

Introdução à Linguagem Haskell

Prof. Dr. Eduardo Takeo Ueda
eduardo.tueda@sp.senac.br

Introdução

- Programação funcional é a aplicação de funções à argumentos
- Linguagem Haskell é uma homenagem ao matemático Haskell Curry, pioneiro no desenvolvimento do Cálculo Lâmbda
- Haskell (1996) é similar a ML (1990), com a diferença de ser puramente funcional

Exemplo

- Para somar os inteiros de 1 a 10 podemos escrever em linguagem C/Java

total = 0;

for (i = 0; i <= 10; i++)

total = total + i;

- O método de computação acima é baseado em atribuição de valores a variáveis

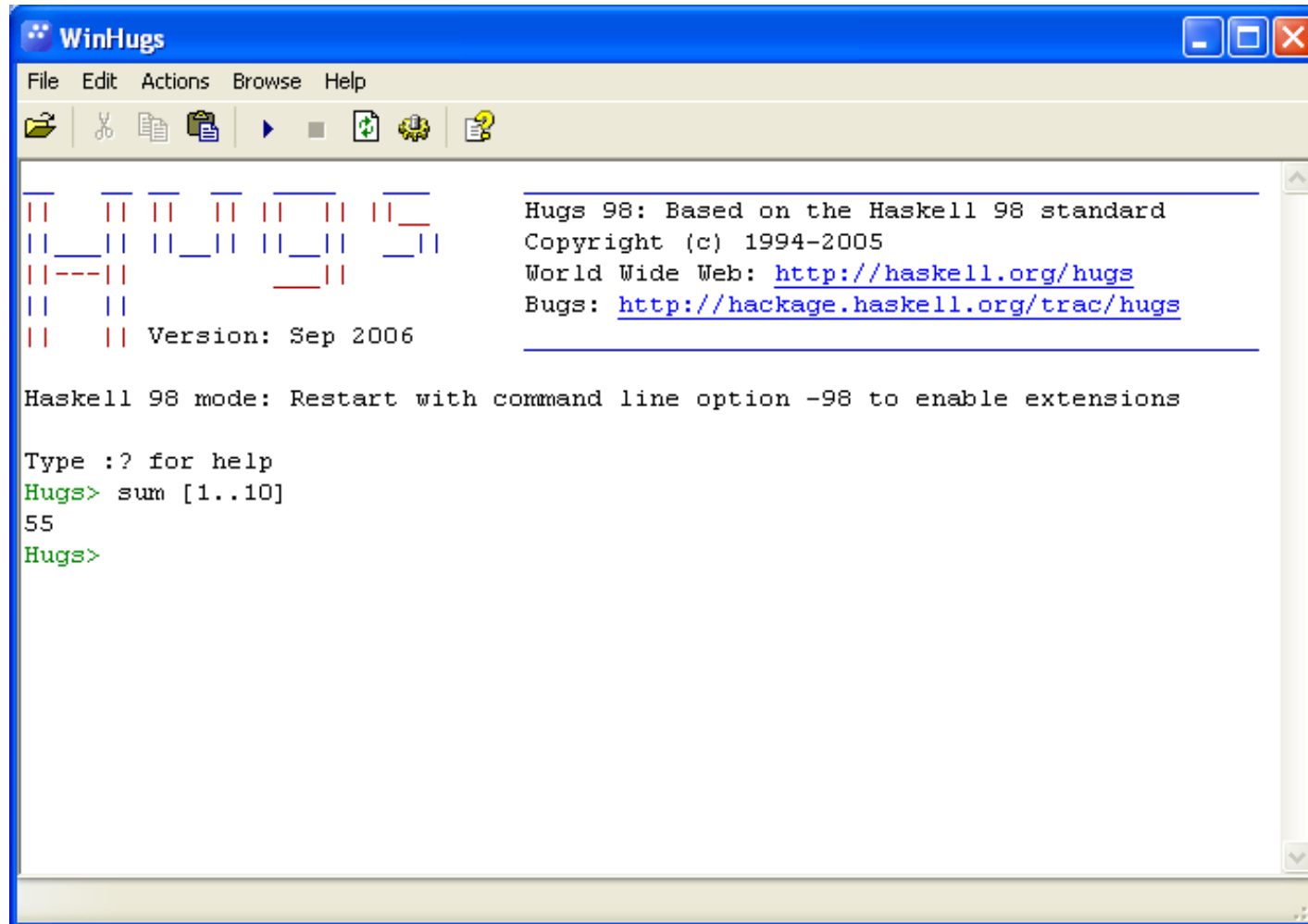
Exemplo

- Para somar os inteiros de 1 a 10 podemos escrever em linguagem Haskell

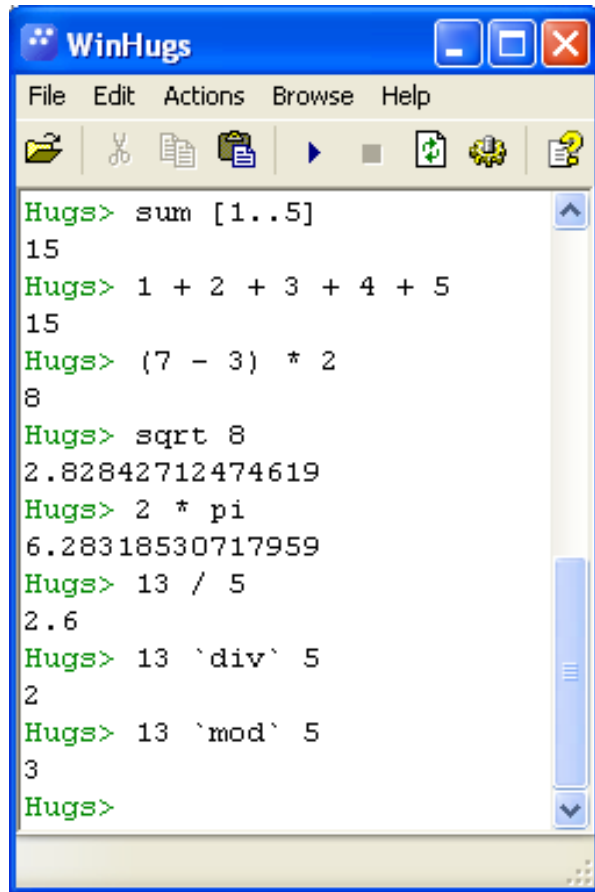
sum [1..10]

- O método de computação acima é baseado em aplicação de funções à argumentos
 - função: *sum*
 - argumento: *[1..10]*

Sistema Hugs



Expressões em Haskell



```
WinHugs
File Edit Actions Browse Help
[Icons]
Hugs> sum [1..5]
15
Hugs> 1 + 2 + 3 + 4 + 5
15
Hugs> (7 - 3) * 2
8
Hugs> sqrt 8
2.82842712474619
Hugs> 2 * pi
6.28318530717959
Hugs> 13 / 5
2.6
Hugs> 13 `div` 5
2
Hugs> 13 `mod` 5
3
Hugs>
```

Hugs é uma implementação da Linguagem Haskell que pode ser executada em Windows ou Linux e pode ser obtida em www.haskell.org/hugs

Funções

- Representação de uma função



- Uma função calcula um valor (saída) que depende de argumentos (entradas)

Funções

- Funções em Haskell são normalmente definidas com a utilização de equações
- Função para somar dois inteiros

soma $x\ y = x + y$

> soma 12 34

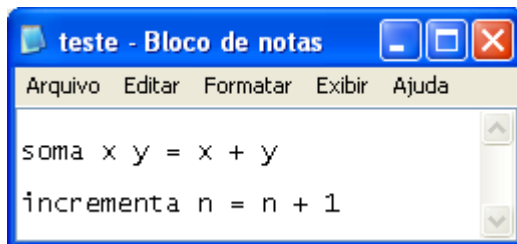
46



Funções

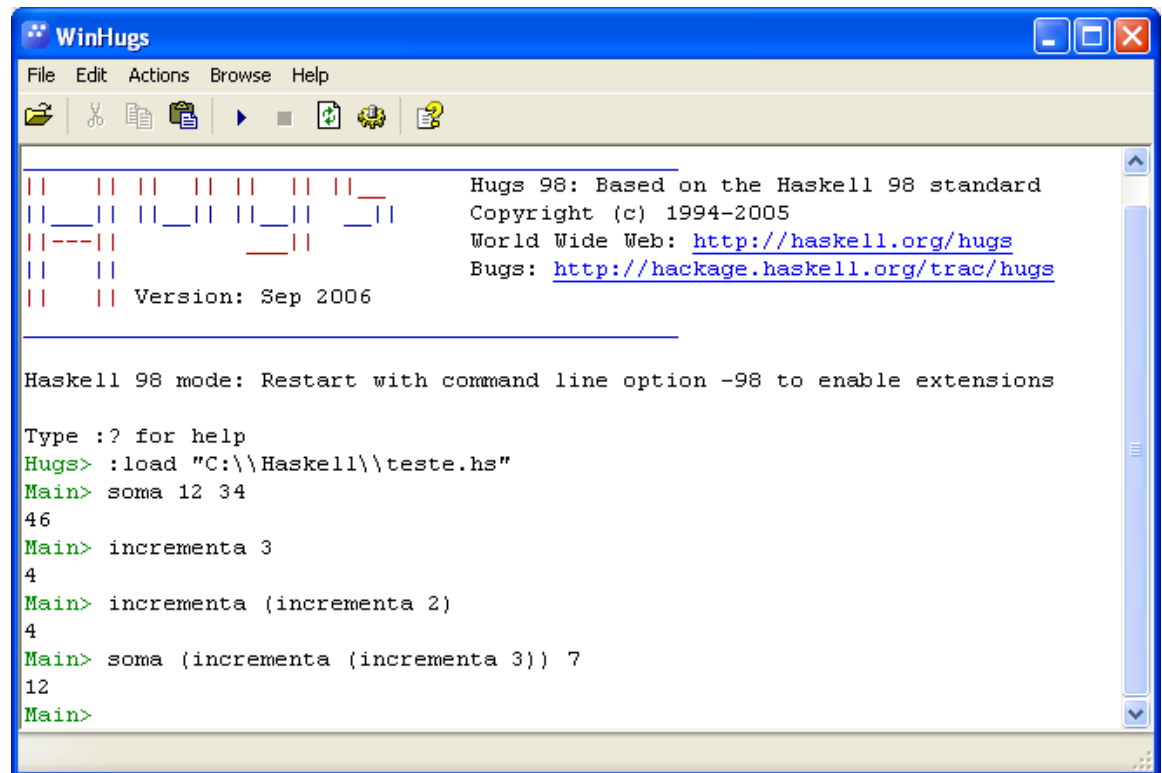
- Função para incrementar um valor inteiro

incrementa $n = n + 1$



```
teste - Bloco de notas
Arquivo  Editar  Formatar  Exibir  Ajuda

soma x y = x + y
incrementa n = n + 1
```



```
WinHugs
File Edit Actions Browse Help

Hugs 98: Based on the Haskell 98 standard
Copyright (c) 1994-2005
World Wide Web: http://haskell.org/hugs
Bugs: http://hackage.haskell.org/trac/hugs
Version: Sep 2006

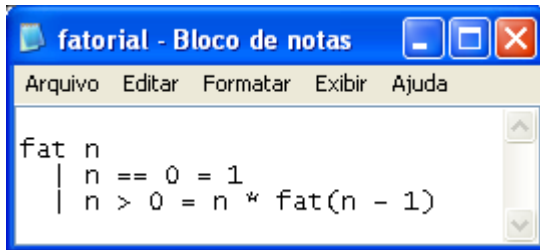
Haskell 98 mode: Restart with command line option -98 to enable extensions

Type :? for help
Hugs> :load "C:\\\\Haskell\\\\teste.hs"
Main> soma 12 34
46
Main> incrementa 3
4
Main> incrementa (incrementa 2)
4
Main> soma (incrementa (incrementa 3)) 7
12
Main>
```

Função Fatorial

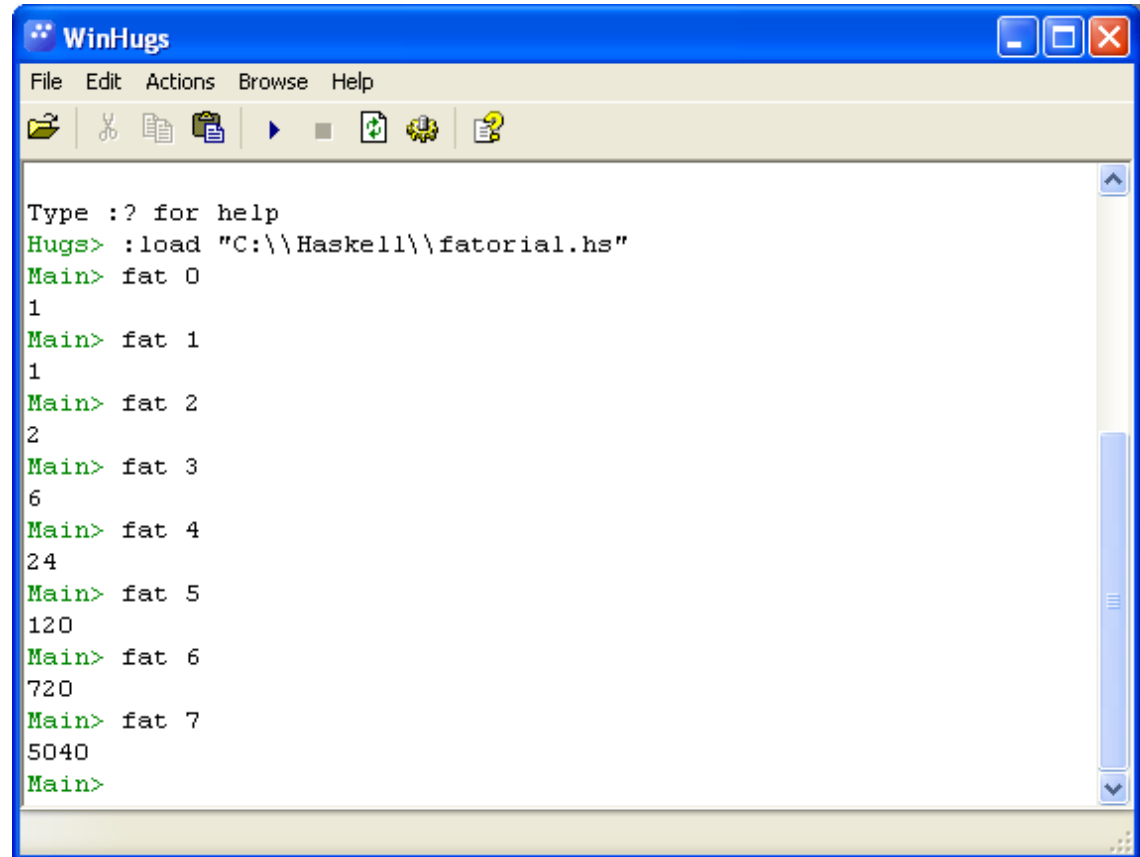
$\text{fat } 0 = 1$

$\text{fat } n = n * \text{fat } (n - 1)$



```
fatorial - Bloco de notas
Arquivo  Editar  Formatar  Exibir  Ajuda

fat n
  | n == 0 = 1
  | n > 0 = n * fat(n - 1)
```



```
WinHugs
File Edit Actions Browse Help

Type :? for help
Hugs> :load "C:\\Haskell\\fatorial.hs"
Main> fat 0
1
Main> fat 1
1
Main> fat 2
2
Main> fat 3
6
Main> fat 4
24
Main> fat 5
120
Main> fat 6
720
Main> fat 7
5040
Main>
```

Comandos úteis no ambiente Hugs

| Comando | Significado |
|------------------------------|---------------------------------|
| <code>:load "arq.ext"</code> | carregar o arquivo |
| <code>:reload</code> | recarregar o arquivo atual |
| <code>:edit "arq.ext"</code> | editar o arquivo pedido |
| <code>:edit</code> | editar o arquivo atual |
| <code>:type expr</code> | mostrar o tipo de uma expressão |
| <code>:?</code> | mostrar todos os comandos |
| <code>:quit</code> | encerrar o Hugs |

Exercícios

- (1) Verifique que *fatorial* $n = \text{product } [1..n]$ em linguagem Haskell
- (2) Escreva uma função em Haskell que retorna o *n-ésimo* termo da sequência de Fibonacci