UFSC / CTC / INE
Disciplina: Paradigmas de Programação

CCO: INE5416 / SIN:INE5636

Prof. Dr. João Dovicchi*

Aula Prática 4 - Haskell: especificação

O objetivo desta aula é compreender a especificação da linguagem HASKELL que é usada como exemplo. Os programas em HASKELL são formados por módulos, cuja estrutura veremos em outra aula. Por exemplo, um módulo pode definir várias funções, como no exemplo da listagem 1.

Além disso, os módulos podem importar outros módulos onde se encontram definidas outras funções. Observe que um módulo começa com a declaração de módulo seguida do nome do módulo, as funções que ele exporta e a palavra `where`. O nome do módulo deve, necessariamente, começar com letra maiúscula e as funções com letra minúsculas ou o caractere "sublinhado".

Em seguida, pode-se declarar quais módulos ele importa. No caso do exemplo da listagem 1 o módulo importa o módulo `Numeric`

Atenção: O módulo "Prelude" é sempre importado por *default*.

As funções podem ser declaradas em qualquer ordem. Em programação funcional o programa não reflete o fluxo de ações como na programação procedimental. Quando há necessidade de se executar um procedimento ele será descrito por meio de funções monádicas que veremos mais tarde.

Observe, ainda, que classes (`Enum, Num, RealFrac`) e tipos (Int, String) iniciam com letra maiúsculas.

*`http://www.inf.ufsc.br/~dovicchi --- dovicchi@inf.ufsc.br`

Listagem 1: Modulo Cent_Tools.hs

```
module Cent_Tools (transp, mulFreq, spectFreq, mergeList,
                   _ConvDat, argNum, sliceList) where

import Numeric

transp :: [[a]] -> [[a]]
transp []  = []
transp ([]:xss) = transp xss
transp ((x:xs):xss) = (x:[h | (h:t) <- xss]) :\
                        transp (xs:[t | (h:t) <- xss])

spectFreq :: (Enum a, Num a) => a -> a -> [a];
spectFreq n x = [ x | x <- map (x*) [1..n]]

mulFreq :: (Num a) => a -> [a] -> [a]
mulFreq x y = map (x*) y

mergeList :: (Num a) => [a] -> [[a]] -> [[a]]
mergeList [] _ = []
mergeList _ [] = []
mergeList x (y:ys) = zipWith (*) x y : mergeList x ys

_ConvDat :: (RealFrac a) => String -> a;
_ConvDat x = head (fst (unzip (readFloat x)))

argNum x = head (fst (unzip (readDec x)))

sliceList :: Int -> [a] -> [[a]]
sliceList _ [] = []
sliceList n x = fst (splitAt n x) :\
                  sliceList n (snd (splitAt n x))
```

O capítulo 3 da apostila traz uma visão geral do HASKELL e alguns exemplos de programação funcional. Os roteiros a seguir fazem referência a textos deste capítulo.

0.1 Roteiro 1

Usando o exemplo da listagem 3.2 do capítulo 3 da apostila, digite o programa e teste no interpretador (ghci ou hugs). Observe que parte da explicação está na seção 3.3.

0.2 Roteiro 2

No final do capítulo 3 da apostila, o aluno encontrará os exercícios que deve resolver usando o programa exemplo das listagens 3.4 e 3.5.