

Disciplina de Programação Funcional Engenharia de Computação

Linguagem Haskell

Mauro Hemerly Gazzani
mauro.hemerly@gmail.com

Universidade Estadual de Minas Gerais
Câmpus de Ituiutaba, 2º semestre de 2018

<https://bit.ly/2A0eUI8>
<https://github.com/mauro-hemerly/UEMG-2018-2>

Linguagem Haskell

- Algumas características de **Haskell**

- ▶ Programas são **concisos**
- ▶ **Tipagem estática**
- ▶ **Sistema de tipos** poderoso
- ▶ Tipos e funções **recursivas**
- ▶ **Funções de ordem superior**
- ▶ Linguagem **pura** (declarativa)
- ▶ Avaliação **lazy**

- **Em 1987:** Um comitê internacional de pesquisadores inicia o desenvolvimento de **Haskell**, uma **linguagem funcional lazy** padrão.
- **Em 2003:** O comitê publica o relatório **Haskell 98**, a definição de uma versão estável da linguagem **Haskell**.
- **Em 2010:** O comitê publica o relatório **Haskell 2010**, uma revisão da definição da linguagem **Haskell**.

Linguagem Haskell

- Algumas características de **Haskell**

- ▶ Programas são **concisos**
- ▶ **Tipagem** estática
- ▶ **Sistema de tipos** poderoso
- ▶ Tipos e funções **recursivas**
- ▶ **Funções de ordem superior**
- ▶ Linguagem **pura** (declarativa)
- ▶ Avaliação **lazy**

- **Em 1987:** Um comitê internacional de pesquisadores inicia o desenvolvimento de **Haskell**, uma **linguagem funcional lazy** padrão.
- **Em 2003:** O comitê publica o relatório **Haskell 98**, a definição de uma versão estável da linguagem **Haskell**.
- **Em 2010:** O comitê publica o relatório **Haskell 2010**, uma revisão da definição da linguagem **Haskell**.

Linguagem Haskell

- Algumas características de **Haskell**

- ▶ Programas são **concisos**
- ▶ **Tipagem estática**
- ▶ **Sistema de tipos** poderoso
- ▶ Tipos e funções **recursivas**
- ▶ **Funções de ordem superior**
- ▶ Linguagem **pura** (declarativa)
- ▶ Avaliação **lazy**

- **Em 1987:** Um comitê internacional de pesquisadores inicia o desenvolvimento de **Haskell**, uma **linguagem funcional lazy** padrão.
- **Em 2003:** O comitê publica o relatório **Haskell 98**, a definição de uma versão estável da linguagem **Haskell**.
- **Em 2010:** O comitê publica o relatório **Haskell 2010**, uma revisão da definição da linguagem **Haskell**.

Linguagem Haskell

- Algumas características de **Haskell**

- ▶ Programas são **concisos**
- ▶ **Tipagem estática**
- ▶ **Sistema de tipos poderoso**
- ▶ Tipos e funções **recursivas**
- ▶ **Funções de ordem superior**
- ▶ Linguagem **pura** (declarativa)
- ▶ Avaliação **lazy**

- **Em 1987:** Um comitê internacional de pesquisadores inicia o desenvolvimento de **Haskell**, uma **linguagem funcional lazy** padrão.
- **Em 2003:** O comitê publica o relatório **Haskell 98**, a definição de uma versão estável da linguagem **Haskell**.
- **Em 2010:** O comitê publica o relatório **Haskell 2010**, uma revisão da definição da linguagem **Haskell**.

Linguagem Haskell

- Algumas características de **Haskell**

- ▶ Programas são **concisos**
- ▶ **Tipagem estática**
- ▶ **Sistema de tipos** poderoso
- ▶ **Tipos e funções recursivas**
- ▶ **Funções de ordem superior**
- ▶ Linguagem **pura** (declarativa)
- ▶ Avaliação **lazy**

- **Em 1987:** Um comitê internacional de pesquisadores inicia o desenvolvimento de **Haskell**, uma **linguagem funcional lazy** padrão.
- **Em 2003:** O comitê publica o relatório **Haskell 98**, a definição de uma versão estável da linguagem **Haskell**.
- **Em 2010:** O comitê publica o relatório **Haskell 2010**, uma revisão da definição da linguagem **Haskell**.

Linguagem Haskell

- Algumas características de **Haskell**

- ▶ Programas são **concisos**
- ▶ **Tipagem estática**
- ▶ **Sistema de tipos** poderoso
- ▶ Tipos e funções **recursivas**
- ▶ **Funções de ordem superior**
- ▶ Linguagem **pura** (declarativa)
- ▶ Avaliação **lazy**

- Em 1987: Um comitê internacional de pesquisadores inicia o desenvolvimento de Haskell, uma **linguagem funcional lazy** padrão.
- Em 2003: O comitê publica o relatório **Haskell 98**, a definição de uma versão estável da linguagem Haskell.
- Em 2010: O comitê publica o relatório **Haskell 2010**, uma revisão da definição da linguagem Haskell.

Linguagem Haskell

- Algumas características de **Haskell**

- ▶ Programas são **concisos**
- ▶ **Tipagem estática**
- ▶ **Sistema de tipos** poderoso
- ▶ Tipos e funções **recursivas**
- ▶ **Funções de ordem superior**
- ▶ **Linguagem pura** (declarativa)
- ▶ Avaliação **lazy**

- Em 1987: Um comitê internacional de pesquisadores inicia o desenvolvimento de Haskell, uma **linguagem funcional lazy** padrão.
- Em 2003: O comitê publica o relatório **Haskell 98**, a definição de uma versão estável da linguagem Haskell.
- Em 2010: O comitê publica o relatório **Haskell 2010**, uma revisão da definição da linguagem Haskell.

Linguagem Haskell

- Algumas características de **Haskell**

- ▶ Programas são **concisos**
- ▶ **Tipagem estática**
- ▶ **Sistema de tipos** poderoso
- ▶ Tipos e funções **recursivas**
- ▶ **Funções de ordem superior**
- ▶ Linguagem **pura** (declarativa)
- ▶ **Avaliação lazy**

- Em 1987: Um comitê internacional de pesquisadores inicia o desenvolvimento de Haskell, uma **linguagem funcional lazy** padrão.
- Em 2003: O comitê publica o relatório **Haskell 98**, a definição de uma versão estável da linguagem Haskell.
- Em 2010: O comitê publica o relatório **Haskell 2010**, uma revisão da definição da linguagem Haskell.

Linguagem Haskell

- Algumas características de **Haskell**
 - ▶ Programas são **concisos**
 - ▶ **Tipagem estática**
 - ▶ **Sistema de tipos** poderoso
 - ▶ Tipos e funções **recursivas**
 - ▶ **Funções de ordem superior**
 - ▶ Linguagem **pura** (declarativa)
 - ▶ Avaliação **lazy**
- **Em 1987:** Um comitê internacional de pesquisadores inicia o desenvolvimento de **Haskell**, uma **linguagem funcional lazy** padrão.
- **Em 2003:** O comitê publica o relatório **Haskell 98**, a definição de uma versão estável da linguagem **Haskell**.
- **Em 2010:** O comitê publica o relatório **Haskell 2010**, uma revisão da definição da linguagem **Haskell**.

Linguagem Haskell

- Algumas características de **Haskell**
 - ▶ Programas são **concisos**
 - ▶ **Tipagem estática**
 - ▶ **Sistema de tipos** poderoso
 - ▶ Tipos e funções **recursivas**
 - ▶ **Funções de ordem superior**
 - ▶ Linguagem **pura** (declarativa)
 - ▶ Avaliação **lazy**
- **Em 1987:** Um comitê internacional de pesquisadores inicia o desenvolvimento de **Haskell**, uma **linguagem funcional lazy** padrão.
- **Em 2003:** O comitê publica o relatório **Haskell 98**, a definição de uma versão estável da linguagem **Haskell**.
- **Em 2010:** O comitê publica o relatório **Haskell 2010**, uma revisão da definição da linguagem **Haskell**.

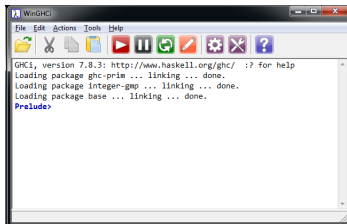
Linguagem Haskell

- Algumas características de **Haskell**
 - ▶ Programas são **concisos**
 - ▶ **Tipagem estática**
 - ▶ **Sistema de tipos** poderoso
 - ▶ Tipos e funções **recursivas**
 - ▶ **Funções de ordem superior**
 - ▶ Linguagem **pura** (declarativa)
 - ▶ Avaliação **lazy**
- **Em 1987:** Um comitê internacional de pesquisadores inicia o desenvolvimento de **Haskell**, uma **linguagem funcional lazy** padrão.
- **Em 2003:** O comitê publica o relatório **Haskell 98**, a definição de uma versão estável da linguagem **Haskell**.
- **Em 2010:** O comitê publica o relatório **Haskell 2010**, uma revisão da definição da linguagem **Haskell**.

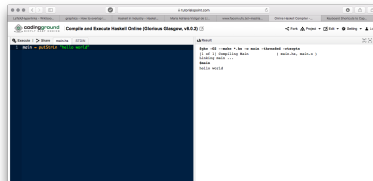
Algumas Empresas que Usam Haskell

- Exemplos de empresas que usam **Haskell**:
 - ▶ **ABN AMRO** análise de riscos financeiros
 - ▶ **AT&T** automatização de processamento de formulários
 - ▶ **Bank of America Merrill Lynch** transformação de dados
 - ▶ **Bump** servidores baseados em Haskell
 - ▶ **Facebook** manipulação da base de código PHP
 - ▶ **Google** infra-estrutura interna de TI
 - ▶ **MITRE** análise de protocolos de criptografia
 - ▶ **NVIDIA** ferramentas usadas internamente
 - ▶ **Qualcomm, Inc** geração de interfaces de programação para Lua
 - ▶ **The New York Times** processamento de imagens
- Para mais detalhes visite a página Haskell na indústria em https://wiki.haskell.org/Haskell_in_industry.

Ambientes de Desenvolvimento Haskell



A screenshot of the WinGHC Haskell IDE window. The title bar says 'WinGHC'. The menu bar includes 'File', 'Edit', 'Actions', 'Tools', and 'Help'. Below the menu is a toolbar with icons for file operations and execution. The main text area shows the GHCi version 7.8.3, the URL <http://www.haskell.org/ghc/>, and the status of loading packages: 'Loading package ghc-prim ... linking ... done.', 'Loading package integer-gmp ... linking ... done.', and 'Loading package base ... linking ... done.'. The prompt is 'Prelude>'.



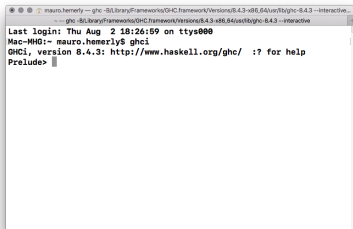
A screenshot of the CodingGround Haskell Online IDE. The browser address bar shows 'codingground.com'. The page title is 'Compile and Execute Haskell Online (Glasgow, v8.0.2)'. The left pane shows a Haskell file named 'Main.hs' with the following code:

```
main = putStrLn "Hello World!"
```

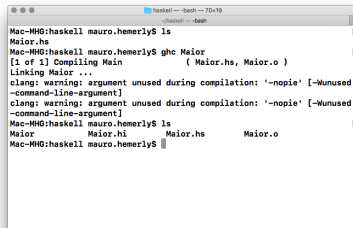
. The right pane shows the output:

```
Alpha <0> -main -i main -i standard -i template [1 of 1] Compiling Main (main.hs, main.o) Linking main ... Hello World
```

.



A screenshot of a terminal window. The prompt is 'mauro.hemerly\$ ghci'. The output shows the last login time, the prompt 'Mac-MHG:~ mauro.hemerly\$ ghci', the GHCi version 8.4.3, the URL <http://www.haskell.org/ghc/>, and the prompt 'Prelude>'.



A screenshot of a terminal window. The prompt is 'Mac-MHG:haskell mauro.hemerly\$ ls'. The output shows the files 'Maior.hs' and 'Maior.o'. The prompt is 'Mac-MHG:haskell mauro.hemerly\$ ghc Maior'. The output shows the compilation process: '[1 of 1] Compiling Main (Maior.hs, Maior.o)', 'Linking Maior ...', and two warnings from clang: 'warning: argument unused during compilation: '-nopie' [-Wunused-command-line-argument]' and 'warning: argument unused during compilation: '-nopie' [-Wunused-command-line-argument]'. The prompt is 'Mac-MHG:haskell mauro.hemerly\$ ls'. The output shows the files 'Maior.hs' and 'Maior.o'. The prompt is 'Mac-MHG:haskell mauro.hemerly\$'.

Linguagem Haskell

O que é um Programa Funcional

- Um programa funcional é uma **expressão**.

- Exemplo:

$(2 * 3) + 4$

- Executar um programa funcional significa **avaliar** a expressão:

$(2 * 3) + 4$ avalia para 10.

Linguagem Haskell

O que é um Programa Funcional

- A Programação Funcional é um estilo de programação em que o método básico de computação é a **expressão**.
- 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.

Java vs Haskell

- 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.

- 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 funções à argumentos.

Java vs Haskell

- 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.

- 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 funções à argumentos.

Java vs Haskell

- 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.

- 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 funções à argumentos.

Como Definir uma Função

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

```
soma x y = x + y  
Prelude> soma 12 34  
46
```



- A função **incrementar** pode ser escrita e testada:

```
incrementar x = x + 1  
Prelude> incrementar 99  
100
```

Como Definir uma Função

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

```
soma x y = x + y  
Prelude> soma 12 34  
46
```



- A função **incrementar** pode ser escrita e testada:

```
incrementar x = x + 1  
Prelude> incrementar 99  
100
```

Como Definir uma Função

Assinatura e Parâmetro Formal

- Retomando a função **incrementar** do slide anterior.

```
incrementar :: Int -> Int
```

```
incrementar x = x + 1
```

- Detalhes da definição da função:

1. A linha 1 declara `incrementar` como uma função recebendo um inteiro como entrada e dando ao usuário um inteiro como saída.
2. A linha 2 declara que a função `incrementar` determina um inteiro $x+1$ a partir de um inteiro x recebido.
3. O símbolo `=` é um operador de definição.

Como Definir uma Função

Assinatura e Parâmetro Formal

- Retomando a função **incrementar** do slide anterior.

```
incrementar :: Int -> Int
```

```
incrementar x = x + 1
```

- Detalhes da definição da função:
 - linha 1: a **assinatura** declara **incrementar** como uma função esperando um **inteiro na entrada** e tendo na **saída um inteiro**.
 - linha 2: **define** que a função **incrementar** determina um inteiro $x + 1$ a partir de um inteiro x qualquer.
 - o símbolo x é um **parâmetro formal**.

Como Definir uma Função

Assinatura e Parâmetro Formal

- Retomando a função **incrementar** do slide anterior.

```
incrementar :: Int -> Int
```

```
incrementar x = x + 1
```

- Detalhes da definição da função:
 - ▶ linha 1: a **assinatura** declara **incrementar** como uma função esperando um **inteiro na entrada** e tendo na **saída um inteiro**.
 - ▶ linha 2: **define** que a função **incrementar** determina um inteiro $x + 1$ a partir de um inteiro x qualquer.
 - ▶ o símbolo x é um **parâmetro formal**.

Como Definir uma Função

Assinatura e Parâmetro Formal

- Retomando a função **incrementar** do slide anterior.

```
incrementar :: Int -> Int
```

```
incrementar x = x + 1
```

- Detalhes da definição da função:
 - ▶ linha 1: a **assinatura** declara **incrementar** como uma função esperando um **inteiro na entrada** e tendo na **saída um inteiro**.
 - ▶ linha 2: **define** que a função **incrementar** determina um inteiro $x + 1$ a partir de um inteiro x qualquer.
 - ▶ o símbolo x é um **parâmetro formal**.

Como Definir uma Função

Assinatura e Parâmetro Formal

- Retomando a função **incrementar** do slide anterior.

```
incrementar :: Int -> Int
```

```
incrementar x = x + 1
```

- Detalhes da definição da função:
 - ▶ linha 1: a **assinatura** declara **incrementar** como uma função esperando um **inteiro na entrada** e tendo na **saída um inteiro**.
 - ▶ linha 2: **define** que a função **incrementar** determina um inteiro $x + 1$ a partir de um inteiro x qualquer.
 - ▶ o símbolo x é um **parâmetro formal**.

Como Definir uma Função

Convenções de Nomeação Obrigatórias

- Exemplo:

```
incrementar :: Int -> Int
```

```
incrementar x = x + 1
```

- Convenções:

• Funções e parâmetros devem iniciar com letra minúscula.

• Tipos de dados devem iniciar com letra maiúscula.

• Constantes devem iniciar com letra maiúscula e underline.

• Variáveis devem iniciar com letra minúscula.

• Funções devem iniciar com letra minúscula.

• Funções devem iniciar com letra minúscula.

Como Definir uma Função

Convenções de Nomeação Obrigatórias

- Exemplo:

```
incrementar :: Int -> Int
```

```
incrementar x = x + 1
```

- Convenções:

- ▶ Funções e parâmetros formais iniciam com letra **minúscula**.
- ▶ Tipos iniciam com letra **maiúscula**.

Como Definir uma Função

Convenções de Nomeação Obrigatórias

- Exemplo:

```
incrementar :: Int -> Int
```

```
incrementar x = x + 1
```

- Convenções:

- ▶ **Funções e parâmetros formais** iniciam com letra **minúscula**.
- ▶ **Tipos** iniciam com letra **maiúscula**.

Como Definir uma Função

Convenções de Nomeação Obrigatórias

- Exemplo:

```
incrementar :: Int -> Int
```

```
incrementar x = x + 1
```

- Convenções:

- ▶ **Funções** e **parâmetros formais** iniciam com letra **minúscula**.
- ▶ **Tipos** iniciam com letra **maiúscula**.

Como Definir uma Função

Definição vs Atribuição

- A definição

`incrementar x = x + 1`

Não deve se confundida com a atribuição

`x = x + 1`

em linguagens imperativas.

Como Definir uma Função

Funções com Mais de um Parâmetro Formal

- Exemplo de uma função definida com dois parâmetros:

```
dobraSoma :: Int -> Int -> Int
dobraSoma x y = 2 * (x + y)
```

- A combinação das funções **incrementa** e **dobraSoma**:

```
f :: Int -> Int -> Int
f x y = incrementa(dobraSoma x y)
```


Como Definir uma Função

Funções com Mais de um Parâmetro Formal

- Exemplo de uma função definida com dois parâmetros:

```
dobraSoma :: Int -> Int -> Int
dobraSoma x y = 2 * (x + y)
```

- A combinação das funções **incrementa** e **dobraSoma**:

```
f :: Int -> Int -> Int
f x y = incrementa(dobraSoma x y)
```

Outros Exemplos de Funções

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

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

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

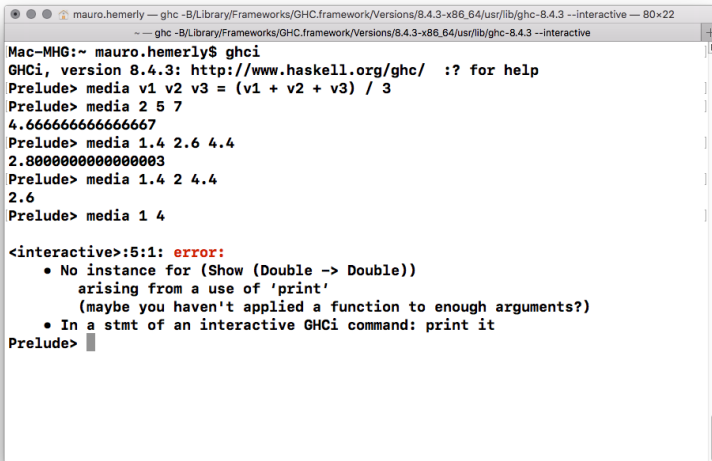
$$\text{areaCirculo } r = \text{pi} * r * r$$

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



Outros Exemplos de Funções

- Função para calcular a média entre três números:



```
mauro.hemerly — ghc -B/Library/Frameworks/GHC.framework/Versions/8.4.3-x86_64/usr/lib/ghc-8.4.3 --interactive — 80x22
~ — ghc -B/Library/Frameworks/GHC.framework/Versions/8.4.3-x86_64/usr/lib/ghc-8.4.3 --interactive
Mac-MHG:~ mauro.hemerly$ ghci
GHCi, version 8.4.3: http://www.haskell.org/ghc/  :? for help
Prelude> media v1 v2 v3 = (v1 + v2 + v3) / 3
Prelude> media 2 5 7
4.666666666666667
Prelude> media 1.4 2.6 4.4
2.8000000000000003
Prelude> media 1.4 2 4.4
2.6
Prelude> media 1 4
<interactive>:5:1: 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
Prelude>
```

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 exercício anterior.
- 3 Defina uma função para calcular a distância entre dois pontos (num plano).
- 4 Dadas as medidas dos catetos de um triângulo retângulo, calcular o valor de sua hipotenusa.
- 5 Considere as seguintes definições de funções:

inc $x = x + 1$

dobro $x = x + x$

quadrado $x = x * x$

media $x\ y = (x + y)/2$

Efetuando reduções passo-a-passo, calcule os valores das expressões seguintes:

(a) `inc (quadrado 5)`

(b) `quadrado (inc 5)`

(c) `media (dobro 3) (inc 5)`