Introdução à Linguagem Funcional Haskell

UFRN, 2018

Programação Funcional

- **Um programa funcional consiste de uma série de definições de funções.**
- **A execução de um programa funcional consiste em calcular o valor de uma expressão, usando as funções definidas.**
- **Linguagem funcional considerada: Haskell.**
  - **Os programas Haskell são chamados de scripts.**

Exemplo de script em Haskell

```
1  ------------------------------
2  --  example.hs
3  ------------------------------
4  answer :: Int
5  answer = 42
6
7  square :: Int -> Int
8  square x = x * x
9
10 allEqual :: Int -> Int -> Int -> Bool
11 allEqual m n p = (m==n) && (n==p)
12
13 maxi :: Int -> Int -> Int
14 maxi m n
15    | m >= n     = m
16    | otherwise  = n
```

Cálculos em Haskell

Como cálculos são efetuados em Haskell?

```
allEqual m n p = (m==n) && (n==p)
```

```
allEqual 2 3 3
= (2==3) && (3==3)
= False && True
= False
```

```
allEqual 5 5 5
= (5==5) && (5==5)
= True && True
= True
```

Cálculos em Haskell

Exemplo envolvendo várias funções

```
allEqual (square 3) answer (square 2)
= ((square 3) == answer) && (answer == (square 2))
= ((3*3) == 42) && (42 == (2*2))
= (9 == 42) && (42 == 4)
= False && False
= False
```

Cálculos em Haskell

Exemplo envolvendo equação condicional

```
maxi m n
  | m >= n      = m
  | otherwise   = n
```

```
maxi 3 1
   ? 3>=1 = True = 3
```

```
maxi 3 4
   ? 3>=4 = False
   ? otherwise = True = 4
```

Cálculos em Haskell

Outro exemplo envolvendo equação condicional:

```
allEqual (maxi 1 5) 5 (maxi 4 2)

= ((maxi 1 5) == 5) && (5 == (maxi 4 2))
  ? 1>=5 = False
  ? otherwise = True
= (5 == 5) && (5 == (maxi 4 2))
  ? 4>=2 = True
= (5 == 5) && (5 == 4)
= True && False
= False
```

Hugs: Interpretador Haskell

- Ao executar Hugs, uma sessão é iniciada.
- O sistema carrega funções pré-definidas (Prelude.hs) e passa a esperar comandos:

```
hugs
Reading file
"/usr/local/share/hugs/lib/Prelude.hs":

Hugs session for:
/usr/local/share/hugs/lib/Prelude.hs
Type :? for help
Prelude>
```

Hugs: Interpretador Haskell

Exemplo de interação:

```
Prelude> 2+3
5
Prelude> (1*6) == (3 `div` 5)
False
Prelude> sum [1..10]
55
Prelude> reverse "hugs is cool"
"looc si sguh"
Prelude>
```

Hugs: Interpretador Haskell

- Cada linha digitada é tratada como um comando.
- Se a linha for uma expressão, então é tratada como um comando para avaliar a expressão.
- Alguns comandos importantes:
  - :? imprime a lista de todos os comandos;
  - :q abandona o interpretador;
  - :load carrega definições a partir de um arquivo.

Hugs: Interpretador Haskell

- Novas definições não podem ser criadas a partir da linha de comando. Elas devem ser carregadas a partir de arquivos.

```
Prelude> :load example.hs
Reading file "example.hs":

Hugs session for:
/usr/local/share/hugs/lib/Prelude.hs
example.hs
Main>
```

Hugs: Interpretador Haskell

Seja o arquivo "example.hs" a seguir:

```
------------------------------
--  example.hs
------------------------------
module Exemplo1 where

answer :: Int
answer = 42
...
```

Define o nome do módulo que será carregado.

Hugs: Interpretador Haskell

Carregando novamente o arquivo:

```
Prelude> :l example.hs
Reading file "example.hs":

Hugs session for:
/usr/local/share/hugs/lib/Prelude.hs
example.hs
Exemplo1> maxi 3 4
4
Exemplo1>
```