Linguagem de Programação
Introdução à Linguagem Hugs98

TURMA A
PROF. MARCELO LADEIRA CIC/UNB

Tipo Booleano

```
(&&), (||)      :: Bool -> Bool -> Bool

True  && x      =  x
False && _      =  False

True  || _      =  True
False || x      =  x

not             :: Bool -> Bool
not True        =  False
not False       =  True

otherwise       :: Bool
otherwise       =  True
```

Tipo Char

É uma enumeração e consiste de valores de 16 bits, conforme o padrão unicode.

É representado pelo caractere entre aspas simples: ‘a’, ‘b’, ‘A’, ‘0’, ‘1’, ..., ‘9’, etc.

Cada um dos caracteres de controle ascii tem mais de uma representação possível:

- Ex.:
  ‘\7’, ‘\a’ e ‘\BEL’, ‘\b’ e ‘\BS’, ‘\f’ e ‘\FF’, ‘\r’ e ‘ \CR’, ‘\t’ e ‘\HT’, ‘\v’ e ‘\VT’, ‘\n’ e ‘\LF’

Funções de conversão: chr e ord

- disponível após “import Data.Char” ou “:l Data.Char”
  chr :: Int → Char
  ord :: Char → Int

Tipo String

É uma lista de caracteres

type String = [Char] é um tipo sinônimo

Strings podem ser abreviadas envolvendo os caracteres por aspas

" string” abrevia a notação [‘ ‘,’s’,’t’,’r’, ‘i’,’n’,’g’]

Todas as operações para lista se aplicam a strings

Strings e I/O

String são objetos visíveis: podem ser lidos ou impressos.

Exemplo:

```
Prelude> let { leia = do
                        putStr "informe uma string >";
                  str <- getLine; putStr str}
         in leia
informe uma string > lah vai a string, pega!
 lah vai a string, pega! :: IO ()
Prelude> putStr "Isto eh uma string"
Isto eh uma string :: IO ()
```

Strings e I/O

Qualquer objeto para ser impresso deve antes ser convertido em string

Exemplo:

```
Prelude> putStr (show [1,2,3,4,5])
  [1,2,3,4,5] :: IO ()
Prelude> read "[1,2,3,4]" :: [Int] -- converte string em objeto
  [1,2,3,4] :: [Int]
Prelude> show (23,5.4) -- converte objeto em string
  "(23,5.4)" :: [Char]
Prelude> read "[('a','b',4.5),('c','d',6.0)]" ::[(Char,Char,Float)]
  [('a','b',4.5),('c','d',6.0)] :: [(Char,Char,Float)]
```

Tipos Numéricos

Int e Integer

- Int tem valores limitados
- Integer tem valores ilimitado

Operadores para inteiros

+, -, *, /, ^, negate, div, rem, mod, odd, even, abs, etc

Funções diversas existem no prelude para inteiros

Float

Complex

Tipo Listas

É um tipo algébrico de dois construtores: '[]' e ':'

Há muitas funções no Prelude para manipular listas

Exemplos:

```
[ ]                                  lista vazia ou nula
[3]   == 3:[]                                lista com 1 elemento
[1,2,3,4,5] == 1:2:3:4:5:[]      lista com 5 elementos numéricos
Prelude> [1,3..15]
  [1,3,5,7,9,11,13,15] :: [Integer]
Prelude> ['a'..'z']
  "abcdefghijklmnopqrstuvwxyz" :: [Char]
Prelude> [0.1,0.3 .. 2.0]
  [0.1,0.3,0.5,0.7,0.9,1.1,1.3,1.5,1.7,1.9] :: [Double]
Prelude> length [1..10]
  10 :: Int
```

Tipo Listas

É um conjunto ordenado e homogêneo de elementos com repetição permitida.

Se t é um tipo, então [t] é uma lista de elementos do tipo t.

Operadores e funções sobre listas

length, ++, :, concat, filter, head, tail, last, take,
drop, replicate, reverse, zip, unzip, elem, and, or,
foldl1, foldl, etc.

Operadores e Funções em Tipos Listas

length [1..1000]
1000

[1,3..10]++[2,4..8]
[1,3,5,7,9,2,4,6,8]

[1,2]:[2,3]:[]
[[1,2],[2,3]]

filter even [1..10]
[2, 4, 6, 8, 10]

concat [['1'],['2'],['3']," abc"," efg"]
"123 abc efg“

head (tail [1,5..200])
5

drop 2 (take 4 ["jan","fev","mar","abr","mai","jun"])

last (reverse [1,10..200])

replicate 5 ‘a’
“aaaaa”

replicate 3 “a”
[“a”, “a”, “a”]

elem 13 [1,3..20]
True

Operadores e Funções em Tipos Listas

zip [1,3..10] [0,2..15]
[(1,0),(3,2),(5,4),(7,6),(9,8)]

unzip (zip [1,3..10] [0,2..15])
([1,3,5,7,9],[0,2,4,6,8])

and (map even [1,2,3,4])
False

or (map even [1,2,3,4])
True

foldl (+) 2 [-2..2]
2

foldl1 (+) [-2..2]
0

foldr (-) 2 [-2..2]
-2

foldr1 (-) [-2..2]
0

foldl (-) 2 [-2..2]
2

foldl1 (-) [-2..2]
4

Exemplos de Fold

foldl (op) arg0 [a,b,c] = ((arg0 op a) op b) op c

foldl1 (op) [a,b,c] = (a op b) op c

foldr (op) arg0 [a,b,c] = a op (b op (arg0 op c))

foldr1 (op) [a,b,c] = a op (b op c)

foldr (-) 2 [-2..2]

-2

foldl (-) 2 [-2..2]

2

foldr1 (-) [-2..2]

0

foldl1 (-) [-2..2]

-4

Tipo Tuplas

É uma estrutura do tipo registro.

Uma relação fixa de campos de tipos quaisquer.

Se t_1, t_2, ..., t_n são tipos, então o tipo da n-tupla é (t_1, t_2, ..., t_n).

Exemplo:

(("Nome","Leda"),"mulher","casada",("idade", 62))

(5, [1,2,3], "Brasília")

Entrada/Saída

```
entrada = do  putStr "\n dados> "
              dados <- getLine
              putStr “Digitado: "
               putStr (concat [dados, "\n"])
              putStr "continua, (s/n)?"
              carac <- getChar
              if carac=='S'  || carac=='s' then entrada
              else return ()
Main> entrada
 dados> primeira
Digitado: primeira
continua, (s/n)?s
 dados> Segunda
Digitado: Segunda
continua, (s/n)?T :: IO [Char]
```

Entrada/Saída

```
import Data.Char
tecla = do putStr "\n Qual a tecla? > "
           carac <- getChar
           putStr (show (ord carac))
           if carac == '\ESC' then (return ())
                              else tecla
Main> tecla
 Qual a tecla? > a97
 Qual a tecla? > b98
 Qual a tecla? > B66
 Qual a tecla? > 27 :: IO ()
```