

# Roteiro

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

# Introdução

- ❑ Na programação de computadores, uma linguagem de programação (LP) serve como **meio de comunicação entre o indivíduo que deseja resolver um determinado problema e o computador** escolhido para ajudá-lo na solução.



Base	Offset	Values
00001000	+0	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000
00001010	+16	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000
00001020	+32	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000
00001030	+48	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000
00001040	+64	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000
00001050	+80	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000
00001060	+96	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000
00001070	+112	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000
00001080	+128	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000
00001090	+144	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000
000010A0	+160	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000
000010B0	+176	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000
000010C0	+192	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000
000010D0	+208	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000
000010E0	+224	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000
000010F0	+240	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000

Refresh Close

# Introdução

- ❑ A LP deve fazer a **ligação** entre o **pensamento humano** (muitas vezes de natureza não estruturada) e a **precisão** requerida para o **processamento** pela **máquina**.

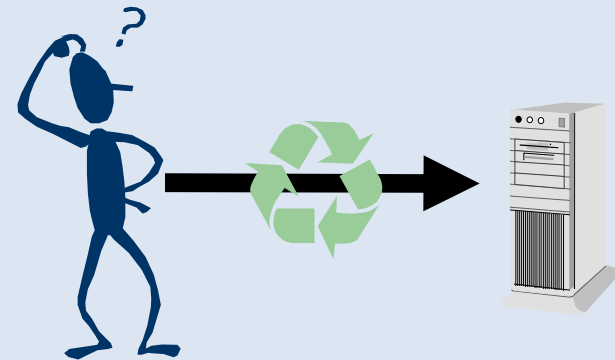


Base	Offset	Values
00001000	+0	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000
00001010	+16	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000
00001020	+32	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000
00001030	+48	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000
00001040	+64	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000
00001050	+80	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000
00001060	+96	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000
00001070	+112	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000
00001080	+128	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000
00001090	+144	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000
000010A0	+160	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000
000010B0	+176	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000
000010C0	+192	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000
000010D0	+208	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000
000010E0	+224	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000
000010F0	+240	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000

## Objetivos de uma LP

❑ Auxiliar o programador no processo de desenvolvimento de software. Isso inclui auxílio no:

- Projeto
- Implementação
- Teste
- Verificação e
- Manutenção do software



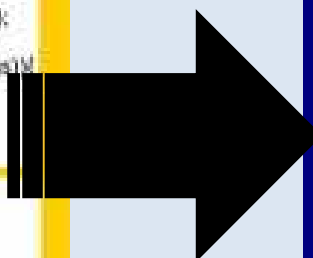
# Definição

- ❑ Uma LP é uma linguagem destinada para ser usada por uma **pessoa** para expressar um **processo** através do qual um **computador** possa resolver um **problema**.
- ❑ Os quatro modelos (**paradigmas: Imperativo, Orientado a Objeto, Funcional e Lógico**) 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.

# Definição

- ❑ Para que se tornem operacionais, os programas escritos em **linguagens de alto nível** devem ser traduzidos para **linguagem de máquina**.

PASCAL	JAVA
<pre> program Hello; var mensagem : string; begin   mensagem := 'Hello World!';   write(mensagem); End.</pre>	<pre> public class Main {   public Main(){     System.out.println("Hello World");   }   public static void main(String [] args){     Main m = new Main();   } }</pre>
C	COBOL
<pre> #include &lt;stdio.h&gt; #include &lt;stdlib.h&gt;  int main() {   printf("HELLO WORLD!!!");   return(0); }</pre>	<pre> function Hello ()(   alert("Hello World!") )</pre>



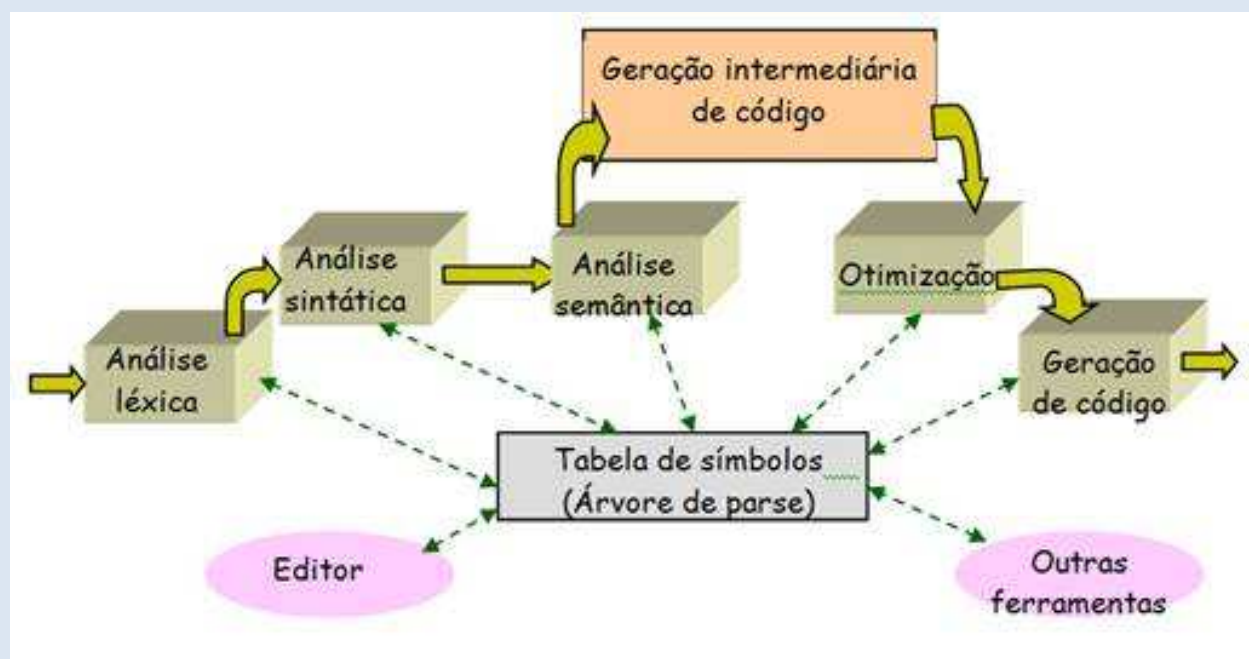
Endereço	Código	Assembly	
1B8D:0100	01D8	ADD	AX,BX
1B8D:0102	C3	RET	
1B8D:0103	16	PUSH	SS
1B8D:0104	B03A	MOV	AL,3A
1B8D:0106	380685D5	CMP	[D585],AL
1B8D:010A	750E	JNZ	011A
1B8D:010C	804E0402	OR	BYTE PTR [BP+04],02
1B8D:0110	BF86D5	MOV	DI,D586
1B8D:0113	C6460000	MOV	BYTE PTR [BP+00],00
1B8D:0117	E85F0B	CALL	0C79
1B8D:011A	8B7E34	MOV	DI,[BP+34]
1B8D:011D	007C1B	ADD	[SI+1B],BH

# Definição

## Linguagem alto nível para linguagem de máquina

- ❑ Essa conversão é realizada através de sistemas especializados – compiladores ou interpretadores – que aceitam (como entrada) uma representação textual da solução de um problema, expresso em uma **linguagem fonte**, e produzem uma representação do mesmo algoritmo expresso em outra linguagem, dita **linguagem objeto**.

# Definição





## Razões para estudar os conceitos de LP

- ☐ Aumento da capacidade de expressar ideias
- ☐ Maior conhecimento para escolha de linguagens apropriadas
- ☐ Entender melhor a importância da implementação
- ☐ Maior capacidade para aprender novas linguagens
- ☐ Aumento da capacidade de projetar novas linguagens
- ☐ Avanço global da comunicação

# Razões para estudar os conceitos de LP

## ❑ Aumento da capacidade de expressar ideias

- Capacidade intelectual pode ser influenciada pelo poder expressivo da linguagem
- Uma maior compreensão de uma LP pode aumentar nossa habilidade em pensar em como atacar os problemas.

# Razões para estudar os conceitos de LP

## ❑ Aumento da capacidade de expressar ideias

- Conhecimento amplo dos recursos de linguagem reduz as limitações no desenvolvimento de softwares
- A melhor compreensão das funções e implementação das estruturas de uma LP nos leva a usar a LP de modo a extrair o máximo de sua funcionalidade e eficiência
- Recursos ou facilidades podem ser simulados

# Razões para estudar os conceitos de LP

## ❑ **Maior conhecimento para escolha de linguagens apropriadas**

➤ Escolher a melhor linguagem para um problema específico devido ao conhecimento de novos recursos é difícil para:

- ✓ Programadores antigos
- ✓ Desenvolvedores sem educação formal

# Razões para estudar os conceitos de LP

## ❑ Entender melhor a importância da implementação:

- Leva um entendimento do **PORQUÊ** das linguagens serem projetadas de determinada maneira.
- Melhora as escolhas que podemos fazer entre as construções de LP e as consequências das opções.

# Razões para estudar os conceitos de LP

## ❑ **Maior capacidade para aprender novas linguagens:**

- Aprendizado contínuo é fundamental, a computação é uma ciência nova
- Compreender os conceitos gerais das linguagens torna mais fácil entender como eles são incorporados na linguagem que está sendo aprendida.

# Razões para estudar os conceitos de LP

## ❑ Aumento da capacidade de projetar novas linguagens:

- Ajuda no desenvolvimento de sistemas complexos

# Razões para estudar os conceitos de LP

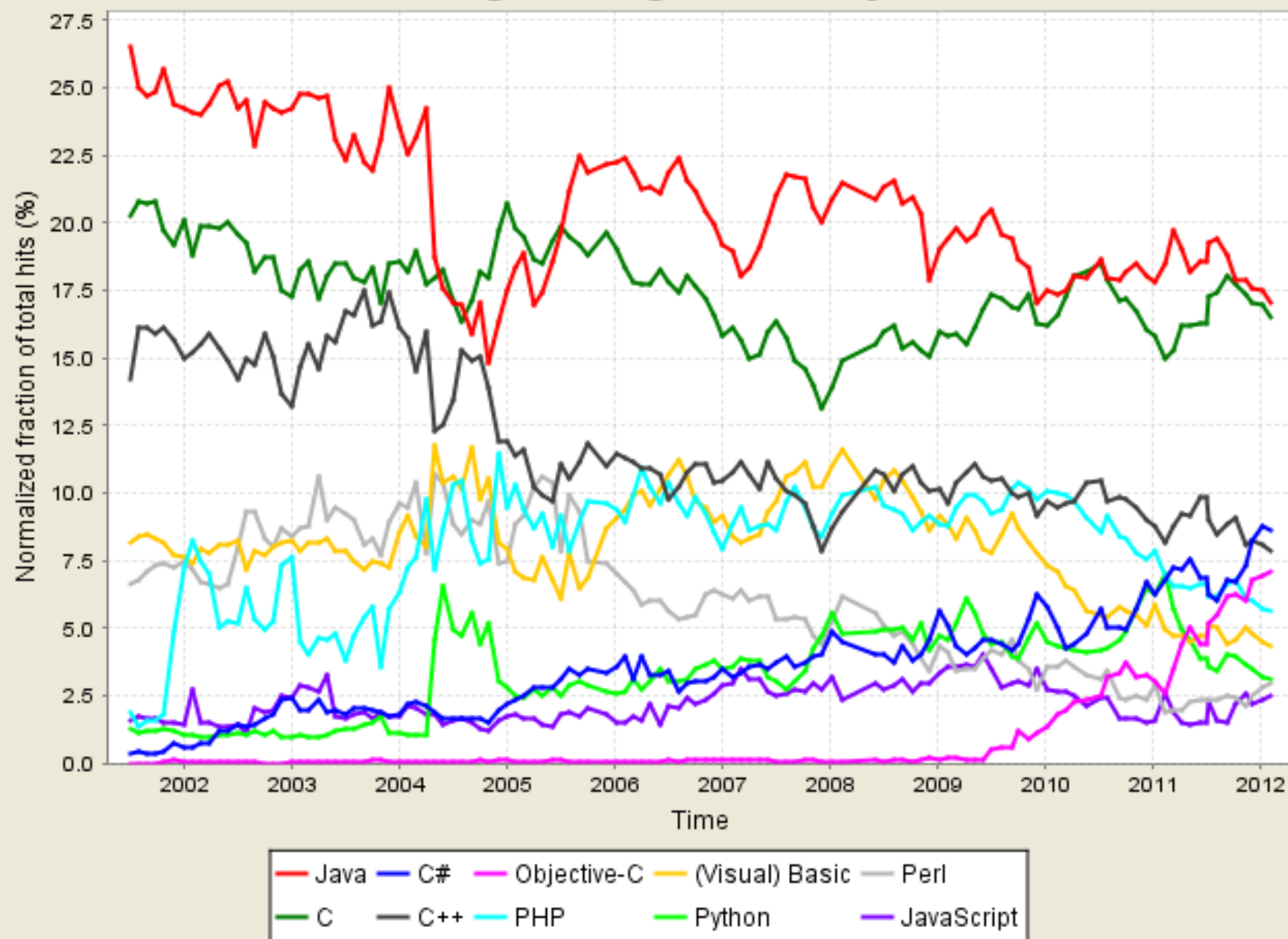
❑ **Aumento da capacidade de projetar novas linguagens:**

➤ Indicador de popularidade das LP – **Tiobe**  
(**[www.tiobe.com](http://www.tiobe.com)**)



Position Feb 2012	Position Feb 2011	Delta in Position	Programming Language	Ratings Feb 2012	Delta Feb 2011	Status
1	1	=	Java	17.050%	-1.43%	A
2	2	=	C	16.523%	+1.54%	A
3	6	↑↑↑	C#	8.653%	+1.84%	A
4	3	↓	C++	7.853%	-0.33%	A
5	8	↑↑↑	Objective-C	7.062%	+4.49%	A
6	5	↓	PHP	5.641%	-1.33%	A
7	7	=	(Visual) Basic	4.315%	-0.61%	A
8	4	↓↓↓↓	Python	3.148%	-3.89%	A
9	10	↑	Perl	2.931%	+1.02%	A
10	9	↓	JavaScript	2.465%	-0.09%	A
11	13	↑↑	Delphi/Object Pascal	1.964%	+0.90%	A
12	11	↓	Ruby	1.558%	-0.06%	A
13	14	↑	Lisp	0.905%	-0.05%	A
14	26	↑↑↑↑↑↑↑↑	Transact-SQL	0.846%	+0.29%	A
15	17	↑↑	Pascal	0.813%	+0.08%	A
16	22	↑↑↑↑↑	Visual Basic .NET	0.796%	+0.21%	A--
17	32	↑↑↑↑↑↑↑↑	PL/SQL	0.792%	+0.38%	A
18	24	↑↑↑↑↑	Logo	0.677%	+0.10%	B
19	16	↓↓↓	Ada	0.632%	-0.17%	B
20	25	↑↑↑↑↑	R	0.623%	+0.06%	B

## TIOBE Programming Community Index



# Razões para estudar os conceitos de LP

## ❑ Avanço global da comunicação:

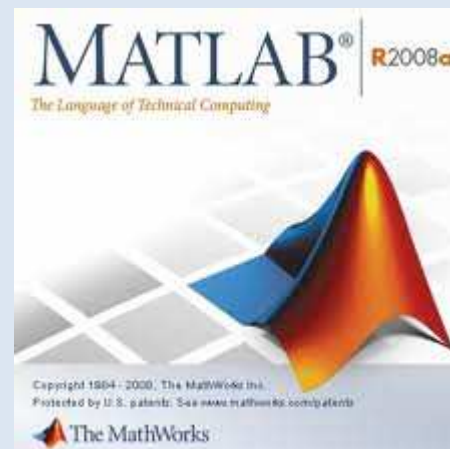
- Nem sempre as linguagens mais populares são melhores, por quê?
  - **Imposição!!**
- Por que existem várias linguagens de programação?
  - Resolução específica de problemas

# 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

# Domínios de programação

- ❑ Computadores têm sido aplicados a uma infinidade de áreas



# Domínios de programação

USP

## Metodologias para estimação da idade óssea

Escolha uma metodologia

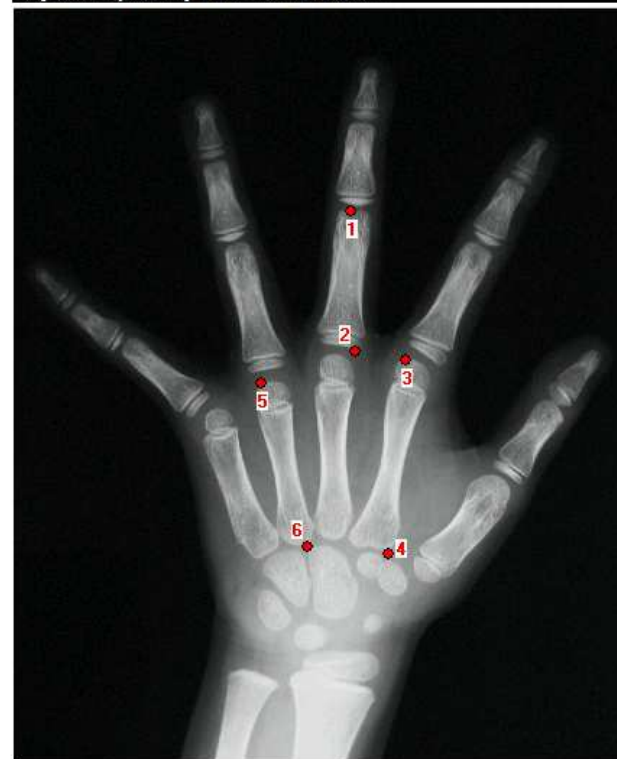
☐ E & R - 10 ☐ E & R - 5 ☒ E & R - 3

☐ N-Ouro-1 ☐ N-Ouro-2 ☐ N-Ouro-3 ☐ N-Ouro-4

Abrir imagem (jpg, jpeg, bmp, gif ou png)

(Tamanho máximo - 2.5Mb)

Disponibilização dos pontos - Método E&R-3



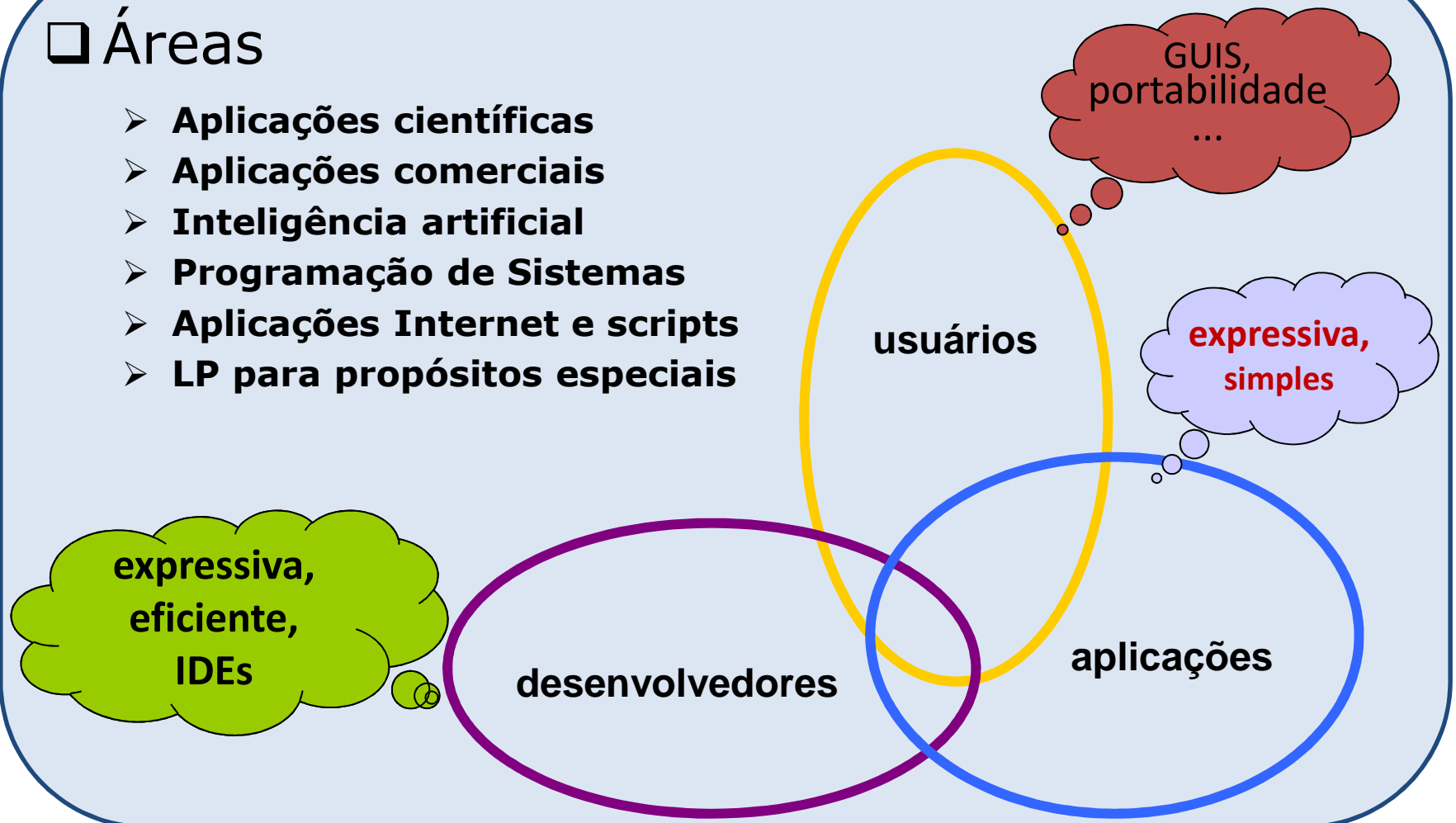
Linguagens de programação



# Domínios de programação

## □ Áreas

- Aplicações científicas
- Aplicações comerciais
- Inteligência artificial
- Programação de Sistemas
- Aplicações Internet e scripts
- LP para propósitos especiais





# Domínios de programação

## aplicações científicas

❑ Início: década de 40.

❑ Foco: eficiência (*assembly*).

Endereço	Código	Assembly	
1B8D:0100	01D8	ADD	AX,BX
1B8D:0102	C3	RET	
1B8D:0103	16	PUSH	SS
1B8D:0104	B03A	MOV	AL,3A
1B8D:0106	3B0685D5	CMF	[D585],AL
1B8D:010A	750E	JNZ	011A
1B8D:010C	804E0402	OR	BYTE PTR [BP+04],02
1B8D:0110	BF86D5	MOV	DI,D586
1B8D:0113	C6460000	MOV	BYTE PTR [BP+00],00
1B8D:0117	E85F0B	CALL	0C79
1B8D:011A	8B7E34	MOV	DI,[BP+34]
1B8D:011D	B07C1B	ADD	[SI+1B],BH

❑ Nesta categoria se enquadram todos os problemas que necessitam um grande volume de processamento, com **operações** geralmente feitas em **ponto flutuante**, e com poucas exigências de entrada e saída.

❑ As estruturas de dados mais comuns são as matrizes e *arrays*; as estruturas de controle mais comuns são os laços de contagem e de seleções



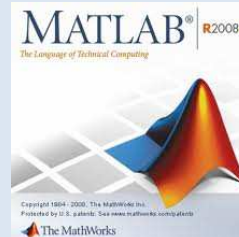
# Domínios de programação aplicações científicas

- ❑ As aplicações científicas incentivaram a criação de algumas linguagens de alto nível, como por exemplo o **FORTRAN**

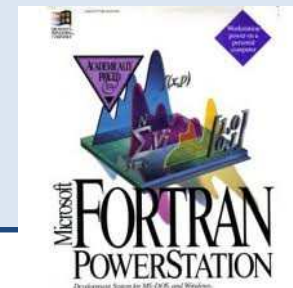
- ❑ O **ALGOL 60** e a maioria de suas descendentes também se destinam a serem usadas nessa área, ainda que projetadas para outras áreas relacionadas

```
start : p := 0;  
        i := n;  
loop :  if i = 0 then go to done;  
        p := p + m;  
        i := i - 1;  
        go to loop;  
done :
```

- ❑ Exemplo: **MATLAB**



- ❑ Para aplicações científicas cuja eficiência é altamente prioritária, nenhuma linguagem subsequente é significativamente melhor do que **FORTRAN**



## Domínios de programação aplicações comerciais

- ❑ Iniciou-se na década de 50.
- ❑ **Foco:** produção de relatórios elaborados.
- ❑ Impulsionou o desenvolvimento de equipamentos especiais.
- ❑ A primeira linguagem bem sucedida foi o **COBOL** (em 1960).

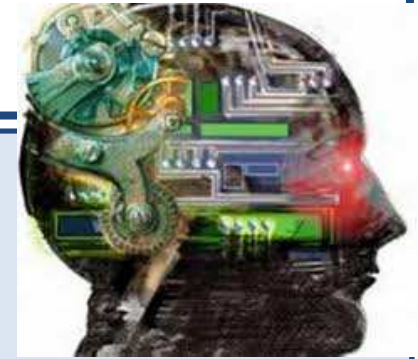


## Domínios de programação aplicações comerciais

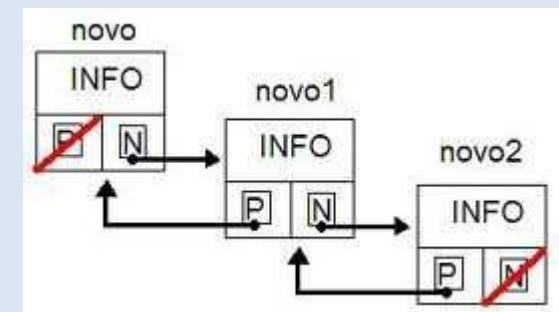
- ❑ As linguagens comerciais se caracterizam pela facilidade de elaborar relatórios e armazenar números decimais e dados de caracteres.
- ❑ Com o advento dos microcomputadores surgiram as planilhas e os sistemas de banco de dados amplamente utilizados hoje em dia.
- ❑ **Exemplos:** Planilhas eletrônicas e Sistemas Gerenciadores de Banco de Dados.



# Domínios de programação inteligência artificial

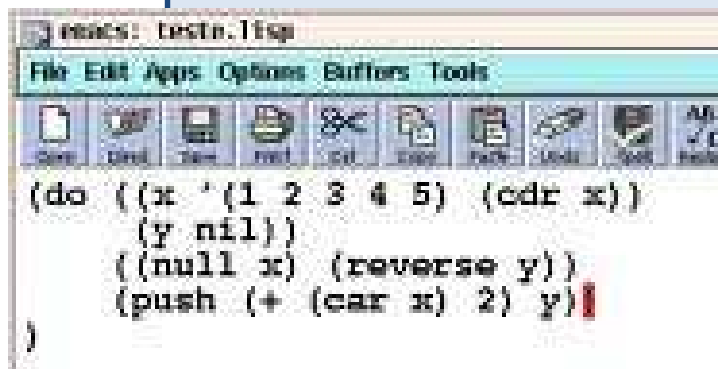


- ❑ Início: fim da década de 50
- ❑ Foco: manipulação flexível de informações, mesmo com pouca eficiência, em muitos casos.
  - Exemplo: classificação de padrões (reconhecimento da placa de veículos)
- ❑ Estruturas de dados: listas encadeadas de dados (em oposição a *arrays*).



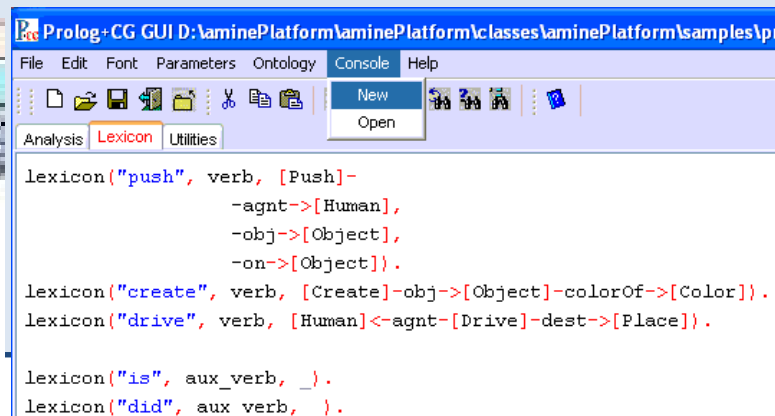
# Domínios de programação inteligência artificial

- ❑ Caracterizam-se pelo uso de computações simbólicas em vez de numéricas (são manipulados nomes e não números);
- ❑ A primeira linguagem desenvolvida para IA foi a funcional LISP (1959). No início de 70 surge a programação lógica: PROLOG.



```

macs: teste.lisp
File Edit Apps Options Buffers Tools
[Icons]
(do ((x '(1 2 3 4 5) (cdr x))
    (y nil))
    ((null x) (reverse y))
    (push (+ (car x) 2) y))
)
  
```



```

Prolog-CG GUI D:\aminePlatform\aminePlatform\classes\aminePlatform\samples\pr
File Edit Font Parameters Ontology Console Help
[Icons]
Analysis Lexicon Utilities
lexicon("push", verb, [Push]-
    -agnt->[Human],
    -obj->[Object],
    -on->[Object]).
lexicon("create", verb, [Create]-obj->[Object]-colorOf->[Color]).
lexicon("drive", verb, [Human]<-agnt-[Drive]-dest->[Place]).

lexicon("is", aux_verb, _).
lexicon("did", aux_verb, _).
  
```

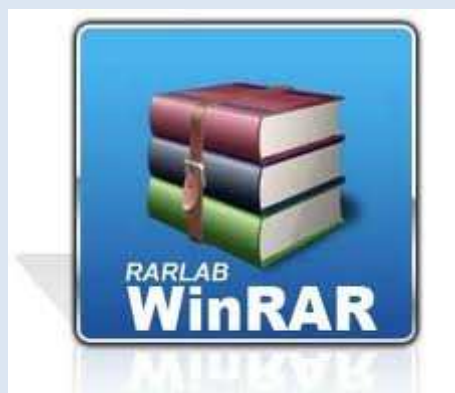


## Domínios de programação programação de sistemas

- ❑ Denominados **Software básico** (SO, utilitários), devem possuir eficiência na execução por propiciar suporte a execução de outros aplicativos.
- ❑ Devem oferecer execução rápida e ter recursos de baixo nível que permitam ao software fazer interface com os dispositivos externos.
- ❑ **Linguagem de Programação:** orientada a software básico, para execução rápida, com recursos de baixo nível.

# Domínios de programação

## programação de sistemas

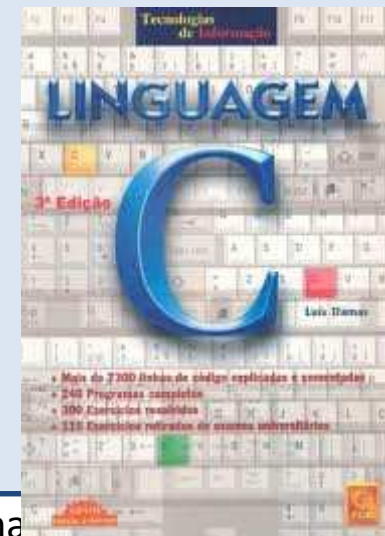




# Domínios de programação

## programação de sistemas

- ❑ O sistema operacional UNIX foi desenvolvido quase inteiramente em C (tornando-o fácil de portar para diferentes máquinas).
- ❑ Ling. C pode ser considerada “baixo nível”, com execução eficiente e leve.





## Domínios de programação propósitos especiais

- ❑ Se distinguem por serem criadas para oferecer recursos de computação para hardwares de uso específico. (tornos CNC, calculadoras HP, etc.)
- ❑ Exemplos de linguagem de uso específico: RPG (relatórios comerciais), GPSS (simulação de sistemas).

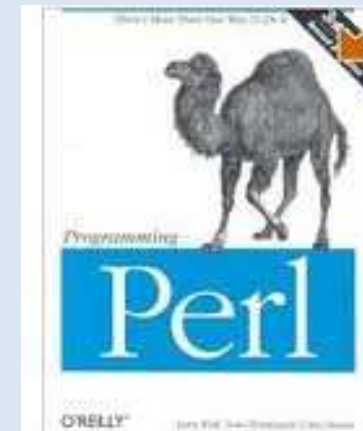
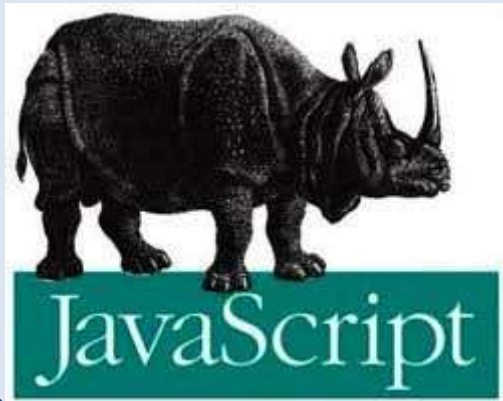
## **Domínios de programação** linguagens de scripts

- ❑ Início: década de 80.
- ❑ Script = lista de comandos em arquivo
- ❑ Execução do arquivo para, p.ex., exercer funções utilitárias do sistema, tais como bloquear uma sequência de endereços IP em uma determinada rede.

# Domínios de programação

## linguagens de scripts

- ❑ **Nestas linguagens** é assumida a existência de um conjunto de componentes já desenvolvidos em outras linguagens, de forma que o objetivo destas linguagens passa a ser **combinar componentes e não desenvolver programas** a partir de estruturas de dados elementares.



# Domínios de programação

## linguagens de scripts

- ❑ Linguagens de script modernas (Perl, Tcl/Tk e Python) suportam:
  - Manipulação textual simples para interpretação das entradas do usuário;
  - Interfaceamento com software já existente;
  - Acesso a sistemas operacionais, por meio de variáveis de ambiente e chamadas ao sistema.
- ❑ Aplicação: linguagens de script são muito usadas para implementar CGIs (*Common Gateway Interfaces*), para criar páginas dinâmicas de Web.

# **Critérios de avaliação de linguagens**

**1) Legibilidade**

**2) Capacidade de Escrita**

**3) Confiabilidade**

**4) Custo**

# Critérios de avaliação de linguagens

## 1) Legibilidade

- ❑ Um dos critérios mais importantes para julgar uma **LP** é a **facilidade** com que os **programas são lidos e entendidos**
- ❑ Antes de 70: pensado em termos de escrita de código.
  - Principais características: **eficiência** e **legibilidade** de máquina.
  - LP foram projetadas mais do ponto de vista do computador do que do usuário
- ❑ Na década de 70 foi desenvolvido o conceito de ciclo de vida de software  
→ manutenção



# Critérios de avaliação de linguagens

## 1) Legibilidade

- ❑ A facilidade de manutenção é determinada em grande parte, pela **legibilidade** dos programas, ela se tornou uma medida importante da qualidade dos programas e das linguagens
- ❑ A legibilidade deve ser considerada no contexto do domínio do problema (Ex. um programa escrito em uma linguagem não apropriada se mostra antinatural e “enrolado”, difícil de ser lido)

# **Critérios de avaliação de linguagens**

## **1) Legibilidade**

- 1.1) Simplicidade Geral
- 1.2) Ortogonalidade
- 1.3) Instruções de Controle
- 1.4) Tipos de Dados e estruturas
- 1.5) Considerações sobre sintaxe



# Critérios de avaliação de linguagens

## 1) Legibilidade

### 1.1) Simplicidade geral

- ❑ A **simplicidade geral** de uma LP **afeta** fortemente sua **legibilidade**
- ❑ *Uma linguagem com um grande número de componentes básicos é mais difícil de ser manipulada do que uma com poucos desses componentes.*
  - Os programadores que precisam usar uma linguagem grande tendem a aprender um subconjunto dela e ignorar seus outros recursos.

# **Critérios de avaliação de linguagens**

## **1) Legibilidade**

### **1.1) Simplicidade geral**

- ❑ Esse padrão de aprendizagem pode ocasionar problemas quando o leitor do programa aprende um conjunto diferente de recursos daquele que o autor aplicou em seu programa.

# Critérios de avaliação de linguagens

## 1) Legibilidade

### 1.1) Simplicidade geral

- ❑ Uma **segunda** característica que **complica** a legibilidade é a **multiplicidade de recursos** (mais que uma maneira de realizar uma operação particular)

**$i = i + 1$**

**$i + = 1$**

**$i ++$**

**$++i$**

Mesmo significado  
quando usadas em  
expressões separadas!!

# Critérios de avaliação de linguagens

## 1) Legibilidade

### 1.1) Simplicidade geral

❑ Um **terceiro** problema é a **sobrecarga de operadores**, na qual um único símbolo tem mais que um significado.

➤ Apesar de ser um recurso útil, pode ser prejudicial a legibilidade se for permitido aos usuários criar suas próprias sobrecargas.

➤ **Ex: sobrecarregar o + para adicionar inteiros, reais, concatenar strings, somar vetores...**

# Critérios de avaliação de linguagens

## 1) Legibilidade

### 1.1) Simplicidade geral

❑ A simplicidade de linguagens, no entanto pode ser levada ao extremo, por exemplo a forma e o significado da maioria das instruções em *Assembly* são modelos de simplicidade, entretanto torna os programas em *Assembly* menos legíveis.

- Falta instruções de controle mais complexas, torna necessário o uso de mais códigos para expressar problemas do que os necessário em linguagens de alto nível.

# Critérios de avaliação de linguagens

## 1) Legibilidade

Endereço	Código	Assembly
1B8D:0100	01D8	ADD AX,BX
1B8D:0102	C3	RET
1B8D:0103	16	PUSH SS
1B8D:0104	B03A	MOV AL,3A
1B8D:0106	380685D5	CMP [D585],AL
1B8D:010A	750E	JNZ 011A
1B8D:010C	804E0402	OR BYTE PTR [BP+04],02
1B8D:0110	BF86D5	MOV DI,D586
1B8D:0113	C6460000	MOV BYTE PTR [BP+00],00
1B8D:0117	E85F08	CALL 0C79
1B8D:011A	8B7E34	MOV DI,[BP+34]
1B8D:011D	B07C1B	ADD [SI+1B],BH

# Critérios de avaliação de linguagens

## 1) Legibilidade

### 1.2) Ortogonalidade

❑ Possibilidade de **combinar** entre si, sem restrições, os **componentes básicos da LP**.

➤ **Exemplo:** permitir combinações de estruturas de dados, como *arrays* de registros

➤ **Contra exemplo:** não permitir que um *array* seja usado como parâmetro de um procedimento

# Critérios de avaliação de linguagens

## 1) Legibilidade

### 1.2) Ortogonalidade - Exemplos

#### ❑ Falta de ortogonalidade em C:

- A linguagem **C** possui dois tipos de dados estruturados: **arrays** e registros (**struct**), sendo que :
  - ✓ registros podem ser retornados de funções, *arrays* não.
  - ✓ um membro de estrutura pode ser qualquer tipo de dado, exceto *void* ou uma estrutura do mesmo tipo.
  - ✓ um elemento de *array* pode ser qualquer tipo de dado, exceto *void* ou uma função.
  - ✓ Parâmetros são passados por valor, a menos que sejam *arrays* – que obrigatoriamente são passados por referência.



# Critérios de avaliação de linguagens

## 1) Legibilidade

### 1.2) Ortogonalidade - Exemplos

- ❑ Falta de ortogonalidade C

$A + B$

- ❑ Valores de A e B são obtidos e adicionados juntos
- ❑ Se A for um **ponteiro** afeta o valor de B
  - ❑ Se A aponta para um valor de ponto flutuante que ocupa 4 bytes, o valor de B deve ser ampliado (multiplicado por 4) antes que seja adicionado a A
    - ❑ Logo o tipo de A afeta o tratamento do valor de B
    - ❑ O contexto de B afeta seu significado

# **Critérios de avaliação de linguagens**

## **1) Legibilidade**

### **1.3) Instruções de controle**

- ❑ A revolução da programação estruturada da década de 70 foi, em parte, uma reação à má legibilidade causada pelas limitadas instruções de controle das linguagens das décadas de 50 e 60.

# Critérios de avaliação de linguagens

## 1) Legibilidade

### 1.3) Instruções de controle

#### ❑ Uso indiscriminado de **goto**.

- Em certas linguagens, entretanto, instruções **goto** que se ramificam para cima, às vezes, são necessárias;
- Ex: elas constroem laços **while** em FORTRAN 77. Restringir instruções **goto** das seguintes maneiras pode tornar os programas mais legíveis:
  - Elas devem preceder seus alvos, exceto quando usadas para formar laços;
  - Seus alvos nunca devem estar tão distantes;
  - Seu número deve ser limitado.

# Critérios de avaliação de linguagens

## 1) Legibilidade

### 1.3) Instruções de controle

- ❑ A partir do final de 60, as linguagens projetadas passaram a ter instruções de controle suficientes e portanto a necessidade da instrução ***goto*** foi quase eliminada.
  - O projeto da estrutura de controle de uma linguagem é agora um fator menos importante na legibilidade do que no passado.

**while   do...while   repeat...until   for..next**

# Critérios de avaliação de linguagens

## 1) Legibilidade

### 1.4) Tipos de dados e estruturas

❑ A presença de facilidades adequadas para definir tipos de dados e estruturas de dados em uma linguagem é outro auxílio significativo para a legibilidade.

➤ Ex: supõe-se que um tipo numérico seja usado para um sinalizador porque não há nenhum tipo booleano na linguagem:

**Terminou=1, não é tão claro como      Terminou=true**

# Critérios de avaliação de linguagens

## 1) Legibilidade

### 1.5) Considerações sobre sintaxe

- ❑ A sintaxe ou a forma dos elementos de uma linguagem tem um efeito significativo sobre a legibilidade dos programas. Exemplos de opções de projeto sintático que afetam a legibilidade :

- **Formas identificadoras.**

- Restringir os identificadores a tamanhos muito pequenos prejudica a legibilidade.
- Tamanhos de identificadores muito pequenos impedem as vezes de nomear variáveis com nomes conotativos. (Ex: **FORTRAN 77, máximo 6 caracteres; BASIC ANSI, uma letra ou uma letra e um número**)

# Critérios de avaliação de linguagens

## 1) Legibilidade

### 1.5) Considerações sobre sintaxe

#### ❑ Palavras especiais.

- A aparência de um programa e sua consequente legibilidade são fortemente influenciadas pelas formas das palavras especiais de uma linguagem (**ex: begin, end e for**).
- ❑ O Pascal exige pares de **begin/end** para formar grupos em todas as construções de controle (exceto *repeat*), a linguagem C usa chaves.

# Critérios de avaliação de linguagens

## 1) Legibilidade

### 1.5) Considerações sobre sintaxe

- ❑ Ambas as linguagens sofrem porque os grupos de instruções são sempre encerrados da mesma maneira, o que torna difícil determinar qual grupo está sendo finalizado quando um **end** ou **}** aparece
- ❑ O FORTAN 90 e o ADA tornam isso mais claro, usando uma sintaxe de fechamento distinta para cada tipo de grupo de instrução. (**if...end if / loop...end loop**)



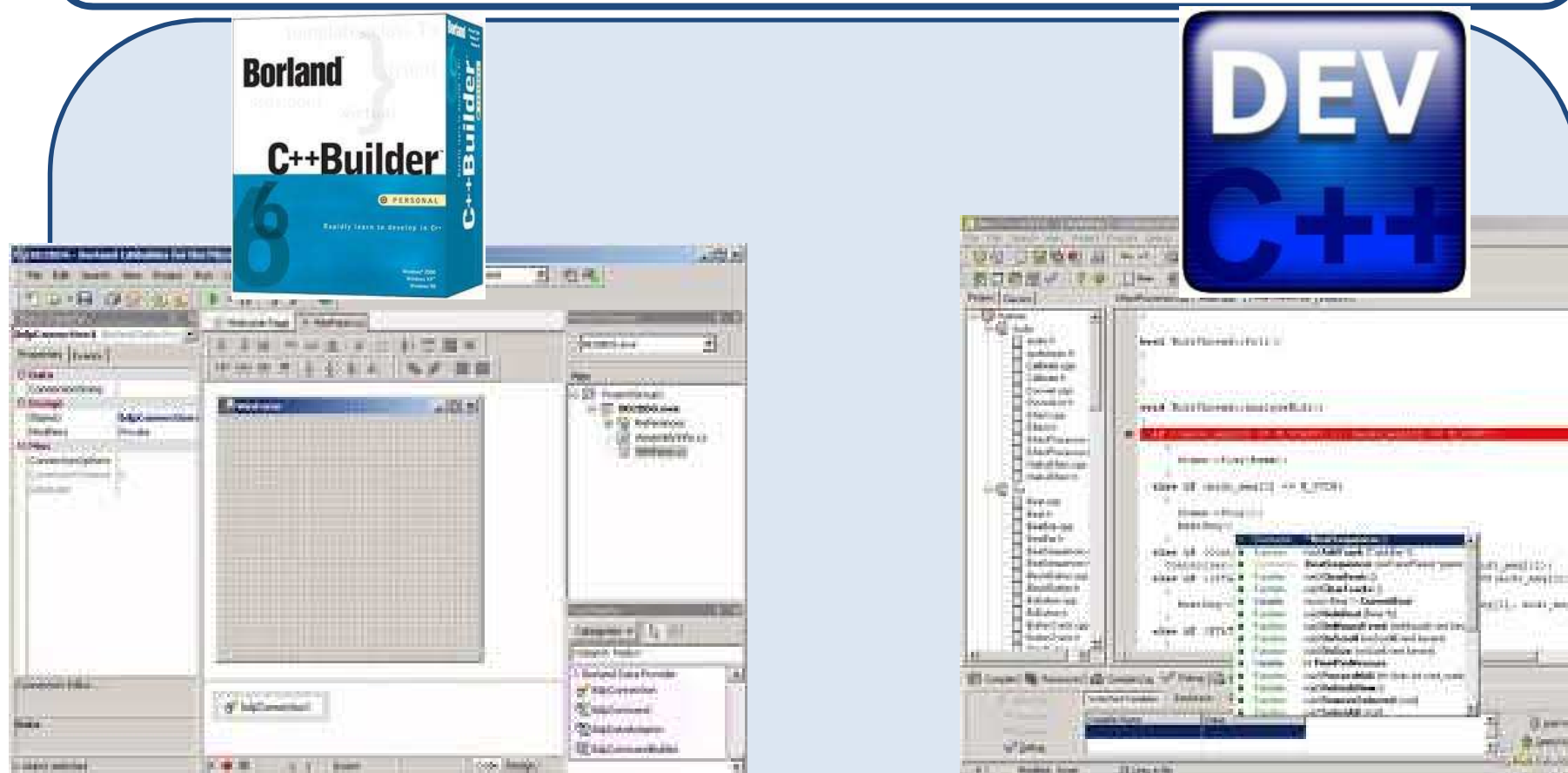
# **Critérios de avaliação de linguagens**

## **2) Capacidade de escrita (CE)**

- ☐ É a medida da facilidade em que uma linguagem pode ser usada para criar programas para um domínio de problema escolhido.
- ☐ A maioria das características da linguagem que afetam a legibilidade também afetam a CE.
- ☐ Deve ser considerada no contexto do domínio de problema-alvo da linguagem.

# Critérios de avaliação de linguagens

## 2) Capacidade de escrita (CE)



**Construção de uma interface gráfica**  
C++ Builder *versus* Dev C++

## **Critérios de avaliação de linguagens**

### **2) Capacidade de escrita**

2.1) Simplicidade e Ortogonalidade

2.2) Suporte para Abstração

2.3) Expressividade

# **Critérios de avaliação de linguagens**

## **2) Capacidade de escrita**

### **2.1) Simplicidade e ortogonalidade**

- ❑ Se uma LP tem um grande número de construções, alguns programadores não estarão familiarizados com todas.
- ❑ Pode acarretar o uso incorreto de alguns recursos e uma utilização escassa de outros que podem ser mais elegantes ou eficientes do que os usados

# Critérios de avaliação de linguagens

## 2) Capacidade de escrita

### 2.2) Suporte para abstração

- ❑ **Abstração:** capacidade de definir e, depois usar estruturas ou operações complicadas de uma maneira que permita ignorar muito dos detalhes.
- ❑ Exemplo: **uso de subprogramas** (algoritmo de ordenação)
- ❑ Tipos de Abstração:
  - **de Processo:** algoritmos de classificação, elementos de interface gráfica
  - **de Dados:** tipo moeda, tipo string, tipo data

# Critérios de avaliação de linguagens

## 2) Capacidade de escrita

### 2.3) Expressividade

#### ❑ Formas convenientes de especificar computações

➤ Uma expressão representa muitas computações

✓ Exemplos:

`i++`, ao invés de `i=i+1`

**for** ao invés do **while**

**Readln** do Pascal ao invés de **readLine** do Java

#### (Java)

```
BufferedReader teclado;  
String linha;  
teclado = new BufferedReader(  
    new InputStreamReader(System.in) );  
linha = teclado.readLine();
```

#### (Pascal)

```
linha: string[20]  
readln(linha)
```

## **Critérios de avaliação de linguagens**

### **3) Confiabilidade**

- ❑ Um programa é **confiável** se ele se comportar de acordo com suas especificações sob todas as condições.

## **Critérios de avaliação de linguagens**

### **3) Confiabilidade**

3.1) Verificação de Tipos

3.2) Manipulação de Exceções

3.3) *Aliasing* (apelidos)

3.4) Legibilidade e facilidade de Escrita



# Critérios de avaliação de linguagens

## 3) Confiabilidade

### 3.1) Verificação de tipos

- ❑ Testar se existem erros de tipos em determinado algoritmo, ou por meio do compilador ou durante a execução do programa.

- ❑ A verificação de tipos durante a compilação é a mais indicada

- ❑ Quanto antes for detectado, menos caro é fazer todos os reparos necessários!!

Ex: **Java**

- ❑ A verificação de tipos em C é bastante fraca:

```
int vet[50];  
vet[100]=10.3;
```

## **Critérios de avaliação de linguagens**

### **3) Confiabilidade**

#### **3.2) Manipulação de exceções**

- ☐ Capacidade de um programa de interceptar erros em tempo de execução, pôr em prática medidas corretivas e, depois, prosseguir.
- ☐ Exemplos:

# Critérios de avaliação de linguagens

## 3) Confiabilidade

### 3.2) Manipulação de exceções

Java

```
try
{
...
}
catch( Exception e )
{
...
}
```

C++

```
try
{
...
}
catch (Exception &exception)
{
...
}
```

# Critérios de avaliação de linguagens

## 3) Confiabilidade

### 3.3) *Aliasing* (apelido)

- ❑ É ter um ou mais métodos, ou nomes, distintos para fazer referência à mesma célula de memória.

- ❑ Exemplos em C

```
char i=`x`;  
char *p;  
p=&i;  
*p=`z`;
```

Ponteiros

```
union reg  
{  
    long i;  
    float f;  
}r;  
...  
r.f=1000;
```

Union

# **Critérios de avaliação de linguagens**

## **3) Confiabilidade**

### **3.4) Legibilidade e facilidade de escrita**

- ☐ Tanto a legibilidade como a facilidade de escrita influenciam a confiabilidade.
- ☐ Um programa escrito em uma linguagem que não suporta maneiras naturais de expressar os algoritmos exigidos usará necessariamente métodos não naturais, menos prováveis de serem corretas.

## Critérios de avaliação de linguagens

### 3) Confiabilidade

#### 3.4) Legibilidade e facilidade de escrita

- ☐ Quanto mais fácil é escrever um programa, mais probabilidade ele tem de estar correto.
- ☐ **Programas de difícil leitura complicam também sua escrita e sua modificação.**

## **Critérios de avaliação de linguagens**

### **4) Custo**

- ❑ “O custo final de uma linguagem de programação é uma função de muitas de suas características”

# **Critérios de avaliação de linguagens**

## **4) Custo**

### **4.1) Treinamento**

- ❑ Em função da simplicidade e da ortogonalidade da linguagem e da experiência dos programadores.



## Critérios de avaliação de linguagens

### 4) Custo

#### 4.2) Custo da escrita

- ❑ Os esforços originais para projetar e implementar linguagens de alto nível foram motivados pelos desejos de diminuir os custos para criar software.

## Critérios de avaliação de linguagens

### 4) Custo

#### 4.3) Sistema de implementação

- ❑ LP cujo sistema de implementação seja caro, ou rode somente em hardware caro, terá muito menos chance de tornar-se popular.
- ❑ Sucesso de Java.



## **Critérios de avaliação de linguagens**

### **4) Custo**

#### **4.4) Projeto da linguagem**

- ❑ Se uma LP exigir muitas verificações de tipos durante a execução, proibirá a execução rápida do código.

# Critérios de avaliação de linguagens

## 4) Custo

### 4.5) Compilação

❑ Problema amenizado com o surgimento de compiladores otimizados e de processadores mais rápidos.

# Critérios de avaliação de linguagens

## 4) Custo

### 4.6) Má confiabilidade

- ☐ Falhas podem ocasionar insucesso do software e ações judiciais.

### 4.7) Manutenção

- ☐ Depende principalmente da legibilidade.
- ☐ O custo de manutenção pode atingir de duas a quatro vezes o custo de desenvolvimento.

## Fatores que influenciam na escolha de uma LP

### ❑ Implementação

- Disponibilidade quanto à plataforma
- **Eficiência:** velocidade de execução do programa objeto

## Fatores que influenciam na escolha de uma LP

### ❑ Competência na LP

- Experiência do programador
- Competência do grupo envolvido

### ❑ Portabilidade

- Necessidade de executar em várias máquinas

## Fatores que influenciam na escolha de uma LP

### ☐ Sintaxe

- Certos tipos de aplicação acomodam-se melhor em certas sintaxes

### ☐ Semântica

- Aplicação X Facilidades
- Por exemplo, para processamento concorrente pode-se usar ADA, para utilização de recursividade pode-se usar Pascal.



## Fatores que influenciam na escolha de uma LP

### ❑ Ambiente de programação

- Ferramentas para desenvolvimento de software diminuem o esforço de programação
- Bibliotecas

### ❑ Modelo de computação

- Aplicação X modelo de computação
- Por exemplo, para realização de busca heurística é adequado o Paradigma Lógico, para simulações, o Paradigma Orientado a Objeto

# Exercício

- ☐ Pesquise sobre as LP citadas abaixo. Mostre seu histórico, características, importância, estrutura, versões e classificação (nível, geração e paradigma). Insira exemplos de código-fonte no relatório.
  - Algol, Pascal, Fortran
  - Basic, Cobol, Prolog
  - PL/I, Mumps, Clipper
  - Java, Assembly
- ☐ Pesquise sobre linguagens desenvolvidas no Brasil
- ☐ Fazer um relatório (word ou qualquer outro) e gerar PDF
- ☐ Grupo de até 4 pessoas
- ☐ Data limite: 13/03