Linguagem de Programação I

Aula 1 – Linguagens de Programação: conceitos e classificações

Prof Ana Carolina Sokolonski

Bacharelado em Sistemas de Informação
Instituto Federal da Bahia – Campus Feira de Santana

2026

Agenda

Definição

de Linguagem de Programação

O que é uma Linguagem de Programação?

▪ Definição formal

Uma linguagem de programação é um **sistema de comunicação estruturado**, composto por símbolos, palavras-chave, regras semânticas e sintáticas que permitem o entendimento entre um programador e uma máquina. [BlipBlog 2020]

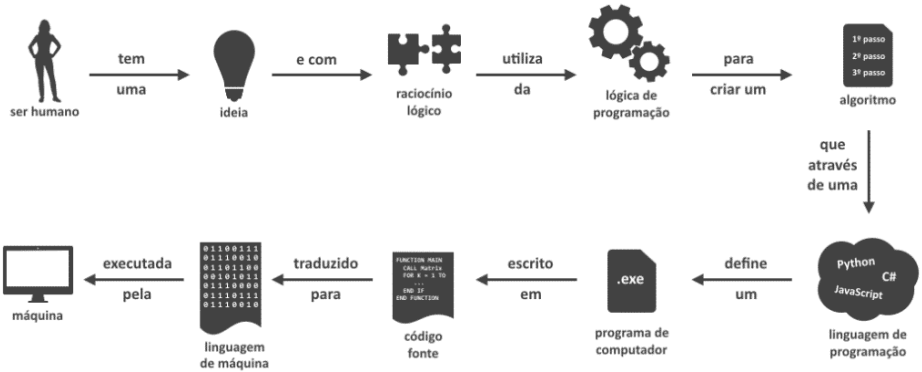
▪ Em outras palavras

É um **padrão de comunicação** que dá instruções a um computador por meio de palavras e símbolos.

▪ Regras

Define **sintaxe** (forma) e **semântica** (significado), traduzidas em programas executáveis.

Fluxo de um Programa



Classificação

das Linguagens de Programação

Categorias principais

Linguagens de programação são normalmente divididas em duas grandes categorias:

Imperativas

Descrevem **como** realizar a tarefa.

- Procedurais
- Orientadas a objetos
- Estruturadas
- Computação distribuída

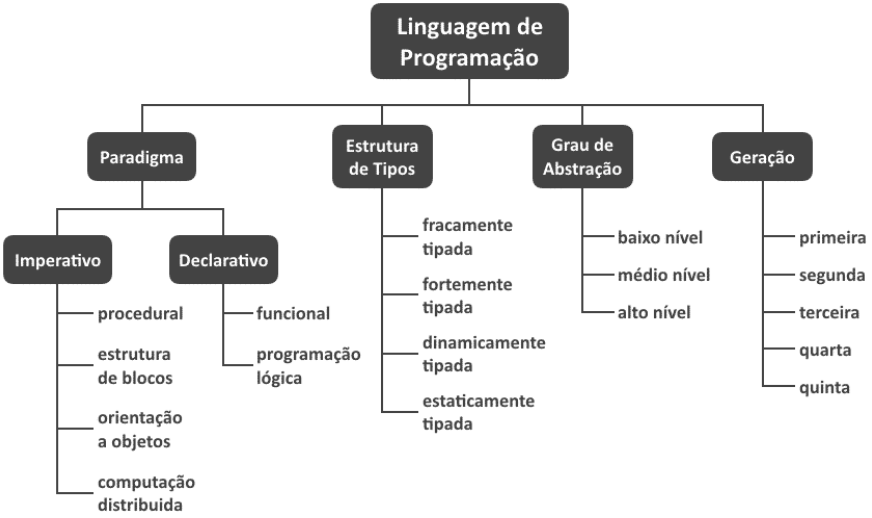
Declarativas

Descrevem **o que** se deseja obter.

- Lógicas (ex: Prolog)
- Funcionais (ex: LISP, Haskell)

Existem outras formas de classificação além dessa divisão clássica.

Mapa de Classificação



Níveis

de uma Linguagem de Programação

Alto e Baixo Nível

A "altura" de uma linguagem está relacionada com a sua **proximidade do hardware**. Um computador executa apenas código de máquina (0s e 1s).

Baixo Nível

Instruções próximas ao código de máquina. Foco no **controle do hardware**.

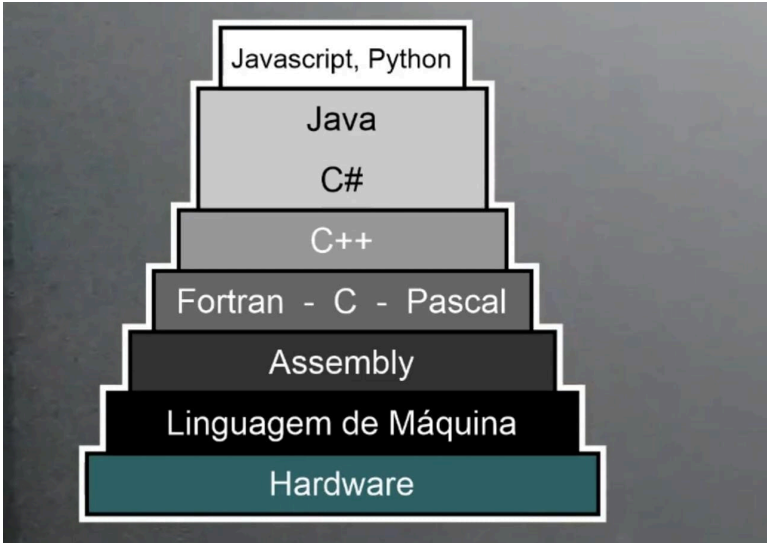
- Ótimo desempenho computacional
- Exemplos: **Assembly**, **C**

Alto Nível

Foco no **usuário**: facilidade de escrita, legibilidade e abstração. A conversão para código de máquina é mais custosa.

- Exemplos: **Java**, **JavaScript**, **Python**

Escala de Níveis



Métodos de Implementação

das Linguagens de Programação

O que é um implementador?

Um **implementador** é um software que converte código escrito em linguagem de programação em um formato executável — um programa que gera novos programas para computadores.

Compilação

Interpretação

Híbrido

Compilação

Como funciona

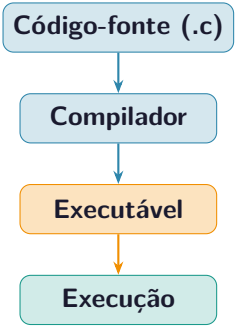
O código-fonte é traduzido integralmente para **código de máquina** antes da execução. Essa tradução é feita pelo **compilador**.

Vantagem

Execução muito rápida após a compilação.

Desvantagem

Necessita recompilar para cada plataforma-alvo.



Exemplos: **C, C++**

Interpretação

▪ Como funciona

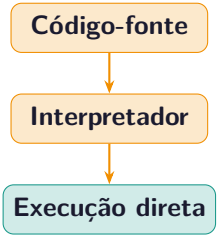
O código é lido e executado linha a linha por um **interpretador**, sem etapa de tradução prévia. O interpretador age como uma máquina virtual.

▪ Vantagem

Facilita a **depuração**: erros são rastreados até a linha exata.

▪ Desvantagem

Tempo de execução de **10 a 100 vezes** maior que os compilados. [Sebesta 2010]



Exemplos: **PHP, JavaScript**

Método Híbrido

O meio-termo

Traduz o programa para uma **linguagem intermediária** (bytecode), mais fácil de interpretar do que o código original. É mais rápido que a interpretação pura.

Exemplo: Java e a JVM

O código Java é compilado para **bytecode**, interpretado pela **JVM** (Java Virtual Machine). Isso garante portabilidade: "escreva uma vez, execute em qualquer lugar".

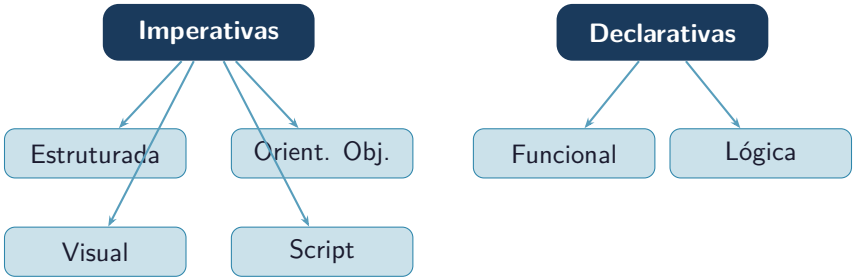
Just-in-Time (JIT)

Compila trechos do bytecode para código de máquina *durante* a execução, quando esses métodos são chamados. Usado em **Java** e em todas as linguagens **.NET** (C#, VB.NET).

Classificação

quanto ao Paradigma

Paradigmas: visão geral



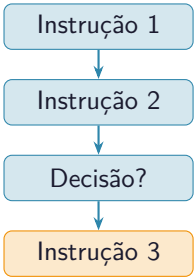
Modelo orientado a processos

O programa é escrito em função *do que acontece* — uma série linear de passos que atuam sobre os dados.

Esse é o **paradigma estruturado**: comandos sequenciais, condicionais e laços de repetição.

Limitação

À medida que os programas crescem, os custos de **manutenção** superam os de desenvolvimento.



▪ Organização em torno dos dados

A POO organiza o programa em torno de **objetos** e de interfaces bem definidas para acessá-los. Surgiu para gerenciar a crescente complexidade dos algoritmos.

▪ Encapsulamento

Esconde os detalhes internos, expõe apenas a interface.

▪ Herança

Permite reutilizar e especializar comportamentos.

▪ Polimorfismo

Diferentes objetos respondem ao mesmo comando de maneiras distintas.

Abstração: a essência da OO

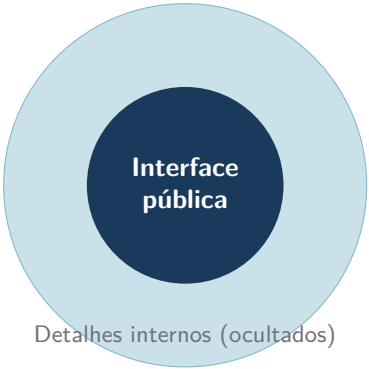
▪ Analogia do carro

Ninguém pensa no carro como um conjunto de peças individuais — todos o veem como um objeto com um comportamento definido.

Para andar de ré, basta colocar na marcha ré: basta **conhecer a interface**, sem entender o motor.

▪ Joe Armstrong (Coders at Work)

"O problema com as linguagens OO é que elas trazem consigo tudo o que carregam. Você queria uma banana, mas recebeu um gorila segurando a banana e toda a floresta."



Arquitetura de von Neumann

Projetadas para a arquitetura de von Neumann: **dados e programas compartilham a mesma memória**; a CPU é separada e recebe/envia dados continuamente. Quase todos os computadores desde os anos 1940 seguem esse modelo.

Recursos centrais

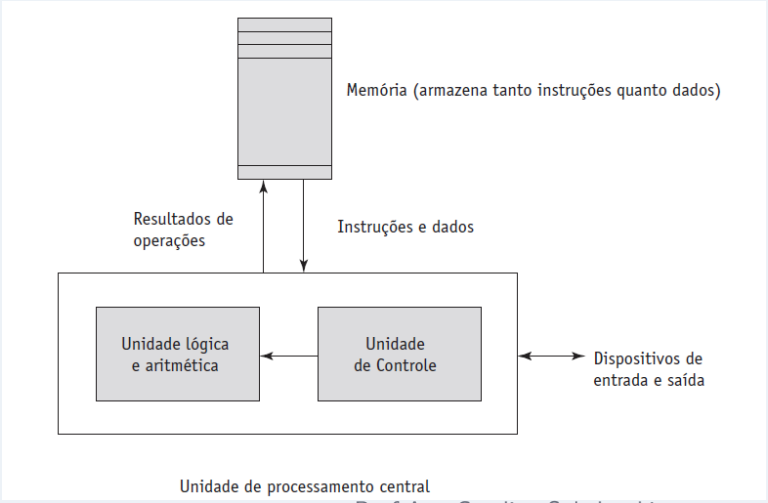
- **Variáveis** (células de memória)
- **Sentença de atribuição**: `x = 10;`
- Comandos sequenciais e controle de fluxo

Exemplos de linguagens imperativas:

C, C++, C#, Java, Python, PHP, Ruby, Cobol, JavaScript

Obs.: linguagens imperativas podem oferecer suporte a OO — Java exige OO; C++ aceita ambos os paradigmas.

Arquitetura de von Neumann



▪ Linguagens Visuais

Subcategoria das imperativas. Permitem criar interfaces gráficas por meio de **cliques e arrastar-e-soltar**, gerando código automaticamente.
Exemplo mais popular: **VB.NET**.

Disponíveis no ecossistema .NET: C#, JScript, J#, C++ gerenciado.

▪ Linguagens de Script

Subcategoria das imperativas, unidas principalmente pelo método de implementação: **interpretação parcial ou completa**.
São imperativas em todos os sentidos.

Exemplos: **Perl, JavaScript, Ruby**. [Sebesta 2010]

Baseadas em funções matemáticas

Organizadas em torno de **chamadas a funções**, não de sentenças de atribuição. Não há necessidade de variáveis nem de estados mutáveis.

As computações são realizadas pela **aplicação de funções a parâmetros**.

Exemplo: robô de limpeza

```
Ande(1);
Vire(esquerda);
Ande(2);
Aspire(15);
```

Prós e Contras

+ Simplicidade e elegância
+ Suporte a OO possível
– Eficiência menor que imperativas
– Uso ainda restrito na indústria

Representantes

LISP — pioneira e mais usada em IA
Scheme — dialeto de LISP, ideal para ensino
Haskell — funcional pura moderna

Peter Norving recomenda *Scheme* ou *Python* como primeira linguagem.

Baseadas em regras

Utilizam **cálculo de predicados** para comunicar processos ao computador. Ao invés de um procedimento com resultado fixo, aceitam **informações e regras de inferência** para computar resultados.

Exemplo: relação de parentesco

```
mae(Leandro, Leda).
mae(Leda, Luzia).

avo(X, Z) :- mae(X,Y), mae(Y,Z).
```

Com isso, o sistema **infere** que Luzia é avó de Leandro.

Representante: Prolog

Linguagem lógica mais usada. Para as linguagens lógicas, Prolog é o que LISP é para as funcionais: a grande referência.

Área de maior uso: **Inteligência Artificial**.

Referências I

BLIPBLOG. *Conheça os Tipos e as Linguagens de Programação mais usadas*. 2020. Https://www.blip.ai/blog/devs/linguagens-de-programacao/.

SEBESTA, R. *Conceitos de Linguagens de Programação*. [S.l.]: Bookman, 2010. ISBN 9788536301716.