Linguagens de Programação

Aula 2

Prof. Bonfim Amaro Junior
bonfimamaro@ufc.br

Relembrando...

- Aspectos preliminares das linguagens de programação
  - Capítulo 1 – Livro: Conceitos de linguagens de programação - Sebesta

- Introdução; Objetivos; Definição e Razões para estudar conceitos de LP

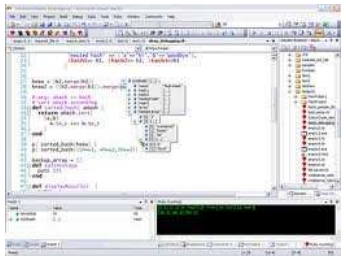
- Domínios de Programação (científicas, comerciais, IA, desenvolvimento de sistemas, Web, Scripts...) e Critérios de Avaliação de uma LP (legibilidade, capacidade de escrita, confiabilidade e custo).

Roteiro

- Introdução; Objetivos; Definição e Razões para estudar os conceitos de LP
- Domínios de Programação; Critérios de Avaliação;
- **Histórico da Evolução das Linguagens de Programação: tradução (interpretação e compilação); Categorias de LP e Classificação de LP**
- Os Paradigmas: Paradigma Imperativo, Orientado a Objeto, Funcional e Lógico; principais representantes de cada um dos paradigmas

Histórico das LP

- Década de 80: **modularização**
  - Ênfase em mecanismos de LP e abstrações
  - Correção de programas: verificação de tipos, exceções
  - Programação concorrente e distribuída e tempo real
  - Programação baseada em objetos (TADs)
  - Programação orientada a objetos (herança)



- Exemplos de linguagens
  - Uso acadêmico: Pascal / Modula
  - Programação de tempo real: Ada 83
  - Orientada a objetos: Smalltalk

Histórico das LP

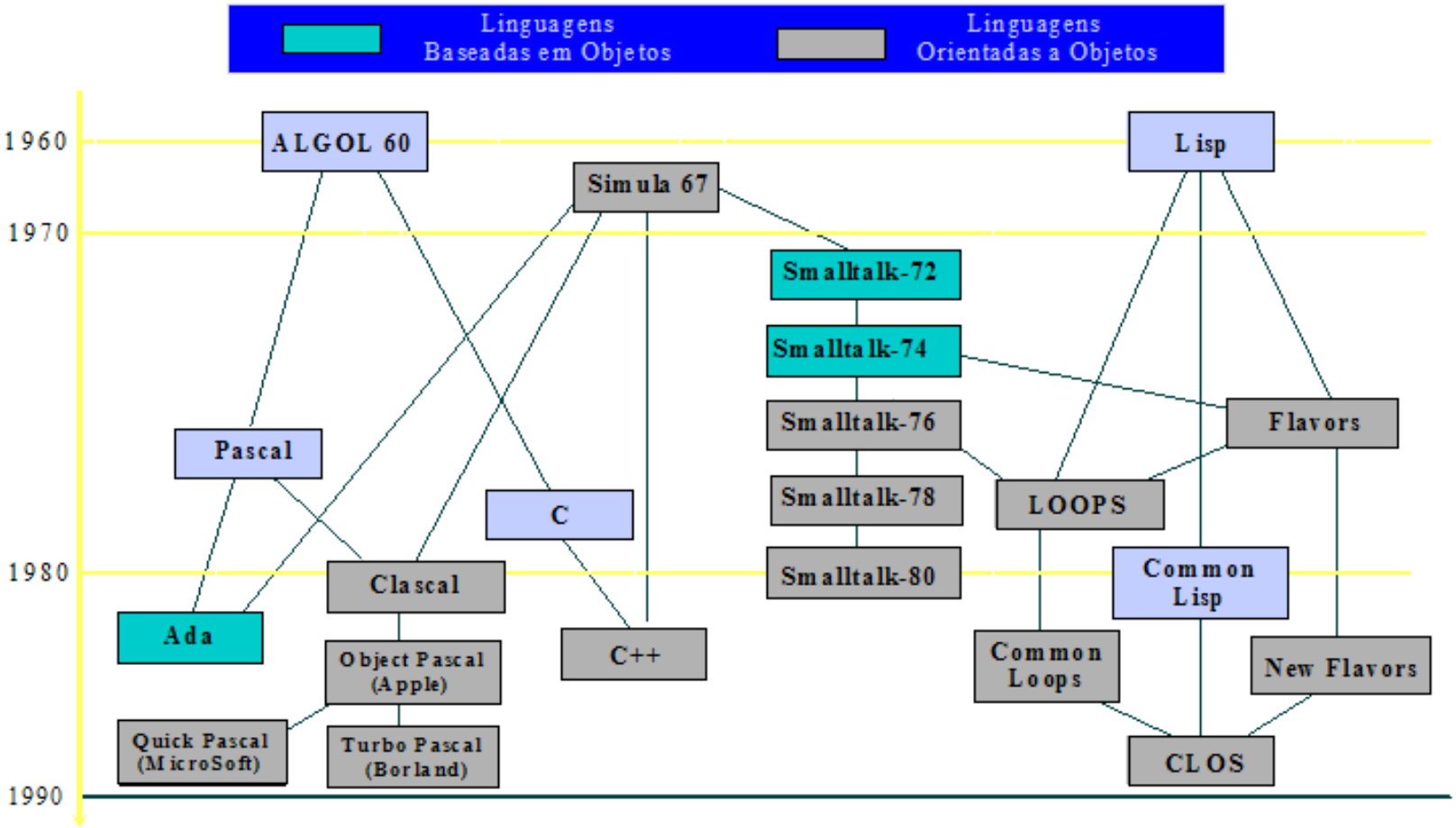
- Década de 90: **base na estrutura**
  - Estruturação de dados: encapsulamento
  - Estruturação da computação: classe
  - Estruturação do programa: classes e objetos
  - Programação para Internet: plataforma neutra

- Exemplos de linguagens
  - Pascal / Object Pascal
  - C / C++
  - Ada83 / Ada95
  - Java

Ada
The Language For A Complex World

Java

Quadro histórico das LP



Classificação das LP

1) Quanto ao nível

2) Quanto à geração

3) Quanto ao paradigma

Classificação das LP
1) quanto ao nível

1.3) Alto nível

1.2) Médio nível

1.1) Baixo nível

Classificação das LP
1) quanto ao nível

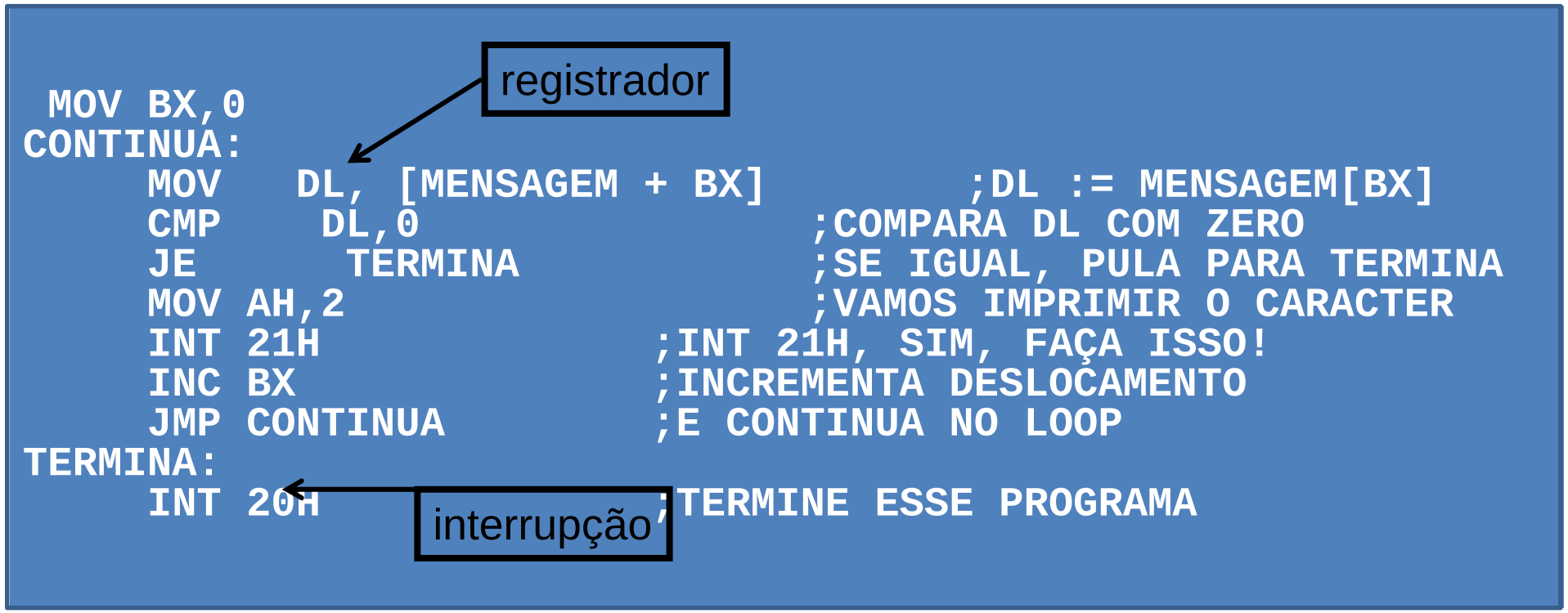
1.1) baixo nível

- As linguagens de Baixo Nível são aquelas voltadas para a máquina, ou seja as que são escritas utilizando as instruções do microprocessador do computador.

- São genericamente chamadas de linguagens *Assembly*. Os programas escritos com Alto Nível geralmente podem ser convertidos com programas especiais para Baixo Nível.

Classificação das LP
1) quanto ao nível

1.1) baixo nível




```
 MOV BX,0
CONTINUA:
     MOV   DL, [MENSAGEM + BX]          ;DL := MENSAGEM[BX]
     CMP    DL,0                      ;COMPARA DL COM ZERO
     JE      TERMINA                  ;SE IGUAL, PULA PARA TERMINA
     MOV AH,2                         ;VAMOS IMPRIMIR O CARACTER
     INT 21H                  ;INT 21H, SIM, FAÇA ISSO!
     INC BX                   ;INCREMENTA DESLOCAMENTO
     JMP CONTINUA             ;E CONTINUA NO LOOP
TERMINA:
     INT 20H                  ;TERMINE ESSE PROGRAMA
```

Classificação das LP
1) quanto ao nível

1.1) baixo nível

- **Vantagens:** Programas são executados com maior velocidade de processamento. Os programas ocupam menos espaço na memória.

- **Desvantagens:** Em geral, programas em *Assembly* tem pouca portabilidade, isto é, um código gerado para um tipo de processador não serve para outro. Códigos *Assembly* não são estruturados, tornando a programação mais difícil.

Classificação das LP
1) quanto ao nível

1.2) médio nível

- São linguagens voltadas ao ser humano e a máquina
  - Estas linguagens são uma **mistura** entre as linguagens de **Alto Nível** e as de **Baixo Nível**.

- Estas LP contêm comandos muito simples e outros muito complicados, o que torna a sua programação "complicada".

Classificação das LP
1) quanto ao nível

1.2) médio nível

- Ex: linguagem C
  - Pode-se acessar aos registros do sistema, trabalhar com endereços de memória - **características de linguagens de baixo nível** - e ao mesmo tempo **realizar operações de alto nível (if...else; while; for)**

Classificação das LP
1) quanto ao nível

1.2) médio nível

- **Vantagens:** Geralmente são linguagens mais poderosas, permitindo a criação de diversos softwares, desde jogos a programas com qualidade profissional.

- **Desvantagens:** Alguns comandos têm uma sintaxe muito difícil de compreender.

Classificação das LP
1) quanto ao nível

1.3) alto nível

- São linguagens voltadas para o ser humano. Em geral utilizam sintaxe mais estruturada tornando o seu código mais fácil de entender e de editar programas.

- Trata-se de linguagens independentes da arquitetura do computador.
  - um programa escrito em uma linguagem de alto nível, pode ser migrado de uma máquina a outra sem nenhum tipo de problema.

- Estas linguagens **permitem ao programador se esquecer completamente do funcionamento interno da máquina** para a que está desenvolvendo o programa. Somente necessita de um tradutor que entenda o código fonte como as características da máquina.

Classificação das LP
1) quanto ao nível

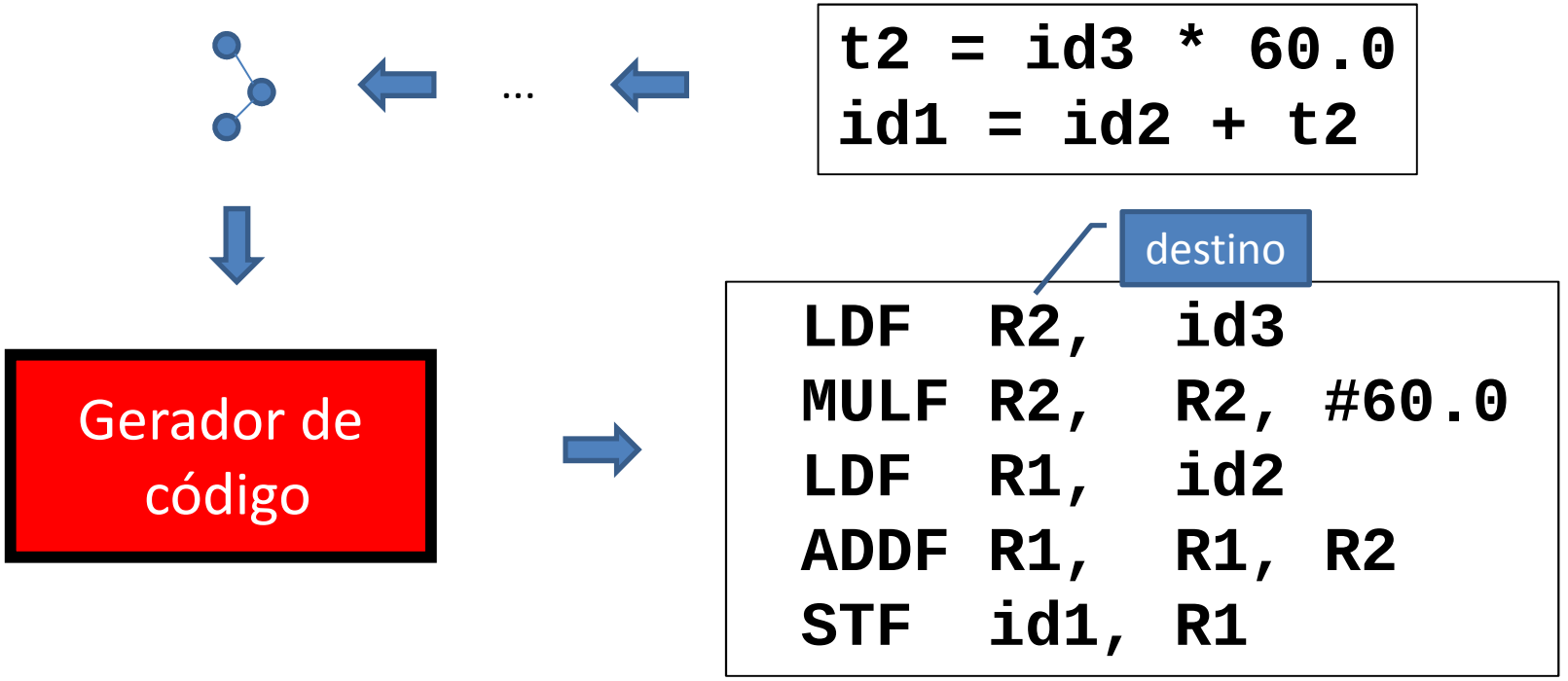
1.3) alto nível

- **Vantagens:** Por serem compiladas ou interpretadas, têm maior portabilidade, podendo ser executados em várias plataformas com pouquíssimas modificações. Em geral, a programação torna-se mais fácil por causa do maior ou menor grau de estruturação de suas linguagens.

- **Desvantagens:** Em geral, as rotinas geradas (em linguagem de máquina) são mais genéricas e, portanto, mais complexas e por isso são mais lentas e ocupam mais memória.

Classificação das LP
1) quanto ao nível

1.3) alto nível – Exemplo: conversão para baixo nível



Classificação das LP
2) quanto à geração

2.1) 1ª Geração

2.2) 2ª Geração

2.3) 3ª Geração

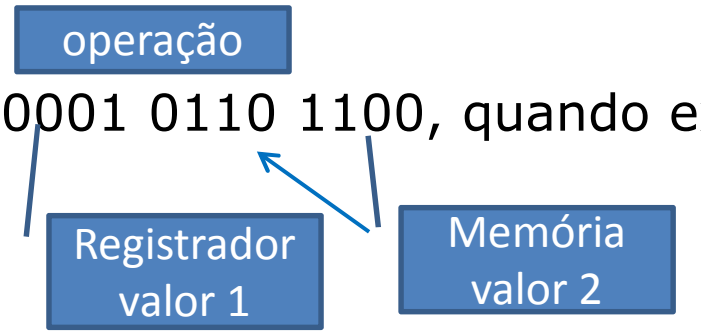
2.4) 4ª Geração

2.5) 5ª Geração

Classificação das LP
2) quanto à geração

2.1) 1ª Geração: linguagens em nível de máquina

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



- A instrução 0010 0001 0110 1100, quando executada, realiza a soma:
  - (código de operação 0010) do dado armazenado no registrador 0001, com o dado armazenado na posição de memória 108 (01101100).

Classificação das LP
2) quanto à geração

2.1) 1ª Geração: linguagens em nível de máquina

- Cada instrução de máquina é, em geral, formada por um código de operação e um ou dois endereços de registradores ou de memória.

- As linguagens de máquina permitem a comunicação direta com o computador em termos de "bits", registradores e operações de máquina bastante primitivas.

- Um programa em linguagem de máquina nada mais é que uma sequência de zeros e uns, a programação de um algoritmo complexo usando esse tipo de linguagem é complexa, cansativa e fortemente sujeita a erros.

Classificação das LP
2) quanto à geração

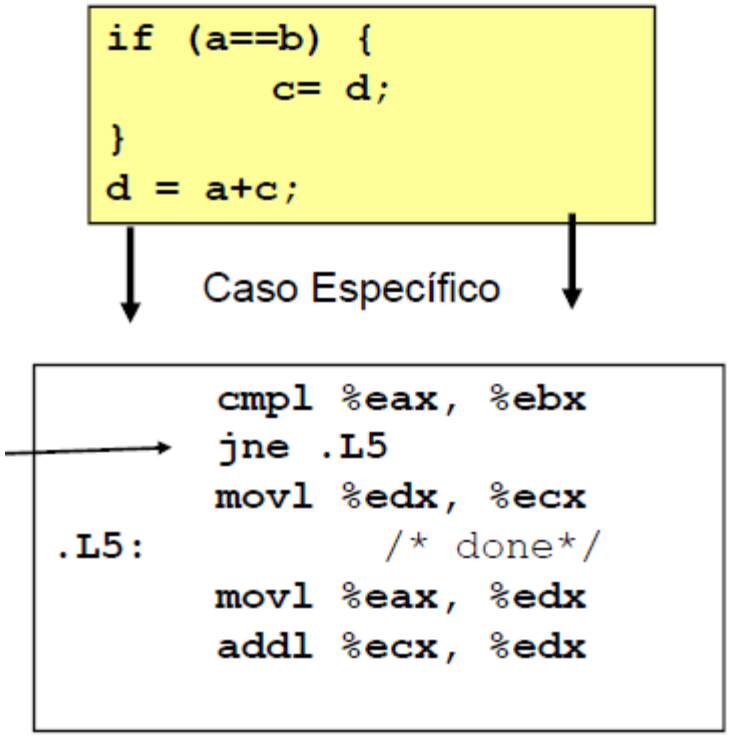
2.2) 2ª Geração: linguagens de montagem (*Assembly*)

- Compreende as linguagens simbólicas ou 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

Classificação das LP
2) quanto à geração

2.2) 2ª Geração: linguagens de montagem (*Assembly*). Exemplo: tradução do IF para *Assembly*



Classificação das LP
2) quanto à geração

2.2) 2ª Geração: linguagens de montagem (*Assembly*)

mov mul add label goto

- Nas linguagens de montagem, a maioria das instruções são representações simbólicas de instruções de máquina. O processamento de um programa em linguagem simbólica requer tradução para linguagem de máquina antes de ser executado.

Classificação das LP
2) quanto à geração

2.2) 2ª Geração: linguagens de montagem (*Assembly*)

- Códigos de operação e endereços binários foram substituídos por **abreviações**. Assim, a instrução de máquina (0010 0001 0110 1100) evoluiu para:

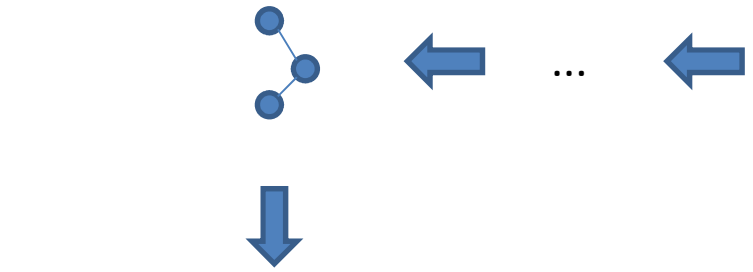
ADD R1, TOTAL

- R1 representa o registrador 1 e TOTAL é o nome atribuído ao endereço de memória 108.

Classificação das LP
2) quanto à geração

2.2) 2ª Geração: linguagens de montagem (*Assembly*)

- Exemplo 2 em Assembly




```
t2 = id3 * 60.0
id1 = id2 + t2
```

Código *Assembly*

```
LDF   R2,   id3
MULF  R2,   R2, #60.0
LDF   R1,   id2
ADDF  R1,   R1, R2
STF   id1,  R1
```

Classificação das LP
2) quanto à geração

2.2) 2ª Geração: linguagens de montagem (*Assembly*)

- **Curiosidade:** um projeto Delphi com 22 linhas de código revertido para *Assembly*, terá aproximadamente 15.000 linhas!

Classificação das LP
2) quanto à geração

2.3) 3ª Geração: linguagens orientadas ao usuário

- Surgiram na década de 60.

- Algumas delas são orientadas à solução de problemas científicos, tais como FORTRAN, PASCAL e ALGOL; outras, tal como COBOL, são usadas para aplicações comerciais.

- Linguagens como PL/I e ADA contêm facilidades para ambos os tipos de computações (científica e comercial).

Classificação das LP
2) quanto à geração

2.3) 3ª Geração: linguagens orientadas ao usuário

- Podem também ser classificadas em:
  - linguagens procedimentais
    - também chamadas "procedurais" ou imperativas
  - linguagens declarativas

Classificação das LP
2) quanto à geração

2.3) 3ª Geração: linguagens orientadas ao usuário

- Nas linguagens procedimentais, um programa especifica um procedimento, isto é, uma sequência de passos a serem seguidos para solucionar um problema.

- As instruções oferecidas por essas linguagens pertencem, em geral, a três classes:
  - instruções entrada/saída,
  - instruções de cálculos aritméticos ou lógicos
  - instruções de controle de fluxo de execução (desvios condicionais, incondicionais e processamento iterativo)

Classificação das LP
2) quanto à geração

2.3) 3ª Geração: linguagens orientadas ao usuário

- Exemplos de linguagens orientadas ao usuário: BASIC, ALGOL, PL/I, PASCAL, ADA, C, etc. Ex: Programa em Basic:

```
10 INPUT A,B,C
20 LET SOMA = A+B+C
30 LET MEDIA = SOMA/3
40 PRINT MEDIA
50 PRINT "DESEJA CONTINUAR (S/N)?"
60 INPUT RESPOSTA
70 IF RESPOSTA = "S"THEN 10
80 END
```

Controle de fluxo

Classificação das LP
2) quanto à geração

2.3) 3ª Geração: linguagens orientadas ao usuário

- Chamada também de linguagens declarativas dividem-se, basicamente, em duas classes:
  - **funcionais**, as quais se baseiam na teoria das funções recursivas
  - **lógicas**, cuja base é a lógica matemática.

Classificação das LP
2) quanto à geração

2.3) 3ª Geração: linguagens orientadas ao usuário

- A programação funcional envolve, essencialmente, a definição e a chamada de funções. **LISP** é o exemplo mais difundido de linguagem funcional.

- Nas linguagens lógicas, um programa declara **fatos** (dados e relações entre eles) e cláusulas lógicas (**regras** de dedução), que permitirão deduzir novas verdades a partir dos fatos conhecidos. Ex. **PROLOG**

Classificação das LP
2) quanto à geração

2.4) 4ª Geração: linguagens orientadas à aplicação

- As linguagens de 3ª geração foram projetadas para profissionais de processamento de dados e não para usuários finais.

- A depuração de programas escritos nessas linguagens consome tempo, e a modificação de sistemas complexos é relativamente difícil.

- As linguagens de 4ª geração foram projetadas em resposta a esses problemas.

Classificação das LP
2) quanto à geração

2.4) 4ª Geração: linguagens orientadas à aplicação

- Os programas escritos em linguagens de 4ª geração necessitam de menor número de linhas de código do que os programas correspondentes codificados em linguagens de programação convencionais.

- Ex. mostrar uma imagem bmp em CA-dBFast:

```
LOAD IMAGE C:\multimid\LAB1.BMP INTO pIMAGE
@  0,28 SAY pIMAGE
```

Classificação das LP
2) quanto à geração

- Conexão com BD em Java

```
public Connection getConnection() {
 try {
  if (con == null) {
   Class.forName(jdbcDriver);
   con = DriverManager.getConnection(url, userName, password);
  } else if (con.isClosed()) {
   con = null;
   return getConnection();
  }
 } catch (ClassNotFoundException e) {
  //TODO: use um sistema de log apropriado.
  e.printStackTrace();
 } catch (SQLException e) {
  //TODO: use um sistema de log apropriado.
  e.printStackTrace();
 }
```

Classificação das LP
2) quanto à geração

2.4) 4ª Geração: linguagens orientadas à aplicação

- As linguagens de 4ª geração variam bastante no número de facilidades oferecidas ao usuário.
- Algumas são, meramente, geradores de relatórios ou pacotes gráficos;
- Outras são capazes de gerar aplicações completas.
- Em geral, essas linguagens são projetadas para atender a classes específicas de aplicações.

Classificação das LP
2) quanto à geração

2.4) 4ª Geração: linguagens orientadas à aplicação

- **Principais objetivos:**
  - Facilitar a programação de computadores 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.

Classificação das LP
2) quanto à geração

2.4) 4ª Geração: linguagens orientadas à aplicação

- Exemplos de linguagens de 4ª geração:
  - LOTUS 1-2-3, SQL, SUPERCALC, VISICALC, DATATRIEVE, VHML, PHP

- Comando em dBase III Plus:

```
LIST ALL NOME, ENDERECO, TELEFONE FOR CIDADE =
                "PRESIDENTE PRUDENTE"
```

Classificação das LP
2) quanto à geração

2.5) 5ª Geração: linguagens do conhecimento

- São usadas principalmente na área de Inteligência Artificial.
- Facilitam a representação do conhecimento que é essencial para a simulação de comportamentos inteligentes.

Classificação das LP
2) quanto à geração

2.5) 5ª Geração: linguagens do conhecimento

- O termo 5ª geração refere-se, especialmente, a sistemas que usam mecanismos da área de inteligência artificial (IA), ou seja, sistemas especialistas, processadores de língua natural e sistemas com bases de conhecimento.

- Um sistema de 5ª geração armazena conhecimento complexo de modo que a máquina pode obter inferências a partir da informação codificada.

Roteiro

- Introdução; Objetivos; Definição e Razões para estudar os conceitos de LP
- Domínios de Programação; Critérios de Avaliação; Categorias de LP e Classificação de LP
- Histórico da Evolução das Linguagens de Programação: tradução (interpretação e compilação)
- **Os Paradigmas: Paradigma Imperativo, Orientado a Objeto, Funcional, Lógico e Concorrente; principais representantes de cada um dos paradigmas**

Classificação das LP
3) quanto ao paradigma

3.1) Imperativo

3.2) Funcional

3.3) Lógico

3.4) Orientado a Objetos

3.5) Concorrente

Definição (lembrando...)

- Uma LP (Linguagem de Programação) é uma linguagem destinada a ser usada por uma **pessoa** para expressar um **processo** através do qual um **computador** pode resolver um **problema**.
- Os quatro modelos (paradigmas) de LP correspondem aos pontos de vista dos quatro componentes citados.
- A eficiência na construção e execução de programas depende da combinação dos quatro pontos de vista.

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.**

Paradigma na programação

- Algumas linguagens criadas durante a história da evolução das linguagens introduziram novas formas de se pensar sobre programação, resultando em formas distintas de modelagem de soluções para problemas de software.
  - FORTRAN (imperativas)
  - LISP (funcionais)
  - Simula (orientadas a objetos)
  - Prolog (lógicas).

Paradigma na programação

- 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.

Paradigma na programação
exemplos

- **Paradigma imperativo (estado, atribuição, sequência)**

  Basic, Pascal, C, Ada, Assembly.

- **Paradigma funcional (função, aplicação, avaliação)**

  Lisp, ML, Miranda, Haskell, SCHEME.

- **Paradigma lógico (relação, dedução)**

  Prolog.

- **Paradigma orientado a objetos (objeto, mensagem)**

  C++, Java, Eiffel, Ocaml.

- **Paradigma concorrente (processo, comunicação (síncrona ou assíncrona))**

  Ada, Java.

Paradigma na programação
exemplos - combinação

- **C++**

  Paradigma imperativo + paradigma orientado a objetos.

- **Java**

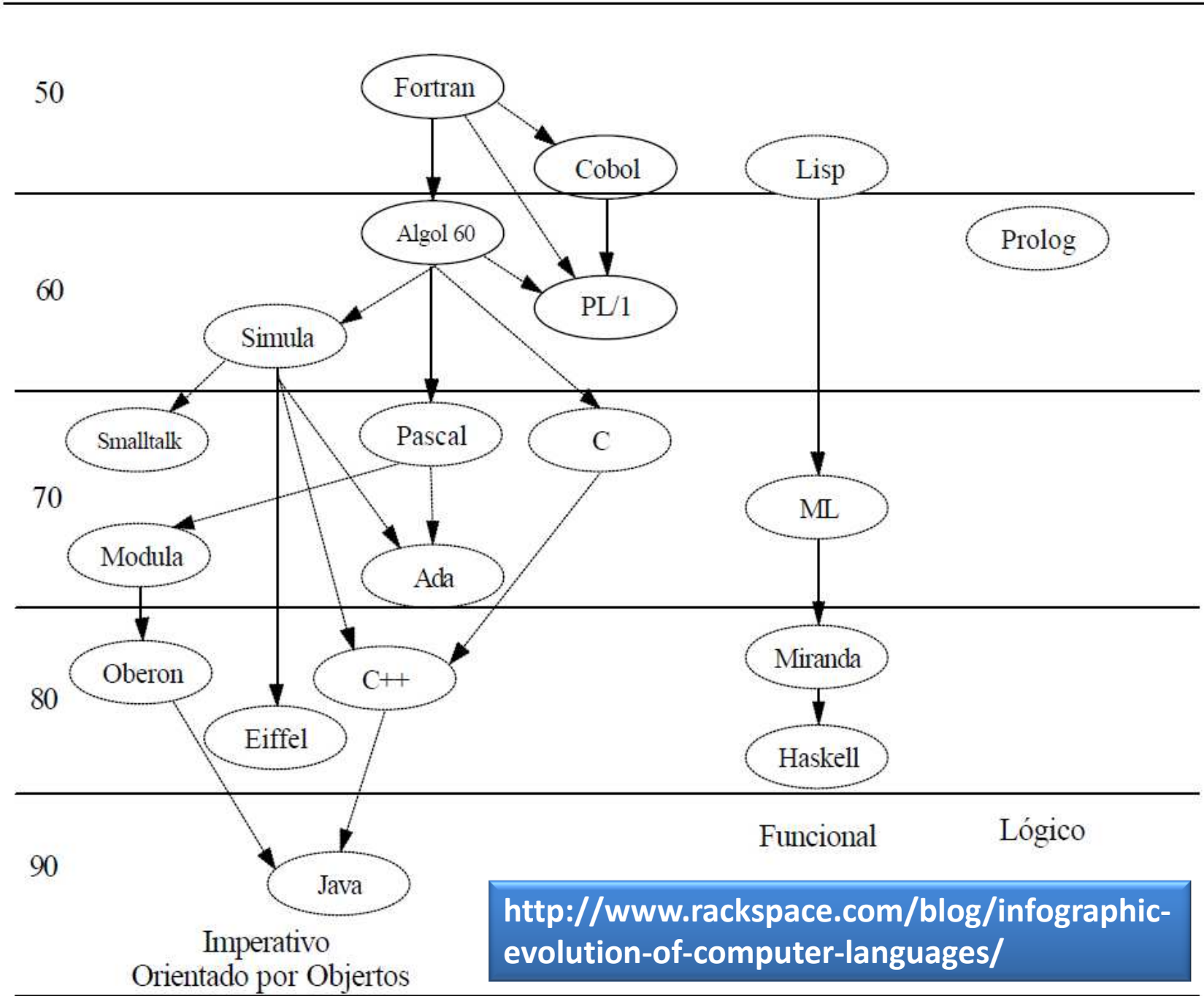
  Paradigma orientado a objetos + paradigma concorrente.

- **Ocaml**

  Paradigma funcional + paradigma orientado a objetos.

- **Ada**

  Paradigma imperativo + paradigma concorrente.



http://www.rackspace.com/blog/infographic-evolution-of-computer-languages/

Classificação das LP
3) quanto ao paradigma

3.1) 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).

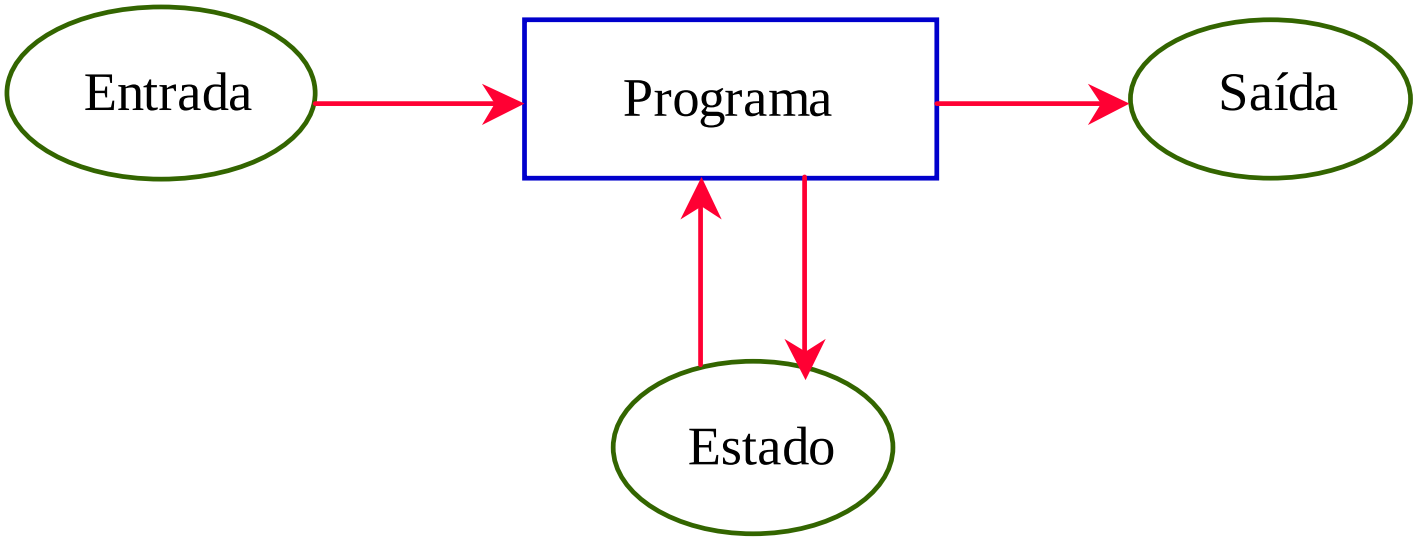
Classificação das LP
3) quanto ao paradigma

3.1) Imperativo

- Programas 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.
- Primeiro paradigma a surgir e ainda muito importante.

Classificação das LP
3) quanto ao paradigma

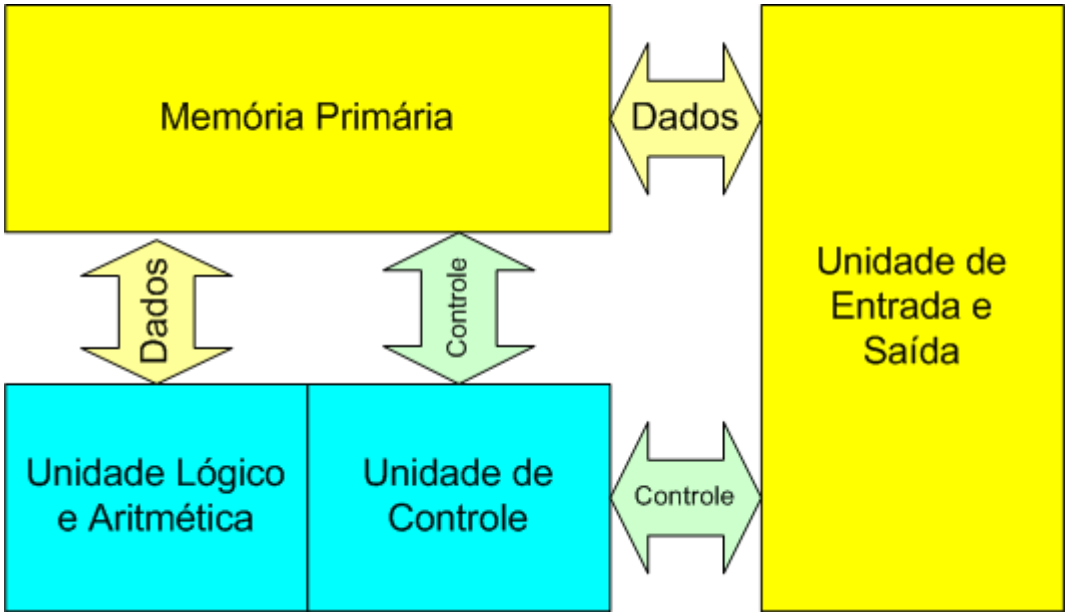
3.1) Imperativo: Modelo computacional



Classificação das LP
3) quanto ao paradigma

3.1) Imperativo

- Paradigma baseado na arquitetura de computadores *Von Neumann*



Classificação das LP
3) quanto ao paradigma

3.1) Imperativo

- 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 LP
3) quanto ao paradigma

3.1) Imperativo

- Subdivide-se em estruturado e não-estruturado
- Exemplo de LP **imperativa não-estruturada**

```
10 INPUT A,B,C
20 LET SOMA = A+B+C
30 LET MEDIA = SOMA/3
40 PRINT MEDIA
50 PRINT "DESEJA CONTINUAR (S/N)?"
60 INPUT RESPOSTA
70 IF RESPOSTA = "S"THEN GOTO 10
80 END
```

Exemplo em Basic

```
read(x);
2: if x = 0 then goto 8;
writeln(x);
4: read(next);
if next = x then goto 4;
x := next;
goto 2;
8: ...;
```

Exemplo em Pascal

Classificação das LP
3) quanto ao paradigma

3.1) Imperativa estruturada

- 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 - while

Classificação das LP
3) quanto ao paradigma

3.1) Imperativa estruturada

- Linguagens estruturadas permitem a criação de procedimentos (e funções)
- Blocos de instruções, que formam os elementos básicos de construção de programas
- 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 LP
3) quanto ao paradigma

3.1) Imperativa estruturada

```
read(x);
while x <> 0 do begin
 writeln(x);
 repeat
   read(next);
 until next <> x;
   x := next;
end;
```

Classificação das LP
3) quanto ao paradigma

3.1) Imperativo: visão crítica

- **Vantagens**
  - Eficiência (embute modelo de Von Neumann)
  - Modelagem "natural" de aplicações do mundo real
  - Paradigma dominante e bem estabelecido

- **Problemas**
  - Relacionamento indireto entre E/S resulta em:
    - Difícil legibilidade
    - Erros introduzidos durante manutenção
    - Descrições demasiadamente operacionais
    - Focalizam o como e não o quê

Classificação das LP
3) quanto ao paradigma

3.1) Exemplos de LP imperativa

- FORTRAN
- BASIC
- COBOL
- Python
- Pascal
- C
- ALGOL
- Modula

Classificação das LP
3) quanto ao paradigma

3.2) Funcional

- Sistemas são construídos através da **definição**, **composição** e definição **de funções**

- Foco no processo de resolução do problema. A visão funcional resulta num programa que descreve as operações que devem ser efetuadas para resolver o problema

Classificação das LP
3) quanto ao paradigma

3.2) Funcional

- Trata a programação como uma transformação de dados por funções

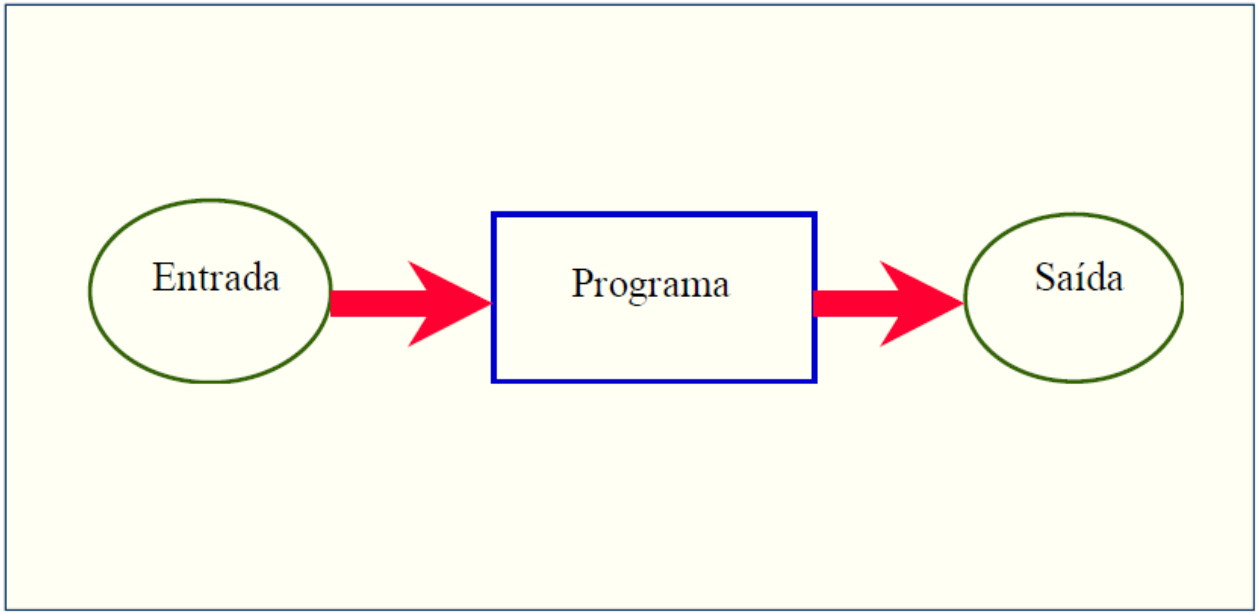
- Que função deve ser aplicada para transformar minha entrada na saída desejada?

- 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-n(...função2(função1(dados))...)

Classificação das LP
3) quanto ao paradigma

3.2) Funcional: modelo computacional



Classificação das LP
3) quanto ao paradigma

3.2) Funcional - características

- 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

- Conceitos sofisticados como polimorfismo, funções de alta ordem e avaliação sob demanda

- Aplicação: prototipação em geral e IA

Classificação das LP
3) quanto ao paradigma

3.2) Exemplos de LP Funcional

- **Scheme**
  - Derivada de LISP
- **Common LISP**
  - Buscando padronização de LISP
- **CLOS (Common LISP Object System)**
  - Dialeto de LISP que incorpora elementos de linguagens orientadas a objetos
- **ML**
  - Linguagem funcional fortemente tipada
- **Miranda**
  - Baseada em ML, mas é puramente funcional

Classificação das LP
3) quanto ao paradigma

3.2) Exemplos de LP Funcional HASKELL

```
1  fibonacci 0 = 1
2  fibonacci 1 = 1
3  fibonacci n = fibonacci (n-1) + fibonacci (n-2)
4
5  fatorial 0 = 1
6  fatorial n = n*fatorial(n-1)
```

```
*Main> fibonacci 5
8
*Main> fatorial 5
120
*Main> fibonacci (fatorial 3)
13
```

Classificação das LP
3) quanto ao paradigma

3.2) Funcional – visão crítica

Vantagens	Desvantagens
• Manipulação de programas mais simples - prova de propriedades - transformação (exemplo: otimização) • Concorrência explorada de forma natural	• Problema: o mundo não é funcional!! • Implementações ineficientes • Mecanismos primitivos de E/S e formatação

Classificação das LP
3) quanto ao paradigma

3.3) Lógico

- O modelo Lógico está relacionado à perspectiva da pessoa: ele encara o problema de uma perspectiva lógica. Um programa lógico é equivalente à descrição do problema expressa de maneira formal, similar à maneira que o ser humano raciocinaria sobre ele.

- Programação é baseada em fatos, que podem ser relações (associações) entre coisas, e regras, que produzem fatos deduzidos a partir de outros.

Classificação das LP
3) quanto ao paradigma

3.3) Lógico

- Programação em linguagens declarativas (lógica) requer um estilo mais descritivo.

- O programador deve conhecer os relacionamentos entre as entidades e conceitos envolvidos para descrever os fatos (cláusulas).

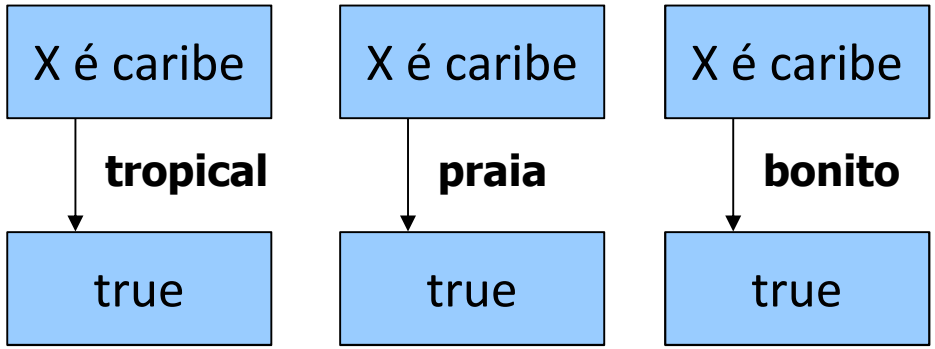
- Programas descrevem um conjunto de regras que disparam ações quando suas premissas são satisfeitas.

- Prolog foi desenvolvida em 1972 para processamento de linguagem natural, na França. O nome vem de **PRO**grammation en **LOG**ique.

Classificação das LP
3) quanto ao paradigma

3.3) Lógico - exemplo

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

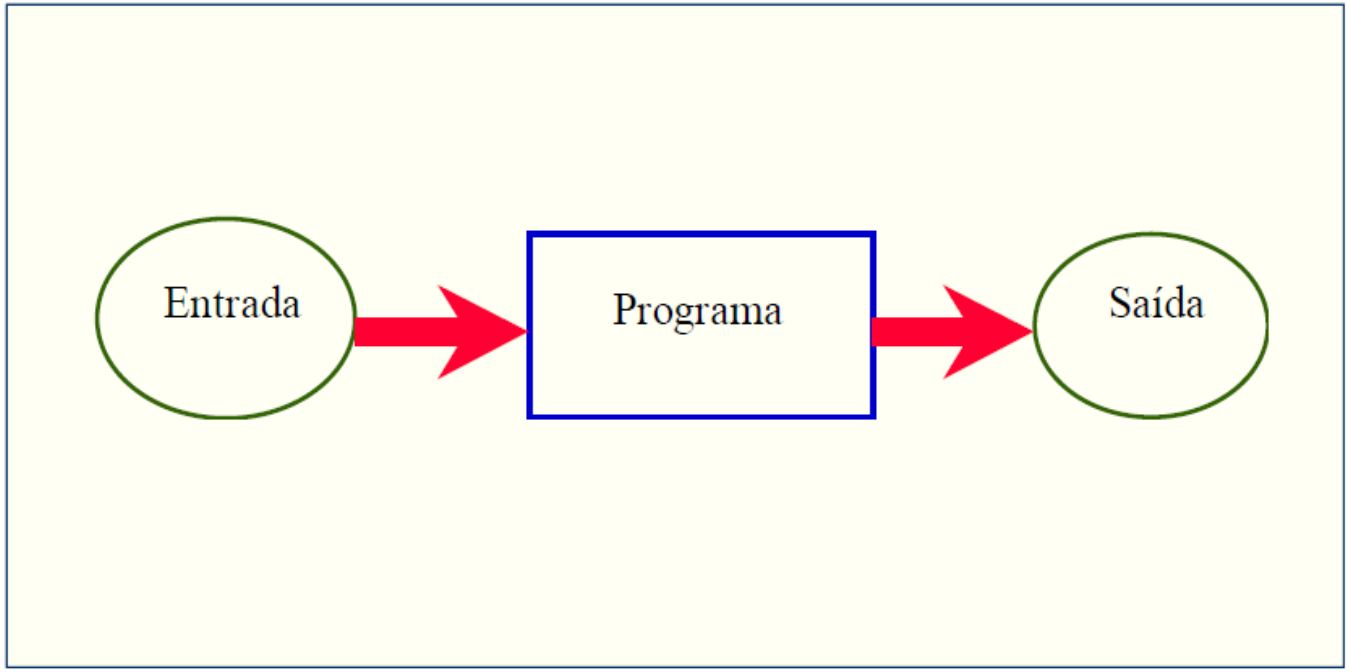


- *Questionando...*

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

Classificação das LP
3) quanto ao paradigma

3.3) Lógico: modelo computacional



Classificação das LP
3) quanto ao paradigma

3.3) Características do paradigma Lógico

- Programas são relações entre E/S
- Estilo declarativo
- Aplicações:
  - Sistemas especialistas (IA)
  - Banco de dados

Classificação das LP
3) quanto ao paradigma

3.3) Lógico: Exemplo em Prolog

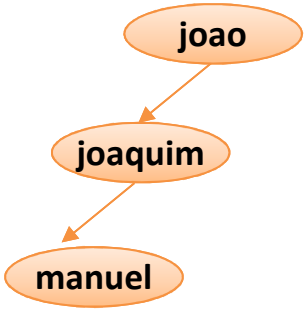
```
pai(joao, joaquim).
pai(joaquim, manuel).

?- pai(joao,Z).
Z = joaquim

?- pai(joao,manuel).
no.
```

Processo de resolução

pai(joao,Z).

No fato pai(joao,Z) a variável Z foi unificada a “joaquim”

No fato pai(joao,manuel) a variável **joao** foi unificada a “joao” e a variável **manuel** não foi unificada a “joaquim”



Classificação das LP
3) quanto ao paradigma

3.3) Lógico: visão crítica

Vantagens	Desvantagens
• Em princípio, todas do paradigma funcional • Permite a concepção da aplicação em alto nível de abstração (através de associações entre E/S)	• Em princípio, todas do paradigma funcional • Linguagens usualmente não possuem tipos, nem são de alta ordem

Classificação das LP
3) quanto ao paradigma

3.4) 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 LP
3) quanto ao paradigma

3.4) Orientado a objetos

- O modelo Orientado a Objeto focaliza mais o problema. Um programa OO é equivalente a objetos que mandam mensagens entre si. Os objetos do programa equivalem aos objetos da vida real (problema).

- A abordagem OO é importante para resolver muitos tipos de problemas através de simulação.

Classificação das LP
3) quanto ao paradigma

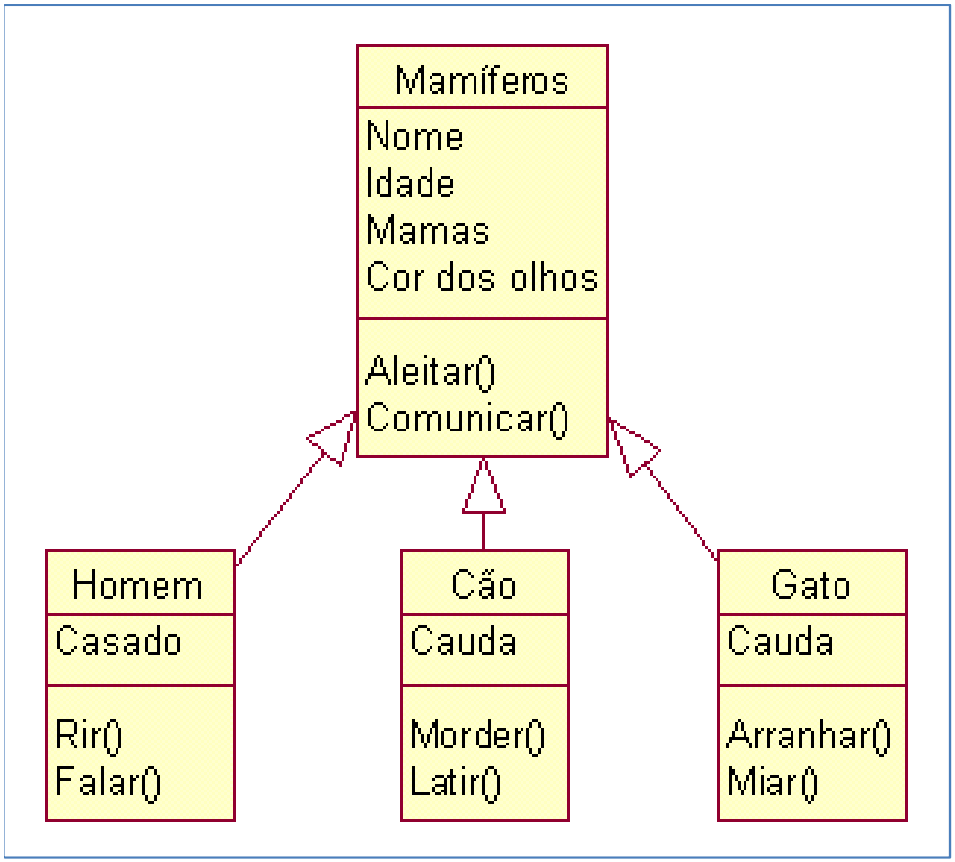
3.4) Orientado a objetos

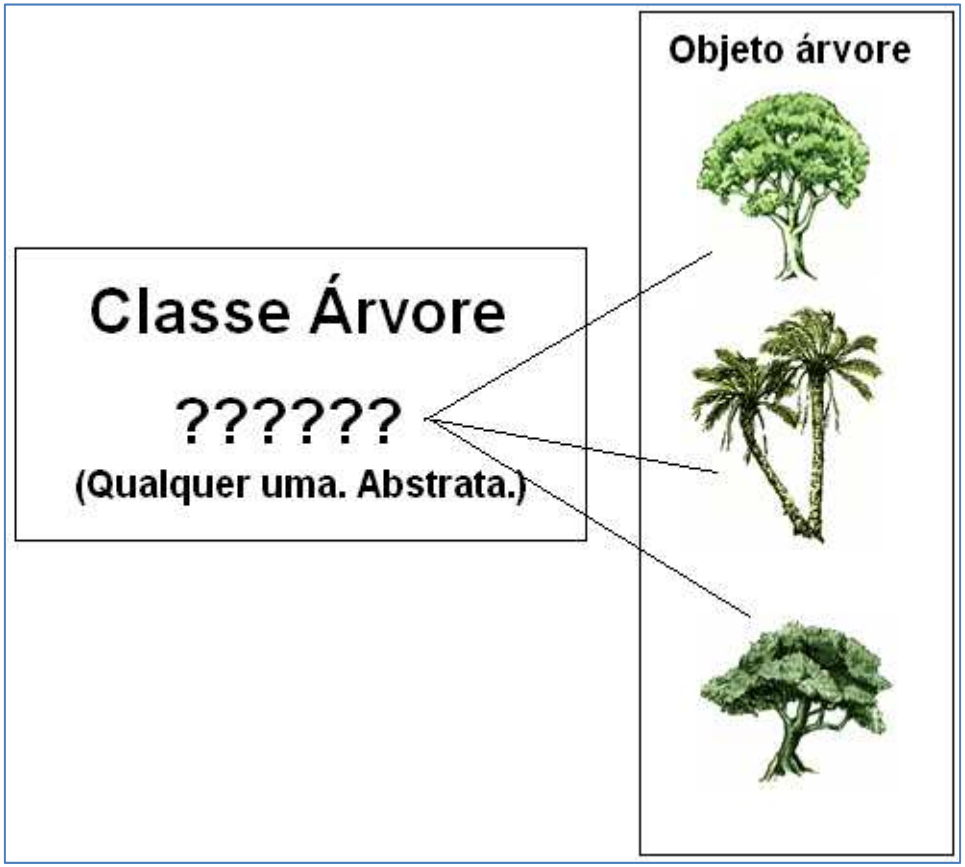
- A grosso modo, uma aplicação é estruturada em módulos (classes) que agrupam um estado e operações (métodos) sobre este.

- Classes podem ser estendidas e/ou usadas como tipos (cujos elementos são objetos).

Classificação das LP
3) quanto ao paradigma

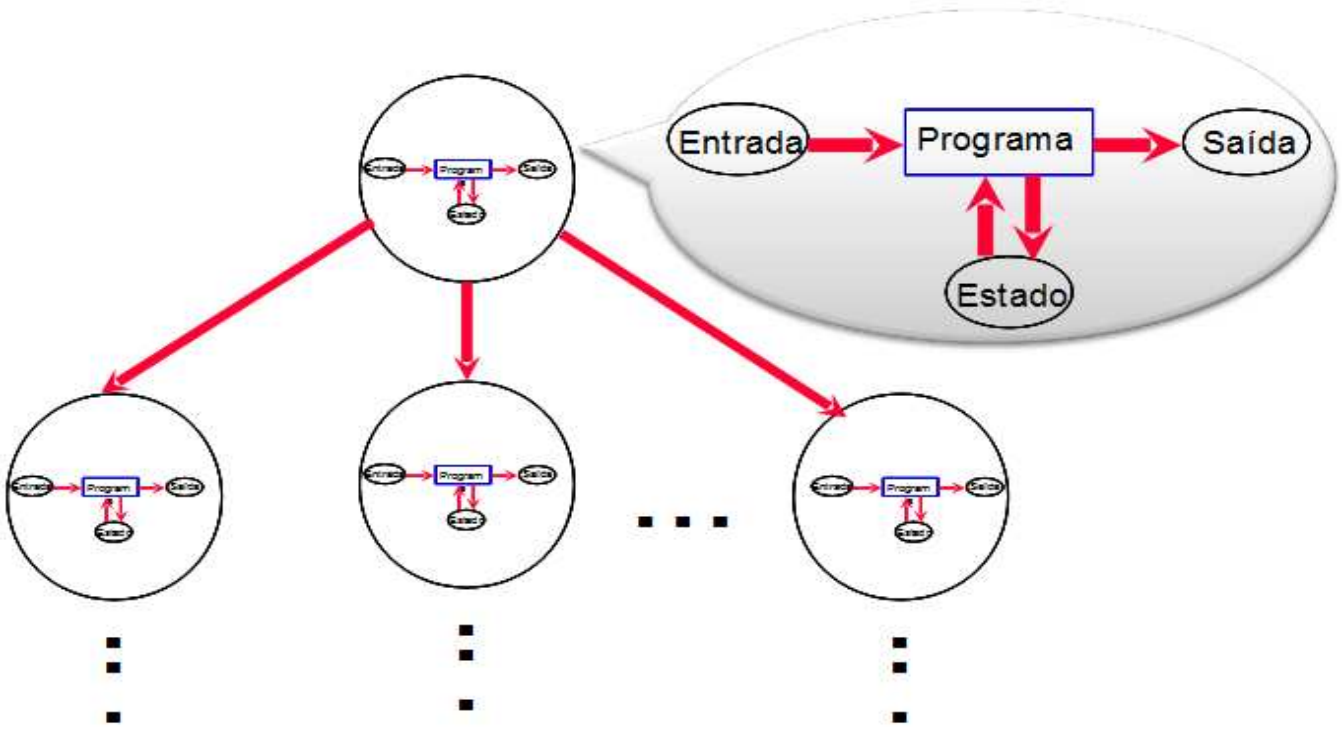
3.4) Orientado a objetos





Classificação das LP
3) quanto ao paradigma

3.4) Orientado a objetos: modelo computacional



Classificação das LP
3) quanto ao paradigma

3.4) Orientado a objetos

- O paradigma orientado a objetos considera objetos e classes como blocos básicos de construção de um sistema.

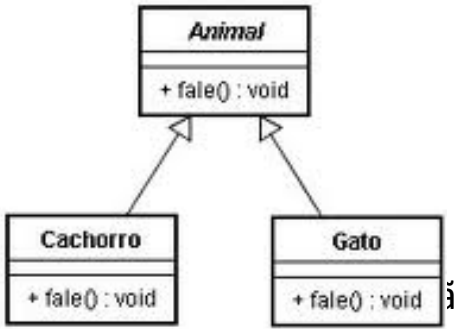
- Sistemas são vistos como coleções de objetos que se comunicam, enviando mensagens, colaborando para dar o comportamento global dos sistemas.

Classificação das LP
3) quanto ao paradigma

3.4) Orientado a objetos

- Uma classe determina o comportamento e a estrutura de objetos similares (Célula, Mamífero, Réptil,...).

- Pode-se dizer que a ideia da orientação a objetos foi importada a partir da observação do comportamento de sistemas complexos do mundo real (animais, plantas, máquinas, ou até mesmo empresas), onde cada objeto possui um determinado conjunto de responsabilidades dentro de um sistema, que normalmente estão relacionadas com a manutenção de conhecimento e com ações que devem executar.



ão

Classificação das LP
3) quanto ao paradigma

3.4) Orientado a objetos: exemplos de LP

- SIMULA 67 foi a primeira linguagem a incorporar estes conceitos, desenvolvida especialmente para a criação de aplicações de simulação.

- Outra linguagem OO importante e pioneira é Smalltalk, originada na dissertação de doutorado de Alan Kay em 1969.

- Outras linguagens OO combinam características de linguagens imperativas e orientadas a objetos: C++, Eiffel, Java, Ada...

Classificação das LP
orientado a objetos – exemplo Java

```
public class Elipse
{
   private int _X, _Y, _raioX, _raioY;
   public Elipse(int x, int y, int rx, int ry)
   {    _X = x; _Y = y; _raioX = rx; _raioY = ry;
   }
   public void DefinirRaios(int rx, int ry)
   {    _raioX = rx; _raioY = ry;
   }
   public int InformarRaioX()
   {    return _raioX;
   }
   public int InformarRaioY()
   {    return _raioY;
   }
   public void Desenhar(Graphics area) {
        area.drawOval(_X, _Y, _raioX, _raioY);
   }
};
```

```
Elipse e1;
e1=new Elipse(10,10,20,25);
e1.desenhar(g);
```

Classificação das LP
3) quanto ao paradigma

3.5) Concorrente

- Múltiplas computações podem ser executadas simultaneamente

- Computações paralelas:
  - Múltiplos processadores compartilham memória

- Computações distribuídas:
  - Múltiplos computadores conectados por uma rede de comunicação

Bibliografia

Capítulo 1: SEBESTA, R. W. ***Conceitos de Linguagens de Programação***, 5ª ed., Bookman, 2003.

Na próxima aula

- Processo de compilação...