INE 5416/5636 - Paradigmas de programação

Turmas 04208/08238

Prof. Dr. João Dovicchi – dovicchi@inf.ufsc.br

http://www.inf.ufsc.br/˜dovicchi

Haskell: Análise léxica

Lexemas

Formados por: Brancos, palavras, símbolos, pontuações

São considerados "Brancos" $\rightarrow$ espaços, `<CR>`, `<LF>`, tabulação, comentários.

Comentário de linha: `--`
Comentário multilinha: `{-  -}`

Haskell: Análise léxica

Lexemas

Formados por: Brancos, palavras, símbolos, pontuações

São considerados "Brancos" → espaços, <CR>, <LF>, tabulação, comentários.

Comentário de linha: `--`
Comentário multilinha: `{-  -}`

```
-- Uma linha de comentário

func x = x*x -- comentário daqui até o final da linha

{- Um comentário
multilinha {- mais um comentário aninhado -} -}
```

Lexemas

Comentários podem estar associados a pragmas do compilador. Exemplo:

```
factorial :: Num a => a -> a
factorial 0 = 0
factorial n = n * factorial (n-1)

{-# SPECIALIZE factorial :: Int -> Int,
               factorial :: Integer -> Integer #-}
```

Haskell: Análise léxica

Identificadores

São letras seguidas de outras letras, números, aspas simples ou o caractere “_”

Identificadores de argumentos e funções: que iniciam, obrigatoriamente, com letra minúscula ou o caractere “_”. Por exemplo: var, _VAR, func, fUNC etc..

```
module Viz (ante, prox) where

ante x = x-1

prox x = x+1
```

Haskell: Análise léxica

Identificadores

São letras seguidas de outras letras, números, aspas simples ou o caractere "_"

Identificadores de construtores: iniciam com letra maiúscula. Por exemplo: Int, Integer, Bool, Char etc..

```
module Primos (primo, primos) where

primo :: Integral a => a -> Bool
primo n = (fator n == [])

primos :: Integral a => [a] -> [a]
primos []     = []
primos (n:ns) = case (primo n) of True  -> n : primos ns
                                  False -> primos ns

fator n = [ x | x <- [2 .. n-1], n `mod` x == 0]
```

Sublinhado

O caractere sublinhado (_) é:

- Considerado caractere minúsculo se associado a um identificador.
- Considerado caractere coringa quando isolado.

Sublinhado

O caractere sublinhado (_) é:

- Considerado caractere minúsculo se associado a um identificador.
- Considerado caractere coringa quando isolado.

Nota: identificadores iniciados com este caractere não são verificados pelo compilador.

Identificadores reservados

case | class | data | default | deriving | do | else | if | import | in | infix | infixl | infixr | instance | let | module | newtype | of | then | type | where | _

Identificadores qualificados

O compilador entende o ponto como construtor de qualificação quando o primeiro termo inicia com maiúscula.

Exemplo:

```
module Main (main) where
import Funcs (func2)

main = print (Funcs.func2 ...)
```

Neste exemplo, `Funcs.func2` é um identificador qualificado. Ou seja, a função "`func2`" do módulo **Funcs** é chamada.

Operadores

Operadores aritméticos

```
+    -    *    /    **    ^    ^^
```

Operadores

Operadores aritméticos

```
+    -    *    /    **    ^    ^^
```

Operadores literais

```
++    \\    !!
```

Operadores

Operadores lógicos

```
||    &&    not
```

Operadores

Operadores lógicos

```
||    &&    not
```

Operadores de Comparação

```
==    !=    >    >=    <    <=
```

Construtores

Construtores de listas:

```
:     [     ]     ..
```

Construtores

Construtores de listas:

```
:    [    ]    ..
```

Construtures lambda

```
\    ->
```

Expressões

Principal elemento das linguagens funcionais (funções)

nome argumento(s) = expressão

“expressão” deve conter o(s) argumento(s), valores e operadores.

Roteiros da aula prática 04