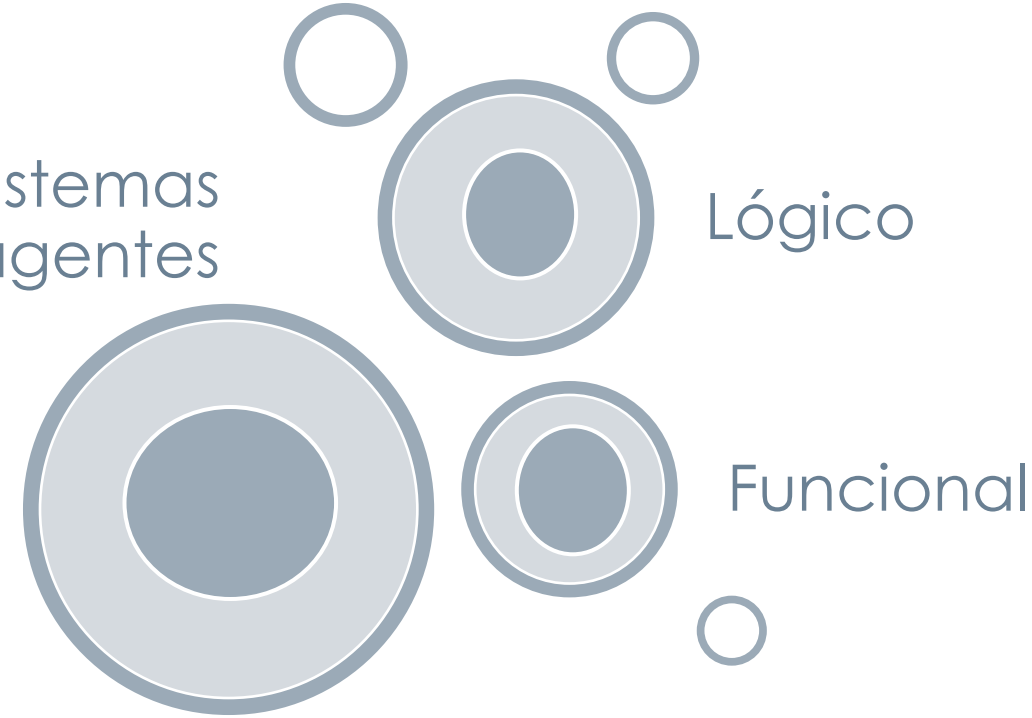


Paradigmas – Aula 01b

Professora: Milene Serrano

Agenda

- **Aula de HOJE:**
  - Visão Geral sobre Paradigmas de Programação;
    - Paradigma Imperativo;
    - Paradigma Estruturado;
    - Paradigma Orientado a Objetos;
    - Paradigma Funcional;
    - Paradigma Lógico;
    - Paradigma Concorrente, e
    - Paradigma Multiagentes.
  - Considerações Finais;
  - Exercícios, e
  - Referências.

Considerações Iniciais

Considerações Iniciais

Já vimos que um Paradigma pode ser entendido como um **modelo conceitual** que orienta soluções de projeto e implementação.

Esse modelo conceitual "enxerga" a realidade ou o problema sob um determinado ponto de vista, ou seja, considerando princípios e conceitos em particular.

Portanto, vale ressaltar que dado um problema, um determinado Paradigma de Programação pode ser mais adequado do que outro para lidar com esse problema.

Considerações Iniciais

Alguns Paradigmas de Programação conhecidos: **Imperativo**, **Estruturado**, **Orientado a Objetos**, **Funcional**, **Lógico**, **Concorrente**, **Multi-Agentes**, e outros

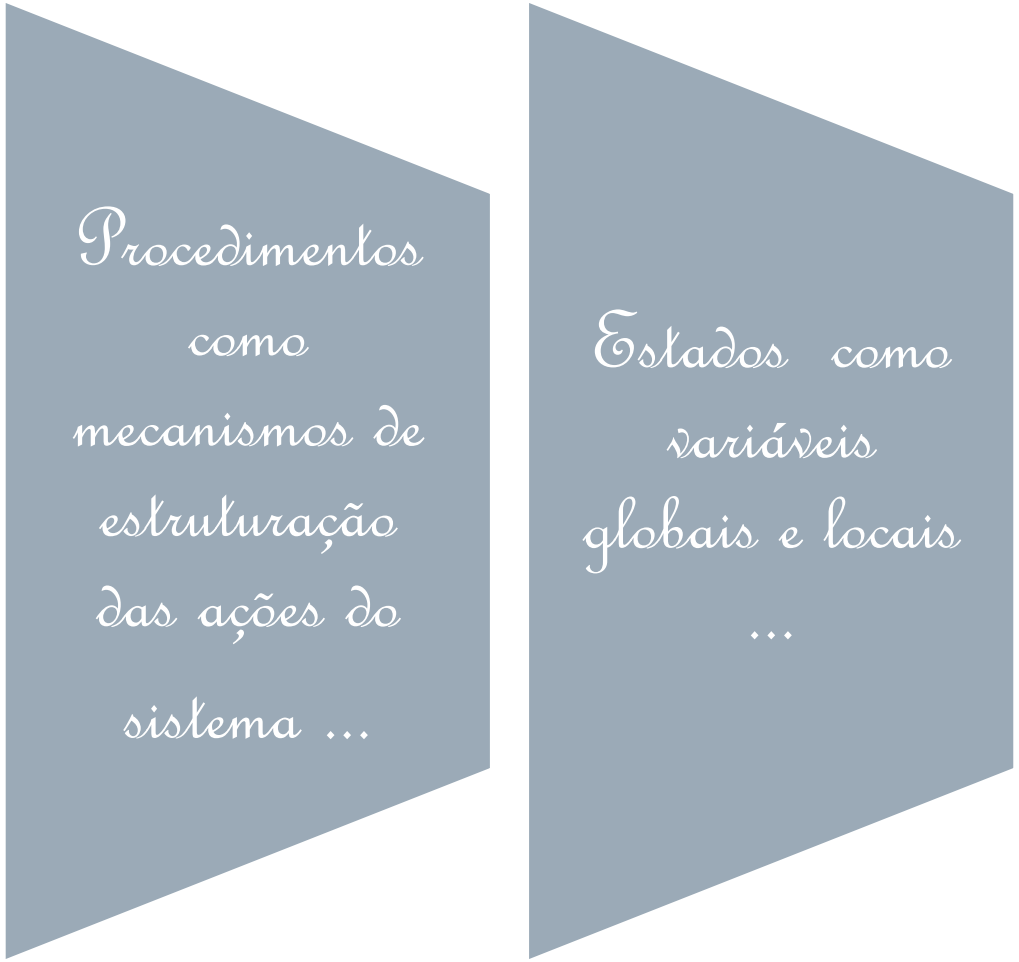
Cada qual determina uma forma particular de abordar os problemas e de formular as respectivas soluções.

Além disso, uma linguagem de programação pode combinar dois ou mais paradigmas para potencializar as análises e soluções.

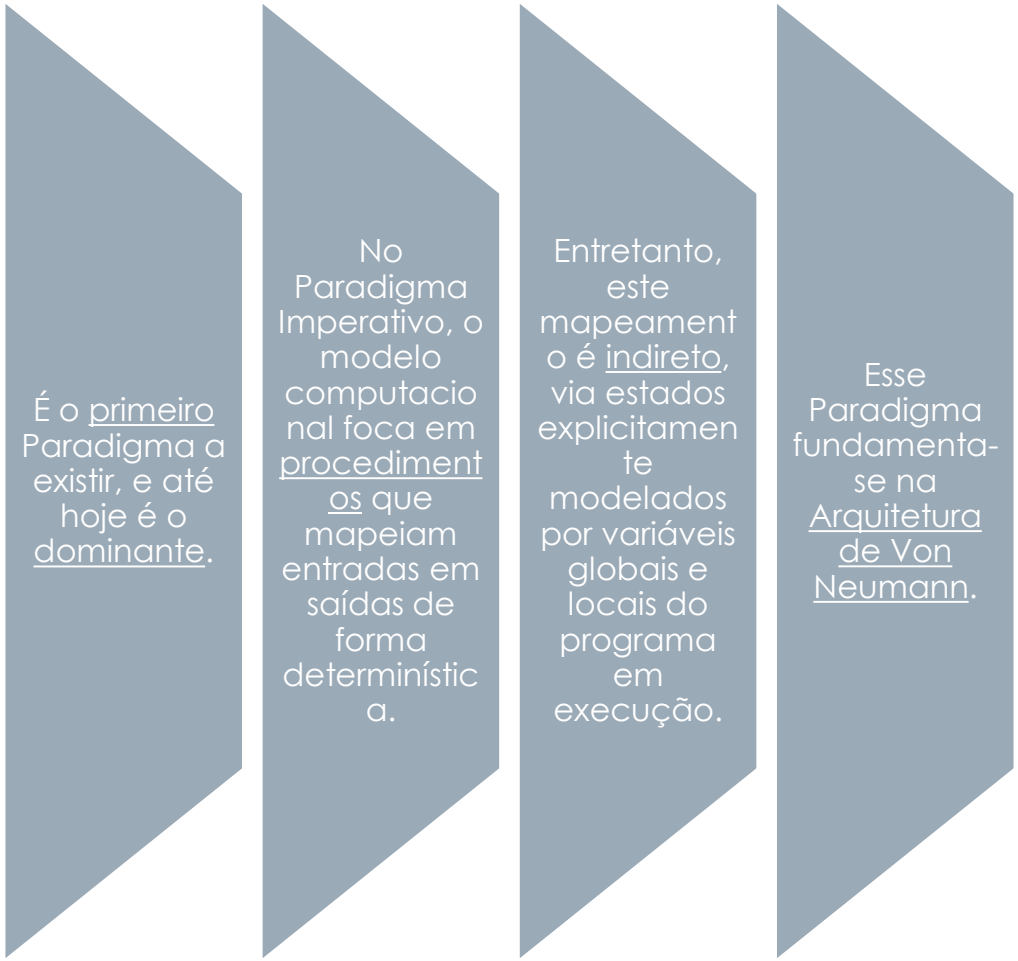
Deste modo, cabe ao programador escolher o Paradigma ou a combinação de Paradigmas mais adequado(a) para analisar e resolver cada problema.

Paradigma de Programação Imperativo

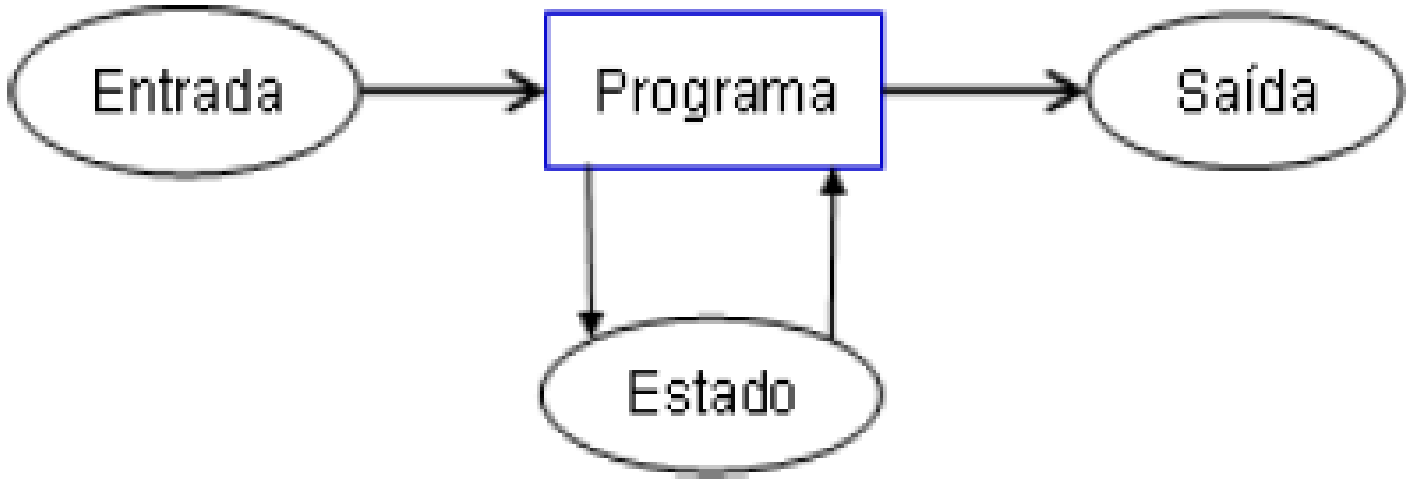
Paradigma de Programação Imperativo



Paradigma de Programação Imperativo



Paradigma de Programação Imperativo



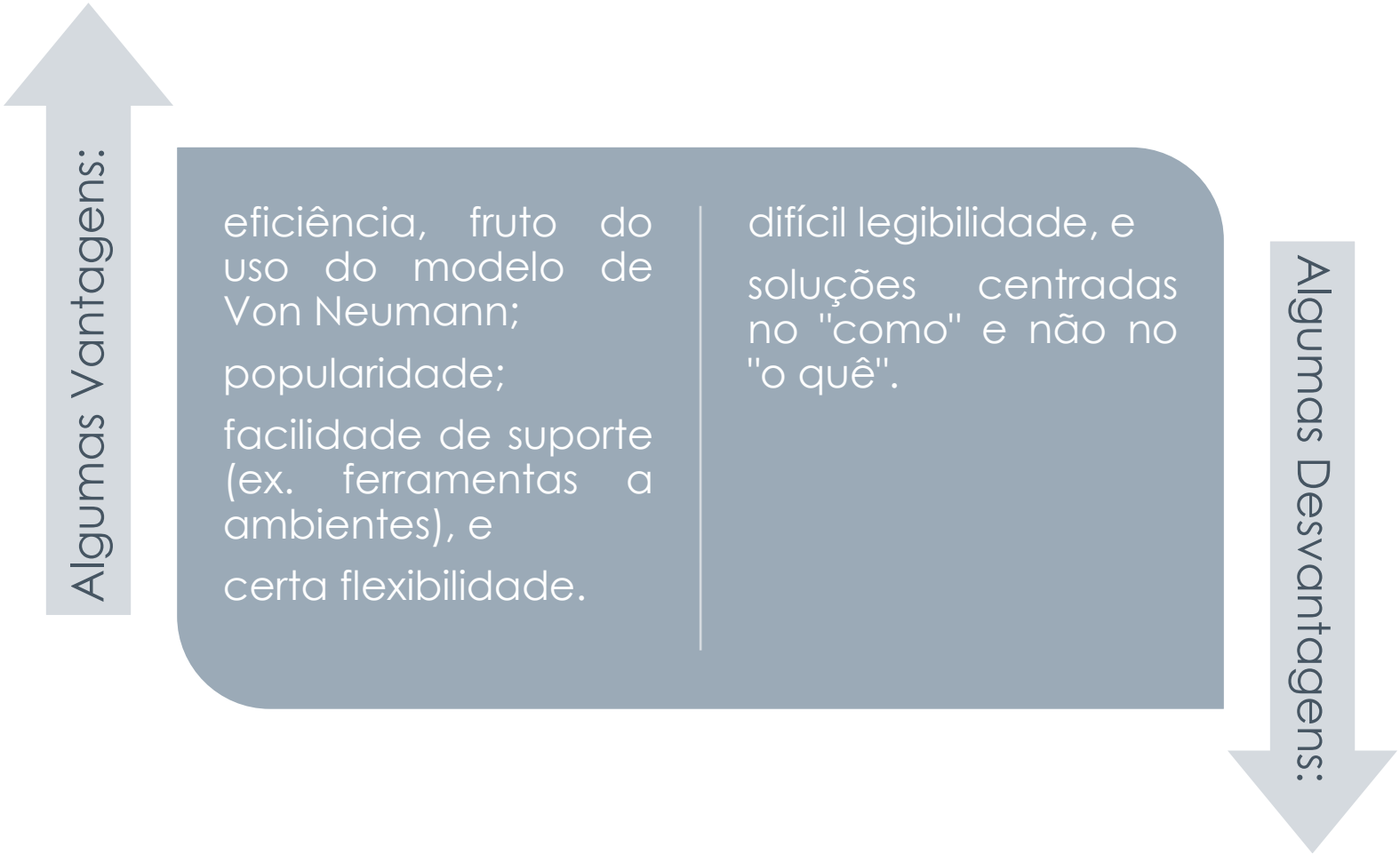
Paradigma de Programação Imperativo

Exemplos de Linguagens orientadas pelo Paradigma Imperativo:

- Ada;
- ALGOL;
- Assembler;
- Basic;
- C;
- Cobol;
- Fortran;
- Pascal;
- Python;
- Lua,
- Outras.

Acredito que sejam as mais conhecidas …

Paradigma de Programação Imperativo



Paradigma de Programação Estrutural

Paradigma de Programação Estrutural

Estruturas Simples, como sequência, decisão, iteração, na modelagem e implementação de Sistemas ...

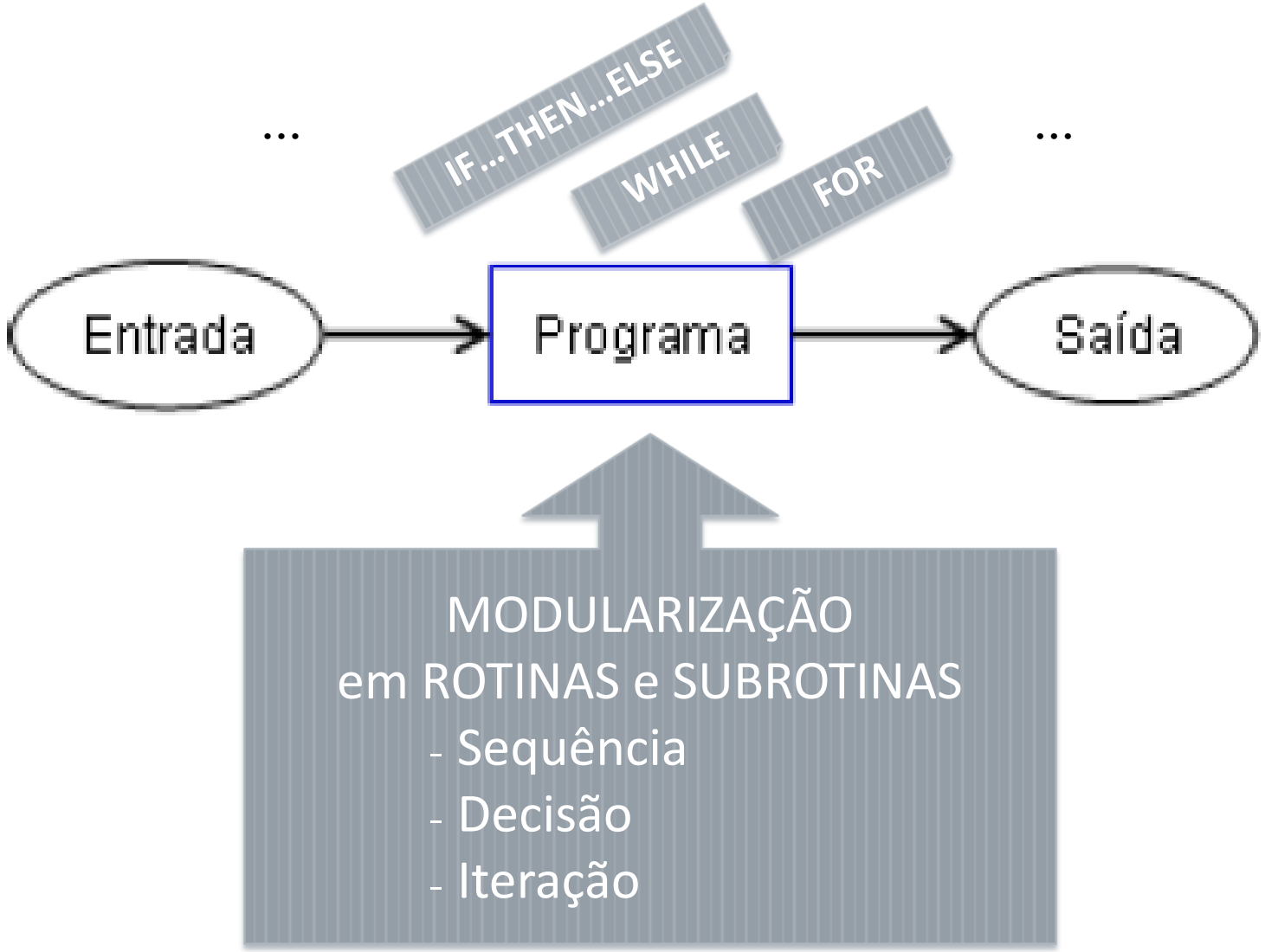
Paradigma de Programação Estrutural

Baseia-se na construção de programas usando estruturas simples, modulares, centradas nas noções de sequência, decisão e iteração.

No Paradigma Estruturado, o modelo computacional foca na **Programação Modular**, orientada pela criação de estruturas simples, usando rotinas e subrotinas.

Foi a forma dominante na criação de software entre a programação linear e a programação orientada por objetos.

Paradigma de Programação Estrutural



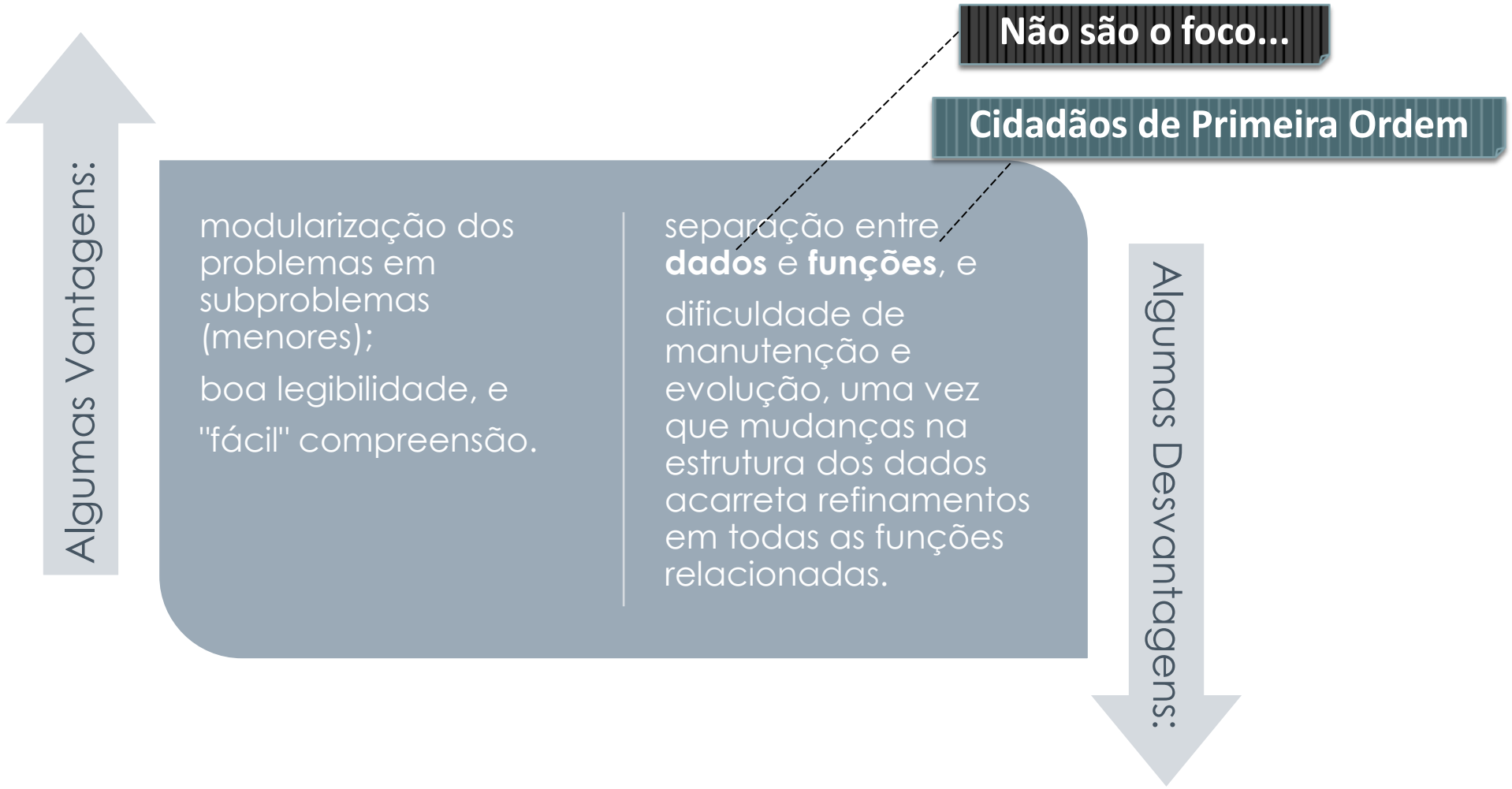
Paradigma de Programação Estrutural

Exemplos de Linguagens orientadas pelo Paradigma Estrutural:

- C;
- Basic;
- Pascal;
- Cobol, e
- Outras.

Acredito que sejam as mais conhecidas ...

Paradigma de Programação Estrutural



Paradigma de Programação OO

Paradigma de Programação OO

Programação baseada em entidades, com propriedades e comportamentos bem definidos ...

Abstração, encapsulamento, herança, polimorfismo e outras noções na elaboração de soluções para problemas reais ...

Paradigma de Programação OO

Por um lado, assim como no Paradigma Imperativo, o modelo computacional do Paradigma Orientado a Objetos baseia-se no conceito de "**estado**".

Cada objeto possui seu estado próprio, formado pela valoração das respectivas propriedades (ou **atributos**) durante a execução do programa.

Portanto, estritamente, este Paradigma pode ser considerado Imperativo.

Paradigma de Programação OO

A programação Orientada a Objetos é baseada na composição e interação de diversas unidades de software denominados objetos.

O funcionamento de um software orientado a objetos se dá através de relacionamentos e trocas de mensagens entre esses objetos.

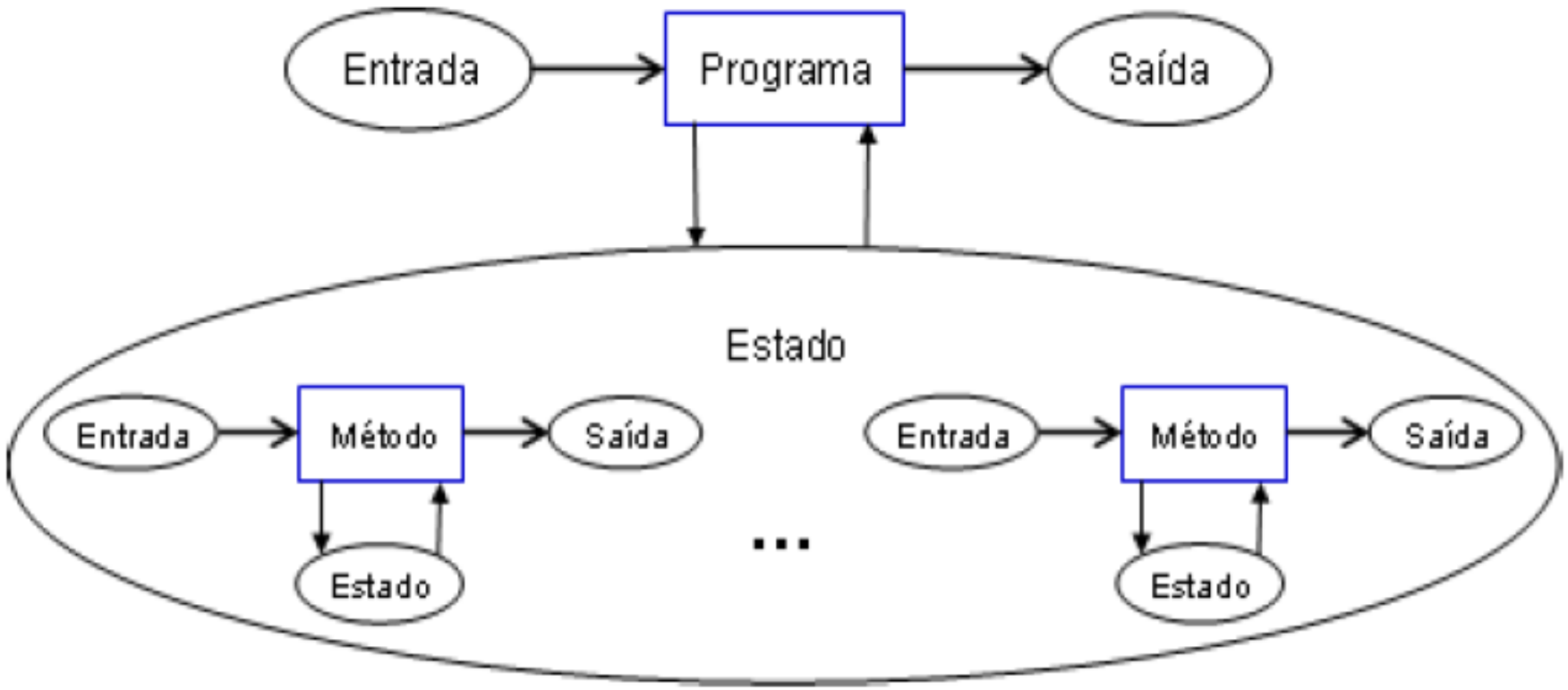
A Orientação a Objetos oferece uma série de mecanismos de estruturação (ex. classes, herança, polimorfismo, mecanismos de visibilidade, entre outros) que conduzem a uma forma particular de modelar as aplicações computacionais.

Paradigma de Programação OO

Em particular, o estado de uma aplicação, ao invés de monolítico como em um programa imperativo típico, é particionado entre os objetos ativos durante a execução.

Os mecanismos de visibilidade (essencialmente os métodos de acesso e modificadores dos atributos privados) permitem um controle de acesso e atualização de cada partição.

Paradigma de Programação OO



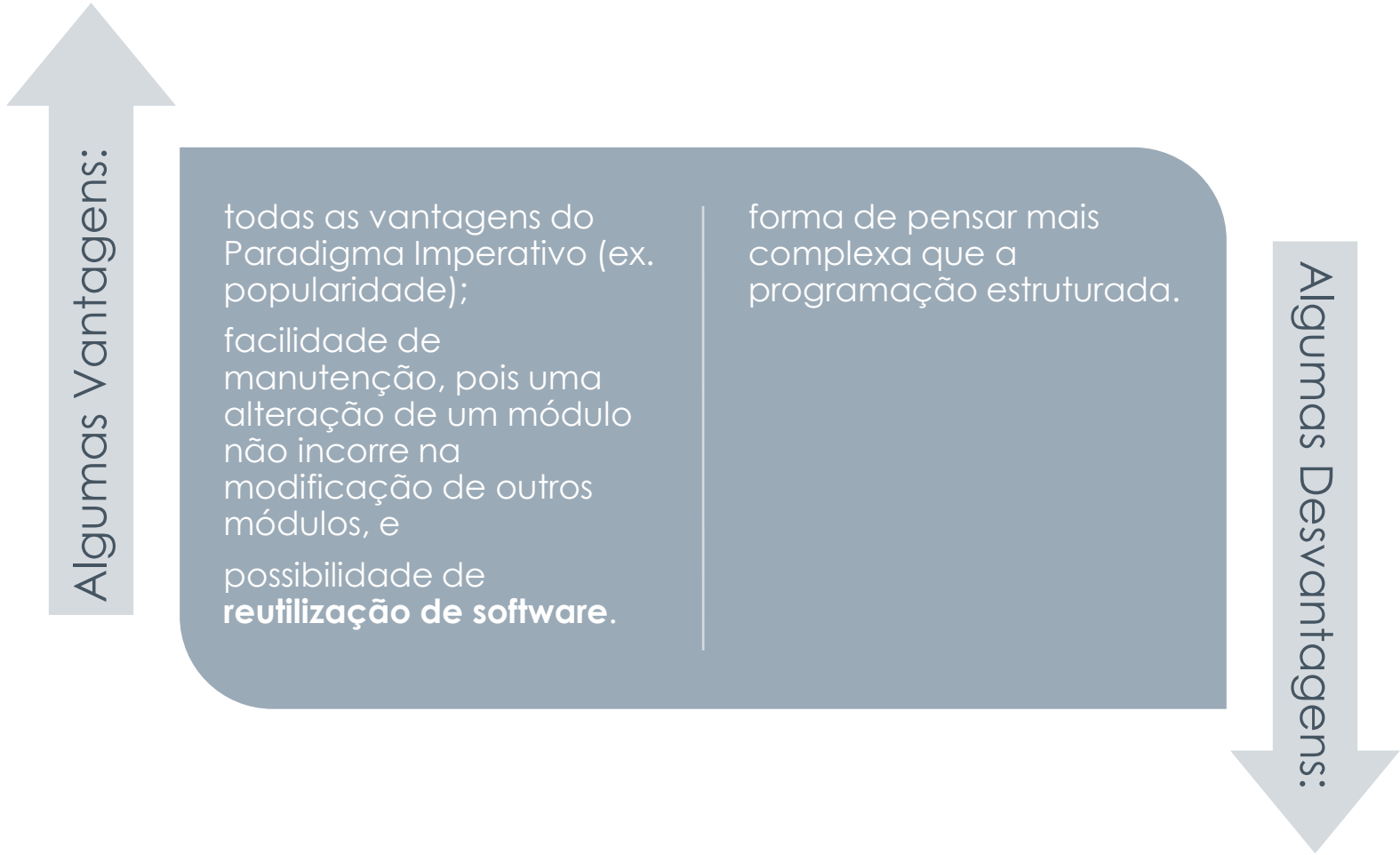
Paradigma de Programação OO

Exemplos de Linguagens orientadas pelo Paradigma OO:

- Smalltalk;
- Python;
- Ruby;
- C++;
- Object Pascal;
- Java;
- C#;
- Oberon;
- Ada;
- Eiffel;
- Simula, e
- Outras.

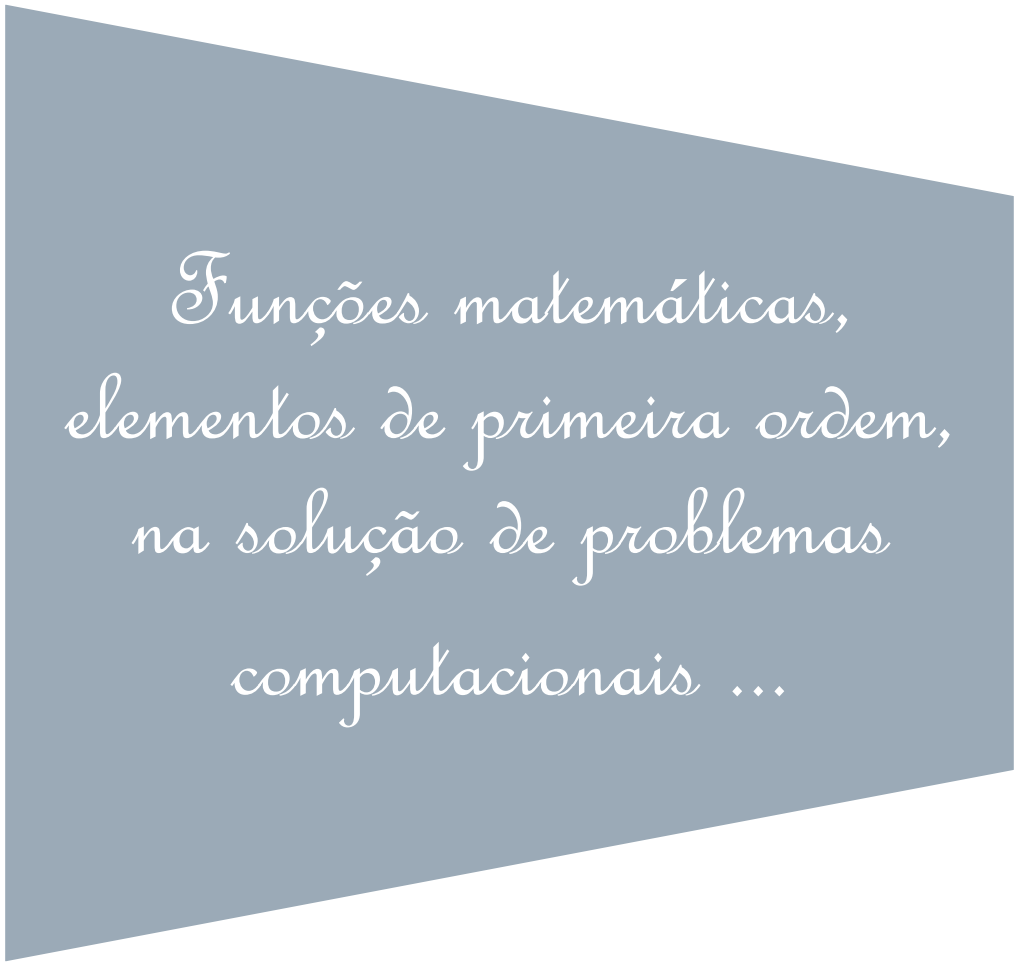
Acredito que sejam as mais conhecidas …

Paradigma de Programação OO



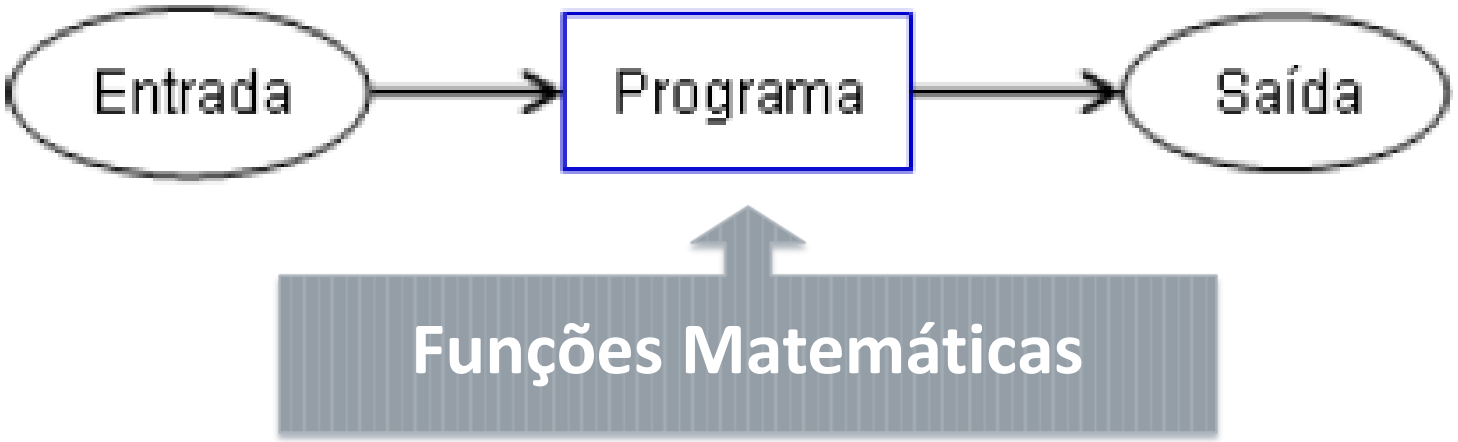
Paradigma de Programação Funcional

Paradigma de Programação Funcional



Paradigma de Programação Funcional

Paradigma de Programação Funcional



Paradigma de Programação Funcional

Exemplos de Linguagens orientadas pelo Paradigma Funcional:

- Lambda (não implementada para computadores);
- LISP;
- Scheme (tentativa de simplificar e melhorar o LISP);
- ML (criada em universidade);
- Miranda (também criada em universidade);
- Haskell, e
- Outras.

Acredito que sejam as mais conhecidas …

Paradigma de Programação Funcional

Algumas Vantagens:

devido ao processo automático de alocação de memória, os efeitos colaterais no cálculo da função não são significativos. Portanto, a linguagem assegura que o resultado da função será o mesmo para um dado conjunto de parâmetros, não importando onde, ou quando, seja avaliada;

amigável para lidar com execuções em paralelo, e

a recursividade na programação funcional pode assumir várias formas, o que torna a linguagem funcional mais poderosa que o uso de laços do Paradigma Imperativo.

carência de diversas construções, frequentemente consideradas essenciais em linguagens imperativas, como C. Por exemplo, não há alocação explícita nem de memória, nem de variáveis.

Algumas Desvantagens:

Paradigma de Programação Lógico

Paradigma de Programação Lógico

Associações de duas vias entre entrada e saída na solução de problemas computacionais.

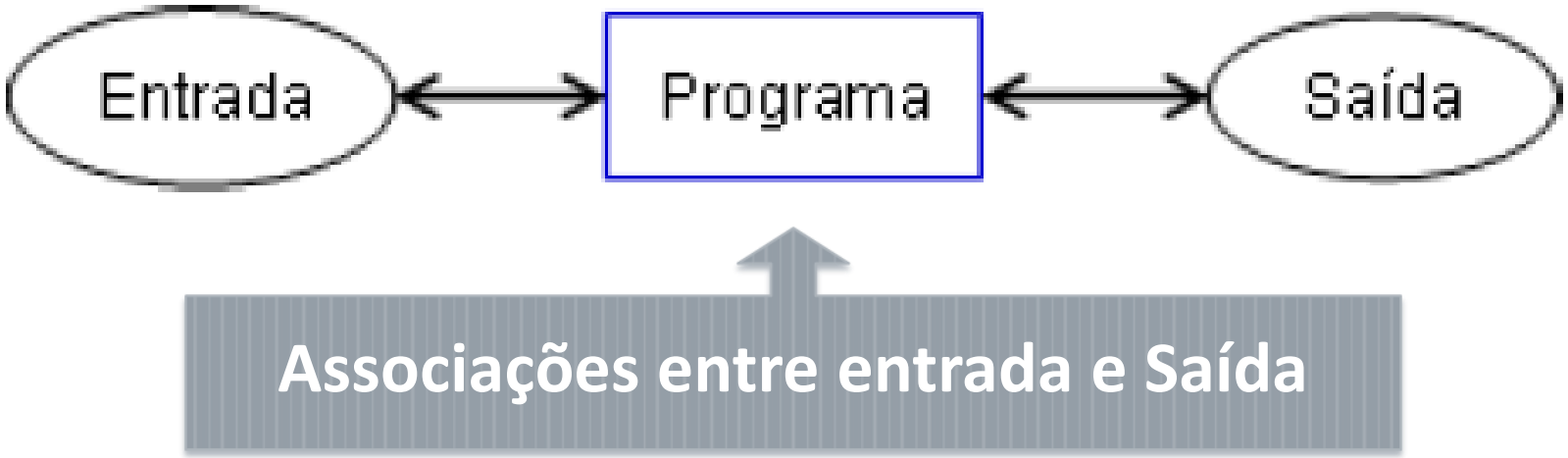
Paradigma de Programação Lógico

No Paradigma Lógico, há uma mudança mais substancial do modelo computacional.

Ao invés de uma função, a execução é baseada em relações entre entradas e saídas, permitindo execuções nos dois sentidos.

Por exemplo, um compilador, modelado como um programa lógico que relaciona código fonte e destino, pode ser utilizado tanto como um compilador quanto como um decompilador, gerando um programa fonte dado um programa destino [Bowen 1993].

Paradigma de Programação Lógico



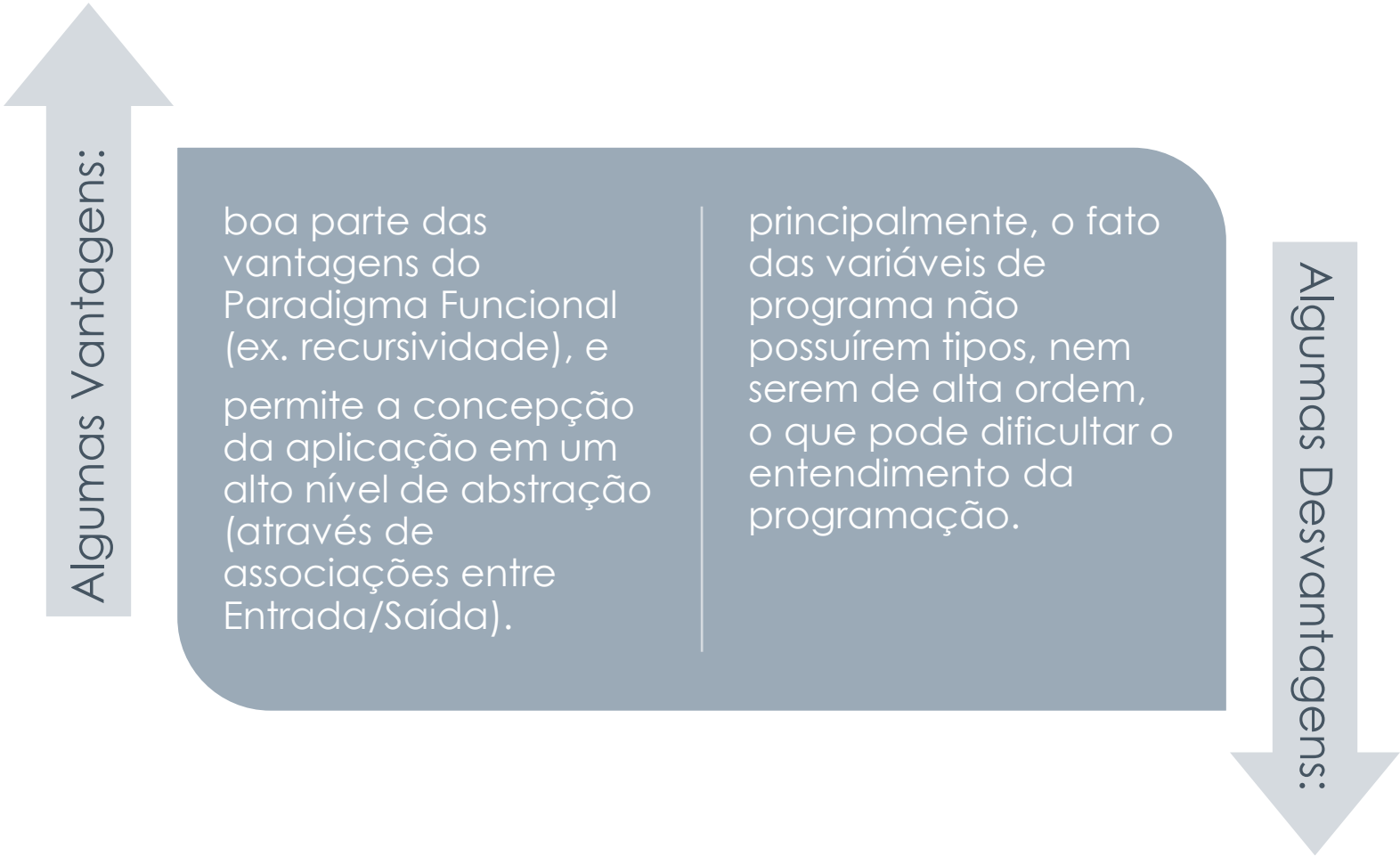
Paradigma de Programação Lógico

Exemplos de Linguagens orientadas pelo Paradigma Lógico:

- Popler;
- Conniver;
- QLISP;
- Planner;
- Prolog;
- Mercury;
- Oz;
- Frill, e
- Outras.

Acredito que sejam as mais conhecidas …

Paradigma de Programação Lógico



Paradigma de Programação Concorrente

Paradigma de Programação Concorrente

Abordagem multitarefas, com instruções executadas em paralelo na solução de problemas computacionais.

Foco no tempo de processamento e na otimização de recursos.

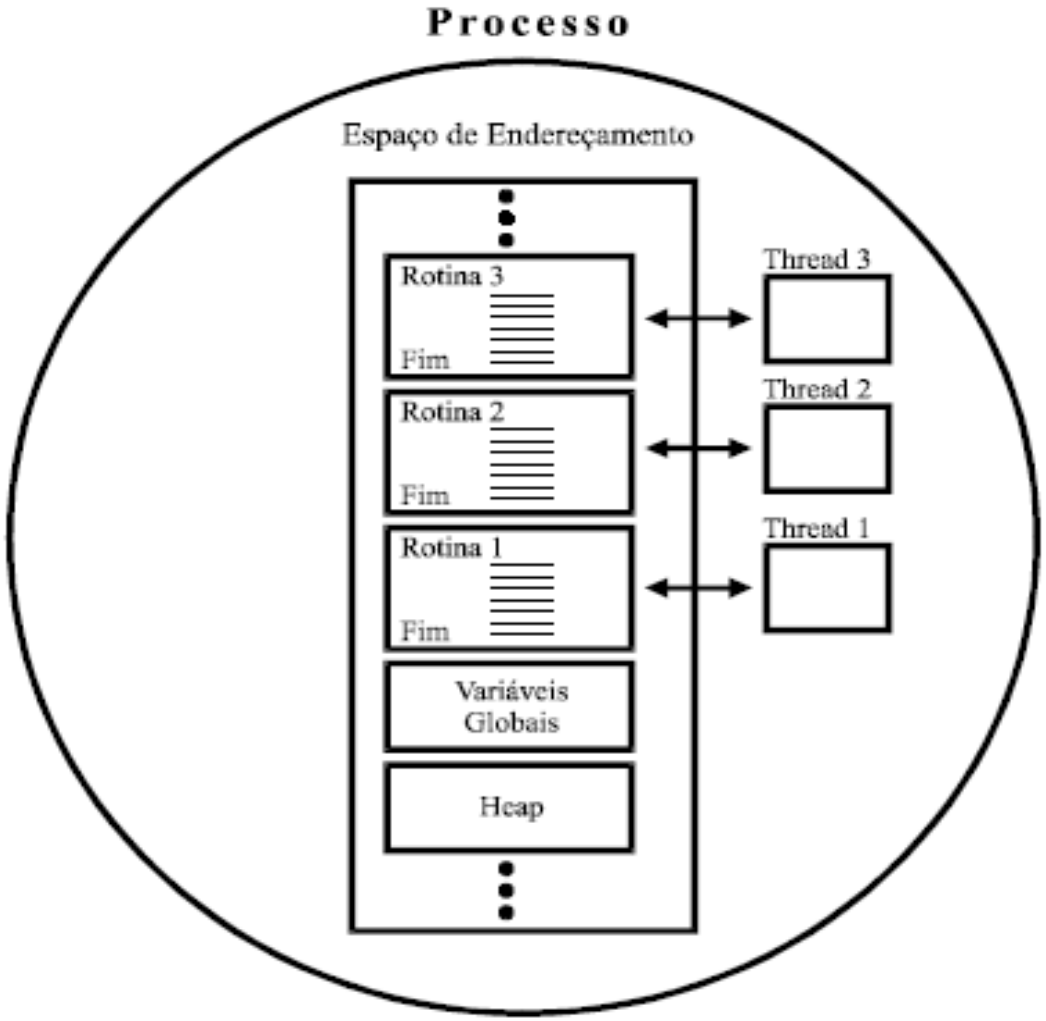
Paradigma de Programação Concorrente

O Paradigma Concorrente fundamenta-se em construções multitarefas, com suporte para sistemas distribuídos, e trocas de mensagens e recursos compartilhados.

Fortemente centrado na execução simultânea de várias tarefas computacionais interativas, que podem ser implementadas como programas separados ou como um conjunto de *threads* criadas por um único programa. Essas tarefas podem ser executadas por um único processador, vários processadores em um único equipamento ou processadores distribuídos em uma rede.

A interação e a comunicação correta entre as diferentes tarefas, além da coordenação do acesso concorrente aos recursos computacionais são as principais questões discutidas durante o desenvolvimento de sistemas concorrentes.

Paradigma de Programação Concorrente



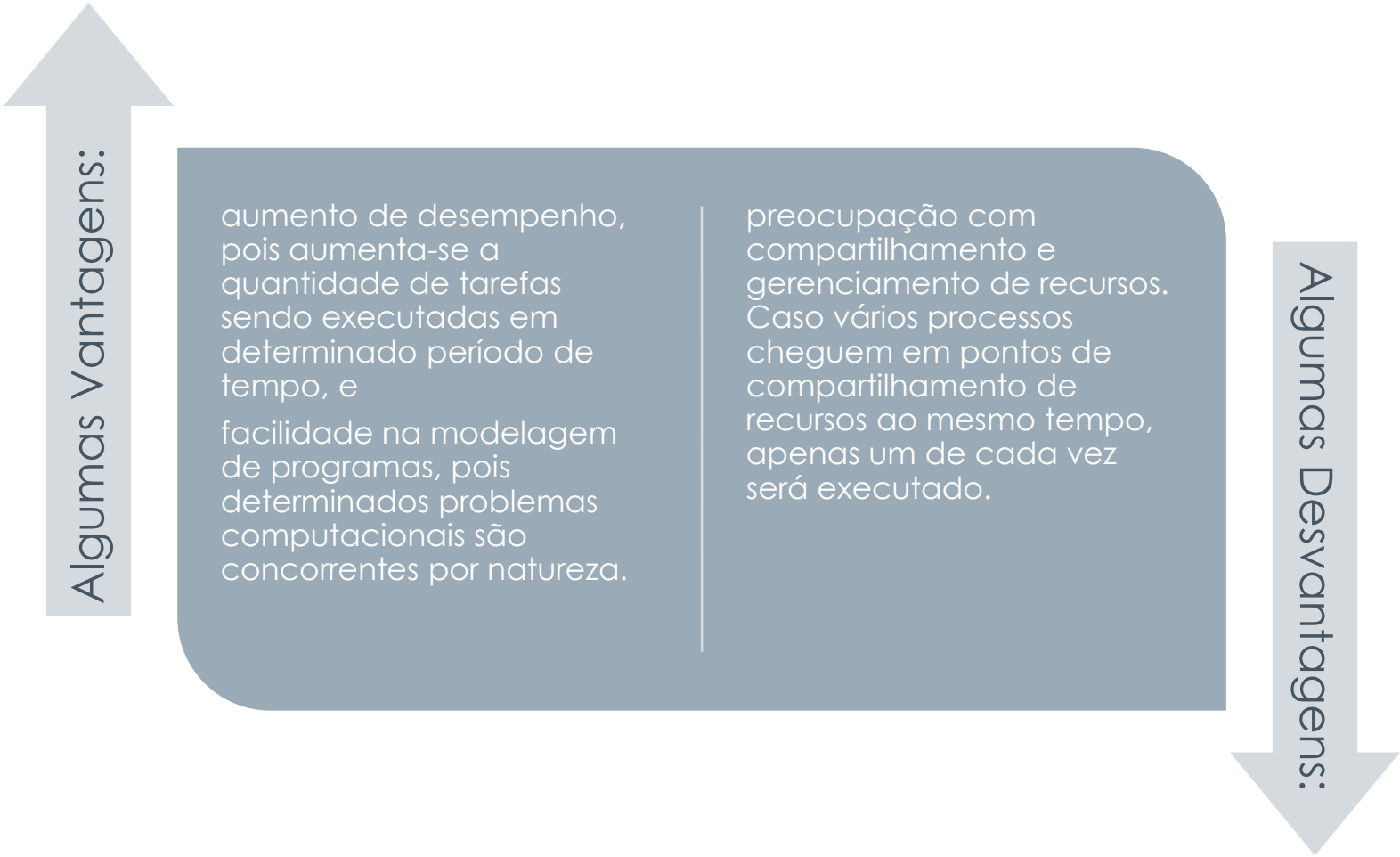
Paradigma de Programação Concorrente

Exemplos de Linguagens orientadas pelo Paradigma Lógico:

- Ada;
- Erlang (modelo de troca de mensagens);
- Limbo;
- Go;
- Occam;
- Scala;
- Java e C# (modelo de memória compartilhada, com o bloqueio sendo fornecido por monitores);
- C e C++ (suporte à concorrência através de bibliotecas), e
- Outras.

Acredito que sejam as mais conhecidas …

Paradigma de Programação Concorrente



Paradigma de Programação SMA

Paradigma de Programação SMA

Autonomia,
Flexibilidade,
Assincronismo,
Aguardem!!!! ;)

Considerações Finais

Considerações Finais

- A aula de hoje conferiu uma visão geral sobre os paradigmas de programação mais conhecidos.
- São eles: Imperativo, Estrutural, OO, Funcional, Lógico, Concorrente e Multiagentes.

Vamos exercitar?

Exercícios

Exercício_01

√Imaginem o problema:

Um cliente contrata a sua empresa de software para desenvolver um Sistema com as seguintes características:

-Primeiramente, o cliente deseja que o Sistema permita que vários dados sejam coletados para posterior processamento;

-Depois, o cliente deseja que esses dados sejam ordenados, visando facilitar o futuro acesso/busca de dados no Sistema, e

- Finalmente, o cliente deseja que a saída de dados seja formatada, melhorando a apresentação dos dados de saída, e facilitando para os funcionários que lidarão no dia-a-dia com esse Sistema.

Exercício_01

- √ Notem que, basicamente, o foco do Sistema está em:

  - coletar;
  - ordenar;
  - formatar, e
  - exibir.

- √ Mas, essas ações representam o que? Como podemos lidar com as mesmas no nível de programação?

Exercício_01

- √ Usando:

Funções

- √ Essas funções representam procedimentos bem definidos, que manipulam os dados de entrada de forma determinística...

- √ Reparem que o foco está nas ações e **não** nos dados ou mesmo nas demais entidades envolvidas no Sistema (ex. Funcionários).

Exercício_01

- √ Vamos pensar em como implementar o Sistema...

Alguém tem alguma ideia de como podemos coletar os dados de entrada usando como base comandos na linguagem C?

Exercício_01

√ Podemos usar:

scanf

Reparem que se trata de uma função...

Funções/Procedimentos são cidadãos de primeira ordem no Paradigma Imperativo!!!

√ Sintaxe:

```
scanf("expressão de controle", argumentos);
```

√ Por exemplo:

```
scanf("%d", &idadeCliente);
scanf("%s", &nomeCliente);
```

Exercício_01

- √ Vamos pensar em como continuar implementando o Sistema...

Alguém tem alguma ideia de como podemos ordenar os dados usando como base a linguagem C?

Exercício_01

- √ Podemos usar:

Novamente uma Função

- √ Entretanto, essa função será baseada em um algoritmo de ordenação, como por exemplo, o *Bubble Sort*.

Veremos mais adiante...

Aguardem...

Exercício_01

- √ Vamos pensar em como continuar implementando o Sistema...

Alguém tem alguma ideia de como podemos formatar/exibir os dados usando como base a linguagem C?

Exercício_01

√ Podemos usar:

printf

Novamente, reparem que se trata de uma função...

Funções/Procedimentos são cidadãos de primeira ordem no Paradigma Imperativo!!!

√ Sintaxe:

```
printf("expressão de controle", argumentos);
```

√ Por exemplo:

```
printf("A idade do cliente é %d", 20);
printf("O nome do cliente é %s", Maria);
```

Exercício_01

√ Podemos ainda caprichar na saída, usando:

Expressões de Controle na função printf

√ Por exemplo:

\n *new line* (pula linha)
\t *tab* (tabulação horizontal)
\b *backspace* (volta um caractere)
\f *form feed* (avanço de página)
**** (imprime a barra invertida)
e outras expressões de controle…

Exercício_02

- Imaginem o problema:

> *Um Sistema que pode ser decomposto em procedimentos, ou sejam, ações bem definidas, bem como pode ser modelado em estados explícitos, usando como base variáveis globais e locais ...*

- Qual seria o melhor Paradigma de Programação para lidar com esse problema?

Exercício_02

√ Resposta:

Paradigma Imperativo
ou Procedimental

Exercício_03

- Imaginem o problema:

> Um Sistema que preferencialmente seja modelado usando estruturas simples, baseadas nas noções de sequencia, decisão e iteração. Portanto, loops e ifs são adequados para modelagem desse Sistema ...

- Qual seria o melhor Paradigma de Programação para lidar com esse problema?

Exercício_03

√ Resposta:

Paradigma Estrutural

Exercício_04

- Imaginem o problema:

> *Um Sistema baseado em troca de mensagens, cujos elementos podem ser agrupados em entidades com características/propriedades bem como comportamentos bem definidos ...*

- Qual seria o melhor Paradigma de Programação para lidar com esse problema?

Exercício_04

√ Resposta:

Paradigma
Orientado a Objetos

Exercício_05

- Imaginem o problema:

> *Um Sistema, cujo foco está nas funções, as quais são tidas como elementos de primeira ordem. Essas funções podem ser usados, inclusive, como parâmetros para outras funções ...*

- Qual seria o melhor Paradigma de Programação para lidar com esse problema?

Exercício_05

√ Resposta:

Paradigma Funcional

Exercício_06

- Imaginem o problema:

> *Um Sistema especialista, declarativo, com fortes relações entre entrada e saída, cuja relação entre as mesmas pode ser executada nos dois sentido ...*

- Qual seria o melhor Paradigma de Programação para lidar com esse problema?

Exercício_06

√ Resposta:

Paradigma Lógico

Exercício_07

- Imaginem o problema:

> Um Sistema preferencialmente modelado com threads, cuja execução deve ser em paralelo, e o controle usando semáforos ...

- Qual seria o melhor Paradigma de Programação para lidar com esse problema?

Exercício_07

√ Resposta:

Paradigma Concorrente

Exercício_08

- Imaginem o problema:

> *Um Sistema distribuído, inteligente, autônomo, focado em entidades móveis que raciocinam e colaboram entre si ...*

- Qual seria o melhor Paradigma de Programação para lidar com esse problema?

Exercício_08

√ Resposta:

Paradigma Multi-Agentes

Referências

Referências

Bibliografia Básica

[EBRARY] Scott, M. L. **Programming Language Pragmatics**. eISBN: 9780080515168. 2ª. Edition. 915 pages. Editor: Morgan Kaufmann. Saint Louis, MO, USA. November 2005.

[BIBLIOTECA FGA – 6 Unidades] Tucker, Allen B.; Noonan, Robert. **Linguagens de Programação: Princípios e Paradigmas**. 2ª. Edição. São Paulo: McGraw-Hill, c2009. xxiii, 599 p. ISBN 9788577260447 OU Tucker, Allen B.; Noonan, Robert. **Programming Languages: Principles and Paradigms**. 2ª. Edition. Boston: McGraw-Hill, c2007. xxiii, 600 p. ISBN 9780072866094.

[BIBLIOTECA FGA – 15 Unidades] Cormen, Thomas H. **Algoritmos: Teoria e Prática**. Rio de Janeiro: Elsevier, c2002. 916 p. ISBN 9788535209266.

Referências

Bibliografia Complementar

[OPEN ACCESS] Paradigma Orientado a Convenção sobre Configuração (Híbrido: Estruturado, OO e Funcional) **Grails Platform**: https://grails.org/documentation.html e https://grails.org/wiki/version/Documentation%20Portuguese/9 (Julho 2017).
* Extra, caso queiram conhecer mais um paradigma.

[OPEN ACCESS] Paradigma Funcional. The Haskell Programming Language: http://book.realworldhaskell.org/read/ e http://learnyouahaskell.com/chapters (principais) e http://www.haskell.org/haskellwiki/Haskell e https://www.haskell.org/platform/ e http://www.haskell.org/haskellwiki/GHC/GHCi (Julho 2017.

[OPEN ACCESS] Paradigma Lógico. LPA WinProlog: http://www.lpa.co.uk/index.htm e http://www.lpa.co.uk/dow_doc.htm (principais LPA WinProlog) OU SWI Prolog: http://www.swi-prolog.org/pldoc/doc_for?object=manual (principal SWI Prlog) e http://www.swi-prolog.org/ OU GNUProlog/gProlog: http://www.gprolog.org/#manual (principal gProlog) e http://www.gprolog.org/ (Julho 2017).

Referências

Bibliografia Complementar

[OPEN ACCESS] Paradigma Multiagentes (Híbrido: Estruturado, OO e Comportamental) **Jade Documentation**. Multiagent Systems: http://jade.tilab.com/dl.php?file=JADE-doc-4.3.2.zip (principal) e http://jade.tilab.com/

(Julho 2017).

[OPEN ACCESS] **Introduction to Computer Science Programming Paradigms**. Stanford Graduate School of Education (Stanford University). Stanford, CA. October 2014.
https://see.stanford.edu/Course/CS107
http://videolectures.net/stanfordcs107s08_programming_paradigms/
https://www.udemy.com/cs-107-programming-paradigms/

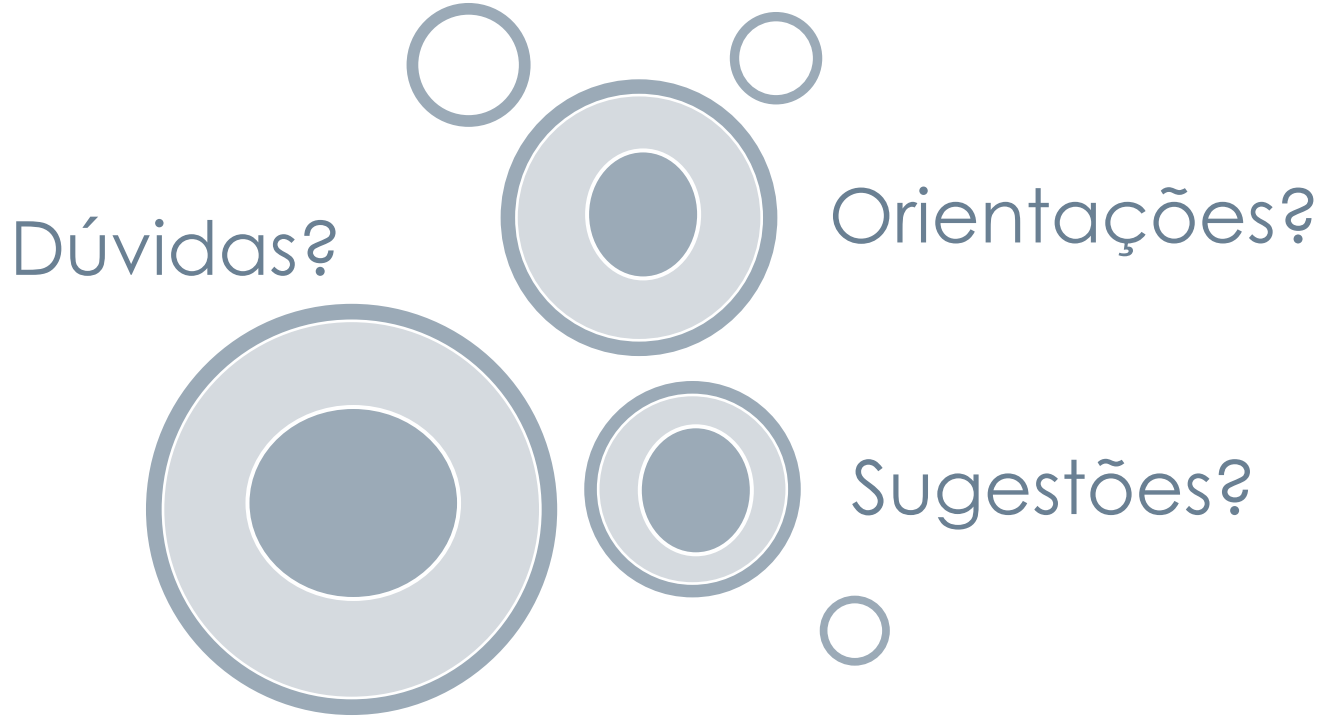
(Julho 2017)

Referências

Biblioteca Virtual Ebrary:

- http://site.ebrary.com/lib/univbrasilia/home.action

Vocês já acessaram?
O acervo de livros virtuais
é muito bom!



FIM

mileneserrano@unb.br ou mileneserrano@gmail.com