

# Paradigmas de Linguagens de Programação



Prof.: Claudio Junior (claudiojns@ufba.br)

Paradigmas de Linguagem de Programação (MATA56)

2023.1

# Na aula anterior...

---

- Métodos de implementação
  - Compilação
  - Interpretação Pura
  - Sistemas de Implementação Híbridos

# Agenda

---

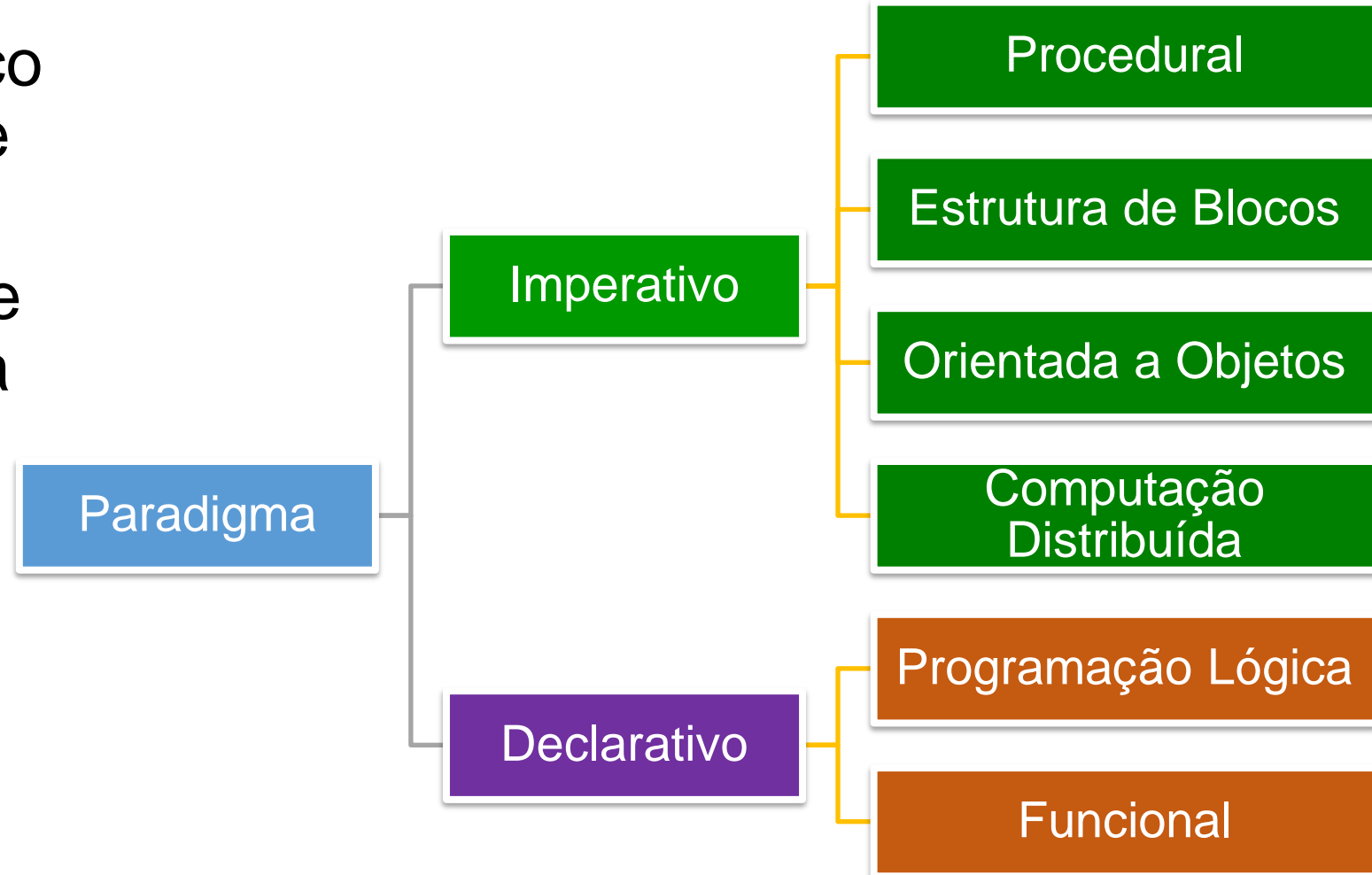
- Classificação das Linguagens de Programação;
- Paradigma Imperativo;
- Paradigma Orientado a Objetos;
- Paradigma Funcional;
- Paradigma Lógico;
- Evolução das Linguagens de Programação.

# Classificação das Linguagens de Programação

Estrutura de Tipos	Abstração	MacLennan	Apley & VandeKopple
<b>Fracamente Tipada</b> Muda de acordo com a situação	<b>Baixo Nível</b> Representação do código de máquina	<b>1º Geração</b> Máquina, GOTO. Fortran	<b>1º Geração</b> Linguagem de máquina
<b>Fortemente Tipada</b> Se mantém o mesmo	<b>Médio Nível</b> Conversão para código máquina (compilador)	<b>2º Geração</b> Estruturação dos controles. Algol 60	<b>2º Geração</b> Linguagem de Montagem
<b>Dinamicamente Tipada</b> Definido em tempo de execução	<b>Alto Nível</b> Inteligível para o ser humano	<b>3ª Geração</b> Simplicidade e eficiência. Pascal	<b>3ª Geração</b> Procedurais
<b>Estaticamente Tipada</b> Definido em tempo de compilação		<b>4º Geração</b> Abstração de dados, encapsula- mento. Ada	<b>4º Geração</b> Aplicativas
		<b>5º Geração</b> OO, Funcional e Lógica	<b>5º Geração</b> IA, Funcionais
			<b>6º Geração</b> Redes neurais

# Classificação quanto aos Paradigmas

- Exemplo típico ou modelo de algo;
- Agrupadas de acordo com a sintaxe e a semântica.



# Classificação quanto aos Paradigmas

---

- Imperativos - Facilitam a computação por meio de estado;
  - Procedural - Os programas são executados por meio de chamadas sucessivas a procedimentos separados:
    - Basic, Fortran
  - Estruturas de Blocos - Possuem os escopos aninhados:
    - Algol 60, Pascal e C
  - Orientação a Objetos - Linguagens que suportam interação entre objetos:
    - C++, Java, Python
  - Computação Distribuída - Mais de uma rotina possa executar independentemente:
    - Ada.

# Classificação quanto aos Paradigmas

---

- Declarativos
  - O programa especifica uma relação ou função;
  - **Funcional:**
    - Não incluem qualquer provisão para atribuição ou dados mutáveis;
    - Lisp, Scheme e Haskell.
  - **Programação Lógica:**
    - Um programa implementa uma relação ao invés de um mapeamento.
    - Prolog.



# Paradigma Imperativo



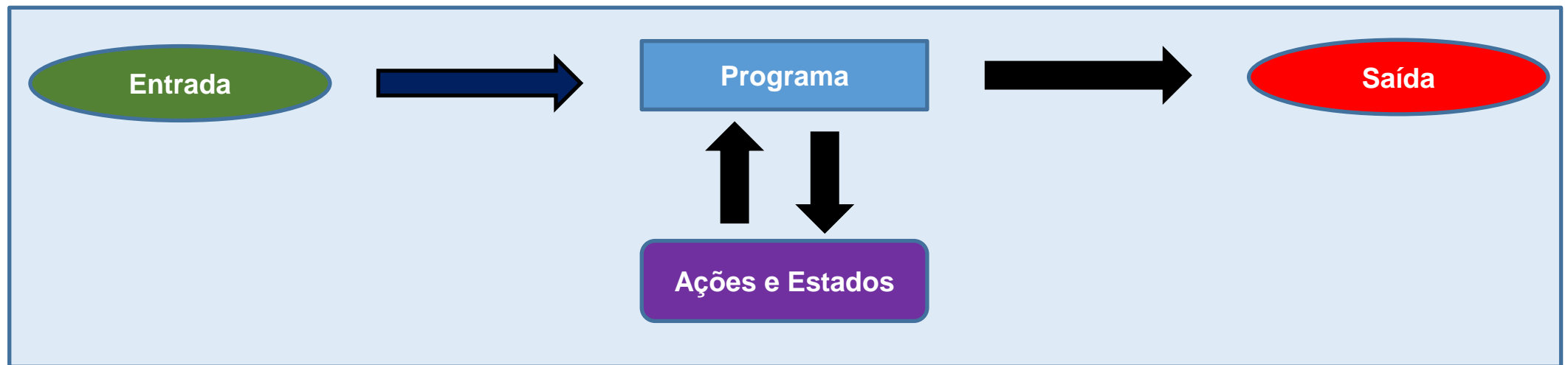
# Paradigma Imperativo

---

- Descreve a computação como ações, enunciados ou comandos que mudam o estado (modelado por variáveis) de um programa. As instruções devem ser passadas ao computador na sequência em que devem ser executadas;
- Paradigma procedural: inclui sub-rotinas ou procedimentos como mecanismo de estruturação;
- Primeiro paradigma a surgir: ainda é muito utilizado

# Paradigma Imperativo – Modelo Computacional

---



Estrutura da Programação Imperativa

Fortran, Cobol, Basic, Pascal, C e Ada.

# Paradigma Imperativo – COBOL x ALGOL X Pascal


IDENTIFICATION DIVISION.

PROGRAM-ID. FATORIAL. 

DATA DIVISION.

WORKING-STORAGE SECTION.

01 NUMERO PIC 9(4). 

01 FATORIAL PIC 9(9) COMP VALUE 1. 

PROCEDURE DIVISION.

DISPLAY "Digite o número: ".

ACCEPT NUMERO.

PERFORM CALCULAR-FATORIAL.

DISPLAY "O fatorial de ", NUMERO, " é ", FATORIAL.

STOP RUN.

CALCULAR-FATORIAL. 

PERFORM VARYING I FROM 1 BY 1 UNTIL I > NUMERO


COMPUTE FATORIAL = FATORIAL \* I 

END-PERFORM.


begin

integer n, i, fat; 

read(n);

fat := 1; 

for i := 2 step 1 until n do 

fat := fat \* i; 


write(fat);

end Fatorial. 

function fatorial (n: integer) : integer;


var f: integer; 

begin

f := 1; 

while n > 0 do 

begin

f := f \* n; n := n - 1 

end;

fatorial := f

end

# Paradigma Imperativo

---

- ✓ Características centrais das Linguagens Imperativas:
  - ✓ As variáveis, que modelam as células de memória;
  - ✓ Comandos de atribuição, que são baseados nas operações de transferências dos dados e instruções;
  - ✓ A execução sequencial de instruções;
  - ✓ A forma iterativa de repetição, que é o modelo mais eficiente desta arquitetura.

# Paradigma Imperativo

---

Vantagens	Desvantagens
Eficiência	Facilita a introdução de erros em sua manutenção
Modelagem “natural” de aplicações do mundo real	Tende a gerar códigos confusos, onde o tratamento dos dados são misturados com o comportamento do programa
Paradigma dominante e bem estabelecido	Descrições demasiadamente operacionais focalizam o “como” e não “o que”
Flexível	
Mais fácil de traduzir para a linguagem de máquina	



# Paradigma Orientado a Objetos

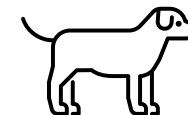
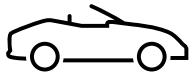
# Paradigma Orientado a Objetos

---

- Conhecido e popularizado na década de 90 com Java. Origem na resolução de problemas da indústria de *software*;
- Não é um paradigma no sentido estrito: subclassificação do imperativo;
- Metodologia de concepção e modelagem do sistema: a aplicação é **estruturada** em módulos (**classes**) que agrupam um estado (**atributos**) e operações (**métodos**) sobre este estado.
- Classes podem ser estendidas (**herança**) e/ou usadas como tipos (cujos elementos são **objetos**).

# Paradigma Orientado a Objetos - Objeto

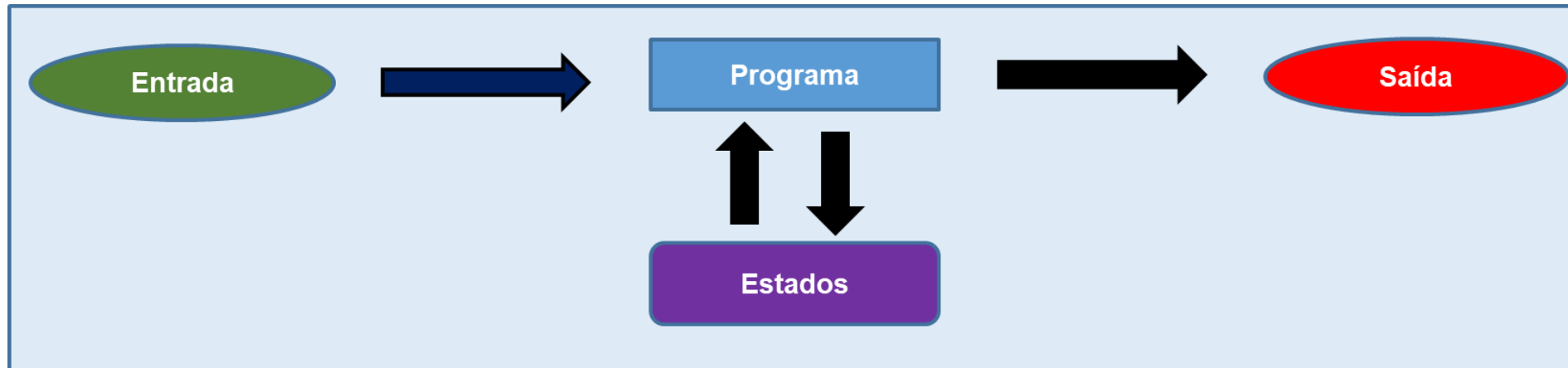
- Entidade autônoma que combina a representação da informação (estruturas de dados) e sua manipulação (procedimentos): Objeto = dados + operações;
- Item identificável, unidade, ou entidade individual, real ou abstrato, com uma regra bem definida;
- Identidade: representa a forma que se conhece o objeto, é sua referência.



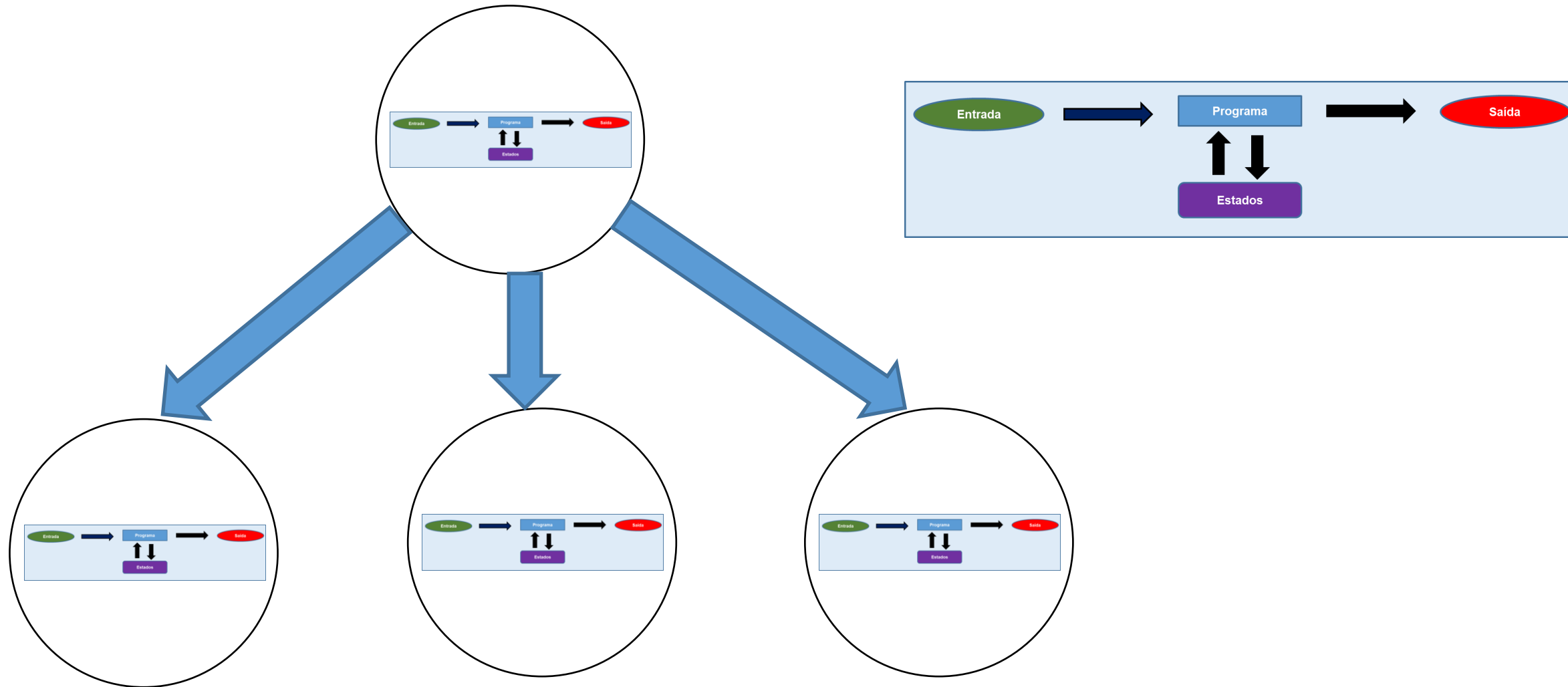


# Paradigma Orientado a Objetos – Modelo Computacional

---



# Paradigma Orientado a Objetos – Modelo Computacional



# Paradigma Orientado a Objetos

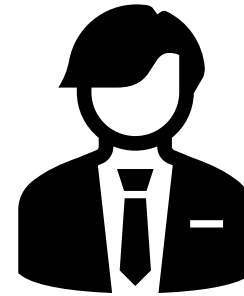
---

Vantagens	Desvantagens
Modularidade	Execução mais lenta
Reusabilidade	Curva de aprendizado
Extensibilidade	Maior esforço para modelagem do sistema
Aceitação comercial crescente	
Confiabilidade	
Manutenabilidade	

# Paradigma Orientado a Objetos

---

- ✓ Objetos do tipo Empregado
  - Identidade única: CPF
  - Atributos:
    - endereço
    - idade
    - dependentes
    - salario
    - cargo
  - Comportamento/Métodos (Operações):
    - aumentar salario
    - listar dependentes
    - contratar
    - demitir



# Paradigma Orientado a Objetos

---



Alicerces da Orientação a Objetos

# Paradigma Orientado a Objetos

---

## Abstração

- Processo de identificar as qualidades ou propriedade importantes do fenômeno que está sendo modelado e ignorar todas as propriedades irrelevantes.

## Encapsulamento

- Separa os aspectos externos de um objeto dos detalhes internos de sua implementação.

## Polimorfismo

- Propriedade que permite que a mesma mensagem seja enviada a diferentes objetos e que cada objeto execute a operação que é apropriada à sua classe

## Herança

- Refere-se ao fato de que uma subclasse herda todos os componentes da classe pai, incluindo uma estrutura de estado interna e pares "mensagem - método"



# Paradigma Funcional

# Paradigma Funcional

---

- A computação é vista como uma avaliação de funções matemáticas. Evita estados ou dados mutáveis;
- Suporte a uma abordagem funcional pura para a solução de problemas. Indicado quando a solução requerida é fortemente dependente de uma base matemática;
- O problema proposto é dividido em blocos e as funções implementadas farão os cálculos matemáticos;
- Conceitos sofisticados como polimorfismo, funções de alta ordem e avaliação sob demanda
- Aplicação: prototipação em geral e IA



# Paradigma Funcional - Características

---

- Não usa variáveis e comandos de atribuição;
- Não usa laços;
- Repetição por recursividade.
- Linguagens: Lambda, LISP, Scheme, ML, Miranda, Haskell.

# Paradigma Funcional

---

ML

```
fun factorial(n) =  
  if n > 0  
  then n * factorial(n-1)  
  else 1
```

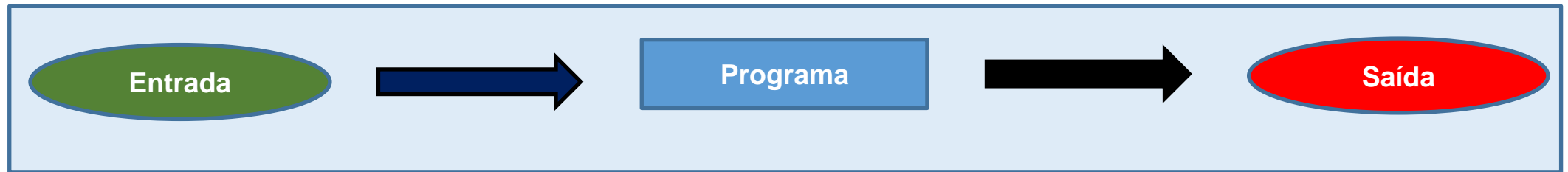
*Programas podem ser feitos sem usar variáveis locais e laços*

# Paradigma Funcional

Vantagens	Desvantagens
Visualização uniforme dos programas como funções	Ineficiência de execução
Tratamento das funções como dados	Devido à sua natureza dinâmica LP funcionais são interpretadas
Limitação do efeito colateral	“O mundo não é funcional”
Uso de gerenciamento de memória automático	
Grande flexibilidade	
Notação concisa	
Semântica simples	

# Paradigma Funcional – Modelo Computacional

---



Estrutura da Programação Funcional



# Paradigma Lógico

# Paradigma Lógico

---

- Utiliza formas de lógica simbólica como padrões de entrada e saída e assim realiza inferências para produzir os resultados;
- Principais elementos: proposições, regras de inferência e busca;
- Estilo Declarativo. Utilizada na solução de problemas que envolvem inteligência artificial, criação de programas especialistas e comprovação de teoremas;
- Na prática, inclui características imperativas, por questão de eficiência
- Linguagens: Popler, Conniver, QLISP, Planner, **Prolog**, Mercury, Oz, Frill

# Paradigma Lógico

Vantagens	Desvantagens
Permite concepção da aplicação em um alto nível de abstração (através de associações entre E/S).	Variáveis de programa não possuem tipos, nem são de alta ordem.
Visualização uniforme dos programas como funções	Ineficiência de execução
Tratamento das funções como dados	Devido à sua natureza dinâmica LP Lógicas são interpretadas
Limitação do efeito colateral	“O mundo não é lógico”
Uso de gerenciamento de memória automático	
Notação concisa	
Semântica simples	

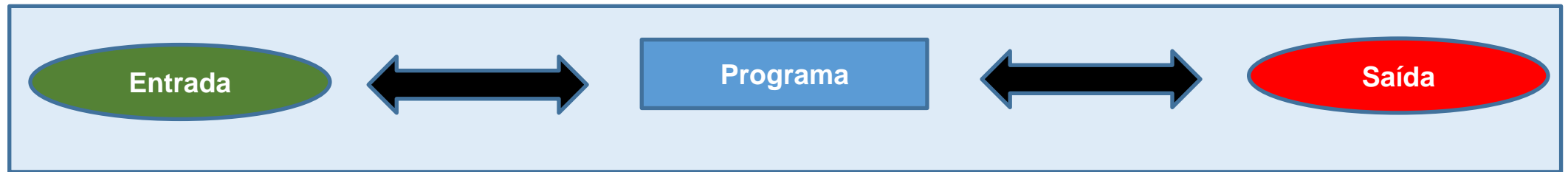
# Paradigma Lógico

Estrutura	Exemplo
Fato	<p>cachorro(tom). <i>(informa que tom é um cachorro)</i></p> <p>?- cachorro(tom). <i>(tom é um cachorro?)</i> yes.</p>
Regras	<p>luz(acesa) :- interruptor(ligado). <i>(luz(acesa) é verdadeiro se interruptor(ligado))</i></p> <p>avo(X,Z) :- pai(X,Y), pai(Y,Z) <i>(X é avô de Z se X é pai de Y e Y é pai de Z)</i></p>



# Paradigma Lógico – Modelo Computacional

---



Estrutura da Programação Lógica



# Comparativo

```
function fatorial n:integer):integer;
  var f: integer;
  begin
    f := 1;
    while n > 0 do
      begin
        f := f * n; n := n - 1
      end;
    factorial := f
  end
```

Imperativo

```
fun fatorial(n) =
  if n > 0
  then n * fatorial(n-1)
  else 1
```

Funcional

```
fatorial(0, 1).
fatorial(N, F) :-
  N > 0, N1 is N - 1,
  fatorial(N1, F1),
  F is N * F1.
```

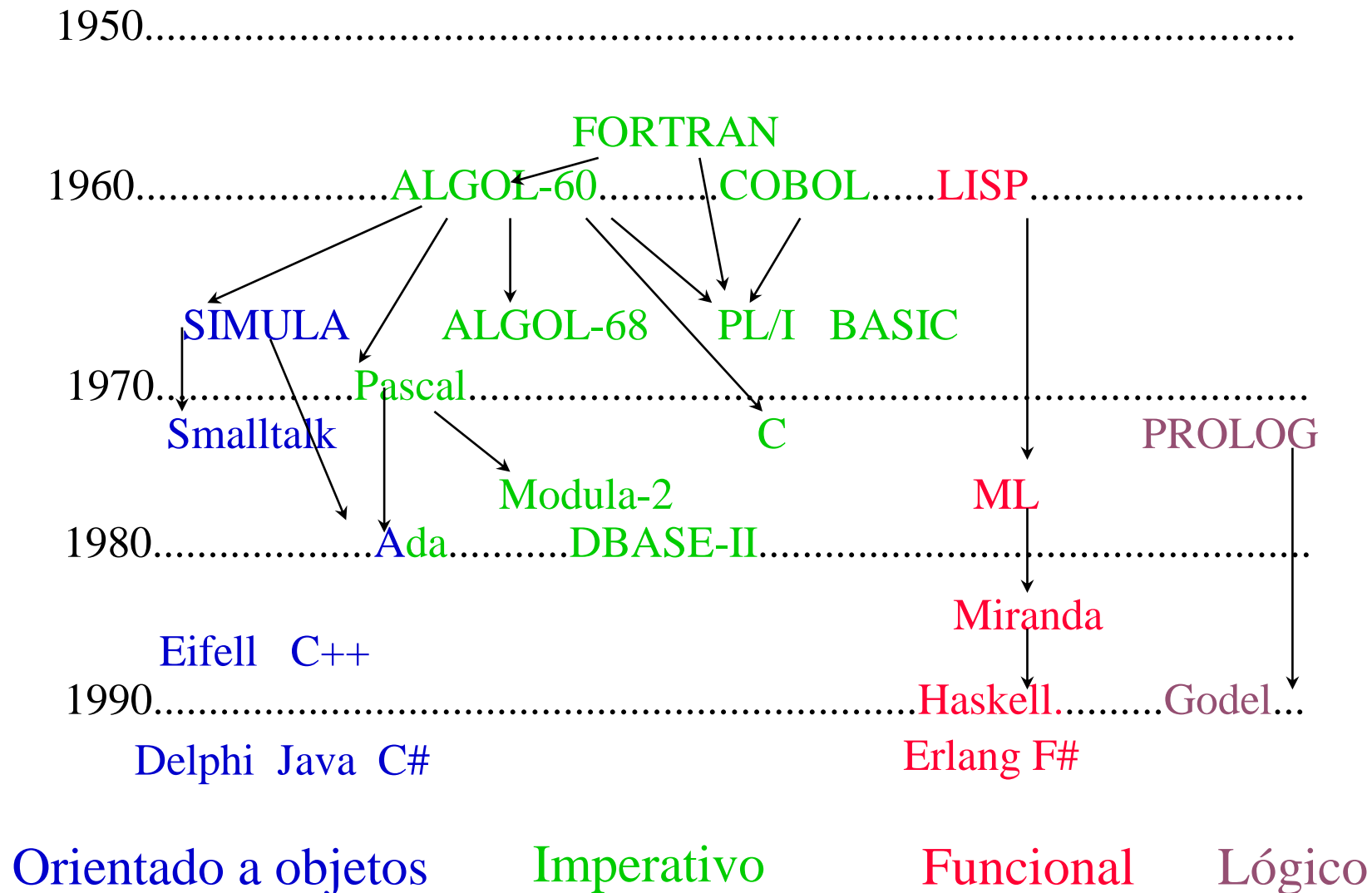
Lógico

```
public class Fatorial {
  public static void main(String[] args) {
    int num = 5;
    BigInteger resultado = calcularFatorial(num);
  }

  public static BigInteger calcularFatorial(int num) {
    BigInteger resultado = BigInteger.valueOf(1);
    for int i = 1; i <= num; i++) {
      resultado = resultado.multiply(BigInteger.valueOf(i));
    }
    return resultado;
  }
}
```

Orientado a Objetos

# Evolução das Linguagens de Programação



# Qual Paradigma devemos usar?

---



# Qual Paradigma devemos usar?

---

- Depende do tipo de problema;
- Das ferramentas de programação (linguagens);
- Da experiência do programador;
- Da equipe...