores.
- Símbolos permitidos: ! # $ % * + . / < = > ? @ \ ^ | : - ~
- infix: define operador não associativo.
- infixl: associativo à esquerda.
- infixr: associativo à direita.

Operadores

```
infixl 7 &&& -- grau de prioridade
(&&&) :: Int -> Int -> Int
a &&& b | a > b = a
        | otherwise = b
```

Exercícios Propostos

- Vamos fazer o 1, 3, e 6