

Programação Funcional



Capítulo 1 Paradigmas de Programação

José Romildo Malaquias

Departamento de Computação
Universidade Federal de Ouro Preto

2017.2

- 1 Paradigmas de programação
- 2 Programação funcional
- 3 A crise do software
- 4 Algumas características de Haskell
- 5 Antecedentes históricos
- 6 Algumas empresas que usam Haskell
- 7 Curso online

- 1 Paradigmas de programação
- 2 Programação funcional
- 3 A crise do software
- 4 Algumas características de Haskell
- 5 Antecedentes históricos
- 6 Algumas empresas que usam Haskell
- 7 Curso online

Paradigmas de programação

- Um **paradigma de programação** fornece e determina a visão que o programador possui sobre a **estruturação** e a **execução** do programa.
- Por exemplo:
 - Em **programação orientada a objetos**, programadores podem abstrair um programa como uma *coleção de objetos* que interagem entre si.
 - Em **programação lógica** os programadores abstraem o programa como um *conjunto de predicados* que estabelecem relações entre objetos (axiomas), e uma *meta* (teorema) a ser provada usando os predicados.
- Diferentes linguagens de programação propõem diferentes paradigmas de programação.
- Algumas linguagens foram desenvolvidas para suportar um paradigma específico.
- Por exemplo:
 - **Smalltalk**, **Eiffel** e **Java** suportam o paradigma orientado a objetos.
 - **Haskell** e **Clean** suportam o paradigma funcional.
 - **OCaml**, **LISP**, **Scala**, **Perl**, **Python** e **C++** suportam múltiplos paradigmas.

Paradigmas de programação

- Um **paradigma de programação** fornece e determina a visão que o programador possui sobre a **estruturação** e a **execução** do programa.
- Por exemplo:
 - Em **programação orientada a objetos**, programadores podem abstrair um programa como uma *coleção de objetos* que interagem entre si.
 - Em **programação lógica** os programadores abstraem o programa como um *conjunto de predicados* que estabelecem relações entre objetos (axiomas), e uma *meta* (teorema) a ser provada usando os predicados.
- Diferentes linguagens de programação propõem diferentes paradigmas de programação.
- Algumas linguagens foram desenvolvidas para suportar um paradigma específico.
- Por exemplo:
 - **Smalltalk**, **Eiffel** e **Java** suportam o paradigma orientado a objetos.
 - **Haskell** e **Clean** suportam o paradigma funcional.
 - **OCaml**, **LISP**, **Scala**, **Perl**, **Python** e **C++** suportam múltiplos paradigmas.

Paradigmas de programação

- Um **paradigma de programação** fornece e determina a visão que o programador possui sobre a **estruturação** e a **execução** do programa.
- Por exemplo:
 - Em **programação orientada a objetos**, programadores podem abstrair um programa como uma *coleção de objetos* que interagem entre si.
 - Em **programação lógica** os programadores abstraem o programa como um *conjunto de predicados* que estabelecem relações entre objetos (axiomas), e uma *meta* (teorema) a ser provada usando os predicados.
- Diferentes linguagens de programação propõem diferentes paradigmas de programação.
- Algumas linguagens foram desenvolvidas para suportar um paradigma específico.
- Por exemplo:
 - **Smalltalk**, **Eiffel** e **Java** suportam o paradigma orientado a objetos.
 - **Haskell** e **Clean** suportam o paradigma funcional.
 - **OCaml**, **LISP**, **Scala**, **Perl**, **Python** e **C++** suportam múltiplos paradigmas.

Paradigmas de programação

- Um **paradigma de programação** fornece e determina a visão que o programador possui sobre a **estruturação** e a **execução** do programa.
- Por exemplo:
 - Em **programação orientada a objetos**, programadores podem abstrair um programa como uma *coleção de objetos* que interagem entre si.
 - Em **programação lógica** os programadores abstraem o programa como um *conjunto de predicados* que estabelecem relações entre objetos (axiomas), e uma *meta* (teorema) a ser provada usando os predicados.
- Diferentes linguagens de programação propõem diferentes paradigmas de programação.
- Algumas linguagens foram desenvolvidas para suportar um paradigma específico.
- Por exemplo:
 - **Smalltalk**, **Eiffel** e **Java** suportam o paradigma orientado a objetos.
 - **Haskell** e **Clean** suportam o paradigma funcional.
 - **OCaml**, **LISP**, **Scala**, **Perl**, **Python** e **C++** suportam múltiplos paradigmas.

Paradigmas de programação

- Um **paradigma de programação** fornece e determina a visão que o programador possui sobre a **estruturação** e a **execução** do programa.
- Por exemplo:
 - Em **programação orientada a objetos**, programadores podem abstrair um programa como uma *coleção de objetos* que interagem entre si.
 - Em **programação lógica** os programadores abstraem o programa como um *conjunto de predicados* que estabelecem relações entre objetos (axiomas), e uma *meta* (teorema) a ser provada usando os predicados.
- Diferentes linguagens de programação propõem diferentes paradigmas de programação.
- Algumas linguagens foram desenvolvidas para suportar um paradigma específico.
- Por exemplo:
 - **Smalltalk**, **Eiffel** e **Java** suportam o paradigma orientado a objetos.
 - **Haskell** e **Clean** suportam o paradigma funcional.
 - **OCaml**, **LISP**, **Scala**, **Perl**, **Python** e **C++** suportam múltiplos paradigmas.

Paradigmas de programação

- Um **paradigma de programação** fornece e determina a visão que o programador possui sobre a **estruturação** e a **execução** do programa.
- Por exemplo:
 - Em **programação orientada a objetos**, programadores podem abstrair um programa como uma *coleção de objetos* que interagem entre si.
 - Em **programação lógica** os programadores abstraem o programa como um *conjunto de predicados* que estabelecem relações entre objetos (axiomas), e uma *meta* (teorema) a ser provada usando os predicados.
- Diferentes linguagens de programação propõem diferentes paradigmas de programação.
- Algumas linguagens foram desenvolvidas para suportar um paradigma específico.
- Por exemplo:
 - Smalltalk, Eiffel e Java suportam o paradigma orientado a objetos.
 - Haskell e Clean suportam o paradigma funcional.
 - OCaml, LISP, Scala, Perl, Python e C++ suportam múltiplos paradigmas.

Paradigmas de programação

- Um **paradigma de programação** fornece e determina a visão que o programador possui sobre a **estruturação** e a **execução** do programa.
- Por exemplo:
 - Em **programação orientada a objetos**, programadores podem abstrair um programa como uma *coleção de objetos* que interagem entre si.
 - Em **programação lógica** os programadores abstraem o programa como um *conjunto de predicados* que estabelecem relações entre objetos (axiomas), e uma *meta* (teorema) a ser provada usando os predicados.
- Diferentes linguagens de programação propõem diferentes paradigmas de programação.
- Algumas linguagens foram desenvolvidas para suportar um paradigma específico.
- Por exemplo:
 - **Smalltalk**, **Eiffel** e **Java** suportam o paradigma orientado a objetos.
 - **Haskell** e **Clean** suportam o paradigma funcional.
 - **OCaml**, **LISP**, **Scala**, **Perl**, **Python** e **C++** suportam múltiplos paradigmas.

Paradigmas de programação

- Um **paradigma de programação** fornece e determina a visão que o programador possui sobre a **estruturação** e a **execução** do programa.
- Por exemplo:
 - Em **programação orientada a objetos**, programadores podem abstrair um programa como uma *coleção de objetos* que interagem entre si.
 - Em **programação lógica** os programadores abstraem o programa como um *conjunto de predicados* que estabelecem relações entre objetos (axiomas), e uma *meta* (teorema) a ser provada usando os predicados.
- Diferentes linguagens de programação propõem diferentes paradigmas de programação.
- Algumas linguagens foram desenvolvidas para suportar um paradigma específico.
- Por exemplo:
 - **Smalltalk**, **Eiffel** e **Java** suportam o paradigma orientado a objetos.
 - **Haskell** e **Clean** suportam o paradigma funcional.
 - **OCaml**, **LISP**, **Scala**, **Perl**, **Python** e **C++** suportam múltiplos paradigmas.

Paradigmas de programação

- Um **paradigma de programação** fornece e determina a visão que o programador possui sobre a **estruturação** e a **execução** do programa.
- Por exemplo:
 - Em **programação orientada a objetos**, programadores podem abstrair um programa como uma *coleção de objetos* que interagem entre si.
 - Em **programação lógica** os programadores abstraem o programa como um *conjunto de predicados* que estabelecem relações entre objetos (axiomas), e uma *meta* (teorema) a ser provada usando os predicados.
- Diferentes linguagens de programação propõem diferentes paradigmas de programação.
- Algumas linguagens foram desenvolvidas para suportar um paradigma específico.
- Por exemplo:
 - **Smalltalk**, **Eiffel** e **Java** suportam o paradigma orientado a objetos.
 - **Haskell** e **Clean** suportam o paradigma funcional.
 - **OCaml**, **LISP**, **Scala**, **Perl**, **Python** e **C++** suportam múltiplos paradigmas.

Paradigmas de programação

- Um **paradigma de programação** fornece e determina a visão que o programador possui sobre a **estruturação** e a **execução** do programa.
- Por exemplo:
 - Em **programação orientada a objetos**, programadores podem abstrair um programa como uma *coleção de objetos* que interagem entre si.
 - Em **programação lógica** os programadores abstraem o programa como um *conjunto de predicados* que estabelecem relações entre objetos (axiomas), e uma *meta* (teorema) a ser provada usando os predicados.
- Diferentes linguagens de programação propõem diferentes paradigmas de programação.
- Algumas linguagens foram desenvolvidas para suportar um paradigma específico.
- Por exemplo:
 - **Smalltalk**, **Eiffel** e **Java** suportam o paradigma orientado a objetos.
 - **Haskell** e **Clean** suportam o paradigma funcional.
 - **OCaml**, **LISP**, **Scala**, **Perl**, **Python** e **C++** suportam múltiplos paradigmas.

Técnicas e paradigmas de programação

- Geralmente os paradigmas de programação são diferenciados pelas **técnicas de programação que permitem ou proíbem**.
- Por exemplo, a **programação estruturada** não permite o uso de *goto*.
- Esse é um dos motivos pelos quais novos paradigmas são considerados mais rígidos que estilos tradicionais.
- Apesar disso, evitar certos tipos de técnicas pode facilitar a prova de correção de um sistema, podendo até mesmo facilitar o desenvolvimento de algoritmos.
- O relacionamento entre paradigmas de programação e linguagens de programação pode ser complexo pelo fato de linguagens de programação poderem **suportar mais de um paradigma**.

Técnicas e paradigmas de programação

- Geralmente os paradigmas de programação são diferenciados pelas **técnicas de programação que permitem ou proíbem**.
- Por exemplo, a **programação estruturada** não permite o uso de *goto*.
- Esse é um dos motivos pelos quais novos paradigmas são considerados mais rígidos que estilos tradicionais.
- Apesar disso, evitar certos tipos de técnicas pode facilitar a prova de correção de um sistema, podendo até mesmo facilitar o desenvolvimento de algoritmos.
- O relacionamento entre paradigmas de programação e linguagens de programação pode ser complexo pelo fato de linguagens de programação poderem **suportar mais de um paradigma**.

Técnicas e paradigmas de programação

- Geralmente os paradigmas de programação são diferenciados pelas **técnicas de programação que permitem ou proíbem**.
- Por exemplo, a **programação estruturada** não permite o uso de *goto*.
- Esse é um dos motivos pelos quais novos paradigmas são considerados mais rígidos que estilos tradicionais.
- Apesar disso, evitar certos tipos de técnicas pode facilitar a prova de correção de um sistema, podendo até mesmo facilitar o desenvolvimento de algoritmos.
- O relacionamento entre paradigmas de programação e linguagens de programação pode ser complexo pelo fato de linguagens de programação poderem **suportar mais de um paradigma**.

Técnicas e paradigmas de programação

- Geralmente os paradigmas de programação são diferenciados pelas **técnicas de programação que permitem ou proíbem**.
- Por exemplo, a **programação estruturada** não permite o uso de *goto*.
- Esse é um dos motivos pelos quais novos paradigmas são considerados mais rígidos que estilos tradicionais.
- Apesar disso, evitar certos tipos de técnicas pode facilitar a prova de correção de um sistema, podendo até mesmo facilitar o desenvolvimento de algoritmos.
- O relacionamento entre paradigmas de programação e linguagens de programação pode ser complexo pelo fato de linguagens de programação poderem **suportar mais de um paradigma**.

Técnicas e paradigmas de programação

- Geralmente os paradigmas de programação são diferenciados pelas **técnicas de programação que permitem ou proíbem**.
- Por exemplo, a **programação estruturada** não permite o uso de *goto*.
- Esse é um dos motivos pelos quais novos paradigmas são considerados mais rígidos que estilos tradicionais.
- Apesar disso, evitar certos tipos de técnicas pode facilitar a prova de correção de um sistema, podendo até mesmo facilitar o desenvolvimento de algoritmos.
- O relacionamento entre paradigmas de programação e linguagens de programação pode ser complexo pelo fato de linguagens de programação poderem **suportar mais de um paradigma**.

Categorias: programação imperativa e declarativa

Programação imperativa

- Descreve a computação como ações, enunciados ou comandos que **mudam o estado** (variáveis) de um programa, enfatizando **como resolver** um problema.
- Muito parecidos com o comportamento imperativo das linguagens naturais que expressam ordens, programas imperativos são uma **sequência de comandos** para o computador executar.
- O nome do paradigma, **imperativo**, está ligado ao tempo verbal imperativo, onde o programador diz ao computador: faça isso, depois isso, depois aquilo...
- Exemplos: paradigmas
 - **procedimental**: C, Pascal, etc, e
 - **orientado a objetos**: Smalltalk, Java, etc

Programação declarativa

- Descreve **o que** o programa faz e não **como** seus procedimentos funcionam.
- Ênfase nos **resultados**, no que se deseja obter.
- Exemplos: paradigmas
 - **funcional**: Haskell, OCaml, LISP, etc.
 - **declarativo**: Prolog, etc.

Categorias: programação imperativa e declarativa

Programação imperativa

- Descreve a computação como ações, enunciados ou comandos que **mudam o estado** (variáveis) de um programa, enfatizando **como resolver** um problema.
- Muito parecidos com o comportamento imperativo das linguagens naturais que expressam ordens, programas imperativos são uma **sequência de comandos** para o computador executar.
- O nome do paradigma, **imperativo**, está ligado ao tempo verbal imperativo, onde o programador diz ao computador: faça isso, depois isso, depois aquilo...
- Exemplos: paradigmas
 - **procedimental**: C, Pascal, etc, e
 - **orientado a objetos**: Smalltalk, Java, etc

Programação declarativa

- Descreve **o que** o programa faz e não **como** seus procedimentos funcionam.
- Ênfase nos **resultados**, no que se deseja obter.
- Exemplos: paradigmas
 - **funcional**: Haskell, OCaml, Elm, etc
 - **base de dados**: Prolog, etc

Categorias: programação imperativa e declarativa

Programação imperativa

- Descreve a computação como ações, enunciados ou comandos que **mudam o estado** (variáveis) de um programa, enfatizando **como resolver** um problema.
- Muito parecidos com o comportamento imperativo das linguagens naturais que expressam ordens, programas imperativos são uma **sequência de comandos** para o computador executar.
- O nome do paradigma, **imperativo**, está ligado ao tempo verbal imperativo, onde o programador diz ao computador: *faça isso, depois isso, depois aquilo...*
- Exemplos: paradigmas
 - **procedimental**: C, Pascal, etc, e
 - **orientado a objetos**: Smalltalk, Java, etc

Programação declarativa

- Descreve **o que** o programa faz e não *como* seus procedimentos funcionam.
- Ênfase nos **resultados**, no que se deseja obter.
- Exemplos: paradigmas

● **funcional**: Haskell, Clojure, Erlang, etc

● **base de dados**: Prolog, etc

Categorias: programação imperativa e declarativa

Programação imperativa

- Descreve a computação como ações, enunciados ou comandos que **mudam o estado** (variáveis) de um programa, enfatizando **como resolver** um problema.
- Muito parecidos com o comportamento imperativo das linguagens naturais que expressam ordens, programas imperativos são uma **sequência de comandos** para o computador executar.
- O nome do paradigma, **imperativo**, está ligado ao tempo verbal imperativo, onde o programador diz ao computador: faça isso, depois isso, depois aquilo...
- Exemplos: paradigmas
 - **procedimental**: C, Pascal, etc, e
 - **orientado a objetos**: Smalltalk, Java, etc

Programação declarativa

- Descreve **o que** o programa faz e não **como** seus procedimentos funcionam.
- Ênfase nos **resultados**, no que se deseja obter.
- Exemplos: paradigmas

Categorias: programação imperativa e declarativa

Programação imperativa

- Descreve a computação como ações, enunciados ou comandos que **mudam o estado** (variáveis) de um programa, enfatizando *como resolver* um problema.
- Muito parecidos com o comportamento imperativo das linguagens naturais que expressam ordens, programas imperativos são uma **sequência de comandos** para o computador executar.
- O nome do paradigma, **imperativo**, está ligado ao tempo verbal imperativo, onde o programador diz ao computador: faça isso, depois isso, depois aquilo...
- Exemplos: paradigmas
 - **procedimental**: C, Pascal, etc, e
 - **orientado a objetos**: Smalltalk, Java, etc

Programação declarativa

- Descreve *o que* o programa faz e não *como* seus procedimentos funcionam.
- Ênfase nos **resultados**, no que se deseja obter.
- Exemplos: paradigmas

Categorias: programação imperativa e declarativa

Programação imperativa

- Descreve a computação como ações, enunciados ou comandos que **mudam o estado** (variáveis) de um programa, enfatizando *como resolver* um problema.
- Muito parecidos com o comportamento imperativo das linguagens naturais que expressam ordens, programas imperativos são uma **sequência de comandos** para o computador executar.
- O nome do paradigma, **imperativo**, está ligado ao tempo verbal imperativo, onde o programador diz ao computador: faça isso, depois isso, depois aquilo...
- Exemplos: paradigmas
 - **procedimental**: C, Pascal, etc, e
 - **orientado a objetos**: Smalltalk, Java, etc

Programação declarativa

- Descreve *o que* o programa faz e não *como* seus procedimentos funcionam.
- Ênfase nos **resultados**, no que se deseja obter.
- Exemplos: paradigmas

Categorias: programação imperativa e declarativa

Programação imperativa

- Descreve a computação como ações, enunciados ou comandos que **mudam o estado** (variáveis) de um programa, enfatizando **como resolver** um problema.
- Muito parecidos com o comportamento imperativo das linguagens naturais que expressam ordens, programas imperativos são uma **sequência de comandos** para o computador executar.
- O nome do paradigma, **imperativo**, está ligado ao tempo verbal imperativo, onde o programador diz ao computador: faça isso, depois isso, depois aquilo...
- Exemplos: paradigmas
 - **procedimental**: C, Pascal, etc, e
 - **orientado a objetos**: Smalltalk, Java, etc

Programação declarativa

- Descreve **o que** o programa faz e não **como** seus procedimentos funcionam.
- Ênfase nos **resultados**, no que se deseja obter.
- Exemplos: paradigmas

Categorias: programação imperativa e declarativa

Programação imperativa

- Descreve a computação como ações, enunciados ou comandos que **mudam o estado** (variáveis) de um programa, enfatizando *como resolver* um problema.
- Muito parecidos com o comportamento imperativo das linguagens naturais que expressam ordens, programas imperativos são uma **sequência de comandos** para o computador executar.
- O nome do paradigma, **imperativo**, está ligado ao tempo verbal imperativo, onde o programador diz ao computador: faça isso, depois isso, depois aquilo...
- Exemplos: paradigmas
 - **procedimental**: C, Pascal, etc, e
 - **orientado a objetos**: Smalltalk, Java, etc

Programação declarativa

- Descreve **o que** o programa faz e não *como* seus procedimentos funcionam.
- Ênfase nos **resultados**, no que se deseja obter.
- Exemplos: paradigmas
 - **funcional**: Haskell, OCaml, LISP, etc, e
 - **lógico**: Prolog, etc.

Categorias: programação imperativa e declarativa

Programação imperativa

- Descreve a computação como ações, enunciados ou comandos que **mudam o estado** (variáveis) de um programa, enfatizando *como resolver* um problema.
- Muito parecidos com o comportamento imperativo das linguagens naturais que expressam ordens, programas imperativos são uma **sequência de comandos** para o computador executar.
- O nome do paradigma, **imperativo**, está ligado ao tempo verbal imperativo, onde o programador diz ao computador: faça isso, depois isso, depois aquilo...
- Exemplos: paradigmas
 - **procedimental**: C, Pascal, etc, e
 - **orientado a objetos**: Smalltalk, Java, etc

Programação declarativa

- Descreve **o que** o programa faz e não *como* seus procedimentos funcionam.
- Ênfase nos **resultados**, no que se deseja obter.
- Exemplos: paradigmas
 - **funcional**: Haskell, OCaml, LISP, etc, e
 - **lógico**: Prolog, etc.

Categorias: programação imperativa e declarativa

Programação imperativa

- Descreve a computação como ações, enunciados ou comandos que **mudam o estado** (variáveis) de um programa, enfatizando *como resolver* um problema.
- Muito parecidos com o comportamento imperativo das linguagens naturais que expressam ordens, programas imperativos são uma **sequência de comandos** para o computador executar.
- O nome do paradigma, **imperativo**, está ligado ao tempo verbal imperativo, onde o programador diz ao computador: faça isso, depois isso, depois aquilo...
- Exemplos: paradigmas
 - **procedimental**: C, Pascal, etc, e
 - **orientado a objetos**: Smalltalk, Java, etc

Programação declarativa

- Descreve **o que** o programa faz e não *como* seus procedimentos funcionam.
- Ênfase nos **resultados**, no que se deseja obter.
- Exemplos: paradigmas
 - **funcional**: Haskell, OCaml, LISP, etc, e
 - **lógico**: Prolog, etc.

Categorias: programação imperativa e declarativa

Programação imperativa

- Descreve a computação como ações, enunciados ou comandos que **mudam o estado** (variáveis) de um programa, enfatizando *como resolver* um problema.
- Muito parecidos com o comportamento imperativo das linguagens naturais que expressam ordens, programas imperativos são uma **sequência de comandos** para o computador executar.
- O nome do paradigma, **imperativo**, está ligado ao tempo verbal imperativo, onde o programador diz ao computador: faça isso, depois isso, depois aquilo...
- Exemplos: paradigmas
 - **procedimental**: C, Pascal, etc, e
 - **orientado a objetos**: Smalltalk, Java, etc

Programação declarativa

- Descreve **o que** o programa faz e não *como* seus procedimentos funcionam.
- Ênfase nos **resultados**, no que se deseja obter.
- Exemplos: paradigmas
 - **funcional**: Haskell, OCaml, LISP, etc, e
 - **lógico**: Prolog, etc.

Categorias: programação imperativa e declarativa

Programação imperativa

- Descreve a computação como ações, enunciados ou comandos que **mudam o estado** (variáveis) de um programa, enfatizando *como resolver* um problema.
- Muito parecidos com o comportamento imperativo das linguagens naturais que expressam ordens, programas imperativos são uma **sequência de comandos** para o computador executar.
- O nome do paradigma, **imperativo**, está ligado ao tempo verbal imperativo, onde o programador diz ao computador: faça isso, depois isso, depois aquilo...
- Exemplos: paradigmas
 - **procedimental**: C, Pascal, etc, e
 - **orientado a objetos**: Smalltalk, Java, etc

Programação declarativa

- Descreve **o que** o programa faz e não *como* seus procedimentos funcionam.
- Ênfase nos **resultados**, no que se deseja obter.
- Exemplos: paradigmas
 - **funcional**: Haskell, OCaml, LISP, etc, e
 - **lógico**: Prolog, etc.

Categorias: programação imperativa e declarativa

Programação imperativa

- Descreve a computação como ações, enunciados ou comandos que **mudam o estado** (variáveis) de um programa, enfatizando *como resolver* um problema.
- Muito parecidos com o comportamento imperativo das linguagens naturais que expressam ordens, programas imperativos são uma **sequência de comandos** para o computador executar.
- O nome do paradigma, **imperativo**, está ligado ao tempo verbal imperativo, onde o programador diz ao computador: faça isso, depois isso, depois aquilo...
- Exemplos: paradigmas
 - **procedimental**: C, Pascal, etc, e
 - **orientado a objetos**: Smalltalk, Java, etc

Programação declarativa

- Descreve **o que** o programa faz e não *como* seus procedimentos funcionam.
- Ênfase nos **resultados**, no que se deseja obter.
- Exemplos: paradigmas
 - **funcional**: Haskell, OCaml, LISP, etc, e
 - **lógico**: Prolog, etc.

- 1 Paradigmas de programação
- 2 Programação funcional
- 3 A crise do software
- 4 Algumas características de Haskell
- 5 Antecedentes históricos
- 6 Algumas empresas que usam Haskell
- 7 Curso online

Programação funcional

- **Programação funcional** é um paradigma de programação que descreve uma computação como uma **expressão** a ser avaliada.
- A principal forma de estruturar o programa é pela definição e aplicação de **funções**.

Exemplo: quick sort em C

```
// To sort array a[] of size n: qsort(a,0,n-1)
```

```
void qsort(int a[], int lo, int hi) {  
    int h, l, p, t;  
  
    if (lo < hi) {  
        l = lo;  
        h = hi;  
        p = a[hi];  
  
        do {  
            while ((l < h) && (a[l] <= p))  
                l = l+1;  
            while ((h > l) && (a[h] >= p))  
                h = h-1;  
            if (l < h) {  
                t = a[l];  
                a[l] = a[h];  
                a[h] = t;  
            }  
        } while (l < h);  
  
        a[hi] = a[l];  
        a[l] = p;  
  
        qsort(a, lo, l-1);  
        qsort(a, l+1, hi);  
    }  
}
```

Exemplo: quick sort em Haskell

```
qs []      = []  
qs (x:xs) = qs (filter (<x) xs) ++  
             [x] ++  
             qs (filter (>x) xs)
```

Observações:

- o símbolo `=` separa os lados esquerdo e direito de uma equação definindo variáveis ou funções.
- `[]` denota a lista vazia.
- `x:xs` denota uma lista não vazia cuja cabeça é `x` e cuja cauda é `xs`.
- Uma lista pode ser escrita enumerando os seus elementos separados por vírgula e colocados entre colchetes.
- A sintaxe para aplicação de função consiste em escrever a função seguida dos argumentos, separados por espaços, como em `max 10 (2+x)`.
- A função `filter` seleciona os elementos de uma lista que satisfazem uma determinada propriedade (expressa por uma função que retorna *verdadeiro* ou *falso*).
- `(<x)` e `(>x)` são funções que verificam se o seu argumento é menor ou maior do que `x`, respectivamente. São funções anônimas construídas pela aplicação parcial dos operadores `<` e `>`.
- O operador `++` concatena duas listas.

- 1 Paradigmas de programação
- 2 Programação funcional
- 3 A crise do software
- 4 Algumas características de Haskell
- 5 Antecedentes históricos
- 6 Algumas empresas que usam Haskell
- 7 Curso online

A Crise do Software

- *Linguagens declarativas* (incluindo linguagens funcionais):
 - permitem que programas sejam escritos de forma **clara, concisa**, e com um **alto nível de abstração**;
 - suportam componentes de software **reutilizáveis**;
 - incentivam o uso de **verificação formal**;
 - permitem **prototipagem rápida**;
 - fornecem **poderosas ferramentas** de resolução de problemas.
- Estas características podem ser úteis na abordagem de dificuldades encontradas no desenvolvimento de software:
 - o **tamanho** e a **complexidade** dos programas de computador modernos
 - o **tempo** e o **custo** de desenvolvimento do programas
 - **confiança** de que os programas já concluídos funcionam **corretamente**



As **linguagens funcionais** oferecem um quadro particularmente elegante para abordar estes objetivos.

A Crise do Software

- *Linguagens declarativas* (incluindo linguagens funcionais):
 - permitem que programas sejam escritos de forma **clara, concisa**, e com um **alto nível de abstração**;
 - suportam componentes de software **reutilizáveis**;
 - incentivam o uso de **verificação formal**;
 - permitem **prototipagem rápida**;
 - fornecem **poderosas ferramentas** de resolução de problemas.
- Estas características podem ser úteis na abordagem de dificuldades encontradas no desenvolvimento de software:
 - o **tamanho** e a **complexidade** dos programas de computador modernos
 - o **tempo** e o **custo** de desenvolvimento do programas
 - **confiança** de que os programas já concluídos funcionam **corretamente**



As **linguagens funcionais** oferecem um quadro particularmente elegante para abordar estes objetivos.

A Crise do Software

- *Linguagens declarativas* (incluindo linguagens funcionais):
 - permitem que programas sejam escritos de forma **clara**, **concisa**, e com um **alto nível de abstração**;
 - suportam componentes de software **reutilizáveis**;
 - incentivam o uso de **verificação formal**;
 - permitem **prototipagem rápida**;
 - fornecem **poderosas ferramentas** de resolução de problemas.
- Estas características podem ser úteis na abordagem de dificuldades encontradas no desenvolvimento de software:
 - o **tamanho** e a **complexidade** dos programas de computador modernos
 - o **tempo** e o **custo** de desenvolvimento do programas
 - **confiança** de que os programas já concluídos funcionam **corretamente**



As **linguagens funcionais** oferecem um quadro particularmente elegante para abordar estes objetivos.

A Crise do Software

- *Linguagens declarativas* (incluindo linguagens funcionais):
 - permitem que programas sejam escritos de forma **clara, concisa**, e com um **alto nível de abstração**;
 - suportam componentes de software **reutilizáveis**;
 - incentivam o uso de **verificação formal**;
 - permitem **prototipagem rápida**;
 - fornecem **poderosas ferramentas** de resolução de problemas.
- Estas características podem ser úteis na abordagem de dificuldades encontradas no desenvolvimento de software:
 - o **tamanho** e a **complexidade** dos programas de computador modernos
 - o **tempo** e o **custo** de desenvolvimento do programas
 - **confiança** de que os programas já concluídos funcionam **corretamente**



As **linguagens funcionais** oferecem um quadro particularmente elegante para abordar estes objetivos.

A Crise do Software

- *Linguagens declarativas* (incluindo linguagens funcionais):
 - permitem que programas sejam escritos de forma **clara**, **concisa**, e com um **alto nível de abstração**;
 - suportam componentes de software **reutilizáveis**;
 - incentivam o uso de **verificação formal**;
 - permitem **prototipagem rápida**;
 - fornecem **poderosas ferramentas** de resolução de problemas.
- Estas características podem ser úteis na abordagem de dificuldades encontradas no desenvolvimento de software:
 - o **tamanho** e a **complexidade** dos programas de computador modernos
 - o **tempo** e o **custo** de desenvolvimento do programas
 - **confiança** de que os programas já concluídos funcionam **corretamente**



As **linguagens funcionais** oferecem um quadro particularmente elegante para abordar estes objetivos.

A Crise do Software

- *Linguagens declarativas* (incluindo linguagens funcionais):
 - permitem que programas sejam escritos de forma **clara, concisa**, e com um **alto nível de abstração**;
 - suportam componentes de software **reutilizáveis**;
 - incentivam o uso de **verificação formal**;
 - permitem **prototipagem rápida**;
 - fornecem **poderosas ferramentas** de resolução de problemas.
- Estas características podem ser úteis na abordagem de dificuldades encontradas no desenvolvimento de software:
 - o **tamanho** e a **complexidade** dos programas de computador modernos
 - o **tempo** e o **custo** de desenvolvimento do programas
 - **confiança** de que os programas já concluídos funcionam **corretamente**



As **linguagens funcionais** oferecem um quadro particularmente elegante para abordar estes objetivos.

A Crise do Software

- *Linguagens declarativas* (incluindo linguagens funcionais):
 - permitem que programas sejam escritos de forma **clara**, **concisa**, e com um **alto nível de abstração**;
 - suportam componentes de software **reutilizáveis**;
 - incentivam o uso de **verificação formal**;
 - permitem **prototipagem rápida**;
 - fornecem **poderosas ferramentas** de resolução de problemas.
- Estas características podem ser úteis na abordagem de dificuldades encontradas no desenvolvimento de software:
 - o **tamanho** e a **complexidade** dos programas de computador modernos
 - o **tempo** e o **custo** de desenvolvimento do programas
 - **confiança** de que os programas já concluídos funcionam **corretamente**



As **linguagens funcionais** oferecem um quadro particularmente elegante para abordar estes objetivos.

A Crise do Software

- *Linguagens declarativas* (incluindo linguagens funcionais):
 - permitem que programas sejam escritos de forma **clara**, **concisa**, e com um **alto nível de abstração**;
 - suportam componentes de software **reutilizáveis**;
 - incentivam o uso de **verificação formal**;
 - permitem **prototipagem rápida**;
 - fornecem **poderosas ferramentas** de resolução de problemas.
- Estas características podem ser úteis na abordagem de dificuldades encontradas no desenvolvimento de software:
 - o **tamanho** e a **complexidade** dos programas de computador modernos
 - o **tempo** e o **custo** de desenvolvimento dos programas
 - **confiança** de que os programas já concluídos funcionam **corretamente**



As **linguagens funcionais** oferecem um quadro particularmente elegante para abordar estes objetivos.

A Crise do Software

- *Linguagens declarativas* (incluindo linguagens funcionais):
 - permitem que programas sejam escritos de forma **clara**, **concisa**, e com um **alto nível de abstração**;
 - suportam componentes de software **reutilizáveis**;
 - incentivam o uso de **verificação formal**;
 - permitem **prototipagem rápida**;
 - fornecem **poderosas ferramentas** de resolução de problemas.
- Estas características podem ser úteis na abordagem de dificuldades encontradas no desenvolvimento de software:
 - o **tamanho** e a **complexidade** dos programas de computador modernos
 - o **tempo** e o **custo** de desenvolvimento do programas
 - **confiança** de que os programas já concluídos funcionam **corretamente**



As **linguagens funcionais** oferecem um quadro particularmente elegante para abordar estes objetivos.

A Crise do Software

- *Linguagens declarativas* (incluindo linguagens funcionais):
 - permitem que programas sejam escritos de forma **clara, concisa**, e com um **alto nível de abstração**;
 - suportam componentes de software **reutilizáveis**;
 - incentivam o uso de **verificação formal**;
 - permitem **prototipagem rápida**;
 - fornecem **poderosas ferramentas** de resolução de problemas.
- Estas características podem ser úteis na abordagem de dificuldades encontradas no desenvolvimento de software:
 - o **tamanho** e a **complexidade** dos programas de computador modernos
 - o **tempo** e o **custo** de desenvolvimento dos programas
 - **confiança** de que os programas já concluídos funcionam **corretamente**



As **linguagens funcionais** oferecem um quadro particularmente elegante para abordar estes objetivos.

- 1 Paradigmas de programação
- 2 Programação funcional
- 3 A crise do software
- 4 Algumas características de Haskell**
- 5 Antecedentes históricos
- 6 Algumas empresas que usam Haskell
- 7 Curso online

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*
- Maior facilidade de raciocínio sobre programas

- 1 Paradigmas de programação
- 2 Programação funcional
- 3 A crise do software
- 4 Algumas características de Haskell
- 5 Antecedentes históricos
- 6 Algumas empresas que usam Haskell
- 7 Curso online

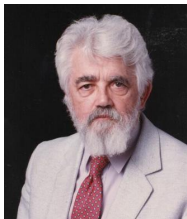
- Década de 1930:



Alonzo Church desenvolve o **cálculo lambda**, uma teoria de funções simples, mas poderosa.

Antecedentes históricos (cont.)

- Década de 1950:



John McCarthy desenvolve **Lisp**, a primeira linguagem funcional, com algumas influências do **cálculo lambda**, mas mantendo as atribuições de variáveis.

Antecedentes históricos (cont.)

- Década de 1960:



Peter Landin desenvolve **ISWIM**, a primeira linguagem funcional pura, baseada fortemente no **cálculo lambda**, sem atribuições.

- Década de 1970:



John Backus desenvolve **FP**, uma linguagem funcional que enfatiza funções de ordem superior e raciocínio sobre programas.

Antecedentes históricos (cont.)

- Década de 1970:



Robin Milner e outros desenvolvem **ML**, a primeira linguagem funcional moderna, que introduziu a inferência de tipos e tipos polimórficos.

Antecedentes históricos (cont.)

- Décadas de 1970 e 1980:



David Turner desenvolve uma série de linguagens funcionais com *avaliação lazy*, culminando com o sistema **Miranda**.

- 1987:

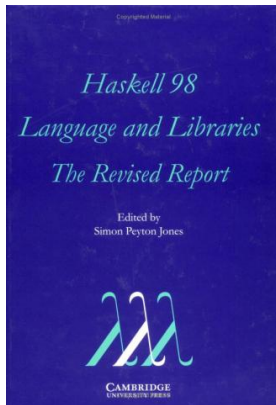
Haskell

A Purely Functional Language

Um comitê internacional de pesquisadores inicia o desenvolvimento de Haskell, uma linguagem funcional *lazy* padrão.

Antecedentes históricos (cont.)

- 2003:



O comitê publica o relatório **Haskell 98**, a definição de uma versão estável da linguagem Haskell.

- 2009:



O comitê publica o relatório **Haskell 2010**, uma revisão da definição da linguagem Haskell.

- 1 Paradigmas de programação
- 2 Programação funcional
- 3 A crise do software
- 4 Algumas características de Haskell
- 5 Antecedentes históricos
- 6 Algumas empresas que usam Haskell
- 7 Curso online

Quem usa 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 maiores detalhes visite a página *Haskell na indústria* em http://www.haskell.org/haskellwiki/Haskell_in_industry.

- 1 Paradigmas de programação
- 2 Programação funcional
- 3 A crise do software
- 4 Algumas características de Haskell
- 5 Antecedentes históricos
- 6 Algumas empresas que usam Haskell
- 7 Curso online



- **Functional Programming in Haskell: Supercharge Your Coding**
- Universidade de Glasgow
- Início: 17 de setembro de 2017
- Duração: 6 semanas
- Dedicção: 4 horas por semana
- <https://www.futurelearn.com/courses/functional-programming-haskell>

Fim