Classificação das Linguagens

- As linguagens de programação podem ser **classificadas** em relação a três critérios:
  - **Em relação ao nível:**
    - Baixo nível, Médio nível, ou Alto nível;
  - **Em relação à geração:**
    - 1ª Geração, 2ª Geração, 3ª Geração, 4ª Geração, ou 5ª Geração;
  - **Em relação ao paradigma:**
    - Imperativo, Funcional, Lógico, Orientado a Objetos, ...;

Classificação das Linguagens: Paradigma

- Paradigma é um **modelo interpretativo** (ou conceitualização) de uma **realidade**.

- Permite organizar as ideias com vista:
  - Ao entendimento dessa realidade;
  - À determinação de qual a melhor forma de atuar sobre essa realidade.

- Pode dizer-se que um paradigma é um **ponto de vista**: um ponto de vista que determina como uma realidade é entendida e como se atua sobre ela.

Classificação das Linguagens: Paradigma

- Algumas linguagens criadas durante a história introduziram **novas formas de se pensar sobre programação**, resultando em formas distintas de modelagem de soluções de software.
  - FORTRAN (imperativa);
  - LISP (funcional);
  - Simula (orientadas a objetos);
  - Prolog (lógica).

- Outras linguagens são o resultado da evolução de linguagens mais antigas, muitas vezes **mesclando** características de diferentes linguagens existentes.
  - Por exemplo, C++ é uma evolução do C com características de orientação a objetos, importadas de Simula.

Classificação das Linguagens: Paradigma

- **Paradigma imperativo** (sequência, atribuição, estado): Basic, Pascal, C, Ada, Fortran, Cobol, Assembly...
- **Paradigma funcional** (função, aplicação, avaliação): Lisp, Haskell, Erlang, Scheme...
- **Paradigma lógico** (relação, dedução): Prolog.
- **Paradigma orientado a objetos** (objeto, estado, mensagem): C++, Java, C#, Eiffel, Smalltalk, Python...
- **Paradigma concorrente** (processo, comunicação (síncrona ou assíncrona)): C++, C#, Java...

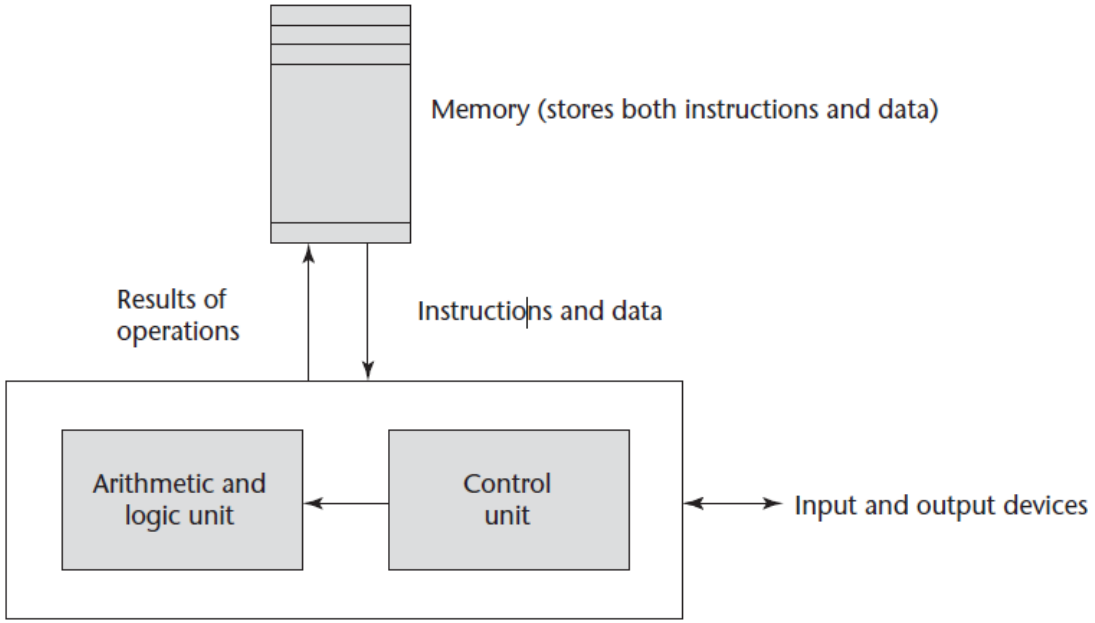
Classificação das Linguagens: Paradigma

1. **Paradigma Imperativo:**
   - As linguagens imperativas são orientadas a **ações**, onde a computação é vista 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.
   - Paradigma também denominado de **procedural**, por incluir subrotinas ou procedimentos como mecanismo de estruturação.

Classificação das Linguagens: Paradigma

1. Paradigma Imperativo:

- Baseia-se na arquitetura de computadores Von Neumann:
  - Programas e dados são armazenados na mesma memória;
  - Instruções e dados são transmitidos da CPU para a memória, e vice-versa;
  - Resultados das operações executadas na CPU são retornadas para a memória.



Classificação das Linguagens: Paradigma

1. **Paradigma Imperativo:**
   - Subdivide-se em **estruturado** e **não-estruturado**.
   - Linguagens não-estruturadas geralmente fazem uso de comandos goto ou jump. 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"
```

Classificação das Linguagens: Paradigma

1. **Paradigma Imperativo:**
   - As linguagens estruturadas surgiram objetivando facilitar a leitura e execução de algoritmos – **não fazem o uso do goto.**
     - 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...).

Classificação das Linguagens: Paradigma

1. **Paradigma Imperativo:**
   - 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;
}
```

Classificação das Linguagens: Paradigma

1. **Paradigma Imperativo:**
   - Linguagens estruturadas permitem a criação de procedimentos (**funções**);
   - **Procedimentos criam um nível de abstração**, onde não é necessário conhecer todos os passos de execução de um procedimento, apenas qual sua função e quais os pré-requisitos para sua execução correta;
   - **Linguagens estruturadas modulares** criam um outro mecanismo de abstração – módulo: composto de definições de variáveis e procedimentos, agrupados de acordo com critérios específicos.

Classificação das Linguagens: Paradigma

1. **Paradigma Imperativo:**
   - **Exemplos de Linguagens Imperativas:**
     - FORTRAN
     - BASIC
     - COBOL
     - Pascal
     - C
     - ALGOL
     - Modula
     - …

Classificação das Linguagens: Paradigma

1. **Paradigma Imperativo:**
   - **Vantagens:**
     - Eficiência;
     - Modelagem "natural" de aplicações do mundo real;
     - Paradigma dominante e bem estabelecido;
   - **Desvantagens:**
     - Possui difícil legibilidade e facilita introdução de erros em sua manutenção;
     - Descrições demasiadamente profissional focaliza o "como" e não o "quê“;
     - Tende a gerar códigos confusos, onde tratamento dos dados são misturados com o comportamento do programa;

Classificação das Linguagens: Paradigma

2. **Paradigma Funcional:**
   - Trata a computação como um processo de avaliação de **funções matemáticas**, evitando o uso de estados ou dados mutáveis;
   - Enfatiza a **aplicação de funções**, em contraste da programação imperativa, que enfatiza mudanças no estado do programa;
   - A visão funcional resulta num programa que descreve as operações que devem ser efetuadas para resolver o problema.

Classificação das Linguagens: Paradigma

2. **Paradigma Funcional:**
   - Programar em uma linguagem funcional consistem em pensar qual função deve ser aplicada para transformar uma entrada qualquer na saída desejada.

   - Ao invés dos passos sucessivos do paradigma imperativo, a sintaxe da linguagem é apropriada para definição de funções compostas que denotam aplicações sucessivas de funções:

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

Classificação das Linguagens: Paradigma

2. **Paradigma Funcional:**
   - **Exemplo:** Distancia entre dois pontos em C

```
#include <stdio.h>
#include <math.h>
float dist(float PX1, float PY1, float PX2, float PY2)
{
  float res = sqrt((PX2 - PX1)*(PX2 - PX1) +
                   (PY2 - PY1)*(PY2 - PY1));
  return res;
}
int main()
{
  float f = dist(2.0, 4.0, 3.0, 1.0);
  printf("Distance = %f", f);
  return 0;
}
```

Classificação das Linguagens: Paradigma

2. **Paradigma Funcional:**
   - **Exemplo:** Distancia entre dois pontos em Haskell

```
dist x1 y1 x2 y2 = sqrt(((x2 - x1)^2) + ((y2 - y1)^2))

main = print(dist 2.0 4.0 3.0 1.0)
```

   - Características da programação funcional:
     - Programas são funções que descrevem uma relação explícita e precisa entre E/S;
     - Estilo declarativo: não há o conceito de estado nem comandos como atribuição;

Classificação das Linguagens: Paradigma

2. **Paradigma Funcional:**
   - **Exemplos de Linguagens Funcionais:**
     - Haskell;
     - Scheme;
     - Common LISP;
     - CLOS (Common LISP Object System);
     - Miranda;
     - ML;
     - Erlang;
     - Ocaml;
     - …

Classificação das Linguagens: Paradigma

2. **Paradigma Funcional:**
   - **Vantagens:**
     - Simplifica a resolução de alguns tipos problemas:
       - Prova de propriedades;
       - Resolução de programas de otimização;
   - **Desvantagens:**
     - Problema: o mundo não é funcional!
     - Implementações ineficientes;
     - Mecanismos primitivos de E/S;

Classificação das Linguagens: Paradigma

3. **Paradigma Lógico:**
   - Paradigma de programação baseado em **lógica formal**;
   - Um programa lógico é equivalente à descrição do problema expressa de maneira formal, similar à maneira que o ser humano raciocinaria sobre ele;
   - A programação lógica consiste em declarar **fatos**, que podem ser **relações** (associações) ou **regras** que produzem fatos deduzidos a partir de outros.

Classificação das Linguagens: Paradigma

3. **Paradigma Lógico:**
   - Programação em linguagens lógicas requer um **estilo mais descritivo**;
   - O programador deve conhecer os **relacionamentos** entre as **entidades** e **conceitos** envolvidos para descrever os fatos relacionados ao problema;
   - Programas descrevem um **conjunto de regras** que disparam ações quando suas premissas são satisfeitas;
   - Principal linguagem lógica: **Prolog**

Classificação das Linguagens: Paradigma

3. Paradigma Lógico:

– Exemplo Prolog:

```
tropical(caribe).
tropical(havai).
praia(caribe).
praia(havai).
bonito(havai).
bonito(caribe).
paraiso_tropical(X) :- tropical(X), praia(X), bonito(X).
```

```
?- paraiso_tropical(X).
```

Compilador Online: http://www.tutorialspoint.com/execute_prolog_online.php

Classificação das Linguagens: Paradigma

3. Paradigma Lógico:

- Exemplo Prolog:

```
pai(fred, marcos).
pai(ricardo, pedro).
pai(pedro, paulo).
avo(X,Y) :- pai(X, Z), pai(Z, Y).
```

```
?- avo(X, paulo).
```

Compilador Online: http://www.tutorialspoint.com/execute_prolog_online.php

Classificação das Linguagens: Paradigma

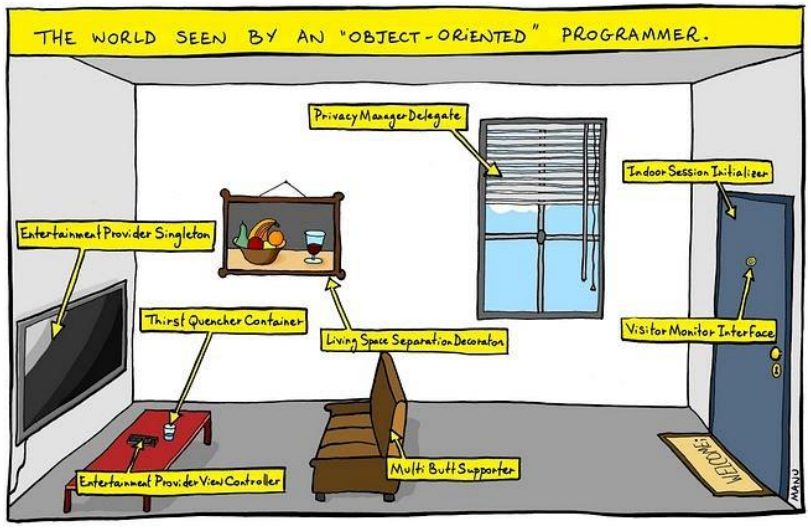
3. Paradigma Lógico:

- **Vantagens:**
  - Permite a concepção da aplicação em alto nível de abstração;
  - Linguagem mais próxima do raciocínio humano;

- **Desvantagens:**
  - Dificuldade em expressar algoritmos complexos;
  - Complexidade exponencial;

Classificação das Linguagens: Paradigma

4. **Paradigma Orientado a Objetos:**
   - Tratam os elementos e conceitos associados ao problema como **objetos**;
   - Objetos são entidades abstratas que embutem dentro de suas fronteiras, as **características e operações** relacionadas com a entidade real;



Classificação das Linguagens: Paradigma

4. **Paradigma Orientado a Objetos:**
   - Sugere a diminuição 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**;
   - Sistemas são vistos como **coleções de objetos** que se **comunicam**, enviando mensagens, colaborando para dar o comportamento global dos sistemas.

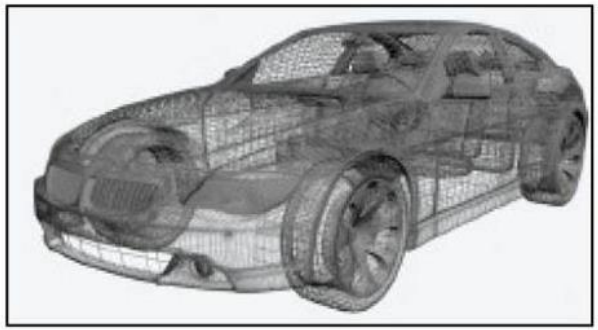
Classificação das Linguagens: Paradigma

4. Paradigma Orientado a Objetos:

- Uma aplicação é estruturada em módulos (**classes**) que agrupam um estado (**atributos**) e operações (**métodos**) sobre este;

- A classe é o **modelo** ou **molde** de construção de **objetos**. Ela define as características e comportamentos que os objetos irão possuir.

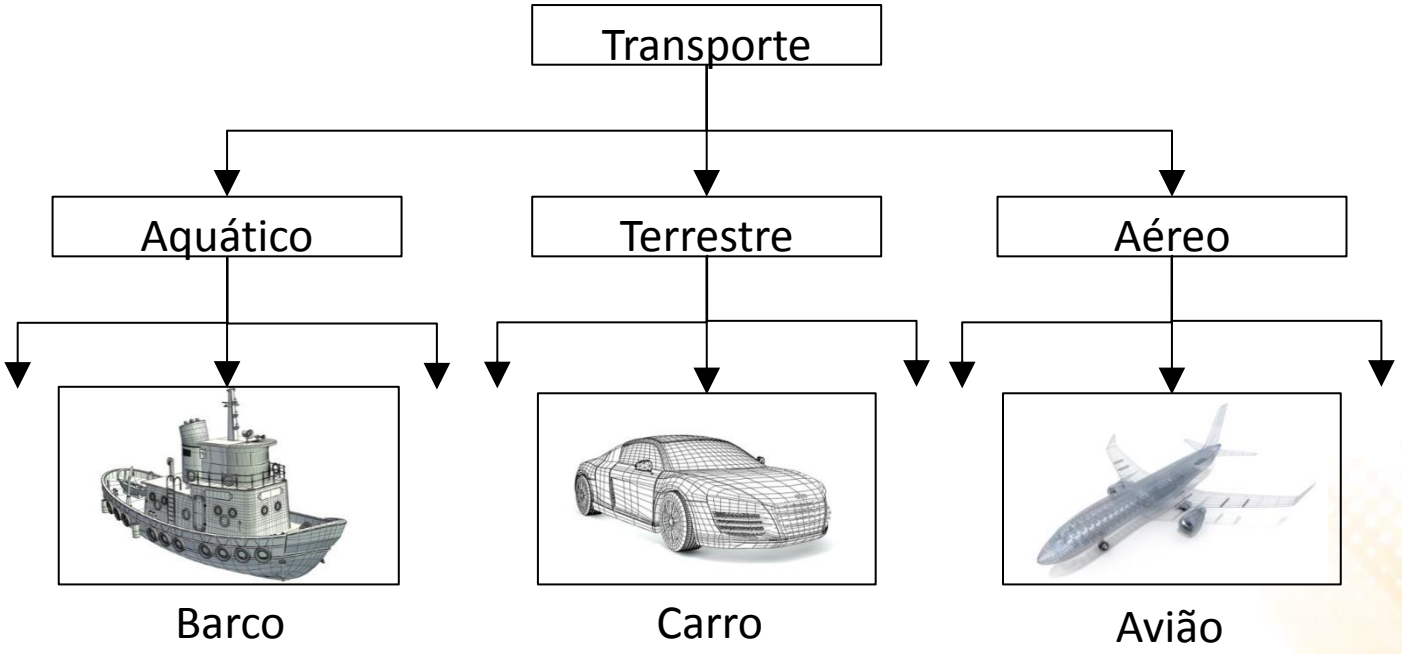
Classe Carro



Objeto Carro2

Classificação das Linguagens: Paradigma

4. **Paradigma Orientado a Objetos:**
   - A orientação a objetos permite que classes possam "herdar" as características e métodos de outra classe para expandi-la ou especializá-la de alguma forma.



Classificação das Linguagens: Paradigma

4. Paradigma Orientado a Objetos:

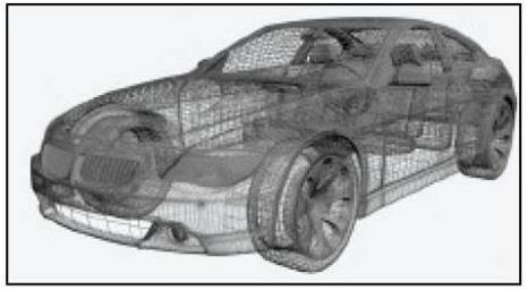
– Exemplo em Java:

```
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 ...

Classificação das Linguagens: Paradigma

4. **Paradigma Orientado a Objetos:**
   - **Exemplos de Linguagens Orientadas a Objetos:**
     - SIMULA 67;
     - Smalltalk;
     - C++;
     - Java;
     - C#;
     - ADA;
     - Eiffel;
     - Perl;
     - Ruby;
     - PHP;
     - …

Classificação das Linguagens: Paradigma

4. Paradigma Orientado a Objetos:

- **Vantagens:**
  - Organização do código;
  - Aumenta a reutilização de código;
  - Reduz tempo de manutenção de código;
  - Ampla utilização comercial;

- **Desvantagens:**
  - Menos eficientes;