

Operadores

- Atribuição
- Aritméticos
- Relacionais
- Lógicos

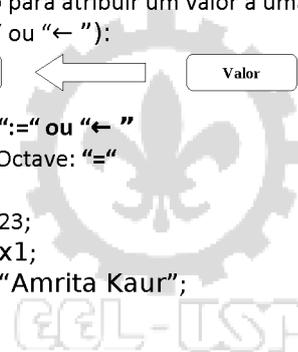


1

Operador de Atribuição

- Utilizado para atribuir um valor a uma variável ("`=`" ou "`:=`" ou "`←`"):

- Visualg: "`:=`" ou "`←`"
- Matlab/Octave: "`=`"
- Notação:
 - x1 ← 23;
 - temp ← x1;
 - nome ← "Amrita Kaur";



2

Linearização de Expressões

- ↳ Para a construção de algoritmos que realizam cálculos matemáticos, todas as expressões aritméticas devem ser linearizadas;
- ↳ Devendo também ser feito o mapeamento dos operadores da aritmética tradicional para os do Português Estruturado.

$\left\{ \left[\frac{2}{3} - (5-3) \right] + 1 \right\} . 5$	$((2/3 - (5-3)) + 1) * 5$
Tradicional	Computacional

3

Operadores Aritméticos

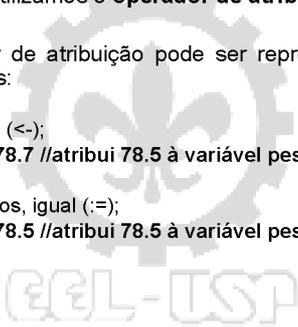
OPERADORES ARITMÉTICOS	PORTUGUÊS ESTRUTURADO
Adição	+
Subtração	-
Multiplicação	*
Divisão	/
Divisão Inteira	\
Exponenciação	^ ou Exp (<base>, <expoente>)
Módulo (resto da divisão)	%



4

Operadores Aritméticos

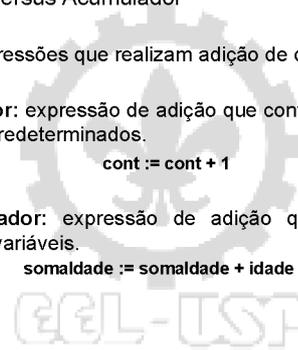
- ↳ Para colocar um valor em uma variável dentro de um algoritmo, utilizamos o **operador de atribuição**;
- ↳ O operador de atribuição pode ser representado de duas formas:
 - ↳ Uma seta (<-);
peso <- 78.7 //atribui 78.5 à variável peso
 - ↳ Dois pontos, igual (:=);
peso := 78.5 //atribui 78.5 à variável peso



5

Operadores Aritméticos

- ↳ Contador versus Acumulador
 - ↳ São expressões que realizam adição de dados
 - ↳ **Contador**: expressão de adição que contabiliza valores fixos e predeterminados.
cont := cont + 1
 - ↳ **Acumulador**: expressão de adição que contabiliza valores variáveis.
somalidade := somalidade + idade



6

Operadores Relacionais

OPERADORES RELACIONAIS	PORTUGUÊS ESTRUTURADO
Maior	>
Menor	<
Maior ou igual	>=
Menor ou igual	<=
Igual	=
Diferente	<>

Os operadores relacionais realizam a comparação entre dois operandos ou duas expressões e resultam em valores lógicos (VERDADEIRO ou FALSO).

7

Operadores Relacionais

- Exemplos:
 - cond1 ← 2 = 3 // (falso)
 - cond2 ← 1.6 <> 5.0 // (verdadeiro)
 - cond3 ← 1 > 5 // (falso)
 - cond4 ← (1 + 2) < 5 // (verdadeiro)
 - cond5 ← 10 >= 3 // (verdadeiro)
 - cond6 ← 1 <= 4 // (verdadeiro)
 - cond7 ← "café" < "expresso" // (verdadeiro)
 - cond8 ← "café" = "café" // (verdadeiro)
 - cond9 ← "café" >= "mocha" // (falso)

8

Tabela Verdade

A	B	A E B	A OU B	NÃO A	NÃO B
VERDADEIRO	VERDADEIRO	VERDADEIRO	VERDADEIRO	FALSO	FALSO
VERDADEIRO	FALSO	FALSO	VERDADEIRO	FALSO	VERDADEIRO
FALSO	VERDADEIRO	FALSO	VERDADEIRO	VERDADEIRO	FALSO
FALSO	FALSO	FALSO	FALSO	VERDADEIRO	VERDADEIRO

9

Operadores Lógicos

- Dados de entrada: tipo lógico
- Resultado: tipo lógico
- E (AND), OU (OR), NAO (NOT)

OPERADORES LÓGICOS	PORTUGUÊS ESTRUTURADO	SIGNIFICADO
Multiplicação lógica	E	Resulta VERDADEIRO se ambas as partes forem verdadeiras.
Adição lógica	Ou	Resulta VERDADEIRO se uma das partes é verdadeira.
Negação	Nao	Nega uma afirmação, invertendo o seu valor lógico: se for VERDADEIRO torna-se FALSO, se for FALSO torna-se VERDADEIRO.

10

Precedência de Operadores

OPERADOR ARITMÉTICO	PRIORIDADE
Exponenciação	3 (maior)
Multiplicação	2
Divisão	2
Adição	1
Subtração	1 (menor)

OPERADOR LÓGICO	PRIORIDADE
e	3
ou	2
nao	1

OPERADOR	PRIORIDADE
Operadores aritméticos	3
Operadores relacionais	2
Operadores lógicos	1

11

Parênteses

- Com eles você assegura que as expressões sejam avaliadas como você espera que sejam.
- Pode evitar erros com precedência de operadores.

Exemplos:

$$\diamond Y = m * x + b$$

$$\diamond Y = (m * x) + b$$

$$\diamond Y = a * b * b + c * b - d$$

$$\diamond Y = (((a * b) * b) + c (c * b)) - d$$

12

Comentários

- Utilizados para descrever trechos do código (documentar)
 - {} (Farrer) ou
 - // (C++, Visualg) ou
 - /* (Java)
 - % (Matlab/Octave)

13

Computação Aplicada à Engenharia

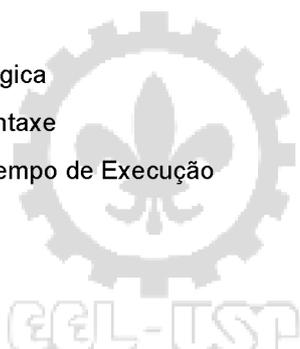
Erros em Programação



14

Erros em Programação

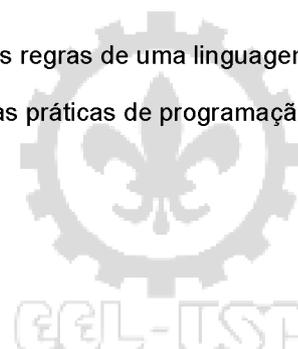
- ↳ Erro de Lógica
- ↳ Erro de Sintaxe
- ↳ Erro em Tempo de Execução



15

Sintaxe x Estilo

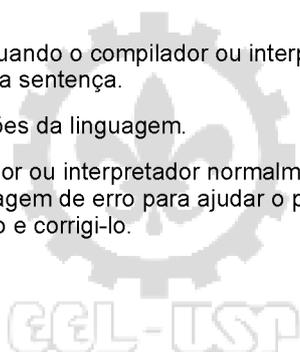
- ↳ Sintaxe: as regras de uma linguagem.
- ↳ Estilo: boas práticas de programação.



16

Erro de Sintaxe

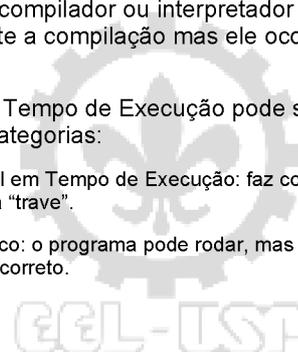
- ↳ Causado quando o compilador ou interpretador não reconhece a sentença.
- ↳ São violações da linguagem.
- ↳ O compilador ou interpretador normalmente retorna uma mensagem de erro para ajudar o programador a localizá-lo e corrigi-lo.



17

Erro em Tempo de Execução e Lógico

- ↳ Quando o compilador ou interpretador não retorna erro durante a compilação mas ele ocorre em execução.
- ↳ O Erro em Tempo de Execução pode ser dividido em duas categorias:
 - ↳ Erro Fatal em Tempo de Execução: faz com que o programa "trave".
 - ↳ Erro Lógico: o programa pode rodar, mas o resultado não está correto.



18

Computação Aplicada à Engenharia

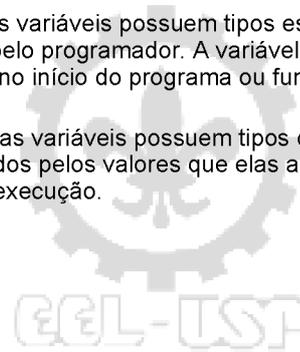
Tipagem Estática x Tipagem Dinâmica



19

Tipagem Estática x Tipagem Dinâmica

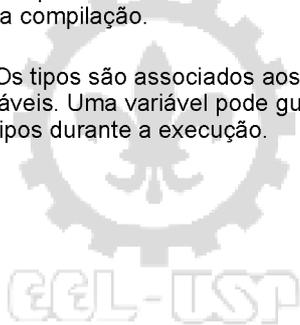
- ♦ Estática: as variáveis possuem tipos estáticos definidos pelo programador. A variável precisa ser declarada no início do programa ou função.
- ♦ Dinâmica: as variáveis possuem tipos dinâmicos determinados pelos valores que elas assumem em tempo de execução.



20

Tipagem Estática x Tipagem Dinâmica

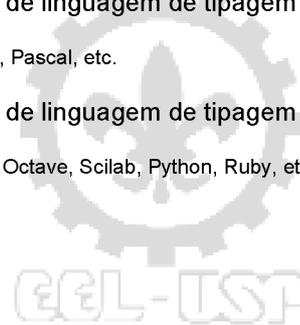
- ♦ Estática: tem tipos declarados – fixados no momento da compilação.
- ♦ Dinâmica: Os tipos são associados aos valores e não as variáveis. Uma variável pode guardar diferentes tipos durante a execução.



21

Tipagem Estática x Tipagem Dinâmica

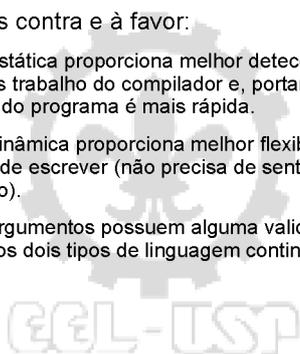
- ♦ Exemplos de linguagem de tipagem estática:
 - ♦ C, Java, Pascal, etc.
- ♦ Exemplos de linguagem de tipagem dinâmica:
 - ♦ Matlab, Octave, Scilab, Python, Ruby, etc.



22

Tipagem Estática x Tipagem Dinâmica

- ♦ Argumentos contra e à favor:
 - ♦ A tipagem estática proporciona melhor detecção de erros, exige mais trabalho do compilador e, portanto, a execução do programa é mais rápida.
 - ♦ A tipagem dinâmica proporciona melhor flexibilidade, é mais fácil de escrever (não precisa de sentenças de declaração).
 - ♦ Ambos os argumentos possuem alguma validade e, portanto, os dois tipos de linguagem continuarão a existir.



23