

Paradigmas de Linguagens de Programação

Sumário

- Referência Bibliografia da Aula
- Objetivo Geral da Aula
- Repositório *Git*
- O que é Paradigma?
- O que é Paradigma de Linguagem de Programação?
- O que é Linguagem de Programação?



Sumário

- O que é Sintaxe?
- O que é Semântica?
- Sintaxe/Semântica de Linguagens de Programação
- Qual a Linguagem de Programação mais popular hoje?
- Por que estudar conceitos de Linguagens de Programação?



Sumário

- Mas por que tantas Linguagens de Programação?
- Critérios de avaliação de Linguagens de Programação
- Evolução das principais Linguagens de Programação
- Linguagens de Alto Nível vs. Baixo Nível
- Paradigmas de Linguagens de Programação

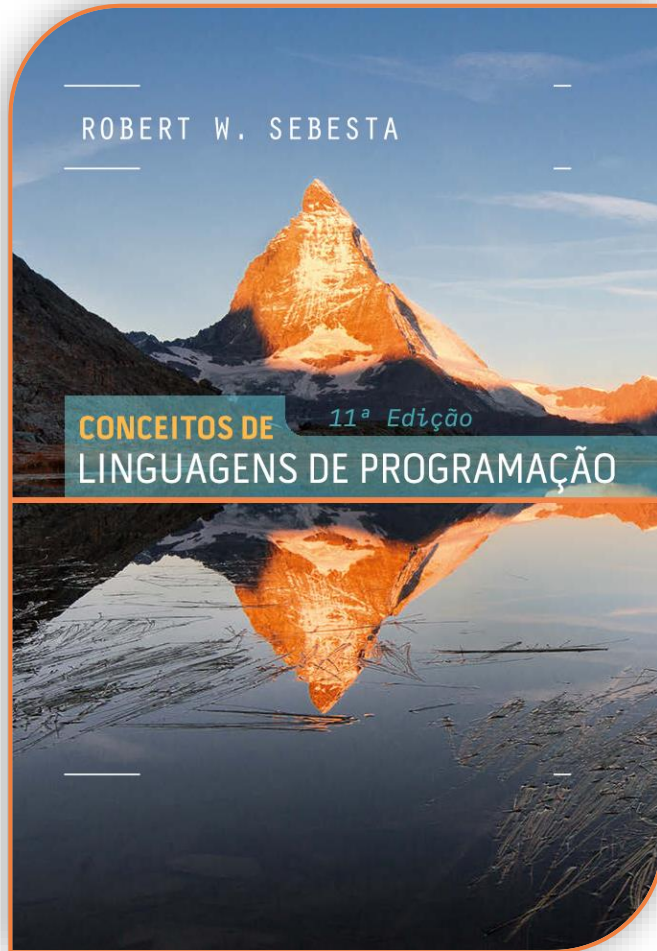



Sumário

- Paradigma de Programação Imperativa
 - Programação Estruturada/Procedural
 - Programação Orientada a Objetos
- Paradigma de Programação Declarativa
- Paradigma de Programação Concorrente e Paralela
- Paradigma de Programação Baseada em Eventos




Referência Bibliografia da Aula



-  SEBESTA, R. W.
Conceitos de Linguagens de Programação. 11^a ed.
Porto Alegre: Bookman,
2018.




Objetivo Geral da Aula

-  Os estudantes devem conhecer os diferentes **Paradigmas de Linguagens de Programação**, comparando suas características, sendo capazes de identificar as diferenças e aplicabilidade dos **Paradigmas**.



Repositório *Git*

- github.com/whoisraibolt/UNIFESO-CCOMP-EDP
-  Repositório com o conteúdo da disciplina **Estrutura de Dados e Paradigmas** do curso **Ciência da Computação** da **UNIFESO** — Centro Universitário Serra dos Órgãos.



O que é Paradigma?

- Algo que serve de exemplo ou modelo; padrão.
- **[GRAMÁTICA]** Modelo de conjugação ou de declinação de uma palavra.
- **[LINGUÍSTICA]** Conjunto de termos comutáveis entre si, em uma mesma posição, numa estrutura.



O que é Paradigma?

- **[FILOSOFIA]** Segundo o filósofo americano Thomas Kuhn (1922-1996), qualquer campo de investigação e de experiência que está na origem da evolução científica.
- **PARADIGMA.** In: MICHAELIS, Dicionário Brasileiro da Língua Portuguesa. © 2022 Editora Melhoramentos Ltda. Disponível em: <https://michaelis.uol.com.br/>. Acesso em: 19/02/2022.



O que é Paradigma de Linguagem de Programação?

- **Modelo** ou **padrão** de **programação** sustentado por **linguagens** que adotam **características semelhantes**.



O que é Linguagem de Programação?

- Linguagem formal:
 - Formada por regras **sintáticas** e/ou **semânticas**.
- Permite a comunicação entre **homem-máquina**:
 - **Código-fonte**: instruções, ordens e ações consecutivas.



O que é Sintaxe?

- Coleção de regras.
- **[GRAMÁTICA]** Parte da gramática que trata da disposição das palavras na frase, da relação entre essas palavras, bem como das combinações e das relações lógicas das frases no enunciado.



O que é Sintaxe?

- **[LINGUÍSTICA]** Cada um dos elementos da estrutura linguística que determina as relações entre os componentes da oração.
- **SINTAXE.** In: MICHAELIS, Dicionário Brasileiro da Língua Portuguesa. © 2022 Editora Melhoramentos Ltda. Disponível em: <https://michaelis.uol.com.br/>. Acesso em: 28/02/2022.



O que é Semântica?

- Estudo do significado dos vocábulos.
- **[LINGUÍSTICA]** Ramo da linguística que estuda a significação das palavras e suas mudanças de sentido ao longo do tempo, bem como a representação do sentido dos enunciados.



O que é Semântica?

- **[LINGUÍSTICA]** O significado dos vocábulos, por oposição à sua forma.
- **[FILOSOFIA]** Ciência que estuda a evolução do significado das palavras, signos e símbolos que estão a serviço da comunicação; semiologia.



O que é Semântica?

- SEMÂNTICA. In: MICHAELIS, Dicionário Brasileiro da Língua Portuguesa. © 2022 Editora Melhoramentos Ltda. Disponível em: <https://michaelis.uol.com.br/>. Acesso em: 28/02/2022.



Sintaxe/Semântica de Linguagens de Programação

Sintaxe:

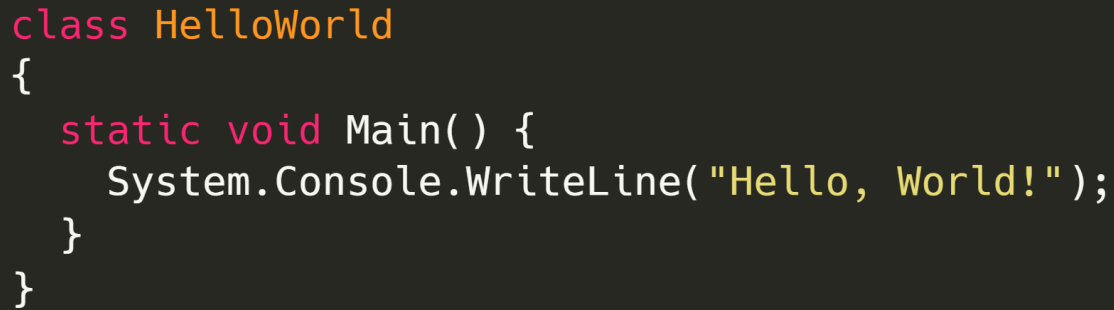
- Forma de escrever a estrutura do código-fonte (palavras reservadas, comandos, etc.).

Semântica:

- Interpretação do código-fonte, ou seja, descreve o significado das expressões e instruções.



Sintaxe/Semântica de Linguagens de Programação



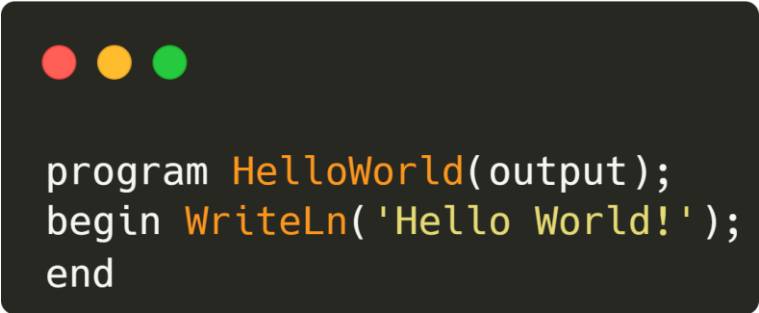
```
class HelloWorld
{
    static void Main() {
        System.Console.WriteLine("Hello, World!");
    }
}
```

C#



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

Haskell



```
program HelloWorld(output);
begin WriteLn('Hello World!');
end
```

Object Pascal



Qual a Linguagem de Programação mais popular hoje?

Python

C




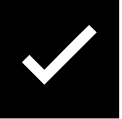

Java



* tiobe.com/tiobe-index

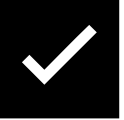
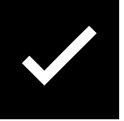
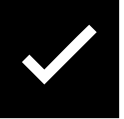


Por que estudar conceitos de Linguagens de Programação?

-  Aumento da capacidade de expressar ideias.
-  Embasamento para escolher linguagens adequadas.
-  Aumento da habilidade para aprender novas linguagens.

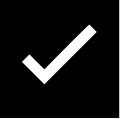
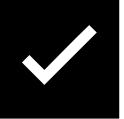
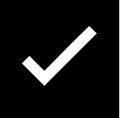


Por que estudar conceitos de Linguagens de Programação?

-  Melhor entendimento da importância da implemen-tação.
-  Melhor uso de linguagens já conhecidas.
-  Avanço geral da computação.



Mas por que tantas Linguagens de Programação?

-  Propósitos diferentes.
-  Avanços tecnológicos.
-  Interesses comerciais ou científicos.

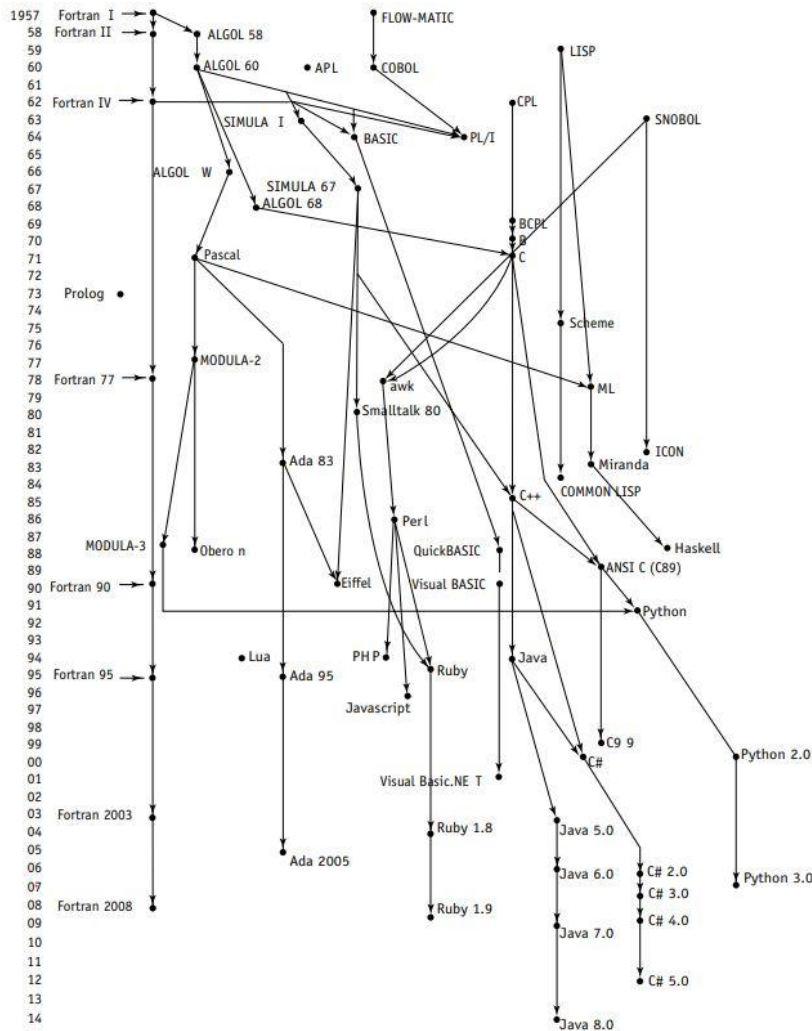



Critérios de avaliação de Linguagens de Programação

- Simplicidade e Legibilidade.
- Confiabilidade.
- Suporte da comunidade.
- Suporte à abstração.
- Ortogonalidade.
- Implementação eficiente.



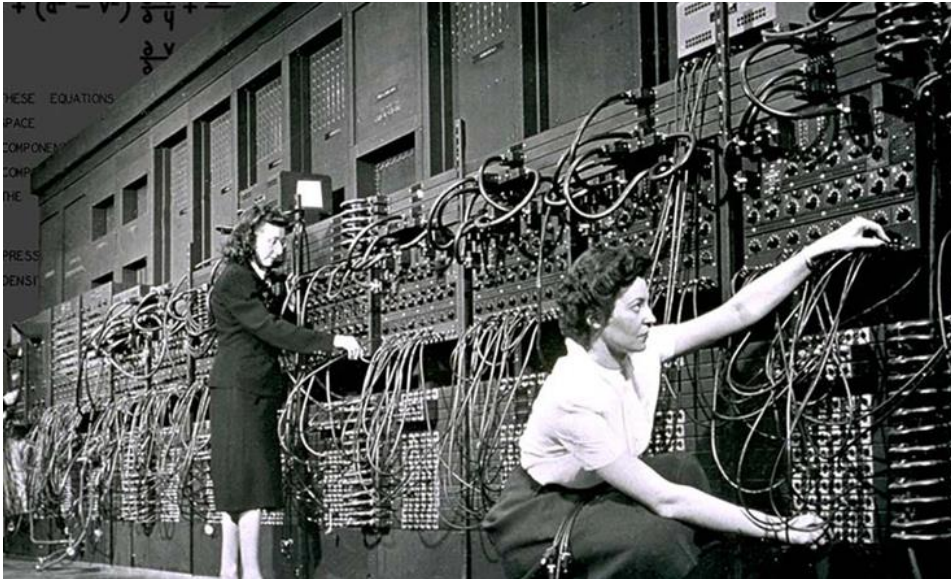
Evolução das principais Linguagens de Programação



-  Antigamente, toda programação era realizada em **Linguagem de Máquina**, evoluindo até as **Linguagens de Alto Nível**, como temos hoje.



Evolução das principais Linguagens de Programação




-  Primeiro computador eletrônico da história - **ENIAC**.





Evolução das principais Linguagens de Programação



-  **Cartão perfurado** contendo código legível por máquina para um computador da época.



Linguagens de Alto Nível vs. Baixo Nível

-  As **Linguagens de Alto Nível** são as que se aproximam da **Linguagem Humana**.
-  As **Linguagens de Baixo Nível** são as que se aproximam da **Linguagem de Máquina**.



Linguagens de Alto Nível vs. Baixo Nível

linguagem de alto nível		linguagem de baixo nível	
PRÓS	CONTRAS	PRÓS	CONTRAS
<ul style="list-style-type: none">■ facilidade de aprendizagem;■ produtividade;■ permite a escrita de mais códigos em menos tempo;■ fácil entendimento;■ manutenção simplificada;■ baixo custo operacional.	<ul style="list-style-type: none">■ desempenho de um programa pode ser prejudicado, pois ele exige maior tempo de processamento;■ ocupa mais memória quando comparadas a uma linguagem de baixo nível.	<ul style="list-style-type: none">■ tempo de processamento mais rápido do que o de uma de alto nível;■ melhor aproveitamento da arquitetura do computador.	<ul style="list-style-type: none">■ maior tempo para compreender e dominar a sintaxe;■ necessidade de conhecer profundamente o hardware da máquina, o que exige mais investimento em estudo e treinamento.



Linguagens de Alto Nível vs. Baixo Nível

```
print('Hello World!')
```

Python

```
section .text align=0

global _start

mensagem db 'Hello world', 0x0a

len equ $ - mensagem

_start:
    mov eax, 4 ;SYS_write
    mov ebx, 1 ;Número do file descriptor (1=stdout)
    mov ecx, mensagem ;Ponteiro para a string.
    mov edx, len ;tamanho da mensagem
    int 0x80

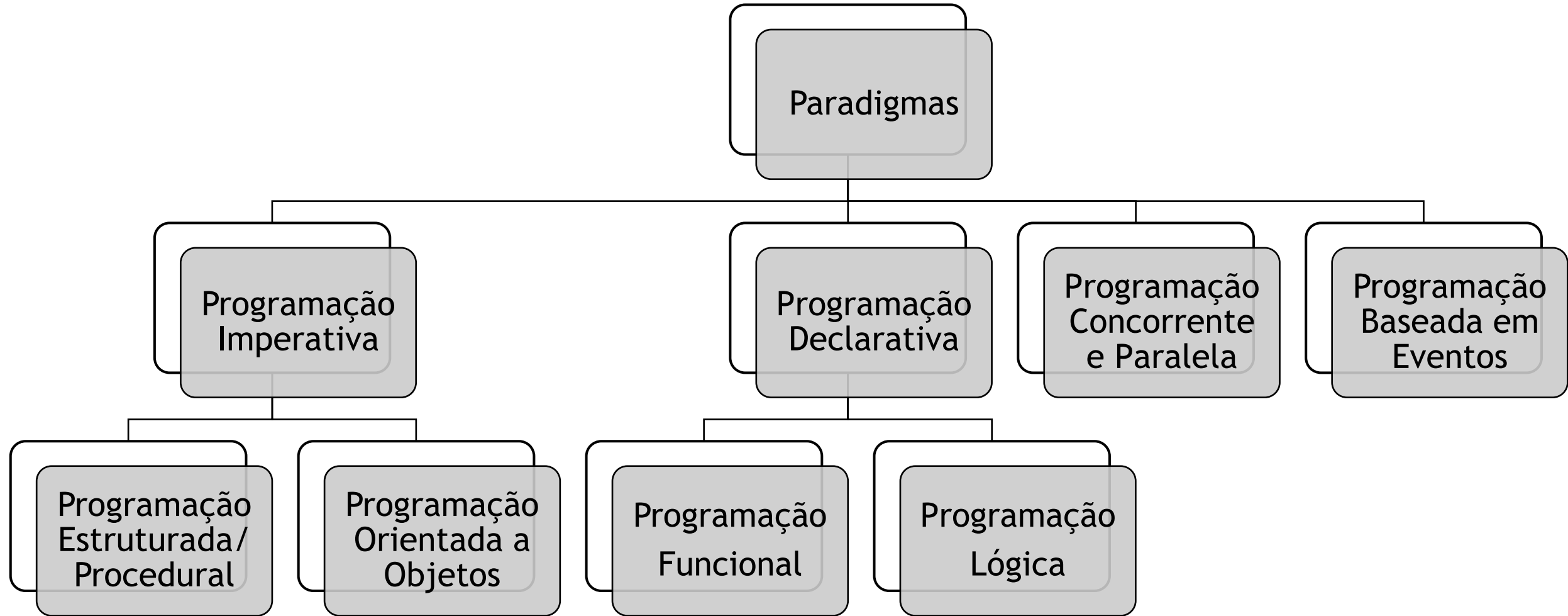
    mov eax, 1
    int 0x80
```

Assembly

* <https://tinyurl.com/299busbu>



Paradigmas de Linguagens de Programação



Paradigma de Programação Imperativa



Paradigma de Programação Imperativa

- Paradigma mais **antigo**:
 - **1940** até os dias de hoje...
- Baseado na arquitetura de **Von Neumann**.
- Exemplos:
 - **Ada, C, COBOL, Fortran**.



Paradigma de Programação Imperativa

- Através do trabalho de **Alan Turing**, John **Von Neumann** percebeu que tanto um **programa** quanto seus **dados** poderiam residir na **memória** de um computador.
- Qual seria a **vantagem** disso?

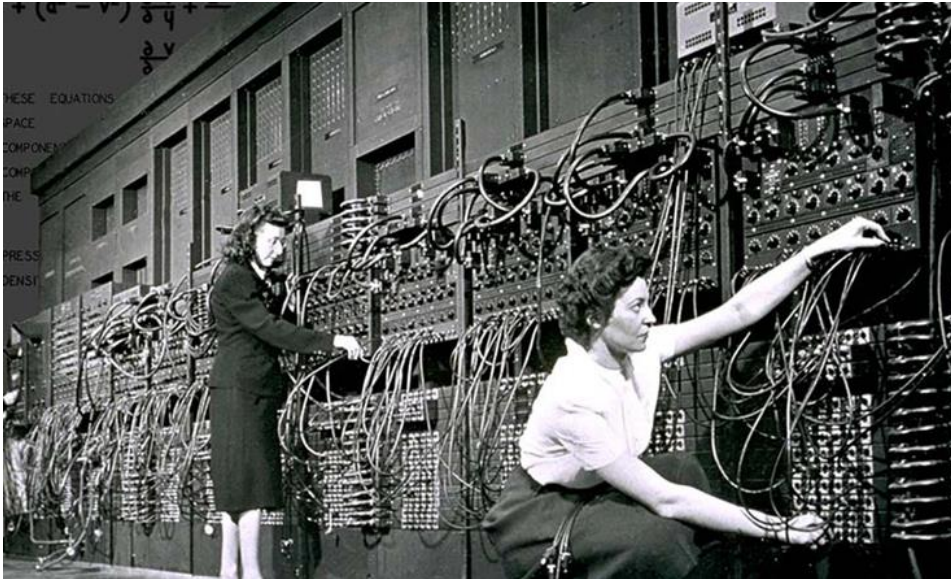


Paradigma de Programação Imperativa

- Através do trabalho de **Alan Turing**, John **Von Neumann** percebeu que tanto um **programa** quanto seus **dados** poderiam residir na **memória** de um computador.
- Qual seria a **vantagem** disso?
 - Potencializar o **uso** dos computadores, tornando-os mais **fáceis** e **versáteis**.



Paradigma de Programação Imperativa




-  Primeiro computador eletrônico da história - **ENIAC**.



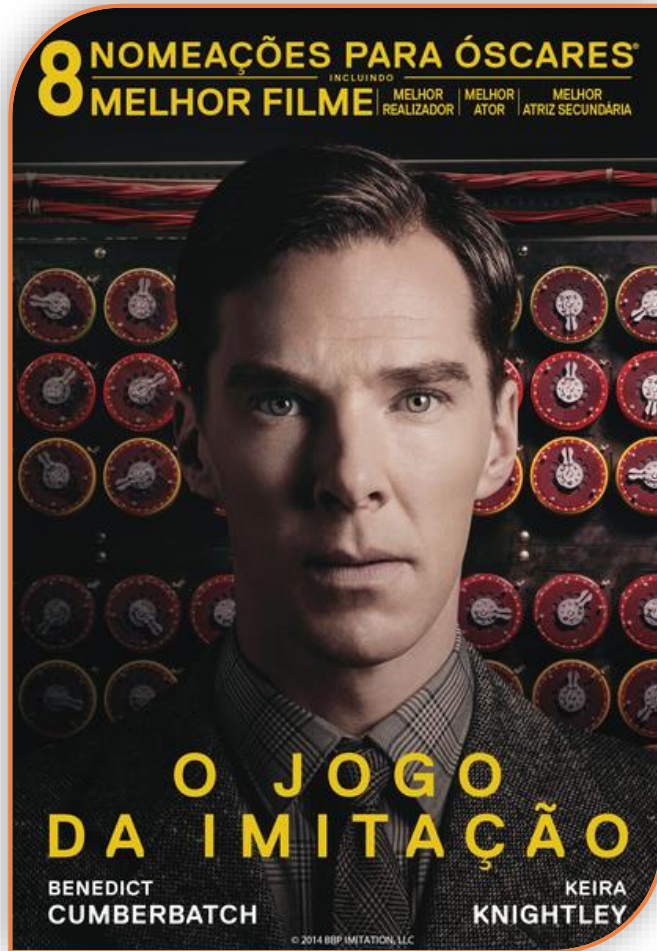
Paradigma de Programação Imperativa



-  **Cartão perfurado** contendo código legível por máquina para um computador da época.



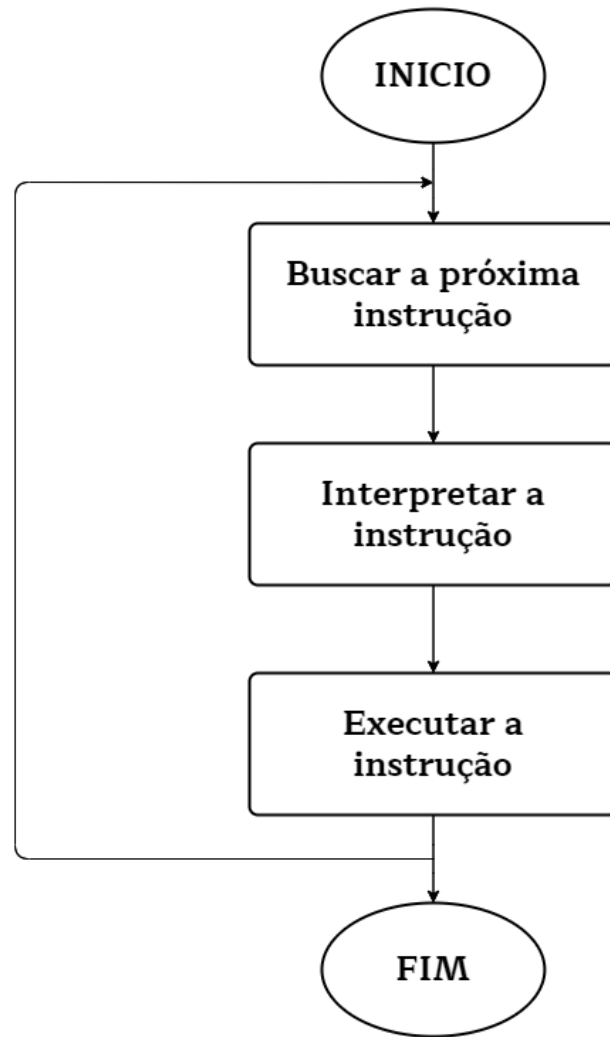
Paradigma de Programação Imperativa




-  **O Jogo da Imitação** (2014).



Paradigma de Programação Imperativa



-  Baseado na arquitetura de **Von Neumann**.



Paradigma de Programação Imperativa

- **Recursos:**

- Declaração de variáveis;
- Expressões;
- Comandos condicionais;
- Comandos de repetições.



Programação Estruturada/Procedural



Programação Estruturada/Procedural

- Programação centrada no conceito de um estado (**modelado por variáveis**) e ações (**comandos**) que manipulam o estado.
- Paradigma também denominado de **procedural**, por incluir **sub-rotinas** ou procedimentos como mecanismo de estruturação.



Programação Estruturada/Procedural



```
# Função que retorna uma série numérica no
# intervalo enviado como argumento
def count(num_start, num_stop, num_step):
    # Gerar lista
    # A convenção de nomenclatura de sublinhado à direita
    # única (list_) é usada para evitar conflitos
    # com palavras-chave do Python
    list_ = list(range(num_start, num_stop, num_step))

    # Printar lista
    print(list_)

# Gerar uma sequência numérica entre 1 e 30, com step igual a 1
count(num_start = 1, num_step = 1, num_stop = 31)

# Gerar uma sequência numérica entre 0 e 50, com step igual a 5
count(num_start = 0, num_step = 5, num_stop = 51)
```



Programação Orientada a Objetos



Programação Orientada a Objetos

- Mapear **objetos** do **mundo real** para modelos computacionais.
- Representando suas **características** e **comportamentos**.
- Exemplos:
 - **Java**, **C++**, **C#**.

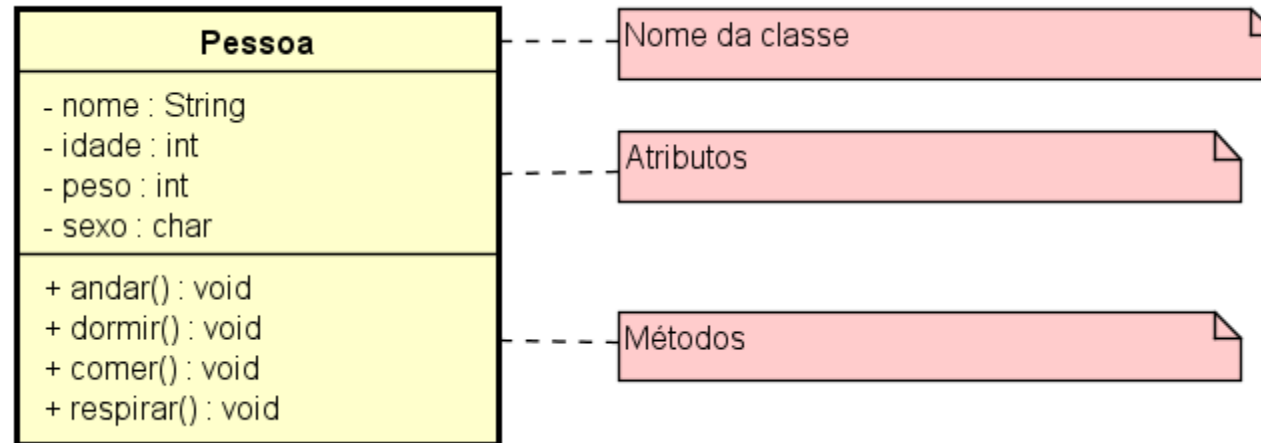


Programação Orientada a Objetos

- Todos os **objetos** têm determinados **estados** e comportamentos. Esses **estados** são descritos pelas **classes** como **atributos**.
- Já a forma como eles se comportam (sua **funcionalidade**) é definida por meio de **métodos**, que nada mais são do que **funções**.



Programação Orientada a Objetos



*<https://tinyurl.com/48hn9px6>



Programação Orientada a Objetos

- Para que uma **Linguagem de Programação** seja do tipo de **Paradigma Orientada a Objetos**, deve-se implementar seus princípios básicos:
 - **Classe, Objeto, Herança, Polimorfismo e Encapsulamento.**

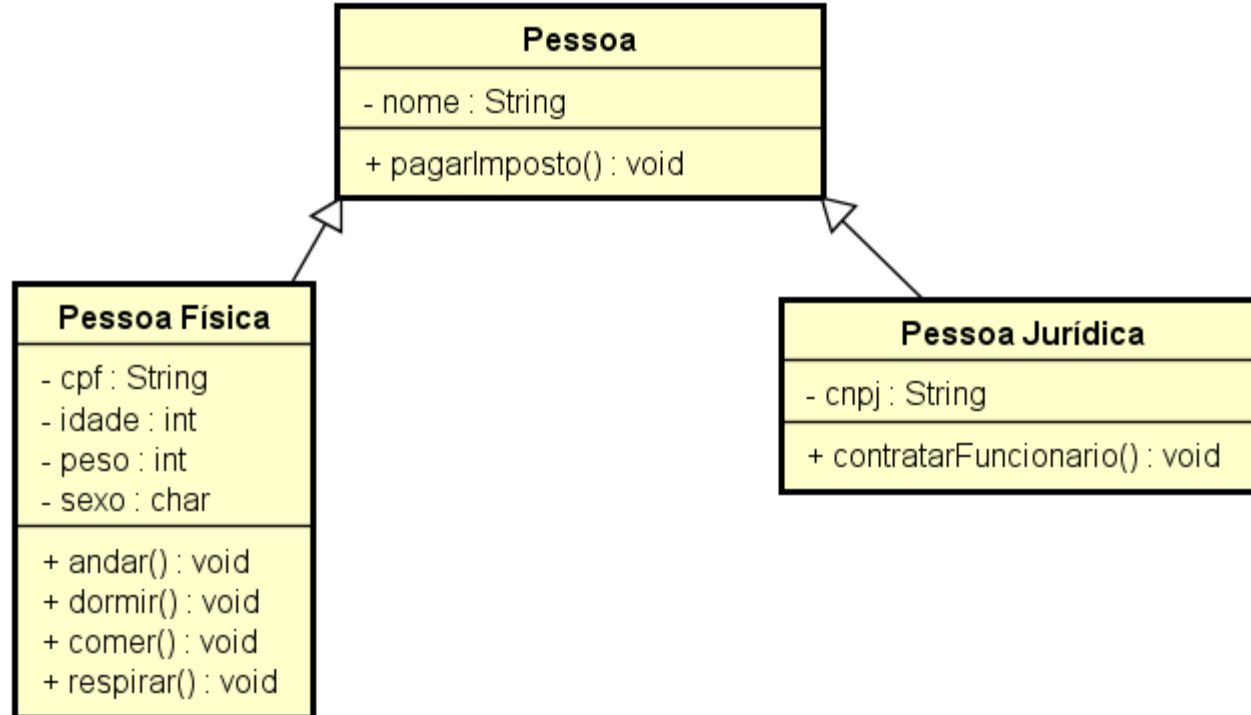


Programação Orientada a Objetos

Classe	Objeto	Herança	Polimorfismo	Encapsulamento
<ul style="list-style-type: none">• Representação de algo abstrato; substantivo, que possui atributos e métodos.	<ul style="list-style-type: none">• Uma instância da Classe.	<ul style="list-style-type: none">• Mecanismo que permite que uma Classe (subclasse) herde os atributos e métodos de uma superclasse (classe mãe).	<ul style="list-style-type: none">• Mecanismo que permite a instanciação de duas ou mais Classes a partir de uma superclasse (classe mãe), entretanto, com assinaturas distintas.	<ul style="list-style-type: none">• Mecanismo que blinda o acesso direto aos atributos de um Objeto, permitindo apenas o acesso a métodos que alteram o estado dos atributos.



Programação Orientada a Objetos



* <https://tinyurl.com/48hn9px6>



Programação Orientada a Objetos



```
# Variável do tipo inteiro  
numero = 14
```

```
# Variável do tipo String  
nome = 'Estrutura de Dados'
```

```
Class Produto:  
    pass
```

```
# Variável do tipo Produto  
xbox = Produto()
```



```
# Variável do tipo inteiro  
numero = 14
```

```
print(type(numero))
```

```
# Variável do tipo String  
nome = 'Estrutura de Dados'
```

```
print(type(nome))
```

```
Class Produto:  
    pass
```

```
# Variável do tipo Produto  
xbox = Produto()
```

```
print(type(xbox))
```



Programação Orientada a Objetos

```
class Produto:
    def __init__(self, nome, descricao, valor):
        self.nome = nome
        self.descricao = descricao
        self.valor = valor

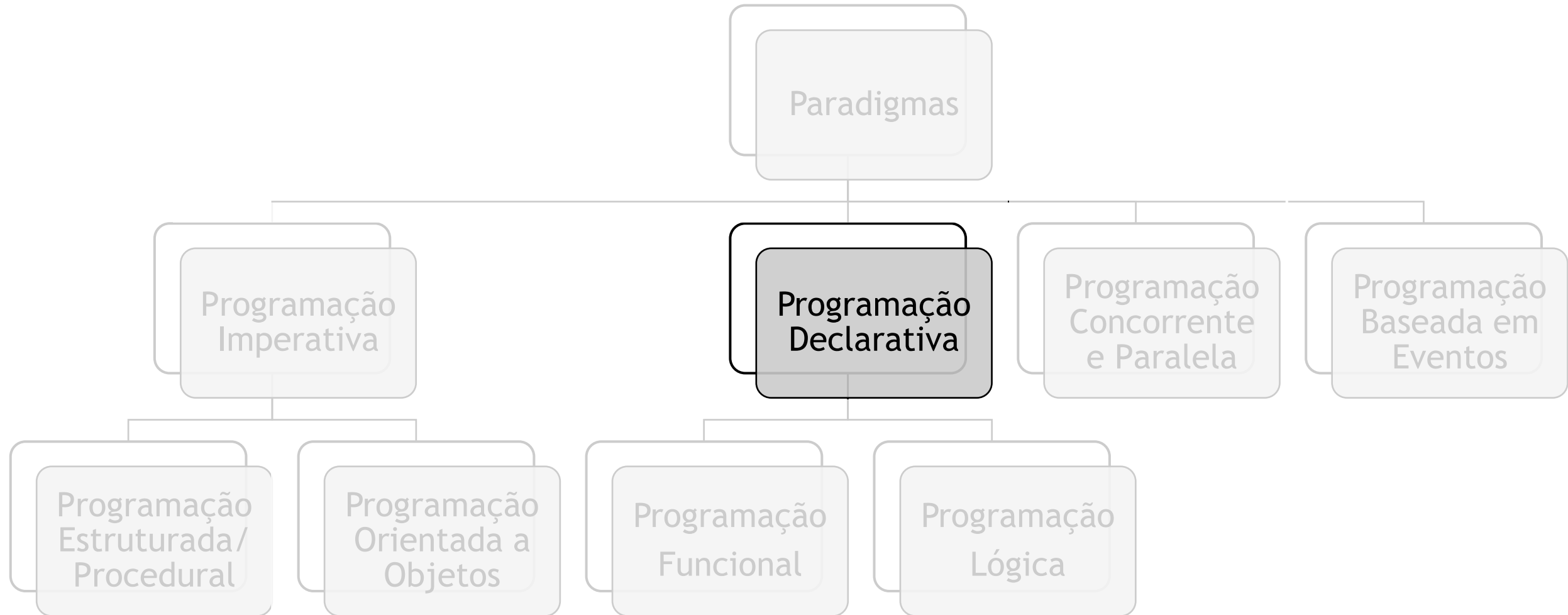
# Variável do tipo Produto
xbox = Produto('Series X', 'N/A', 5000)

print(xbox.nome)

print(type(xbox))
```



Paradigma de Programação Declarativa



Paradigma de Programação Declarativa

- **Não há** preocupação na maneira ou método de execução de uma determinada rotina.
- Foca em “**o quê**” deve ser resolvido e não “**como**” isso deve ser feito.

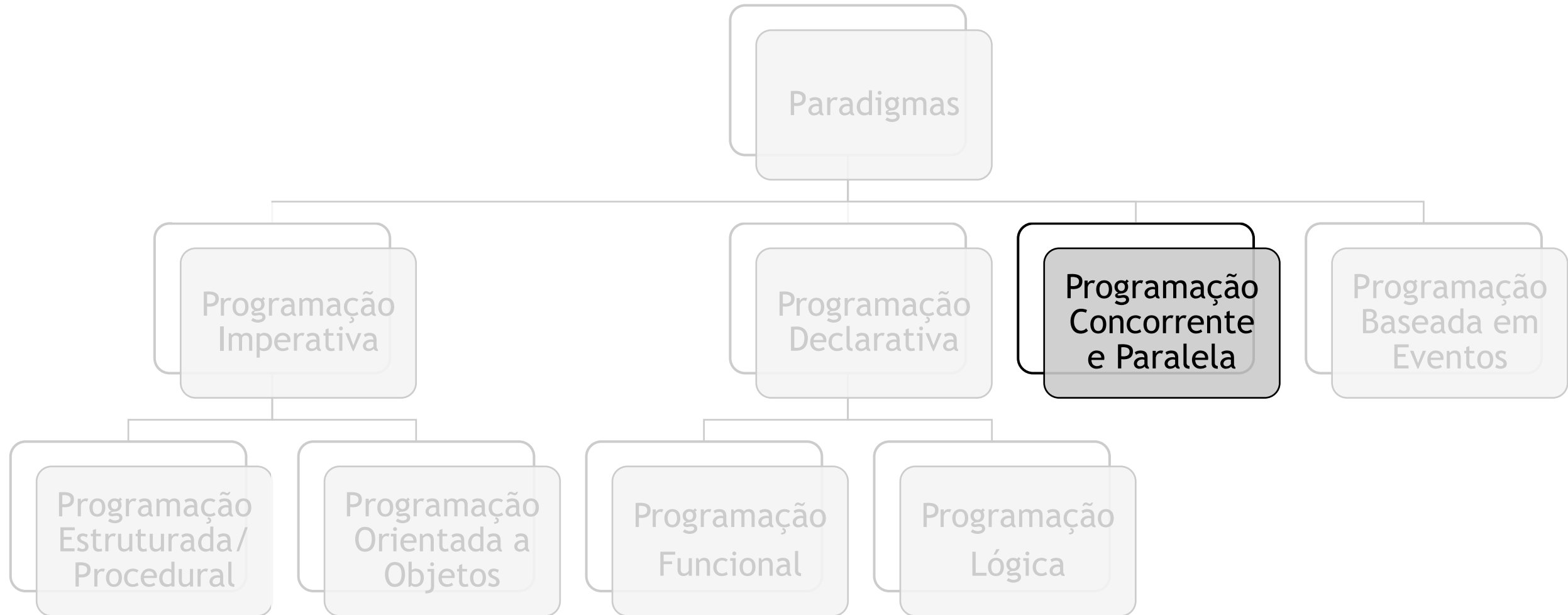


Paradigma de Programação Declarativa

- Exemplo:
 - **SQL**: **SELECT**.
- Exemplos:
 - **LISP**, **Haskell**, **COBOL**, **Fortran**, **Angular**
(Framework JavaScript).



Paradigma de Programação Concorrente e Paralela



Paradigma de Programação Concorrente e Paralela

- **O que é Concorrente?**

- Que concorre; competidor.
- Que tem existência simultânea com outras coisas.



Paradigma de Programação Concorrente e Paralela

- O que é Concorrente?
 - **[ECONOMIA]** Que disputa ou faz concorrência.
 - **[INFORMÁTICA]** Diz-se de ações ou conjuntos que são quase simultâneos.



Paradigma de Programação Concorrente e Paralela

- O que é Concorrente?

- CONCORRENTE. In: MICHAELIS, Dicionário Brasileiro da Língua Portuguesa. © 2022 Editora Melhoramentos Ltda. Disponível em: <https://michaelis.uol.com.br/>. Acesso em: 28/02/2022.



Paradigma de Programação Concorrente e Paralela

- **O que é Paralelo?**

- Que existe ou ocorre ao mesmo tempo de outra coisa; simultâneo.
- Que é parecido ou semelhante; análogo.



Paradigma de Programação Concorrente e Paralela

- O que é Paralelo?

- **[LINGUAGEM FIGURADA]** Diz-se de algo que existe ou ocorre simultaneamente com outra realidade.
- **[INFORMÁTICA]** Diz-se de sistema de computador no qual dois ou mais processadores operam simultaneamente sobre um ou mais itens dados.



Paradigma de Programação Concorrente e Paralela

- **O que é Paralelo?**

- PARALELO. In: MICHAELIS, Dicionário Brasileiro da Língua Portuguesa. © 2022 Editora Melhoramentos Ltda. Disponível em: <https://michaelis.uol.com.br/>. Acesso em: 28/02/2022.



Paradigma de Programação Concorrente e Paralela

- **Programação Concorrente:**

- Execução **simultânea** de **diversas tarefas** que podem ser implementadas como **programas separados** ou como um **único programa**.
- Ou seja, as **tarefas disputam** ou **fazem concorrência** da utilização de **recursos computacionais** do sistema.



Paradigma de Programação Concorrente e Paralela

- **Programação Concorrente :**

- **Uso:**

- Serviços multitarefa, suporte para sistemas distribuídos, troca de mensagens e recursos compartilhados.

- **Exemplos:**

- **C, C++, C#, Java, Erlang.**



Paradigma de Programação Concorrente e Paralela

- Programação Paralela:

- Execução **REALMENTE** simultânea de diversas tarefas que podem ser implementadas como programas separados ou como um único programa.
- Melhora consideravelmente o desempenho.



Paradigma de Programação Concorrente e Paralela

- **Programação Paralela:**

- **Uso:**

- Baseada na arquitetura de memória usada, assim como comunicação e/ou recursos computacionais compartilhados.

- **Exemplos:**

- **C, C++, C#, Java, Python** (Suporte a Programação Paralela).



Paradigma de Programação Baseada em Eventos



Paradigma de Programação Baseada em Eventos

- Utilizado em Linguagens de Programação que possuem **recursos gráfico**, tais quais: **Formulários, Jogos**, entre outros.



Paradigma de Programação Baseada em Eventos

- A **execução** do programa **se dá** a medida que **determinados eventos** são **disparados** pelo **usuário**.
- Exemplos:
 - **C++**, **C#**, **Java**, **Python**.



Paradigmas de Linguagens de Programação

Até a próxima!