

Programação Funcional



Capítulo 1 Introdução

José Romildo Malaquias

Departamento de Computação
Universidade Federal de Ouro Preto

2015.2

1 Paradigmas de programação

2 Programação funcional

3 A Crise do Software

1 Paradigmas de programação

2 Programação funcional

3 A Crise do Software

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, Erlang, etc
 - **lógica**: 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, 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

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

- **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 OCaml

```
let rec qsort lista =  
  match lista with  
  | []          -> []  
  | x :: xs    -> qsort (List.filter (fun y -> y < x) xs) @  
                    [x] @  
                    qsort (List.filter (fun y -> y > x) xs)
```

Observações:

- `[]` 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 ponto-e-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.
- `fun y -> y < x` e `fun y -> y > x` são funções que verificam se o seu argumento é menor ou maior, respectivamente, do que `x`. São funções anônimas, também chamadas de expressões lambda.
- O operador `@` concatena duas listas.
- A expressão `match` permite analisar a estrutura da lista, acessando os componentes de uma lista não vazia.

1 Paradigmas de programação

2 Programação funcional

3 A Crise do Software

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 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.

Fim