

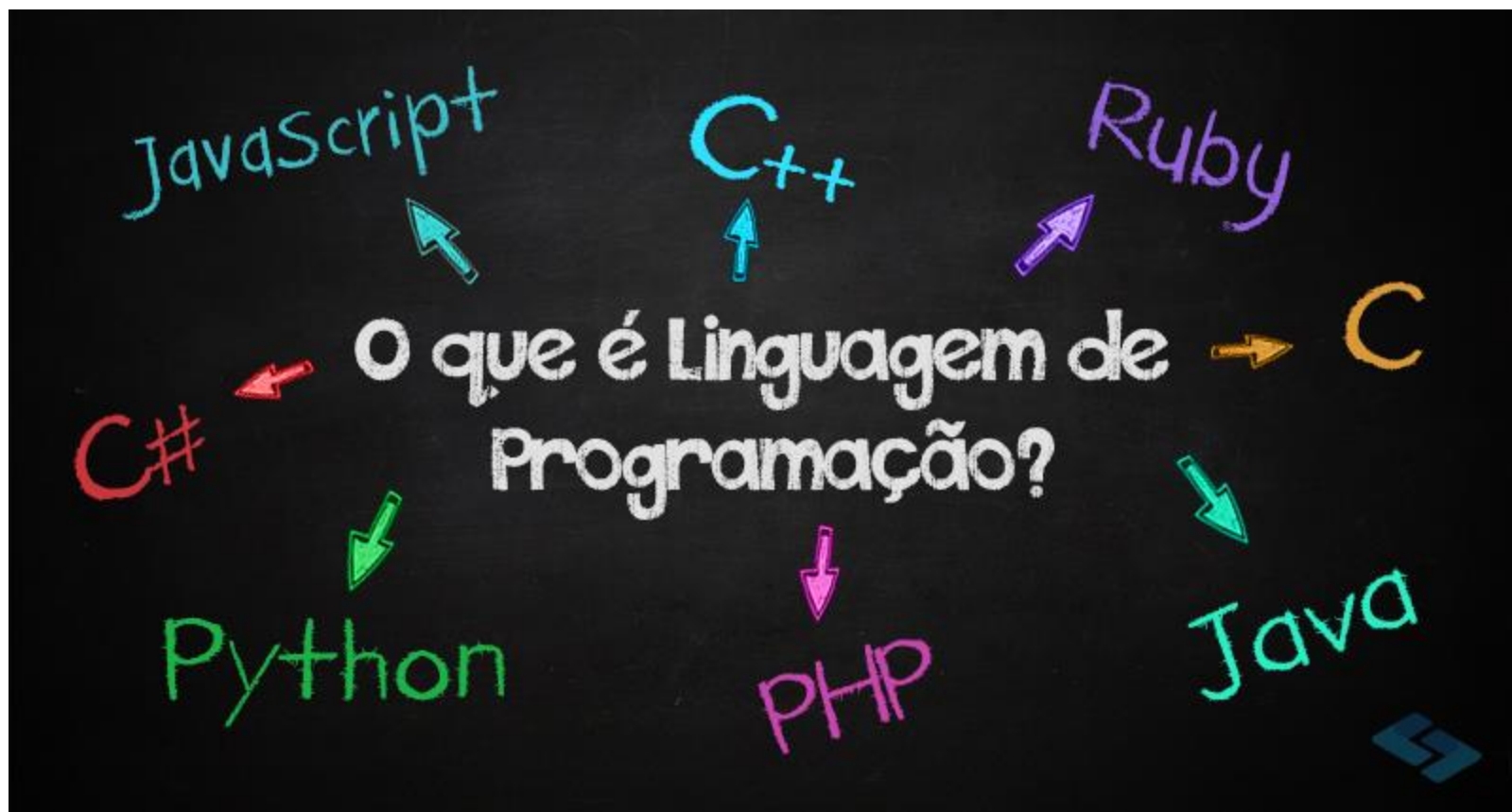


# *Paradigmas de Linguagens de Programação*

Profa. Joyce Miranda

# Paradigmas de Linguagens de Programação

---



# Paradigmas de Linguagens de Programação

---

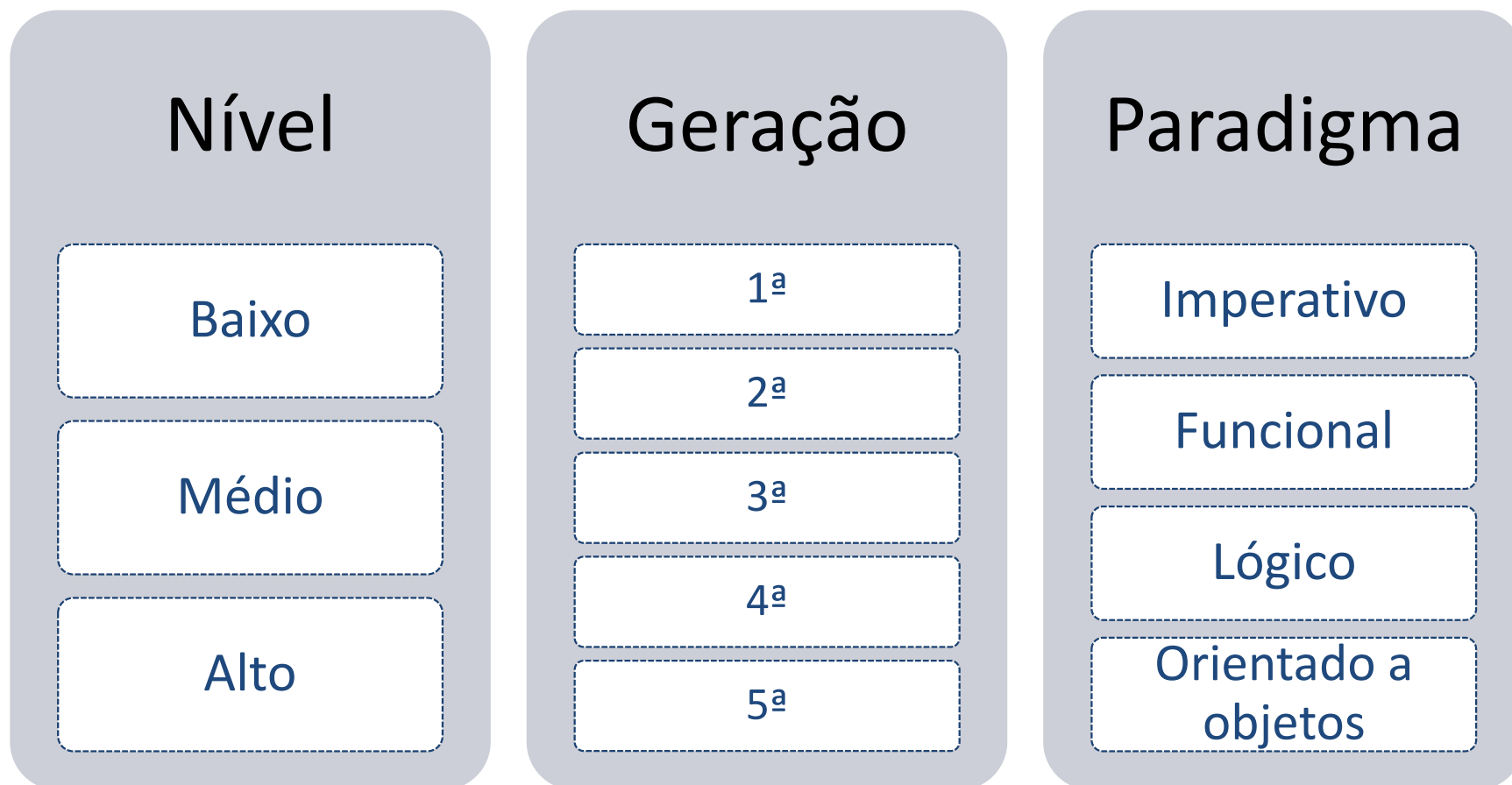
## ► Linguagem de Programação

Linguagem escrita e formal que especifica um conjunto de instruções e regras que são usadas para gerar programas (*software*).

# Paradigmas de Linguagens de Programação

---

## ► Classificação das Linguagens



# Paradigmas de Linguagens de Programação

---

## ► Classificação quanto ao **Nível**

### ► Linguagens de **Baixo** Nível

- São dependentes da arquitetura do computador.
  - Próximas à forma como o computador representa dados e instruções.
- Instruções são executadas pelo processador.
- Linguagem de Máquina.
  - Instruções que o processador é capaz de executar.
    - Sequências de bits, normalmente limitadas pelo número de bits do registrador da CPU (8, 16, 32, 64).
- Linguagem de Montagem/Simbólica (Assembly)
  - Notação legível por humanos para o código de máquina que uma arquitetura de computador específica utiliza.

# Paradigmas de Linguagens de Programação

## ► Classificação quanto ao **Nível**

### ► Linguagens de **Baixo** Nível

#### ► Exemplo código Assembly

$$X = \frac{(A + B) * C}{(D - E) * (F + G)}$$

A:	CONST	0	LD	D
B:	CONST	0	SUB	E
C:	CONST	0	ST	X
D:	CONST	0	LD	F
E:	CONST	0	ADD	G
F:	CONST	0	MULT	X
G:	CONST	0	ST	X
X:	CONST	0	LD	A
Inicio:	READ	A	ADD	B
	READ	B	MULT	C
	READ	C	DIV	X
	READ	D	ST	X
	READ	E	END	Inicio
	READ	F		
	READ	G		

# Paradigmas de Linguagens de Programação

---

## ► Classificação quanto ao **Nível**

### ► Linguagens de **Baixo** Nível

#### ► Vantagens

- ❑ Os programas são executados com maior velocidade de processamento.
- ❑ Os programas ocupam menos espaço na memória.

#### ► Desvantagens

- ❑ Não apresentam portabilidade.
- ❑ Programação mais complexa.

# Paradigmas de Linguagens de Programação

---

## ► Classificação quanto ao **Nível**

### ► Linguagens de **Alto** Nível

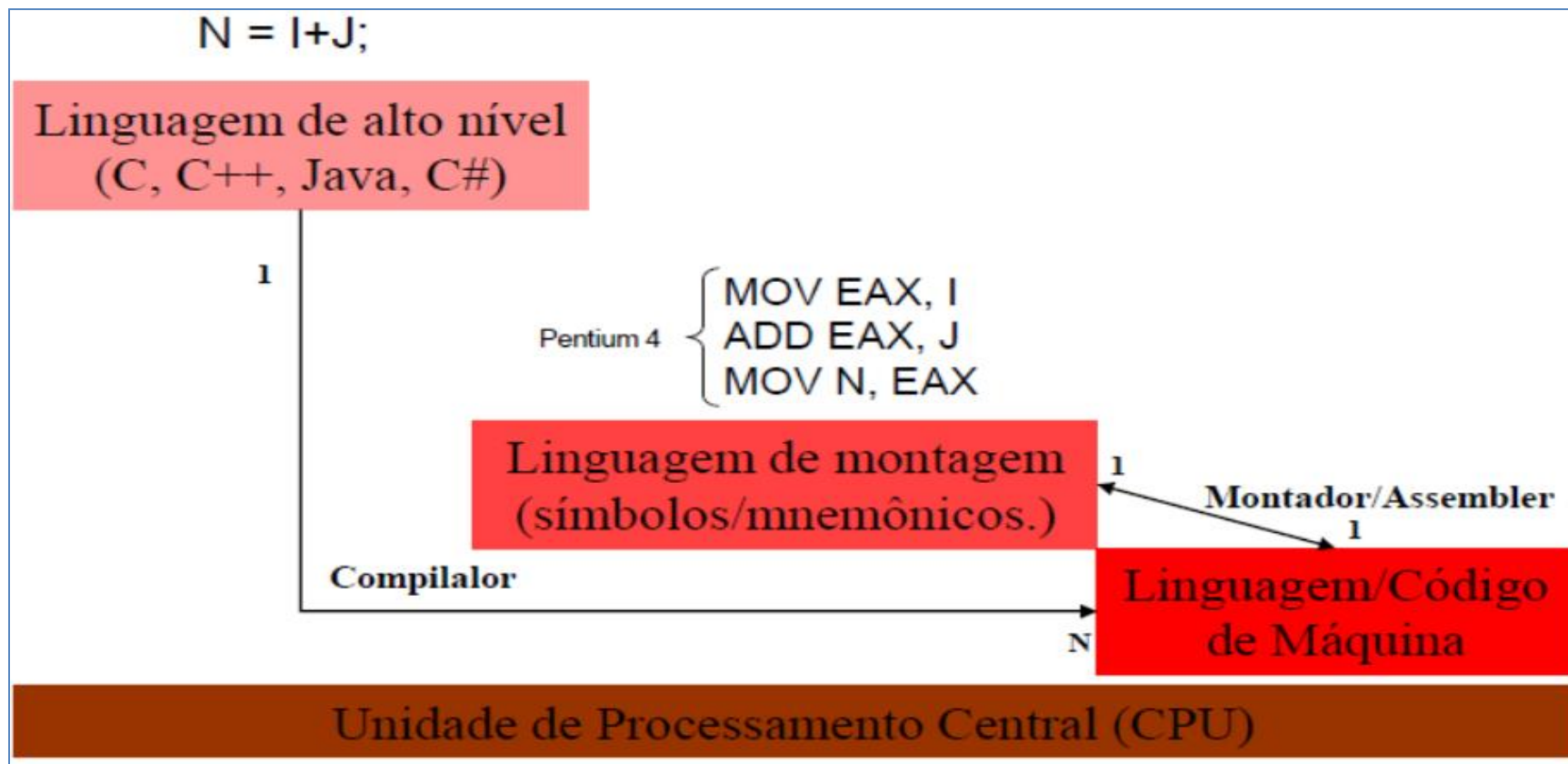
- Possuem um nível de abstração relativamente elevado, longe do código de máquina e mais próximo à linguagem humana.
- Não estão diretamente relacionadas à arquitetura do computador.
  - Características do processador, como instruções e registradores são abstraídas.
- Necessita de um tradutor
  - Traduz o código fonte em código de máquina.
    - Compilador
    - Interpretador



# Paradigmas de Linguagens de Programação

## ► Classificação quanto ao **Nível**

### ► Linguagens de **Alto** Nível



# Paradigmas de Linguagens de Programação

---

## ► Classificação quanto ao **Nível**

### ► Linguagens de **Alto** Nível

- Exemplos: LUA, JAVA, C#, C++

```
nota = io.read()

if nota < 3.0 then
    io.write("Reprovado")
elseif nota >= 5.0 then
    io.write("Aprovado")
else
    io.write("Prova final")
end
```

**LUA**

```
Scanner entrada = new Scanner(System.in);
mes = entrada.nextInt();
switch (mes)
{
    case 1: System.out.println("Janeiro");
            break;
    case 2: System.out.println("Fevereiro");
            break;
    case 3: System.out.println("Marco");
            break;
    default:
        System.out.println("Outro");
        break;
}
```

**JAVA**

# Paradigmas de Linguagens de Programação

---

## ► Classificação quanto ao **Nível**

### ► Linguagens de **Alto** Nível

#### ► Vantagens

- Possuem maior portabilidade.
  - Podem ser compiladas ou interpretadas.
- Programação mais simples.

#### ► Desvantagens

- São mais lentas.
- Ocupam mais memória.

# Paradigmas de Linguagens de Programação

---

## ▶ Classificação quanto ao **Nível**

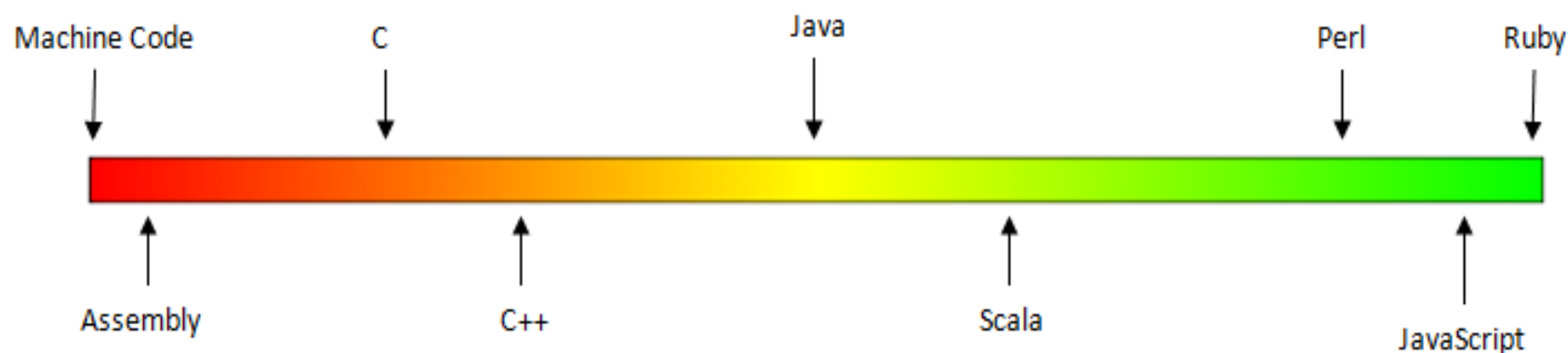
### ▶ Linguagens de **Médio** Nível

- ▶ Combinam características das linguagens de alto nível e as de baixo nível.
- ▶ Exemplo: Linguagem C
  - Baixo Nível: Manipula registros e endereços de memória.
  - Alto Nível: Permite realizar operações de alto nível (if...else; while; for).
- ▶ Vantagens
  - São linguagens mais rápidas em comparação às linguagens de alto nível.
- ▶ Desvantagens
  - Possuem sintaxe de comandos mais complexa em comparação às linguagens de baixo nível.

# Paradigmas de Linguagens de Programação

---

## ► Classificação quanto ao **Nível**



**Baixo Nível >**

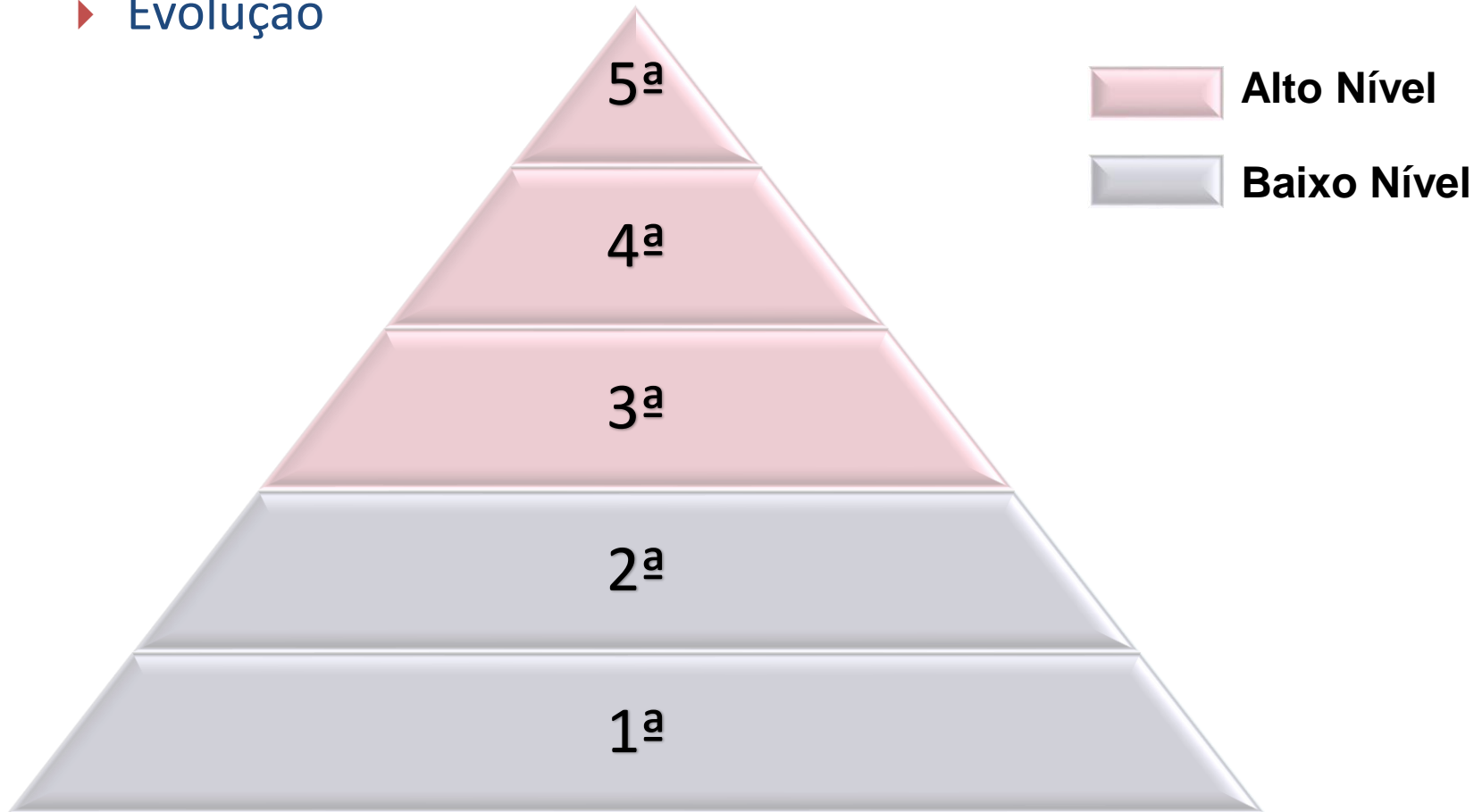
**< Alto Nível**

# Paradigmas de Linguagens de Programação

---

## ► Classificação quanto à **Geração**

### ► Evolução



# Paradigmas de Linguagens de Programação

---

## ► Classificação quanto à **Geração**

### ► 1ª Geração - Linguagens de máquina.

- Os primeiros computadores eram programados em linguagem de máquina, em notação binária.
- Exemplo:

0010 0001 0110 1100

*Realiza a soma (código de operação 0010) do dado armazenado no registrador 0001, com o dado armazenado na posição de memória 108 (0110 1100)*

# Paradigmas de Linguagens de Programação

## ► Classificação quanto à **Geração**

### ► 2ª Geração - Linguagens de montagem (Assembly).

- Projetadas para minimizar as dificuldades da programação em notação binária.
- Códigos de operação e endereços binários foram substituídos por mnemônicos (abreviações).

```
mov  mul  add  label  goto
```

### ► Exemplo

```
0010 0001 0110 1100
```



```
add r1 total
```

*r1 representa o registrador 0001 e total é o nome atribuído ao endereço de memória 108.*



# Paradigmas de Linguagens de Programação

---

## ► Classificação quanto à **Geração**

- 3ª Geração - Linguagens orientadas ao usuário.
  - Surgimento das linguagens de alto nível.
  - Primeiras linguagens dessa geração:
    - Fortran, Cobol, Algol, Basic, Ada, Pascal, C.

```
PROGRAM HELLO  
PRINT *, "HELLO WORLD!"  
END
```

**Fortran**

# Paradigmas de Linguagens de Programação

---

## ► Classificação quanto à **Geração**

### ► 4ª Geração - Linguagens orientadas à aplicação.

- Geram código a partir de expressões de alto nível
  - Dispensa conhecimentos técnicos profundos de programação.
  - O programador fornece apenas o conjunto das tarefas a serem realizadas, não se preocupando com os detalhes da execução.
- Exemplo: DBASE, SQL

**Tabela “Funcionario”**

nome	email	telefone	salario	cargo	*id
João da Silva	jsilva@swhere.com	7363-2334	2300	Gerente	1034
Carlos Ribas	cribas@cblanca.org	8334-3238	1800	Auxiliar	2938
Madalena Reis	mreis@portal.com	6451-5672	2000	Contador	7567

**Código SQL** `select nome, telefone FROM Funcionario;`

# Paradigmas de Linguagens de Programação

---

## ► Classificação quanto à **Geração**

### ► 4ª Geração - Linguagens orientadas à aplicação.

#### ► Objetivos:

- ❑ Facilitar a programação de tal maneira que usuários finais possam resolver seus problemas.
- ❑ Apressar o processo de desenvolvimento de aplicações.
- ❑ Facilitar e reduzir o custo de manutenção de aplicações.
- ❑ Minimizar problemas de depuração.
- ❑ Gerar código sem erros a partir de requisitos de expressões de alto nível.

# Paradigmas de Linguagens de Programação

---

## ► Classificação quanto à **Geração**

### ► 5ª Geração - Linguagens do conhecimento.

- São usadas principalmente na área de Inteligência Artificial.
- Facilitam a representação do conhecimento para a simulação de comportamentos inteligentes..
- Exemplo: Prolog, Lisp, Art.

Todos os homens são  
mortais.

Sócrates é um homem.

Sócrates é mortal.

**Linguagem Natural**

```
mortal(X) :- homem(X).
```

```
homem(sócrates).
```

```
?-mortal(sócrates).
```

**Prolog**

# Paradigmas de Linguagens de Programação

---

## ▶ Classificação quanto ao **Paradigma**

### ▶ Paradigma

- ▶ Modelo de raciocínio para a resolução de problemas.

### ▶ Paradigma de Programação

- ▶ Modelo de programação suportado por linguagens que agrupam certas características em comum.
- ▶ Diferentes formas de pensar sobre a programação.
- ▶ Os paradigmas de programação se diferem uns dos outros em seus conceitos, aplicações e técnicas

# Paradigmas de Linguagens de Programação

---

## ► Classificação quanto ao **Paradigma**

### ► Paradigma Imperativo

- *“Primeiro faça isso e depois faça aquilo.”*
- As linguagens imperativas são orientadas a ações
  - O código é visto como uma sequência de instruções que manipulam valores de variáveis (leitura e atribuição).
  - Os programas são centrados no conceito de um estado (modelado por variáveis) e ações (comandos) que manipulam o estado.
  - Inclui a definição de sub-rotinas ou procedimentos como mecanismo de estruturação.

# Paradigmas de Linguagens de Programação

---

## ► Classificação quanto ao **Paradigma**

### ► Paradigma Imperativo

#### ► Subdivide-se em:

- ❑ Estruturado
- ❑ Não-Estruturado

#### ► **Linguagens Não Estruturadas**

- ❑ Geralmente fazem uso de comandos goto ou jump para um desvio de fluxo de execução (salto).
- ❑ Exemplos – Assembly e Basic:

```
_start:
    cmp eax, ebx
    jne .L7
    mov edx, ecx
.L7:
    mov eax, edx
    add ecx, edx
```

```
10 PRINT "Hello"
20 GOTO 50
30 PRINT "This text will not be printed"
40 END
50 PRINT "Goodbye"
```

# Paradigmas de Linguagens de Programação

---

## ► Classificação quanto ao **Paradigma**

### ► Paradigma Imperativo

#### ► **Linguagens Estruturadas**

- Não fazem o uso do goto ou jump.
  - Objetivam facilitar a leitura e a execução dos algoritmos.
- Instruções são agrupadas em blocos, os quais podem ser considerados como unidades do programa.
- Blocos de instruções
  - Podem ser selecionados para execução através de declarações de seleção como if ... else, ou repetidamente executados através de declarações de repetição (for, while...).



# Paradigmas de Linguagens de Programação

---

## ► Classificação quanto ao **Paradigma**

### ► Paradigma Imperativo

#### ► **Linguagens Estruturadas**

- Exemplo de Linguagem Estruturada: C

```
int busca(int n, int *vet, int elem)
{
    int i;
    for (i = 0; i < n; i++)
    {
        if (elem == vet[i])
        {
            return i;
        }
    }
    return -1;
}
```

# Paradigmas de Linguagens de Programação

---

## ► Classificação quanto ao **Paradigma**

### ► Paradigma Funcional

- Visa estruturar o código seguindo o modelo de funções matemáticas, objetivando a imutabilidade dos dados e o acoplamento de funções.

```
função (... função2(função1(dados)) ...)
```

- Ao invés dos passos sucessivos do paradigma imperativo, a sintaxe da linguagem se define por múltiplas funções que se compõem para resolver um problema.

# Paradigmas de Linguagens de Programação

---

## ► Classificação quanto ao **Paradigma**

### ► Paradigma Funcional (Características)

#### ► **Modelo Declarativo**

##### **Modelo imperativo**

- Linguagens expressam seqüências de comandos que realizam transformações sobre dados
- Base: máquina de von Neumann
  - orientadas a procedimentos
  - orientadas a objetos

##### **Modelo declarativo**

- Linguagens que não possuem os conceitos de
  - seqüências de comandos
  - atribuição
- linguagens funcionais: ênfase em valores computados por funções
- linguagens lógicas: ênfase em axiomas lógicos

# Paradigmas de Linguagens de Programação

---

## ► Classificação quanto ao **Paradigma**

### ► Paradigma Funcional (Características)

#### ► **Imutabilidade**

- ❑ Dados são imutáveis, ou seja, uma vez atribuído o valor a uma “variável”, esse valor não deve ser alterado.
- ❑ A imutabilidade faz sentido dentro da programação funcional pelo seu viés matemático.
  - ❑ ... Se você tem uma expressão como  $f(x) = x + 2$ ;  $x = 3$ ;
  - ❑ .... O número 3 passado para  $x$ , não irá mudar seu valor, ou seja, ele permanece inalterado após o seu uso na função.
  - ❑ Por isso:
  - ❑ .... Não se aconselha o uso de variáveis, mas de constantes.
  - ❑ .... Variáveis globais não devem ser utilizadas.

# Paradigmas de Linguagens de Programação

---

## ► Classificação quanto ao **Paradigma**

### ► Paradigma Funcional (Características)

#### ► **Imutabilidade**

```
var varMutable = 3

console.log(varMutable) // 3

var impureFunction = () => { varMutable = 4 }

impureFunction()

console.log(varMutable) // 4
```

**Mutável**

# Paradigmas de Linguagens de Programação

---

## ► Classificação quanto ao **Paradigma**

### ► Paradigma Funcional (Características)

#### ► **Imutabilidade**

```
const immutableVariable = 3

const pureFunction = number => number + 2

const otherImmutableVariable = pureFunction(immutableVariable)

console.log(number) // number is not defined
console.log(immutableVariable) // 3
console.log(otherImmutableVariable) // 5
```

**Imutável**

# Paradigmas de Linguagens de Programação

---

## ► Classificação quanto ao **Paradigma**

### ► Paradigma Funcional (Características)

#### ► **Transparência Referencial**

- A função não deve sofrer nem exercer influências implícitas no escopo exterior a ela.
- Nenhum dado pode ser capturado pela função a não ser através de seus próprios argumentos. [ Não devem ser usadas variáveis globais]

# Paradigmas de Linguagens de Programação

---

## ► Classificação quanto ao **Paradigma**

### ► Paradigma Funcional (Características)

#### ► Transparência Referencial

```
let name = 'World'

const sayHello = () => {

  // Lê dados de uma variável externa e a modifica.

  name = 'Hello ' + name + '!'

  // Efeito colateral.

  console.log(name)
}

sayHello()
```

Não Transparente

```
let name = 'World'

// Sem efeito colateral.

const hello = name => 'Hello ' + name + '!'

// Transparência referencial: 'name' é um argumento.

console.log(hello(name))
```

Transparente



# Paradigmas de Linguagens de Programação

---

## ▶ Classificação quanto ao **Paradigma**

### ▶ Paradigma Funcional (Características)

#### ▶ **Pureza**

- Uma função não pode produzir nenhum “efeito colateral”.
- Ou seja, persistência de dados, escrita no sistema de arquivos ou renderizações de interface devem ser evitadas dentro de expressões funcionais.

# Paradigmas de Linguagens de Programação

---

## ► Classificação quanto ao **Paradigma**

### ► Paradigma Funcional (Características)

#### ► Pureza

```
let name = 'World'

const sayHello = () => {

  // Lê dados de uma variável externa e a modifica.

  name = 'Hello ' + name + '!'

  // Efeito colateral.

  console.log(name)
}

sayHello()
```

**Não Puro**

```
let name = 'World'

// Sem efeito colateral.

const hello = name => 'Hello ' + name + '!'

// Transparência referencial: 'name' é um argumento.

console.log(hello(name))
```

**Puro**

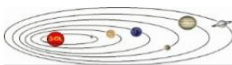
# Paradigmas de Linguagens de Programação

## ► Classificação quanto ao **Paradigma**




### ► Paradigma Lógico

- Utiliza **lógica matemática similar à maneira como o ser humano raciocinaria sobre um problema.**
- Segue estilo **declarativo**, ou seja, especifica o que se deseja e não como deve ser feito.

### FATOS, REGRAS E PERGUNTAS



- Um fato é uma afirmação **sempre verdadeira**.  
A Terra é um planeta.  
O Sol é uma estrela.
- Uma regra é uma afirmação que para ser verdadeira **depende de outras regras** ou fatos.  
A Terra é um planeta **se** ela não for uma estrela.
- Podemos fazer **perguntas** sobre os **fatos ou regras**.  
A Terra é um planeta ?  
Quem é um planeta ?  
A Terra é uma estrela ?



# Paradigmas de Linguagens de Programação

---

## ► Classificação quanto ao **Paradigma**

### ► Paradigma Lógico

#### ► Exemplo Linguagem Prolog

```
avo(X, Y) :- pai(X, Z), pai(Z, Y).  
avo(X, Y) :- pai(X, Z), mae(Z, Y).  
  
pai(carlos, joao).  
pai(joao, jose).  
  
mae(maria, joao).
```

```
?- avo(carlos, jose).  
true .
```

```
?- avo(carlos, joao).  
false.
```

```
?- avo(manoel, carlos).  
false.
```

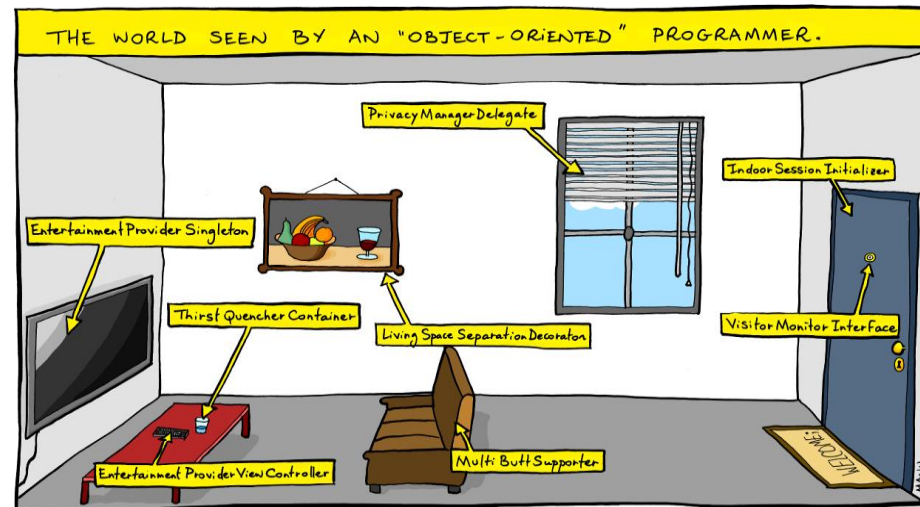
```
?- avo(X, jose).  
X = carlos .
```

# Paradigmas de Linguagens de Programação

## ► Classificação quanto ao **Paradigma**

### ► Paradigma Orientado a Objetos

- Trata os elementos e conceitos associados à solução de um problema como **objetos**.
- Objetos:
  - Entidades abstratas que embutem dentro de suas fronteiras **características** e **operações** relacionadas com a entidade real.



# Paradigmas de Linguagens de Programação

---

## ► Classificação quanto ao **Paradigma**

### ► Paradigma Orientado a Objetos

- Propõe a redução da distância entre a modelagem computacional e o mundo real:
  - ❑ O ser humano se relaciona com o mundo através de conceitos de objetos;
  - ❑ Estamos sempre identificando qualquer objeto ao nosso redor;
  - ❑ Para isso lhe damos nomes, e de acordo com suas características lhes **classificamos** em grupos;

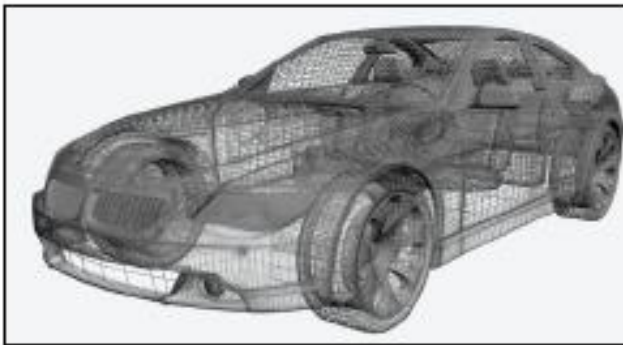
# Paradigmas de Linguagens de Programação

---

## ► Classificação quanto ao **Paradigma**

### ► Paradigma Orientado a Objetos

- O código é organizado em classes que agrupam objetos que apresentam mesmas características (atributos) e mesmos comportamentos (métodos).
- **Classe:**
  - É o modelo (molde) de construção de objetos que define as características e comportamentos que os objetos irão possuir.



**Classe**



**Objeto**

# Paradigmas de Linguagens de Programação

## ► Classificação quanto ao **Paradigma**

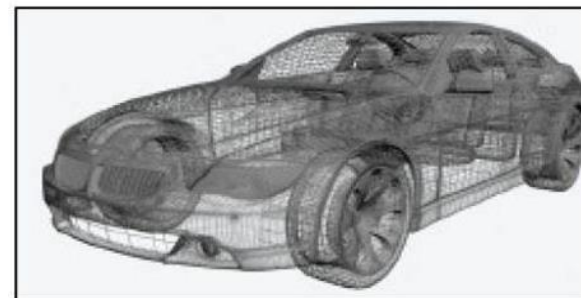
### ► Paradigma Orientado a Objetos

#### ► Exemplo JAVA

```
public class Carro {  
    private String marca;  
    private String cor;  
    private String placa;  
    private int portas;  
    private int marcha;  
    private double velocidade;  
  
    public void Acelerar()  
    {  
        velocidade += marcha * 10;  
    }  
    public void Frear()  
    {  
        velocidade -= marcha * 10;  
    }  
}
```

Atributos

Métodos



Carro
- Marca: Texto - Cor: Texto - Placa: Texto - N° Portas: Inteiro ...
+ Acelerar(): void + Frear(): void + TrocarMarcha(x): void + Buzinar(): void ...



# Paradigmas de Linguagens de Programação

## Vertical

1. Linguagem de Baixo Nível.
3. Tipo de linguagem imperativa, no qual as instruções são agrupadas em funções e/ou procedimentos.
4. Linguagem de Alto Nível.
5. Paradigma que se aproxima da forma como o ser humano raciocina sobre a solução de um problema.

## Horizontal

2. Elemento do paradigma O.O que define propriedades e comportamentos comuns a diferentes objetos.
6. Paradigma que define o código como uma sequência de ações que manipulam variáveis.
7. O código é organizado de acordo com o acoplamento de múltiplas funções que seguem o estilo matemático.
8. Estilo de linguagem que não seguem o conceito de sequência de comandos e atribuição.

