

# Introdução à Programação Funcional

## Linguagem Haskell

Maria Adriana Vidigal de Lima

*Faculdade de Computação - UFU*

*Agosto - 2009*

- 1 Programação Funcional
  - Linguagem Haskell
  - Funções

# Introdução

- A Programação Funcional é um estilo de programação em que o método básico de computação é a *aplicação de funções à argumentos*.
- Haskell é uma linguagem funcional projetada com objetivo de ser utilizada no ensino, pesquisa e construção de sistemas computacionais.
- Haskell deve seu nome ao matemático **Haskell B. Curry**, conhecido por seu trabalho em lógica combinatória e pioneiro no desenvolvimento do Cálculo Lambda (**Cálculo- $\lambda$** ), inspiração aos projetistas da maioria das linguagens funcionais.

# Exemplo

Para somar os números inteiros de 1 a 10 podemos escrever em linguagem Java:

```
total = 0;  
for (i = 1; i <= 10; i++)  
    total = total + i;
```

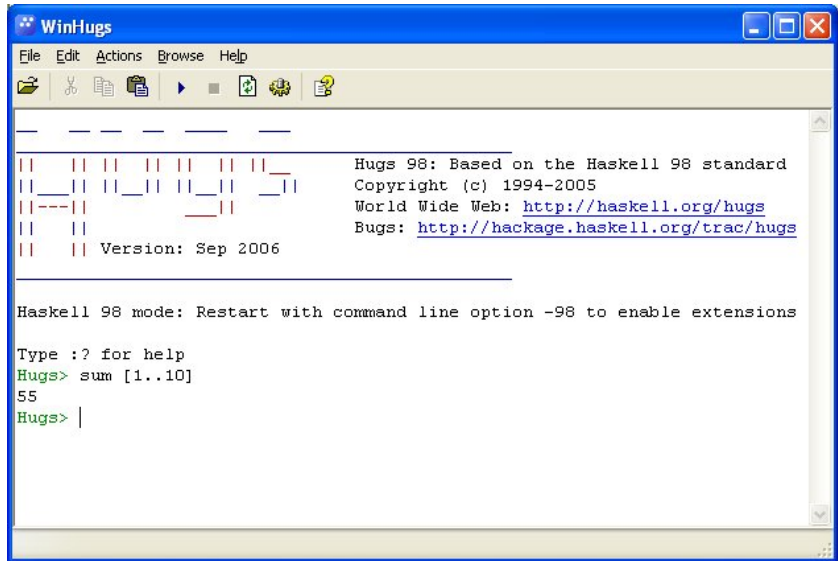
*O método da computação é baseado em atribuição de valores à variáveis*

# Exemplo

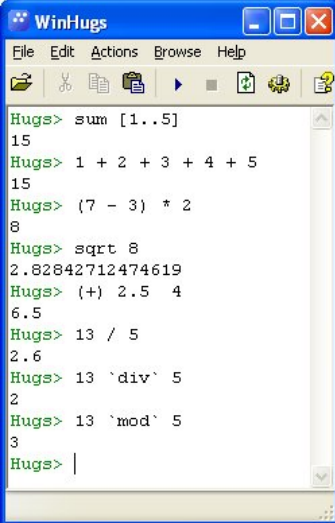
A soma dos números inteiros de 1 a 10 pode ser escrita em Haskell como:

```
sum [1..10]
```

*O método da computação é baseado em aplicação de argumentos à funções*



# 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> (+) 2.5 4
6.5
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 PCs e sistemas Unix, incluindo Linux. Pode ser obtido gratuitamente em [www.haskell.org/hugs](http://www.haskell.org/hugs)

# Funções

Uma função pode ser representada como no desenho abaixo:



A função calcula um valor (o valor de saída) que depende dos valores de entrada.



# Funções

Funções em Haskell são normalmente definidas pelo uso de equações. Por exemplo, a função `soma` pode ser escrita:

```
soma x y = x + y
```

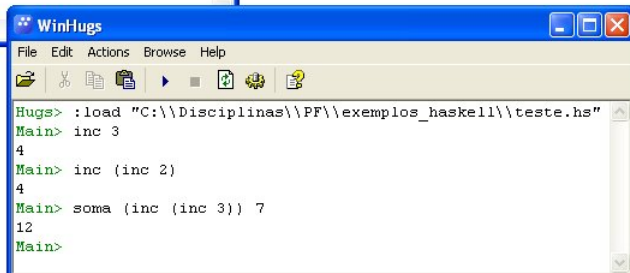
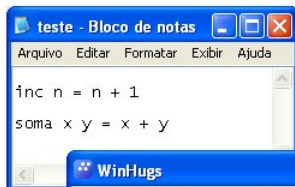
```
> soma 12 34  
46
```



# Funções

A função incrementar pode ser escrita e testada:

```
inc n = n + 1
```



# Outros exemplos de Funções

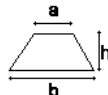
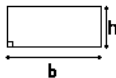
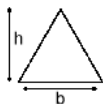
Algumas funções para calcular a área de figuras podem ser definidas:

```
areaTriangulo b h = (b * h) / 2
```

```
areaRetangulo b h = b * h
```

```
areaCirculo r = pi * r * r
```

```
areaTrapezio a b h = (a + b) * h / 2
```



# Outros exemplos de Funções

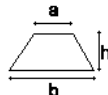
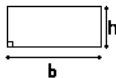
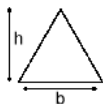
Algumas funções para calcular a área de figuras podem ser definidas:

```
areaTriangulo b h = (b * h) / 2
```

```
areaRetangulo b h = b * h
```

```
areaCirculo r = pi * r * r
```

```
areaTrapezio a b h = (a + b) * h / 2
```



# Outros exemplos de Funções

Função para calcular a média entre três números:

```
media v1 v2 v3 = (v1 + v2 + v3) / 3
```

```
> media 2 5 7
```

```
4.666666666666667
```

```
> media 1.4 2.6 4.4
```

```
2.8
```

```
> media 1.4 4 5.6
```

```
3.666666666666667
```

```
> media 1 4
```

```
ERROR - Cannot find "show" function for:
```

```
*** Expression : media 1 4
```

```
*** Of type      : Double -> Double
```

# Ambiente do Hugs

Alguns comandos úteis no Hugs:

Comando	Significado
:load “arq.ext”	carregar o arquivo
:reload	recarregar o arquivo atual
:edit “arq.ext”	editar o arquivo pedido
:edit	editar o arquivo atual
:type expr	mostrar o tipo de uma expressão
:?	mostrar todos os comandos
:quit	encerrar o Hugs

# Exercícios em Haskell

- 1 Escreva uma função para calcular o dobro de um número.
- 2 Escreva uma função para quadruplicar um número, usando a função definida no item anterior.
- 3 Defina uma função para calcular a distância entre dois pontos (num plano).
- 4 Dadas as medidas dos catetos de um triângulo retângulo, retornar o valor de sua hipotenusa.