

**Prof. Márcio Senhorinha**

**E-mail**

**[marcio.senhorinha@edu.sc.senai.br](mailto:marcio.senhorinha@edu.sc.senai.br)**

# Algoritmos

# Estrutura De Algoritmos

# Definição

**A construção de algoritmos é a etapa mais importante em programação.**

**Se já sabemos a sequência de etapas a serem feitas, a codificação destes comandos em um programa torna-se fácil.**

# Definição

O desafio, portanto, **não é sentar à frente de um computador e programar, mas sim saber o quê programar.**



# ATENÇÃO

Deste ponto em diante,  
**nosso enfoque será**  
**construção de algoritmos**  
**estruturados** para problemas  
passíveis de serem  
executados em um computador.

**Vamos utilizar uma forma escrita de algoritmos que atenda aos seguintes objetivos:**

✓ Conversão mais fácil

Construiremos algoritmos em uma notação que seja **mais próxima das linguagens de programação**



✓ Ser fácil de entender por um programador

Algoritmos devem ser feitos principalmente **para que uma pessoa os entenda**

✓ Ter foco no problema

**Sem se preocupar** com os detalhes técnicos, ou seja, limitações de máquina

# Português Estruturado

## Conceitos Iniciais

# INTRODUÇÃO

Até esse momento, quando escrevemos algoritmos, **nossa opção foi escrever, linha a linha**, uma lista de instruções **ou representar o algoritmo em um fluxograma.**

# INTRODUÇÃO

A partir de agora, passaremos a construir nossos algoritmos em português estruturado (**também chamado de portugol**).

# INTRODUÇÃO

O português estruturado possui as seguintes características:

- ✓ Utiliza um conjunto muito mais limitado de comandos do que a língua normal (*no entanto possibilita a construção de algoritmos simples e complexos*);
- ✓ Cada comando possui interpretação única;
- ✓ Produz uma lista ordenada de comandos.

# INTRODUÇÃO

Na sequência estudaremos alguns conceitos e regras básicas para construção de algoritmos em português

# SINTAXE e SEMANTICA

## Sintaxe

É o nome do conjunto de regras a serem seguidas para a escrita de algoritmos

## Semântica

Refere-se ao que é efetuado pelo computador, ou quem executa o algoritmo, quando é encontrado um comando



# TIPOS DE DADOS

Toda a fundamentação da construção de algoritmos é feita a partir do dado.

Podemos dizer que dado é alguma informação em estado primitivo.

Dado é uma representação de uma situação, cujo processamento pode gerar uma informação útil

# TIPOS DE DADOS

Um **tipo de dado** representa o conjunto de valores possíveis para um dado, e **nós usaremos nos algoritmos cinco tipos de dados básicos:**

# TIPOS DE DADOS

## ➤ NUMÉRICOS

Os dados numéricos dividem-se em dois grupos: ***inteiros e reais.***

# TIPOS DE DADOS

## Inteiro

Representa qualquer valor contido no conjunto dos números inteiros, podendo ser **positivo** ou **negativo**.

**Exemplo, 1, 10, -35, 0, etc...**

## Real

Representa qualquer valor contido no conjunto dos números reais, podendo ser **positivo** ou **negativo** por exemplo, 8.5, 3.14, 10, -0.30, etc...

# TIPOS DE DADOS

## Lógico

Representa um estado, uma situação que só pode ser **verdadeira** ou **falsa**, também são conhecidos como dados booleanos (e nenhuma outra a mais)

## Caractere

Representa um caractere a ser armazenado na memória, por exemplo, **A**, **B**, **7**, **6**, **#**, **\$**, etc...

# TIPOS DE DADOS

## Cadeia de Caracteres

Representa um conjunto de caracteres a ser armazenado na memória, que pode ser composto por letras, dígitos ou outros símbolos, por **exemplo, Sesi, Senai, 321-3131, etc...**

**Obs.:** Em algoritmos, sempre que quisermos escrever uma cadeia de caracteres, ou apenas um caractere, **estes serão envolvidos por aspas simples (').** **Exemplo: 'A', 'senai'**

# Exercício 005

# EXERCÍCIOS 005

Identifique para cada informação abaixo qual o tipo de dado utilizar.

- Salário do funcionário de uma empresa
- Quantidade de vendedores de uma loja
- Número de latas de refrigerante em uma prateleira
- Nome de um assinante de linha telefônica
- Preço de um litro de leite

**CONTINUA...**



# EXERCÍCIOS 005

## ...CONTINUAÇÃO

- Estado de iluminação de uma lâmpada em perfeitas condições
- Número de alunos de uma turma
- Sexo de uma pessoa
- Placa de um automóvel
- Nome de um funcionário
- Um símbolo de “arroba” no endereço eletrônico de um colega
- Saldo bancário de um cliente

**Fim!**

# Identificadores

# Formação de Identificadores

## Os identificadores são:

- Os nomes das variáveis;
- Nomes dos programas;
- Nomes das constantes;
- Nome das rotinas;
- Nome das unidades;
- Tipo;
- Nome dos subprograma;
- Etc.

# Formação de Identificadores

**As regras básicas para a formação dos identificadores são:**

- Os caracteres permitidos são: os números, as letras maiúsculas, as letras minúsculas e o caractere sublinhado;

Os identificadores são formados com letras de “a” até “z”, dos dígitos de “0” até “9” e do **caractere especial “\_”**.

- O primeiro caractere deve ser sempre uma letra ou o caractere sublinhado;

# Formação de Identificadores

**As regras básicas para a formação dos identificadores são:**

- Não são permitidos espaços em branco e caracteres especiais (@, \$, +, -, %, !);
- Não podemos usar as palavras reservadas nos identificadores, ou seja, palavras que pertençam à linguagem de programação.

# Formação de Identificadores

## Exemplos de Identificadores - VÁLIDOS

A  
a  
nota  
NOTA  
x5  
A32

NOTA1  
MATRICULA  
nota\_1  
dia  
IDADE

Palavras reservadas são símbolos que possuem significado definido no algoritmo, não podendo ser redefinidos ou usados como nome de identificador

## Exemplos

- Algoritmo;
- Variáveis;
- Início;
- Fim;
- Se;
- Senão;
- Entre outras, ...

# Palavras Reservadas

## Exemplos de Identificadores – NÃO VALIDOS

**5b** → *por começar com número;*

**e 12** → *por conter espaço em branco;*

**x-y** → *por conter o caractere especial **-** ;*

**prova 2n** → *por conter espaço em branco;*

**nota (2)** → *por conter os caracteres especiais **()***

**case** → *por ser palavra reservada;*

**SET** → *por ser palavra reservada.*



A indentação é a forma como as linhas de um algoritmo estão fisicamente dispostas.

Uma boa indentação facilita a compreensão dos algoritmos

# Variável

Duas pessoas estão conversando e precisam realizar uma conta.

**A primeira pessoa** diz: “Vamos somar dois números”.

E continua: “O primeiro número é 5”.

**A segunda pessoa** guarda o primeiro número na cabeça, ou seja, na memória.

**A primeira pessoa diz:** “O segundo número é 3”.

**A segunda pessoa** também guarda o segundo número na cabeça, sem esquecer o primeiro número, ou seja, cada número foi armazenado em posições diferentes da memória humana, sem sobreposição.

**A primeira pessoa** pergunta:

“ - Qual é o resultado da soma?”

**A segunda pessoa** resgata os valores armazenados na memória, realiza a conta e **responde dizendo que o resultado é 8.**

Uma **variável** é um local com um nome dentro da memória do computador, **criado em um algoritmo para se armazenar um determinado dado.**

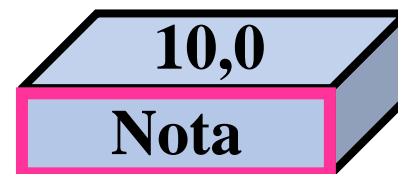
# VARIAVEL

Uma analogia muito comum é **imaginar uma variável como uma caixinha dentro da memória do computador**

Esta caixa tem a **capacidade de guardar um dado de certo tipo**, e o **nome da variável é como uma etiqueta colocada na frente da caixa**

# VARIÁVEL

## MEMÓRIA



As variáveis **existem na memória durante a execução do algoritmo.**

**Quando ele termina**, é como se **todos os dados fossem apagados, e as caixas destruídas**



As regras para utilização de variáveis em algoritmos são as seguintes:

- Toda variável deve ter um nome definido pelo programador, que deve ser único dentro de um mesmo algoritmo;
- O nome de uma variável é construindo seguindo a regra de construção dos identificadores

# VARIÁVEL

- O nome de uma variável é construído seguindo a regra de construção dos identificadores

## Alguns exemplos de nomes de variáveis válidos

**valor\_pago**, **soma**, **num**, **a**, **b**, **nome\_do\_funcionario**, **idade**

## Alguns exemplos de nome de variáveis inválidos

**1nota**, **média**, **quantidade inicial**, **valor\_em\_R\$**

Toda variável deve ter um tipo a ela relacionado

A declaração de uma variável segue a seguinte sintaxe:

```
<variável1>, <variável2>, ... : <tipo>;
```

## Alguns exemplos de declaração de variáveis:

- numerador, denominador : **inteiro**;
- resultado, saldo : **real**;
- nome\_funcionario : **cadeia**;

# Operadores

# OPERADORES

Para que possamos ter **processamento entre variáveis**, e até mesmo constantes, é **preciso conhecermos os operadores** que **são passíveis de uso em algoritmos**.

# OPERADORES

## ARITMÉTICOS

Os operadores aritméticos são símbolos conectivos usados **para efetuar algum cálculo numérico**:

SIMBOLO	CÁLCULO
+	Soma
-	Subtração
*	Multiplicação
/	Divisão
%	Resto da Divisão Inteira

## ARITMÉTICOS

Sabendo que **A** e **B** são duas  
variáveis inteiras, observe alguns exemplos  
da utilização de operadores aritméticos:

➤ **A + B**

➤ **A % B**

➤ **A \* 10**



## RELACIONAIS

Um operador relacional existe para estabelecer uma relação entre dois elementos, cujo resultado será sempre **FALSO** ou **VERDADEIRO**

SIMBOLO	RELAÇÃO
=	Igual a
>	Maior que
<	Menor que
>=	Maior ou igual a
<=	Menor ou igual a
< >	Diferente de

## RELACIONAIS

Sabendo que **A** e **B** são duas variáveis inteiras, observe alguns exemplos da utilização de operadores relacionais:

**A >= 10**

**A <> B**

**11 > 12**

# OPERADORES

## LÓGICO

Estes operadores permitem **estender o uso dos operadores relacionais**, estabelecendo composições lógicas mais sofisticadas.

SIMBOLO	RELAÇÃO
E	Conjunção
OU	Disjunção
NÃO	Negação

## LÓGICO

- **E** : Retorna **VERDADEIRO**, se e somente se, os dois operandos forem **VERDADEIROS**, caso contrário, retorna **FALSO**
- **OU** : Retorna **VERDADEIRO** se pelo menos um dos operandos for **VERDADEIRO**, caso contrário, retorna **FALSO**
- **NÃO** : Retorna o valor contrário do operando

## LÓGICO

Sabendo que **L1** e **L2** são duas variáveis lógicas, observe alguns exemplos da utilização de operadores lógicos:

**L1 E L2**

**NÃO 4=4**

**L1 OU L2**

# Funções

# FUNÇÕES

Da mesma forma que os operadores, uma função também permite efetuar um cálculo, porém as funções **não são conectivos**

Elas atuam como um pequeno programa que, tendo valores de entrada, geram uma resposta (retorno de valor)

As funções que podem ser utilizadas em nossos algoritmos são:

FUNÇÃO	RETORNO
RAIZ(x)	Raiz quadrada de x
TRUNC(x)	Valor inteiro de x sem parte decimal
ARRED(x)	Valor inteiro mais próximo de x
ABS(x)	Valor de x sem sinal



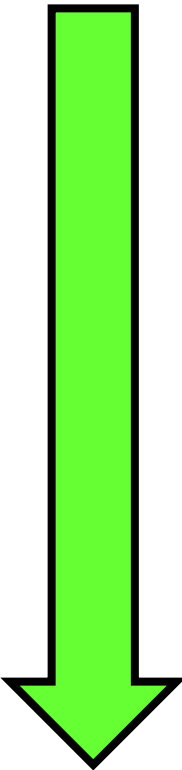
Toda expressão poderá ser montada utilizando apenas os operadores mostrados anteriormente

Os operandos de uma expressão podem ser variáveis, constantes e funções (utilizar apenas as funções definidas anteriormente)

**Toda expressão pode possuir parênteses para priorizar determinados cálculos**

**A tabela a seguir apresenta a prioridade assumida para o cálculo de expressões**

# MONTAGEM DE EXPRESSÕES



Prioridade no Cálculo de Expressões	
1º Lugar	Parênteses Internos
2º Lugar	Funções
3º Lugar	Operadores Aritméticos
4º Lugar	Operadores Relacionais
5º Lugar	Operadores Lógicos

1º os (\*, /, %)

2º os (+, -)

1º o NÃO

2º o E

3º o OU

Havendo algum empate em uma expressão, **deve-se fazer primeiro o cálculo da esquerda**

**Colchetes** e **chaves** utilizados na matemática para priorizar determinados cálculos serão substituídos por **parênteses**

## Exemplo

- $3 + 4 * 9 \rightarrow (resp. 39)$
- $(3 + 4) * 9 \rightarrow (resp. 63)$
- $RAIZ(4) * 10 + 3 \rightarrow (resp. 23)$
- $RAIZ(4) * (10 + 3) \rightarrow (resp. 26)$
- $((5 + 3) - 2) * 6 \rightarrow (resp. 36)$
- $10 > 9 \text{ E } 5 + 4 > 14 \rightarrow (resp. falso)$

# Exercício 006

**Indique a ordem e o resultado das expressões a seguir**

a)  $2 + 3 - 5 * 8 / 4$

b)  $7 * 4 / 2 + 9 - 6$

c)  $RAIZ(16) * 5 \% 2$

d)  $3 * ARRED(1.09 + 4 / 10)$

e)  $7 \% 3 + 8 - 4$

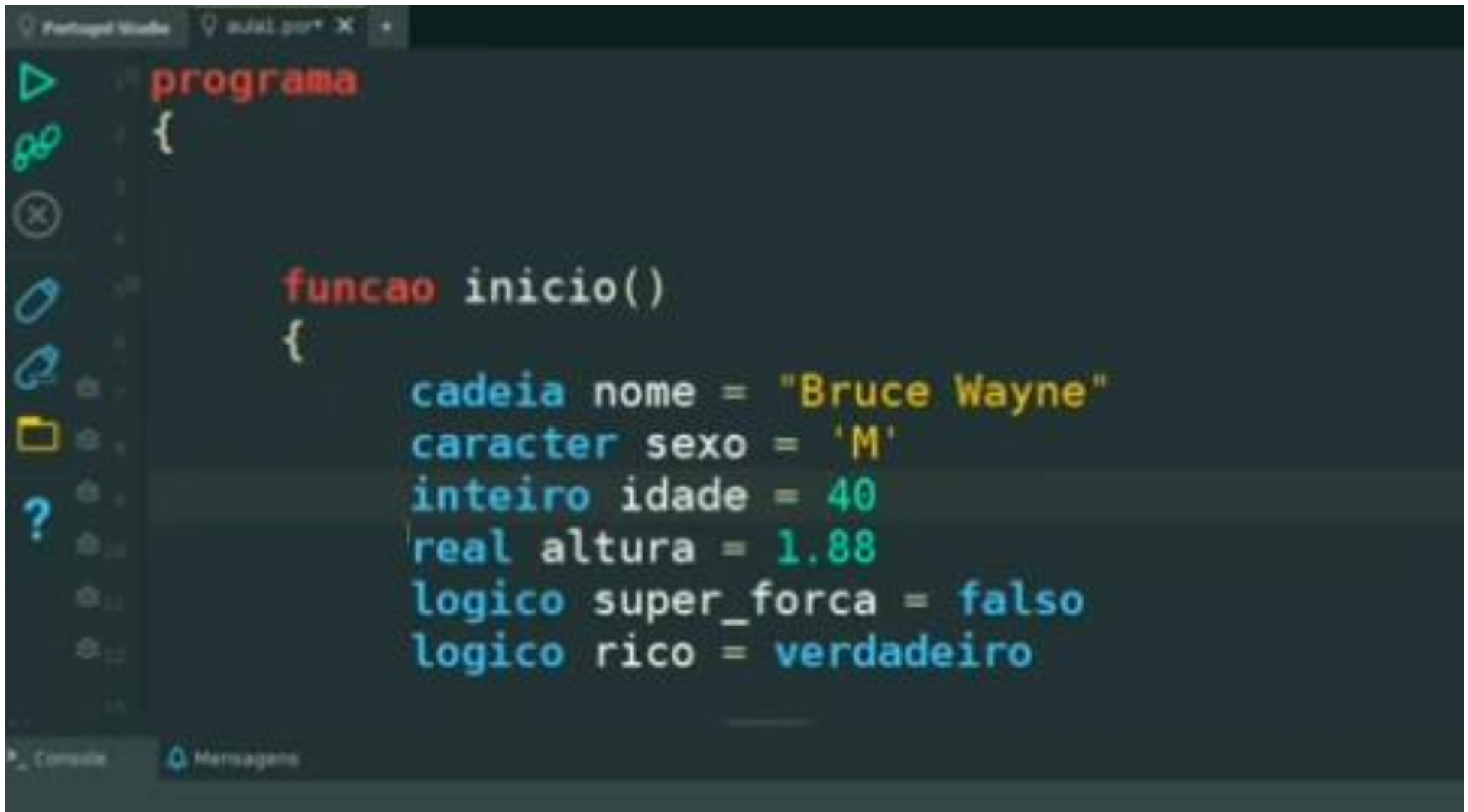
f)  $10 > 11 \text{ E } 11 < 12$

g)  $10 > 11 \text{ OU } 11 < 12$

h)  $N\tilde{A}O(10 > 11) \text{ E } 11 < 12$

i)  $N\tilde{A}O(3 + 5 < > 5 / 2 + 1)$

j)  $11 / ((2*3) + (2+2*(9-5))/2)$



```

programa
{

    funcao inicio()
    {
        cadeia nome = "Bruce Wayne"
        caracter sexo = 'M'
        inteiro idade = 40
        real altura = 1.88
        logico super_forca = falso
        logico rico = verdadeiro
    }
}
  
```



# FIM

# REFERENCIAS

**Slide Logica de Programação** – Carlos Iran Chiarello  
chiarello@spei.br

**Fundamentos da Programação de Computadores /**  
ASCENCIO, Ana Fernanda Gomes; CAMPOS, Edilene aparecida  
Veneruchi ; 3ª. ed. – São Paulo : Pearson Addison Wesley,  
2011.