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**.

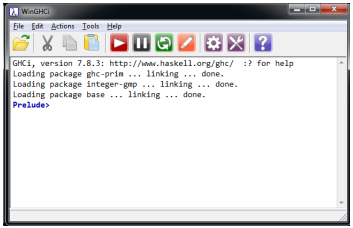
Linguagem Haskell

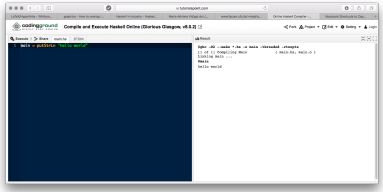
- Algumas características de **Haskell**
  - Programas são **concisos**
  - **Tipagem estática**
  - **Sistema de tipos** poderoso
  - Tipos e funções **recursivas**
  - **Funções de ordem superior**
  - Linguagem **pura** (declarativa)
  - Avaliação **lazy**
- **Em 1987:** Um comitê internacional de 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**.

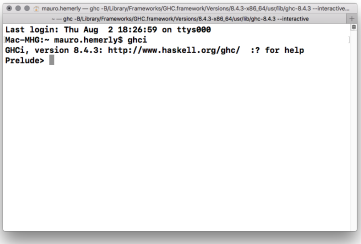
Algumas Empresas que Usam Haskell

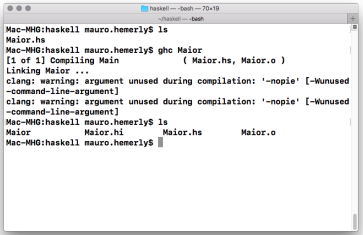
- Exemplos de empresas que usam **Haskell**:
  - **ABN AMRO** análise de riscos financeiros
  - **AT&T** automatização de processamento de formulários
  - **Bank of America Merril Lynch** transformação de dados
  - **Bump** servidores baseados em Haskell
  - **Facebook** manipulação da base de código 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









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:
  - linha 1: a assinatura declara que uma função esperando um inteiro na entrada e sendo na saída um inteiro.
  - linha 2: define que a função 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 $\mathbf{x + 1}$ 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 $\mathbf{x} + \mathbf{1}$ 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 $\mathbf{x} + \mathbf{1}$ 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 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

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)
```

Como Definir uma Função

Matemática vs Haskell

Maths	Haskell
$\max(x, y)$	`max x y`
$3(x^2 + y^2) + 5$	`3*(x^2+y^2)+5`
$\mathrm{ar} : \mathrm{Float} \rightarrow \mathrm{Float}$	`ar :: Float -> Float`

Linguagem Haskell

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
```