

# Modelos de Linguagem de Programação

## Unidade 2 – Critérios de Avaliação de Linguagens



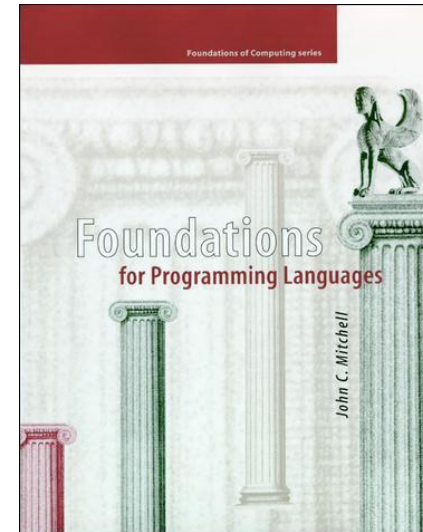
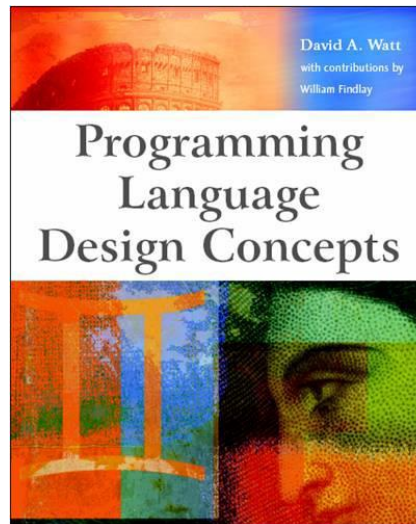
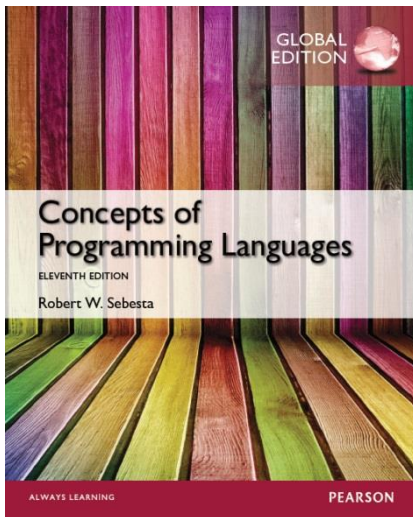
Prof. Aparecido V. de Freitas  
Doutor em Engenharia  
da Computação pela EPUSP  
[aparecidovfreitas@gmail.com](mailto:aparecidovfreitas@gmail.com)





# Bibliografia

- ❖ Sebesta, Robert W. Concepts of Programming Languages – Eleventh Edition
- ❖ Watt, D. Programming Language Design Concepts. John Wiley and Sons, 2004.
- ❖ Mitchell, J. Foundations for Programming Languages, MIT Press, 1996.





## CrITÉrios de Avaliação de Linguagens

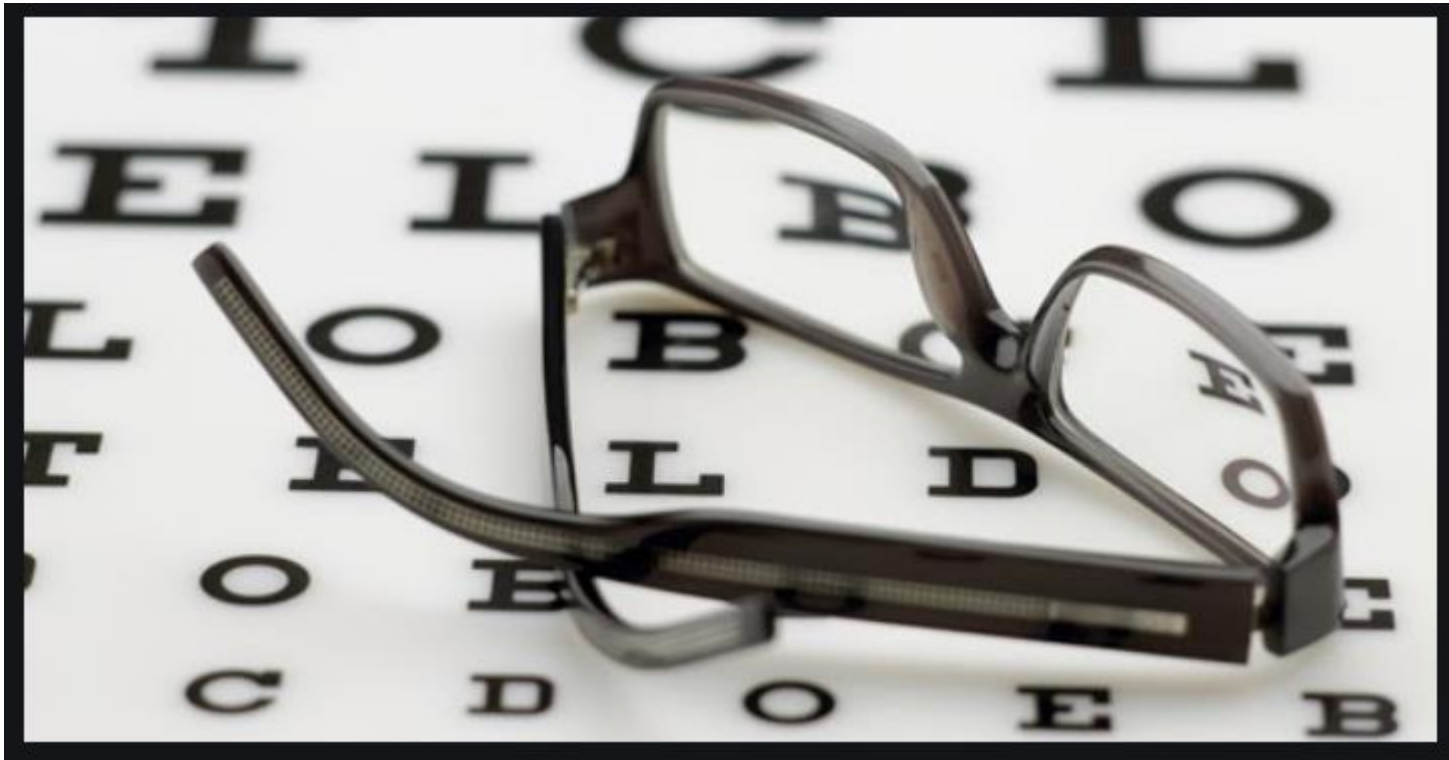
- Um dos critérios mais importantes para se julgar uma linguagem de programação é a facilidade com que os programas são **lidos e entendidos**;
- Antes dos anos **70**: linguagens foram criadas pensando-se em termos de **escrita** de código.
  - ✓ Eficiência e legibilidade de máquina;
  - ✓ As linguagens 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 (Crise do Software)**





# Critérios de Avaliação de Linguagens

## Legibilidade





# Critérios de Avaliação de Linguagens

## Legibilidade

- A facilidade de manutenção é determinada em grande parte, pela **legibilidade** dos programas, dessa forma ela se tornou uma medida importante da qualidade dos programas e das linguagens.
- A legibilidade deve ser considerada no contexto do **domínio do problema**.
  - Um programa escrito em uma linguagem não apropriada para o domínio do problema se mostra antinatural, "enrolado" e difícil de ser lido.







# Critérios de Avaliação de Linguagens

## Legibilidade

# 1

### Simplicidade geral:

- A simplicidade geral de uma linguagem de programação 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.
  - Isso pode ser um problema quando o leitor do programa aprende um conjunto diferente de recursos daquele que o autor aplicou em seu programa.



# Critérios de Avaliação de Linguagens

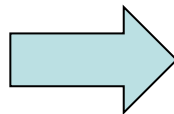
## Legibilidade

# 1

## PL/I – 1965

### Comentários:

- Muitas características novas tinham problemas de projeto
- Muito grande e complexo
- Foi de fato usado para aplicações científicas e comerciais
- Um sucesso parcial



### Contribuições:

- Primeiro tratamento de exceção (23 tipos)
- Tipo de dado ponteiro
- Referência a seções de arrays
- Etc.



# Critérios de Avaliação de Linguagens

## Legibilidade

# 2

### **Simplicidade geral:**

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

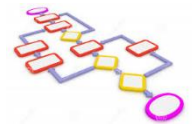
- Exemplo em C:

```
cont = cont + 1;  
cont += 1;  
cont++;  
++cont;
```

Mesmo significado  
quando usadas em  
expressões separadas!







# Critérios de Avaliação de Linguagens Legibilidade

## 3

### 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.
  - Exemplo: sobrecarregar o + para adicionar inteiros, reais, concatenar strings, somar vetores...



# Critérios de Avaliação de Linguagens

## Legibilidade

### 4

#### Simplicidade Geral

- ✓ A simplicidade de linguagens, no entanto, pode ser levada ao **extremo**, por exemplo a **forma** e o **significado** da maioria das instruções da **Linguagem Assembly** são modelos de simplicidade;
- ✓ No entanto, tornam os programas escritos em **Assembly ilegíveis**;
- ✓ **Faltam** instruções de controle mais **complexas**, exigindo que se usem mais comandos que o necessário para se expressar problemas, quando comparado à linguagens de alto nível.

```

00000000      push    ebp
00000001      mov     ebp, esp
00000003      movzx   ecx, [ebp+arg_0]
00000007      pop     ebp
00000008      movzx   dx, cl
0000000C      lea     eax, [edx+edx]
0000000F      add     eax, edx
00000011      shl     eax, 2
00000014      add     eax, edx
00000016      shr     eax, 8
00000019      sub     cl, al
0000001B      shr     cl, 1
0000001D      add     al, cl
0000001F      shr     al, 5
00000022      movzx   eax, al
00000025      ret     0
  
```



# Critérios de Avaliação de Linguagens

## Legibilidade

Programa 1 (Linguagem <i>Assembly</i> )	Programa 2 (Linguagem C)	Programa 3 (Linguagem C)
<pre>.INCLUDE "M32DEF.inc" .EQU Somatorio1a5 = 0x100 .EQU NLOOPS = 5 .EQU RLOOPS = R20 .EQU CNT = R17  .ORG 0     LDI    R16, 0     LDI    CNT, 1 LOOP1:     ADD    R16, CNT     INC    CNT     LDI    RLOOPS, NLOOPS     SUB    RLOOPS, CNT     TST    RLOOPS     BNEQ   LOOP1     STS    Somatorio1a5, R16 LOOPINFINITO:     JMP    LOOPINFINITO</pre>	<pre>#include &lt;avr/io.h&gt;  int main(void){      int Somatorio1a5 = 0, i;      for(i=1; i&lt;=5; i++)         Somatorio1a5 += i;      while(1){     } }</pre>	<pre>#include &lt;avr/io.h&gt;  int main(void){      int Somatorio1a5 = 0, i;      i=1;     while(i&lt;=5){         Somatorio1a5 += i;         i++;     }      while(1){     } }</pre>






# Critérios de Avaliação de Linguagens

## Legibilidade

### Linguagem Assembly



**codingground** SIMPLY EASY CODING | COMPILER AND EXECUTE ASSEMBLY ONLINE

New Project-20170809 | Compile | Execute | Share Code | main.asm x

```
1 section .text
2     global _start           ;must be declared for using gcc
3     _start:                 ;tell linker entry point
4     mov edx, len            ;message length
5     mov ecx, msg            ;message to write
6     mov ebx, 1              ;file descriptor (stdout)
7     mov eax, 4              ;system call number (sys_write)
8     int 0x80                ;call kernel
9     mov eax, 1              ;system call number (sys_exit)
10    int 0x80                ;call kernel
11
12 section .data
13
14 msg db 'Hello, world!',0xa ;our dear string
15 len equ $ - msg           ;length of our dear string
16
```



# Critérios de Avaliação de Linguagens Legibilidade



## Ortogonalidade:

- A ortogonalidade diz respeito a **possibilidade de combinar** entre si, sem restrições, os **componentes básicos** da linguagem para construir estruturas de controle e dados.
- **Exemplo:** permitir combinações de estruturas de dados, como arrays de estruturas;
- **Contra exemplo:** não permitir que um array seja usado como parâmetro para uma função;





# Critérios de Avaliação de Linguagens Legibilidade

## 6

### Ortogonalidade:

- A linguagem C possui dois tipos de dados estruturados, arrays e registros (struct), sendo que:
  - Registros podem ser retornados de funções, arrays não.
  - Parâmetros são passados por valor, a menos que sejam arrays – que obrigatoriamente são passados por referência.





# Critérios de Avaliação de Linguagens

## Legibilidade

# 7

### 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.
- Reconheceu-se que o uso indiscriminado de instruções `goto` reduz criticamente a legibilidade de programas.



# Critérios de Avaliação de Linguagens

## Legibilidade

# 7

### Instruções de controle:

- Restringir instruções `goto` das seguintes maneiras pode tornar os programas mais legíveis:
  - As instruções `goto` devem preceder seus alvos, exceto quando usadas para formar laços;
  - Os seus alvos nunca devem estar tão distantes;
  - O número de usos deve ser limitado;
- A partir do final da década de 60, as linguagens projetadas passaram a ter instruções de controle suficientes e portanto a necessidade da instrução `goto` foi quase eliminada.





# Critérios de Avaliação de Linguagens Legibilidade





# Critérios de Avaliação de Linguagens Legibilidade

## 8

### 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.
- **Exemplo:** supondo 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`
- **Outro avanço:** tipos enumerados.







# Critérios de Avaliação de Linguagens Legibilidade

## 9

### Considerações sobre sintaxe:

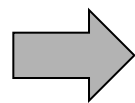
- A sintaxe ou a forma dos elementos de uma linguagem tem um efeito significativo sobre a legibilidade dos programas.
- Exemplos de opções de projeto sintático que afetam a legibilidade:
  - **Formas identificadoras:** restringir os identificadores a tamanhos muitos pequenos prejudica a legibilidade, impedindo que variáveis sejam nomeadas com nomes conotativos. Exemplos:
    - FORTRAN 77: máximo 6 caracteres;
    - BASIC ANSI: uma letra ou uma letra e um número;





# Critérios de Avaliação de Linguagens

## Legibilidade – Sintaxe



Formato Rígido na Linguagem **RPG**  
Nomes de Identificadores : **6 caracteres**

```

XMT D  DNome++++++ETDsDesde++A+/L+++IDc.Palabras clave+++++
0034.56 DnuContador      S              5  0
0034.57 D*****
0034.58 DfeedArr        S              N
0034.59 D*****
0034.60 D array          S             14  4 dim(31)
0034.61 Darreglo         S              5  0 DIM(10)
0034.62 Darreglo1        S              2  0 DIM(21)
0034.63 Darreglo2        S             4a   DIM(21)
0034.64 Darreglo3        S             2a   DIM(21)
0034.65 Dpa1             S              3   DIM(200)
0034.66 Dpa2             S             17   DIM(200)
0034.67 Dpa3             S              2   DIM(200)
0034.68 Dpa4             S              5   DIM(200)
0034.69 Darr1           S              5  0 DIM(200)
0034.70 D SEP            c              ' : : : : F I N : : : : '
0034.71 D barra          c              ' : : : : : : : : '
0034.72 D** AREA DE DATOS
  
```





# Critérios de Avaliação de Linguagens Legibilidade

## Considerações sobre sintaxe:

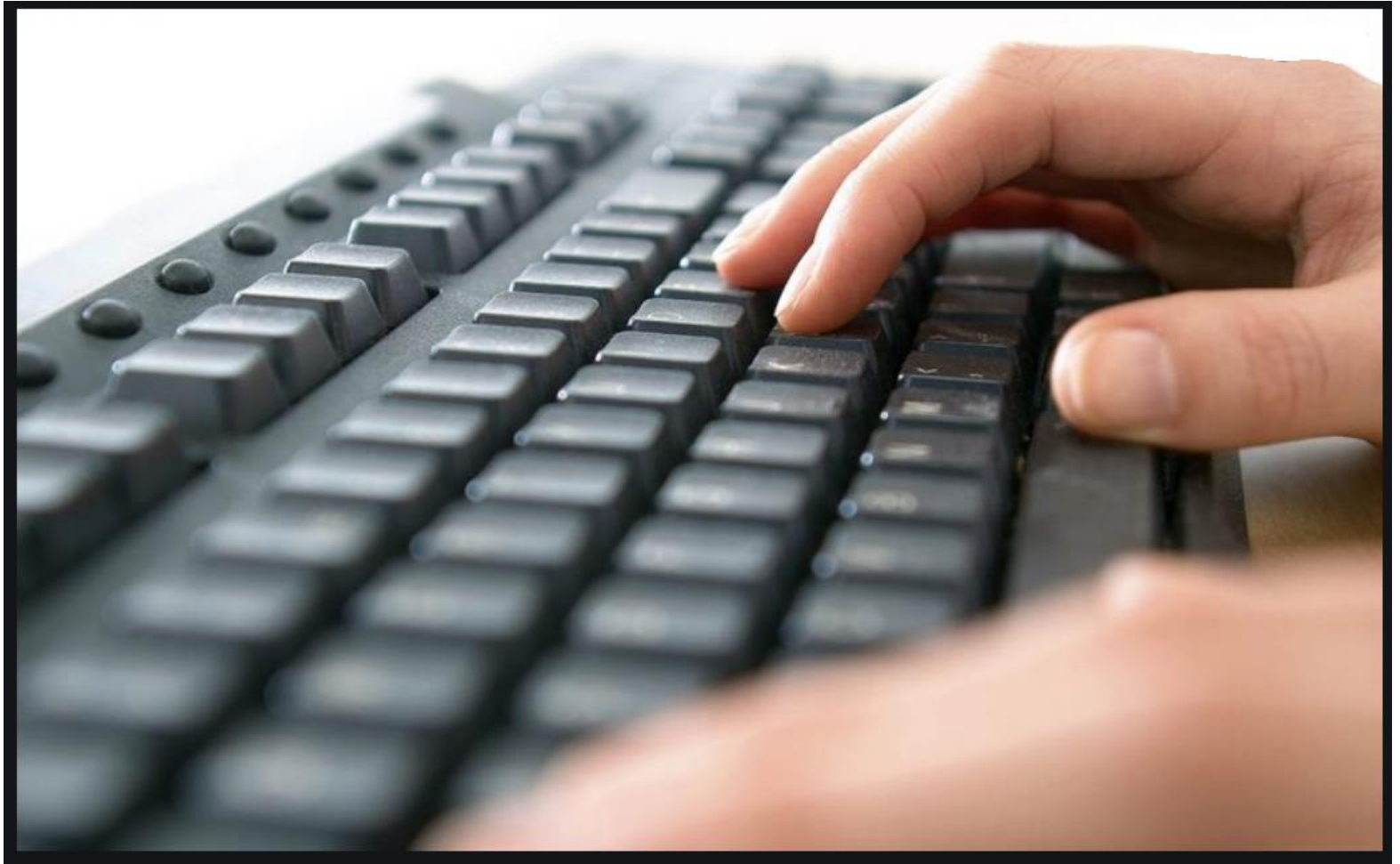
- Exemplos de opções de projeto sintático que afetam a legibilidade:
  - **Palavras especiais:** a aparência de um programa e sua consequente legibilidade são fortemente influenciadas pelas formas das palavras especiais de uma linguagem (begin, end, for...);
  - **Exemplos:** Pascal exige pares de begin/end para formar grupos em todas as construções de controle, a linguagem C usa chaves;





# Critérios de Avaliação de Linguagens

## Capacidade de Escrita





# Critérios de Avaliação de Linguagens

## Capacidade de Escrita

- A **capacidade de escrita** 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 capacidade de escrita;









# Critérios de Avaliação de Linguagens

## Capacidade de Escrita

### **Simplicidade e Ortogonalidade:**

- Se uma linguagem de programação 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;
- Podem ocorrer usos de recursos desconhecidos com resultados não esperados.





# Critérios de Avaliação de Linguagens

## Capacidade de Escrita

### 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 funções provenientes de bibliotecas;
- **Tipos de Abstração:**
  - Abstração de Processo: algoritmos em geral;
  - Abstração de Dados: tipos de dados e estruturas de dados.





# Critérios de Avaliação de Linguagens

## Capacidade de Escrita

### Expressividade:

- Formas convenientes de especificar computações, onde uma expressão representa muitas computações.
- Exemplos:
  - `i++`, ao invés de `i = i + 1;`
  - `for` ao invés do `while`;
  - `cin` do C++ ao invés de `nextInt` do Java

```
int v;  
cin >> v;
```

```
int v;  
Scanner entrada;  
entrada = new Scanner(System.in);  
v = entrada.nextInt();
```





# Critérios de Avaliação de Linguagens

## Confiabilidade

- Um programa é **confiável** se ele se comportar de acordo com suas especificações sob todas as condições.
- Recursos que afetam a confiabilidade:
  - Verificação de Tipos;
  - Manipulação de Exceções;
  - Apelidos (Aliasing);
  - Legibilidade e Facilidade de Escrita;





# Critérios de Avaliação de Linguagens

## Confiabilidade

### Verificação de Tipos:

- Visa testar se existem erros de tipos de dados no programa 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 o erro, menos caro é fazer todos os reparos necessários.
  - Exemplo: Java
- A verificação de tipos em C é bastante fraca:

```
int vet[50];  
vet[100] = 8.5;
```





# Critérios de Avaliação de Linguagens

## Confiabilidade

### Manipulação de Exceções:

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

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

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



# Critérios de Avaliação de Linguagens

## Confiabilidade

### Apelidos (Aliasing):

- Consiste em ter um ou mais métodos, ou nomes, distintos para fazer referência à mesma área de memória.

- Exemplos em C:

```
char c = 'x';  
char *p;  
p = &c;  
*p = 'z';
```

```
union Data{  
    int i;  
    float f;  
    char str[20];  
};  
union Data data;  
data.i = 10;  
data.f = 220.5;  
strcpy(data.str, "Programa");
```





# Critérios de Avaliação de Linguagens Confiabilidade

## **Legibilidade e Facilidade de Escrita:**

- Tanto a legibilidade como a facilidade de escrita influenciam a confiabilidade.
- 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

## Custo

- Custo final de uma linguagem de programação é uma função de muitas de suas características.
- **Custo de Treinamento:** cresce em função da simplicidade e da ortogonalidade da linguagem e da experiência dos programadores;
- **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

## Custo

- **Custo do Sistema de Implementação:** uma linguagem de programação cujo sistema de implementação seja caro, ou rode somente em hardware caro, terá muito menos chance de tornar-se popular;
- **Custo do Projeto da Linguagem:** se uma linguagem de programação exigir muitas verificações de tipos durante a execução, proibirá a execução rápida do código;
- **Custo de Compilação:** problema amenizado com o surgimento de compiladores otimizados e de processadores mais rápidos.





# Critérios de Avaliação de Linguagens Custo

- **Custo da Má confiabilidade:** falhas podem ocasionar insucesso do software e ações judiciais;
- **Custo de Manutenção:** depende principalmente da legibilidade. O custo de manutenção pode atingir de duas a quatro vezes o custo de desenvolvimento;



# Como escolher uma linguagem?

- **Implementação:**
  - Disponibilidade quanto à plataforma;
  - Eficiência: velocidade de execução do programa objeto;
- **Competência:**
  - Experiência do programador;
  - Competência do grupo envolvido;
- **Portabilidade:**
  - Necessidade de executar em várias máquinas diferentes;



# Como escolher uma linguagem?

- **Sintaxe:**
  - Certos tipos de aplicação acomodam-se melhor em certas sintaxes;
- **Semântica:**
  - Algumas linguagens são mais adequadas para processamento concorrente, outras são mais otimizadas para recursividade;



# Como escolher uma linguagem?

- **Ambiente de programação:**
  - Ferramentas para desenvolvimento de software diminuem o esforço de programação;
  - Existência de bibliotecas;
- **Modelo de computação**
  - O Paradigma Lógico pode ser mais adequado para algumas aplicações de inteligência artificial, enquanto que o Paradigma Orientado a Objeto é mais adequado para aplicações comerciais;







# Trabalho 1

- Em dupla, pesquise sobre a **linguagem de programação** sorteada para a sua dupla. Mostre o histórico da linguagem, características, importância, estrutura, vantagens/desvantagens, versões e classificação (nível, geração e paradigma). Apresente exemplos de código-fonte.
- **A dupla deverá:**
  - Apresentar a pesquisa em aula (15 minutos para cada dupla);
  - Entregar um relatório (em PDF);

