

Introdução à Programação Funcional

Videoaula 2

Introdução à Linguagem Haskell

Profa. Dra. Gina Maira B. Oliveira

*Fortemente baseado no material de aula da Profa. Maria Adriana Vidigal de Lima Faculdade de Computação - UFU

2020

Haskell

A Programação Funcional é um estilo de programação em que o método básico de computação é a aplicação de funções a 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- λ), inspiração aos projetistas da maioria das linguagens funcionais.

Comparação: Haskell versus C

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

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

O método da computação é baseado em atribuição de valores `as variáveis.

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 `a funções.

Haskell: função

Uma função pode ser representada como no desenho abaixo:



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

Haskell: função *soma*

Funções em Haskell são normalmente definidas pelo uso de equações.

Por exemplo, a função *soma* pode ser escrita:

soma $x\ y = x + y$

Após a função *soma* ser carregada na memória de um interpretador Haskell (por exemplo, GHC), ela pode ser avaliada com parâmetros de entrada numéricos:

> *soma* 12 34

46



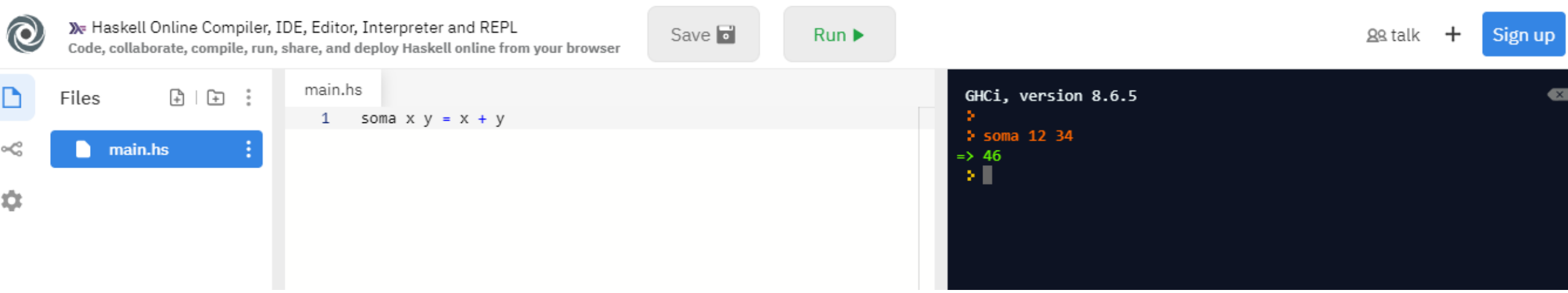
Haskell: compilador GHC

GHC (The Glasgow Haskell Compiler) é uma implementação da linguagem Haskell 2010. mais uma grande variedade de extensões.

O GHC tem um suporte particularmente bom para a concorrência e o paralelismo, incluindo suporte para memória transacional de software (STM).

Uma versão online do GHC pode ser encontrada em:

<https://repl.it/languages/haskell>



The screenshot displays the Haskell Online Compiler interface. At the top, the header reads "Haskell Online Compiler, IDE, Editor, Interpreter and REPL" with the tagline "Code, collaborate, compile, run, share, and deploy Haskell online from your browser". Navigation buttons for "Save" and "Run" are visible. On the left, a file explorer shows a single file named "main.hs". The central code editor contains the following Haskell code:

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

On the right, the GHCi REPL (version 8.6.5) shows the execution of the code:

```
GHCi, version 8.6.5
> soma 12 34
=> 46
```

Haskell: compilador GHC

Editor:

```
main.hs
1  soma x y = x + y
2
```

Interpretador:

```
GHCi, version 8.6.5
> soma 12 34
=> 46
> █
```

Haskell: função *inc*

A função *incrementar* pode ser escrita e testada no GHC, em conjunto com a função *soma*:

$\text{inc } n = n + 1$

main.hs

```
1 soma x y = x + y
2 inc n = n + 1
```

```
GHCi, version 8.6.5
> soma 12 34
=> 46
> inc 46
=> 47
> inc 47
=> 48
> inc (inc 46)
=> 48
> inc (soma 12 34)
=> 47
> inc (inc (soma 12 34))
=> 48
> 
```


Haskell: função media

Função para calcular a média entre três números (v1, v2 e v3):

:

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

Haskell: função media

Função para calcular a média entre três números (v1, v2 e v3):

:

```
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
```

Haskell: função media

Função para calcular a média entre três números:

$\text{media } v1 \ v2 \ v3 = (v1 + v2 + v3) / 3$

> 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:

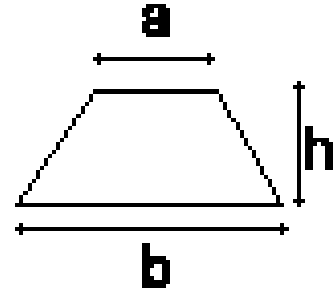
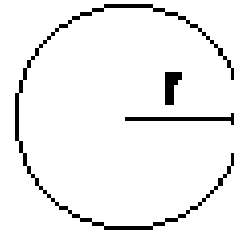
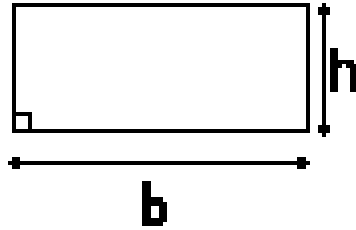
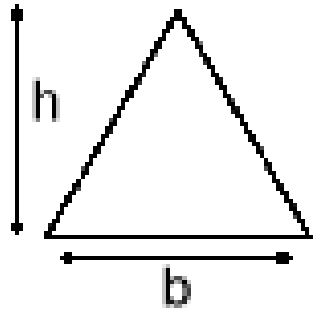
*** Expression : media 1 4

*** Of type : Double -> Double

```
GHCi, version 8.6.5
> media 2 5 7
=> 4.666666666666667
> media 1.4 2.6 4.4
=> 2.8000000000000003
> media 1.4 4 5.6
=> 3.6666666666666665
> media 1 4
<interactive>:10:1: error:
    * No instance for (Show (Double -> Double))
      arising from a use of `print'
      (maybe you haven't applied a function to enough
       arguments?)
    * In a stmt of an interactive GHCi command: print it
> █
```

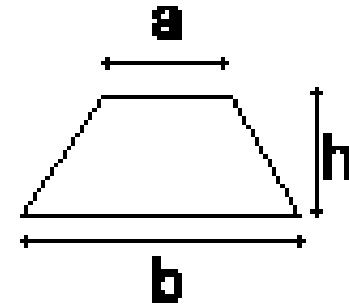
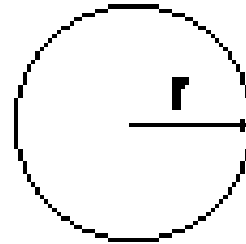
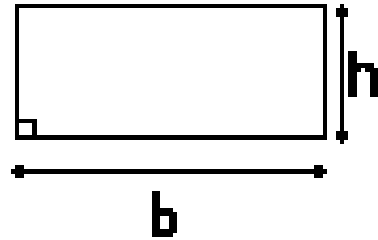
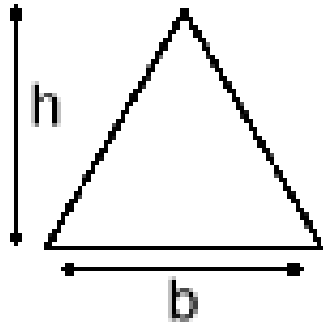
Haskell: funções de área

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



Haskell: funções de área

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



$$\text{areaTriangulo } b \ h = (b * h) / 2$$

$$\text{areaRetangulo } b \ h = b * h$$

$$\text{areaCirculo } r = \text{pi} * r * r$$

$$\text{areaTrapezio } a \ b \ h = (a + b) * h / 2$$

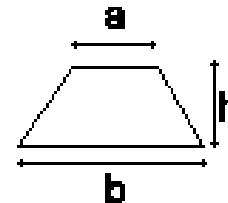
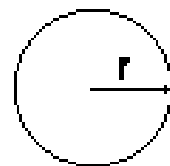
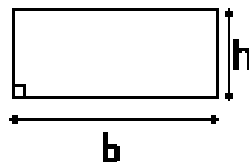
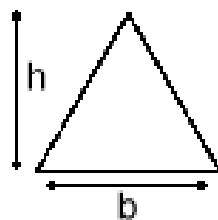
Haskell: funções de área

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

main.hs

```
1  soma x y = x + y
2  inc n = n + 1
3  media v1 v2 v3 = (v1 + v2 + v3) / 3
4  areaTriangulo b h = (b * h) / 2
5  areaRetangulo b h = b * h
6  areaCirculo r = pi * r * r
7  areaTrapezio a b h = (a + b) * h / 2
```

```
GHCi, version 8.6.5
> areaTriangulo 50 10
=> 250.0
> areaRetangulo 50 10
=> 500
> areaCirculo 10
=> 314.1592653589793
> areaTrapezio 10 40 10
=> 250.0
>
```



Fim da Videoaula 2

Introdução à Programação Funcional

Introdução à Linguagem Haskell