

Programação Funcional em Haskell

Tipos de Dados

Maria Adriana Vidigal de Lima

Maio - 2009

1 Tipos Básicos

2 Bibliografia

Introdução

- A linguagem Haskell possui uma disciplina rigorosa de tipos de dados, sendo *fortemente tipada*. Neste caso, toda função, variável, constante tem apenas um tipo de dado, que sempre pode ser determinado.
- Embora *fortemente tipada*, a linguagem Haskell possui um sistema de dedução automática de tipos para funções cujos tipos não foram definidos.
- A partir dos tipos pré-definidos na linguagem Haskell, novos tipos podem ser construídos pelo programador.

Tipos primitivos em Haskell

Tipo	Descrição	Exemplos
Caracter	Char	'a', 'z', 'A', '3'
Booleano	Bool	True, False
Reais	Double	-18412.356626, 12.54
	Float	0.0, 456.235,
Inteiros	Integer	4537687898, -7
	Int	3, 21475
Cadeia de caracteres	String	"Haskell"

Para definir que uma expressão E tem o tipo T (E é do tipo T) escreve-se:

$$E :: T$$

Prototipação de tipos

Toda função definida em Haskell têm uma prototipação de tipos, que segue a sequência dos argumentos da função, sendo o último tipo o do valor de retorno da função.

nome_da_funcao :: Tipo_{arg₁} -> Tipo_{arg₂} ... Tipo_{arg_{saída}}

Prototipação de tipos - Exemplos

nome_funcao :: Tipo_{arg₁} -> Tipo_{arg₂} ... -> Tipo_{arg_{saida}}

-- Função que verifica se um número inteiro é par

par::Int->Bool

par x = if mod x 2 == 0 then True else False

-- Função para converter um valor Fahrenheit em Celsius

converteFC::Float->Float

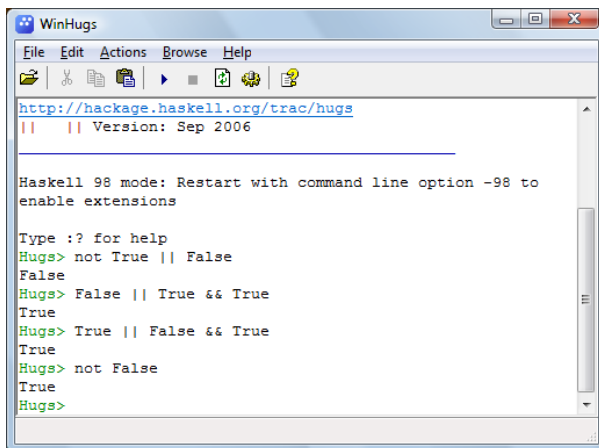
converteFC x = (x - 32)/ 1.8

Tipo Booleano

- O tipo booleano é representado pela abreviatura **Bool** e possui somente dois valores: **True** e **False**.
- Os valores booleanos resultam de testes condicionais de comparações. As operações definidas sobre booleanos são: e (&&), ou (||) e negação (*not*).
- As prototipações são dadas por:

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

Combinações entre True e False



The screenshot shows a window titled "WinHugs" with a menu bar (File, Edit, Actions, Browse, Help) and a toolbar. The main text area contains the following text:

```
http://hackage.haskell.org/trac/hugs  
|| || Version: Sep 2006
```

```
Haskell 98 mode: Restart with command line option -98 to  
enable extensions
```

Type :? for help

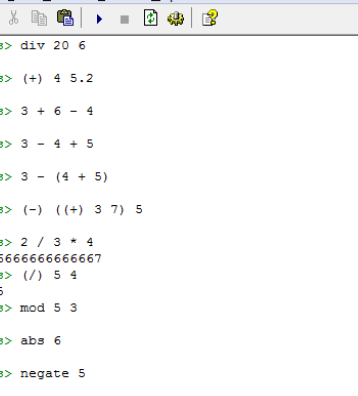
```
Hugs> not True || False  
False  
Hugs> False || True && True  
True  
Hugs> True || False && True  
True  
Hugs> not False  
True  
Hugs>
```


Tipo Inteiro

- Os inteiros são referenciados por *Int*, o domínio matemático dos inteiros (até 2147483647).

Operador	Descrição	Exemplos
+	Soma entre dois inteiros	$1 + 2 = 3$
-	Subtração entre dois inteiros	$(-) 5 - 4 = 1$
*	Multiplicação entre dois inteiros	$2 * 2 = 4$
^	Exponenciação (base qualquer e expoente inteiro)	$3.2^3 = 32.768$
div	Parte inteira da divisão	<code>div 20 3 = 6</code>
mod	Resto da divisão	<code>mod 20 3 = 2</code>
abs	Valor absoluto de um inteiro	<code>abs (-89) = 89</code>
negate	Inverte o sinal de um inteiro	<code>negate (13) = -13</code>

Exemplos de avaliação



The screenshot shows the WinHugs application window. The title bar is 'WinHugs'. The menu bar includes 'File', 'Edit', 'Actions', 'Browse', and 'Help'. The toolbar contains icons for file operations (new, open, save, print, delete), navigation (back, forward), and help. The main text area displays the following R session transcript:

```
Hugs> div 20 6
3
Hugs> (+) 4 5.2
9.2
Hugs> 3 + 6 - 4
5
Hugs> 3 - 4 + 5
4
Hugs> 3 - (4 + 5)
-6
Hugs> (-) ((+) 3 7) 5
5
Hugs> 2 / 3 * 4
2.666666666666667
Hugs> (/) 5 4
1.25
Hugs> mod 5 3
2
Hugs> abs 6
6
Hugs> negate 5
-5
Hugs>
```

Bibliografia Utilizada

Cláudio Cesar de Sá, Márcio Ferreira da Silva, *Haskell - Uma Abordagem Prática*, Novatec Editora, 2006.