

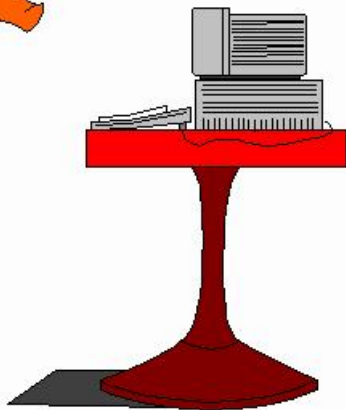
# Algoritmos: Introdução

Prof<sup>a</sup>.Dr<sup>a</sup>.Thatyana de Faria Piola Seraphim (ECO)  
Prof.Dr.Enzo Seraphim (ECO)

Universidade Federal de Itajubá

thatyana@unifei.edu.br  
seraphim@unifei.edu.br

- Harry Farrer. "**Programação Estruturada de Computadores: Algoritmos Estruturados**". 3ª Edição. L.T.C. 2010.
- Ana Fernandes Gomes Ascêncio; Edilene Aparecida Veneruchi de Campos. "**Fundamentos da Programação de Computadores: Algoritmos, Pascal e C/C++**". Prentice Hall. 2002.
- Victorine Viviane Mizrahi. "**Treinamento em Linguagem C**". McGraw-Hill do Brasil. 1990. vol. 2.
- Brian W.Kernighan; Dennis M. Ritchie. "**C a Linguagem de Programação**". Editora Campus. 1986.
- Herbert Schildt. "**C: Completo e Total**". Makron Books do Brasil/McGraw-Hill. 1991.
- Herbert Schildt. "**Linguagem C: Guia do Usuario**". McGraw-Hill do Brasil. 1986.
- Herbert Schildt. "**Linguagem C: Guia Prático e Interativo**". Makron. 1989.



- Lógica, Algoritmos, Programa.
- Portugol: entrada, processamento e saída.
- Variáveis, operadores e expressões.
- Comando de entrada e saída.

### A Lógica:

- trata da correção do pensamento, procurando saber porque pensamos assim e não de outro jeito. Nos ensina a usar corretamente as leis do pensamento;
- é a arte de pensar corretamente, e visto que a forma mais complexo do pensamento é o raciocínio, a Lógica estuda ou tem em vista a *correção do raciocínio*;
- ensina a colocar ordem no pensamento.

### Exemplo

A gaveta está fechada. A chave está dentro da gaveta. É preciso primeiro abrir a gaveta, para depois pegar a chave.

### Lógica de Programação

- É necessária para as pessoas que desejam trabalhar com desenvolvimento de programas e sistemas.
- Permite definir uma sequência natural de atividades com a intenção de atingir um objetivo.

### Lógica de Programação

É a técnica de encadear pensamentos em uma sequência lógica para atingir um determinado objetivo.

- Os pensamentos ou atividades podem ser escritos como uma sequência de instruções que devem ser seguidas para cumprir uma determinada tarefa.

### Sequência Lógica

São os passos que serão executados até atingir um objetivo ou a solução do problema.

- As instruções são um conjunto de regras ou normas definidas para a realização ou emprego de algo. Uma única ordem não permite realizar o processo completo, é necessário um conjunto de instruções colocadas em ordem sequencial e lógica.



- As instruções devem ser executadas em uma ordem adequada.
- Para que seja possível obter um resultado é necessário colocar em prática o conjunto de todas as instruções, na ordem correta.

### Instruções

São um conjunto de regras ou normas definidas para a realização ou emprego de algo. É o que indica a um computador uma ação elementar a executar.



- Um algoritmo é formalmente uma sequência finita de passos que levam a execução de uma determinada tarefa.
- Os algoritmos são muito comuns no nosso dia-a-dia, por exemplo, **receita de bolo**.
  - Uma sequência de diversos passos a serem cumpridos para que se consiga fazer determinado tipo de bolo.

### Algoritmo

É uma sequência de instruções ordenadas de forma lógica para a resolução de uma determinada tarefa ou problema.

### Bolo de Fubá

Colocar em uma panela

- 2 xícaras de chá de açúcar
- 2 " " " " " fubá
- 2 " " " " " leite
- 1 xícara de chá rasa de óleo
- 1 colher de chá deerva doce
- 1 pitada de sal.

(Entrada)  
Inf.  
Iniciais

### Modo de Fazer (Processar)

- misturar tudo muito bem, levar ao fogo mexendo até desprender do fundo da panela;
- Deixar esfriar.
- Bater bem 4 ovos inteiros, misturar na massa e colocar 1 colher de sopa de pó royal, despejar em forma untada e polvilhada com trigo levar para assar 25 a 30 minutos.
- Enrir com capizinho quente ou chocolate bem quente (bom né ?)

### Como trocar uma lâmpada

- Equipamentos: → entrada

- 1 escada;
- 1 lâmpada nova.

- Procedimento:

- Sequências de passos ou instruções*
- pegue uma escada;
  - posicione-a embaixo da lâmpada queimada;
  - busque uma lâmpada nova;
  - suba na escada;
  - retire a lâmpada velha;
  - coloque a lâmpada nova;
  - desce da escada;
  - jogue a lâmpada velha no lixo.

- Resultado:

- iluminação funcionando;
- mamãe feliz.

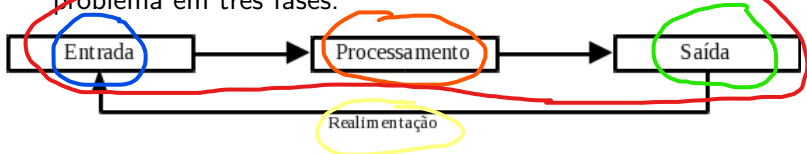
- Os algoritmos podem ser descritos em uma linguagem chamada **pseudocódigo**.
- Pseudocódigo é uma referência à implementação que será realizada em uma linguagem de programação.
  - Por exemplo: um programa escrito na linguagem Pascal, gera um código em Pascal.
- Quando um algoritmo é gerado independente de uma linguagem de programação, então o algoritmo é escrito em pseudocódigo.
- Um algoritmo deve ser fácil de interpretar e codificar, ele deve ser o meio entre a linguagem falada e a linguagem de programação.

- Para escrever um algoritmo é preciso descrever uma sequência de instruções de maneira simples e objetiva.
- Algumas técnicas:
  - Usar somente um verbo por frase.
  - Imaginar que o desenvolvedor faz o algoritmo para pessoas que não entendem do assunto.
  - Usar frases curtas e simples.
  - Ser objetivo.
  - Procurar palavras que não tenham sentido duplo.

# Introdução

## Algoritmo - Fases para Construção

- Qualquer tarefa que siga um determinado padrão pode ser descrita por um algoritmo.
- Ao montar um algoritmo, primeiro é necessário dividir o problema em três fases:



- Exemplo: calcular a média final dos alunos que fizeram 2 provas e 1 trabalho.

$$\underline{MediaFinal} = \frac{P1 + P2 + T}{3}$$

$$MediaFinal = \frac{P1 + P2 + T}{3}$$

- Para montar o algoritmo, é necessário responder a 3 perguntas:
  - 1 Quais são os dados de entrada?
  - 2 Qual será o procedimento a ser utilizado?
  - 3 Quais serão os dados de saída?

# Introdução

## Algoritmo - Fases para Construção

$$MediaFinal = \frac{P1 + P2 + T}{3}$$

- Para montar o algoritmo, é necessário responder a 3 perguntas:

- 1 Quais são os dados de entrada? *P1, P2, T*
- 2 Qual será o procedimento a ser utilizado? *Calculo da média*
- 3 Quais serão os dados de saída? *média das notas*

- Algoritmo

- Receba a nota da prova1
  - Receba a nota da prova2
  - Receba a nota do trabalho
  - Some todas as notas e divida o resultado da soma por 3
  - Mostre o resultado da divisão
- Entrada* (grouping the first three steps)  
*processamento* (grouping the fourth step)  
*Saída* (grouping the fifth step)


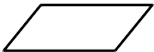
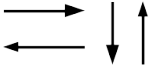
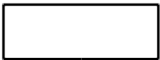
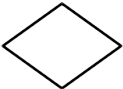


### Fluxograma

- 1 Uma representação gráfica de um procedimento, problema ou sistema, cujas etapas ou módulos são ilustrados de forma encadeada por meio de símbolos geométricos interconectados.
  - 2 Um diagrama para representação de um algoritmo.
- Um fluxograma trata da representação esquemática de um **processo ou uma sequência de passos**, muitas vezes feito através de gráficos que ilustram de forma descomplicada a transição de informações entre os elementos que o compõem.
  - Cada operação ou passo é representado por uma forma geométrica.
  - Os fluxogramas permitem uma visualização global do processo de resolução do problema.
    - Podem se tornar grandes para problemas mais complexos.

# Introdução

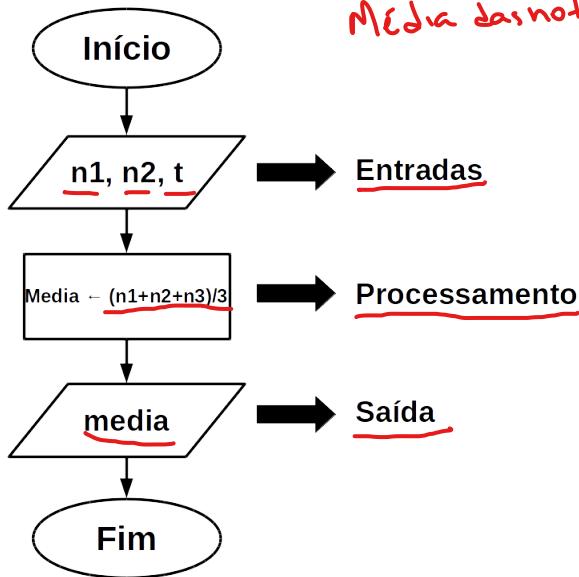
## Algoritmo - Fluxograma

Elemento	Significado e Utilização
	<b><u>Terminal</u></b> : demarca os pontos de <u>início</u> e <u>fim</u> de um algoritmo.
	<b><u>Entrada ou saída de dados</u></b> : mostra dados trocados entre o algoritmo e o ambiente externo
	<b><u>Fluxo</u></b> : indica o sentido (direção) dos passos do algoritmo.
	<b><u>Processo</u></b> : um passo ( <u>operação</u> ) do algoritmo.
	<b><u>Condição</u></b> : indica uma situação na qual o algoritmo deve seguir em uma ou outra direção, conforme resultado.

# Introdução

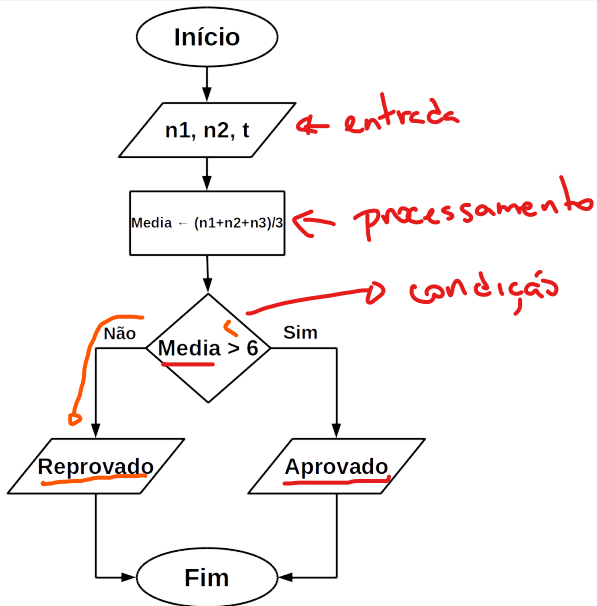
## Algoritmo - Fluxograma

*Média das notas!*



# Introdução

## Algoritmo - Fluxograma



### Teste de Mesa

Significa seguir as instruções do algoritmo de maneira precisa para verificar se o procedimento utilizado está correto. Em geral, o teste de mesa é feito após o desenvolvimento do algoritmo.

- Os **programas** de computadores são algoritmos escritos em uma linguagem de computador.
- As instruções do programa serão interpretadas ou executadas por uma determinada máquina.
- Um programa é por natureza, muito específico e rígido em relação aos algoritmos da vida real.

### Programa

Um algoritmo escrito em uma linguagem computacional.

- Os algoritmos serão desenvolvidos em uma pseudo-linguagem conhecida como Portugol ou **Português Estruturado**.
- **Portugol** é derivado da aglutinação de Português+Algol. Algol é uma linguagem de programação estruturada usada no final de década de 50.
- Para criar um programa que seja executável dentro de um computador, são necessárias três fases:
  - Entrada de dados: com a instrução leia.
  - Saída dos dados: com a instrução escreva.
  - Processamento dos dados: será uma consequência da manipulação das variáveis de ação.

- Uma entrada e uma saída poderão ocorrer dentro de um computador de diversas formas, por exemplo: teclado, modem, discos, leitores ópticos na entrada e vídeo, impressora, discos na saída.
- Independente do meio de entrada e saída, os programas escritos em português estruturado só utilizarão as instruções leia e escreva.

### Corpo do Programa

**algoritmo** "identificador" *nome*

**tipo** <<identificador>> : <<tipo>>

**const** <<identificador>> <- <<dado>>

**var** <<identificador>> : <<tipo>>

**inicio**

— {COMANDOS DE ENTRADA, PROCESSAMENTO E SAÍDA}

— <<comando 1>>

— <<comando N>>

**fimalgoritmo**

### Identação

É a inserção de espaços antes da frase para facilitar a localização do contexto. Os espaços servem como os espaços antes dos parágrafos facilitando a localização do leitor.



- **Comentários:** os comentários são usados em algoritmos para aumentar a sua compreensão. Em algoritmos um bloco de texto comentado fica entre chaves “{}”.

### Exemplo

```
leia(nota1, nota2, nota3) { // Leitura das notas do aluno }
```

- Para facilitar a leitura do algoritmo, no momento de seu desenvolvimento é preciso tomar o cuidado de identá-lo.
- Para comentar apenas uma linha em qualquer parte do algoritmo, pode-se utilizar //.

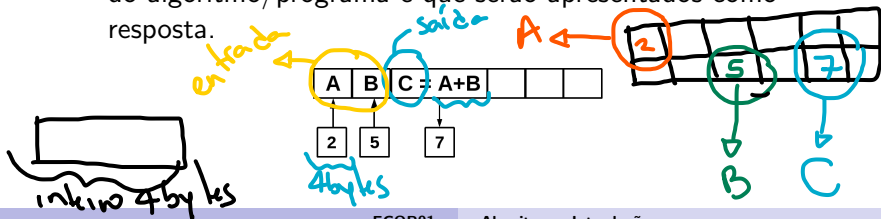
- **Variáveis**: são espaços (como caixas vazias) destinados a armazenar informações temporariamente.
  - Os valores são armazenados em um local disponível na memória do computador.

### Declaração da variável

**var** nome\_da\_variável : tipo\_da\_variável

*B: inteiro*

- **Variáveis de entrada**: armazenam informações fornecidas por um meio externo (usuários ou discos).
- **Variáveis de saída**: armazenam os resultados obtidos através do algoritmo/programa e que serão apresentados como resposta.



- Constantes: armazenam informações fixas, inalteráveis durante a execução do programa.
  - Os valores são armazenados em um local disponível na memória do computador.

### Diferença

A diferença entre **constantes** e **variáveis** é que as constantes não permitem alterar seu valor.

- Por exemplo:

PI <- 3.1416

Empresa <- 'ACME LTDA'

V <- Verdadeiro.

- **Identificadores:** são os nomes dados às variáveis, constantes, tipos de dados, registros, programa
  - servem para lembrar mais facilmente o que representa o conteúdo armazenado nas variáveis e constantes;
  - ou para que servem os programas que serão desenvolvidos.
- Regras para construção de identificadores:
  - 1 Não podem ser nomes de palavras reservadas (comandos da linguagem).
  - 2 Devem possuir como primeiro caracter uma letra ou \_, os demais caracteres podem ser letras, números e \_.
  - 3 Ter no máximo 127 caracteres.
  - 4 Não possuir espaços em branco.
  - 5 A escolha de letras pode ser maiúsculas ou minúsculas.
- Por exemplo: NOME, nome, TELEFONE, PI, idade\_filho.  
↳ CONSTANTE

- Todas as variáveis devem assumir um determinado tipo de informação.
- O **tipo de dado** indica qual tipo de informação pode-se guardar dentro das variáveis.
- Os tipos de dados podem ser:
  - **Primitivo**: pré-definido pela linguagem de programação.
  - **Sub-Faixa**: é uma parte de um tipo já existente.
  - **Escalar**: são os tipos de dados definidos pelo usuário.

Os tipos de dados primitivos são:

- **Inteiro**: toda e qualquer informação numérica que pertença ao conjunto dos números inteiros (negativa, nula ou positiva). Geralmente é usado para representar uma contagem (quantidade).
- **Real**: toda e qualquer informação numérica que pertença ao conjunto dos números reais (negativa, nula ou positiva). Geralmente é utilizado para representar uma medição.
- **Caracter**: toda e qualquer informação composta por um conjunto de caracteres alfanuméricos (0..9) e/ou caracteres especiais (#, \$, %, &, \*, ?).
- **Lógico**: toda e qualquer informação que pode apenas assumir duas situações: verdadeiro ou falso.

### Exemplos

**var**

idade : inteiro

salario : real

sexo : caracter

sexo → 

M	F
---	---

  
caracter

idade → 

--

  
inteiro

salario → 

--

  
real

idade, soma : inteiro

- **Sinal de atribuição:** uma variável nunca é eternamente igual a um valor, o seu conteúdo pode ser alterado a qualquer momento.

- para atribuir valores a variáveis, é preciso usar o sinal de atribuição  $\leftarrow$ .

$\Rightarrow$  atribuição =

### Exemplos

A  $\leftarrow$  2

B  $\leftarrow$  3

C  $\leftarrow$  A + B

A = 

2
---

B = 

3
---

C = 

2+3=5
-------

### Operadores aritméticos

- são o conjunto de símbolos que representa as operações básicas de matemática.

Símbolo	Operador
+	Adição
-	Subtração
*	Multiplicação
/	Divisão
MOD	Resto da divisão entre 2 números inteiros



# Operadores e Expressões

## Operadores Relacionais

### Operadores relacionais

- são utilizados para realizar comparações entre dois valores do mesmo tipo primitivo;
- os valores comparados são representados por constantes, variáveis ou expressões aritméticas.

*igualdade*  
 $A = B$   
 $A < B$

Símbolo	Operador
$>$	Maior que
$<$	Menor que
$\geq$	Maior ou igual
$\leq$	Menor ou igual
$=$	Igual
$\neq$	Diferente

*Sempre comparar  
tipos iguais  
inteiro = inteiro  
real = real  
(A < (-B))*

# Operadores e Expressões

## Operadores Lógicos

**Operadores lógicos:** atuam sobre expressões, retornando sempre valores lógicos como Verdadeiro ou Falso.

- **e** - retorna verdadeiro se ambas as partes forem verdadeiras.
- **ou** - basta que uma parte seja verdadeira para retornar verdadeiro.
- **nao** - inverte o estado, de verdadeiro passa para falso e vice-versa.

$(( )) \in (( ))$   
=  
 $(( )) \text{ ou } (( ))$   
=  
=

<b>A</b>	<b>B</b>	<b>A e B</b>	<b>A ou B</b>	<b>nao (A)</b>
V	V	V	V	F
V	F	F	V	F
F	V	F	V	V
F	F	F	F	V

a: lógico  $\downarrow$  F

$\neg(a)$   
=  
=

# Operadores e Expressões

## Expressões Lógicas

- **Expressão lógica**: são expressões compostas de operadores relacionais que sempre retornam um valor lógico.

### Exemplos

$2 + 4 > 4 \Rightarrow \text{Verdadeiro}$  ✓

$3 < 3 \Rightarrow \text{Falso}$  ✓

- De acordo com a necessidade, as expressões podem ser unidas pelos operadores lógicos.

### Exemplos

$2 > 1 \text{ ou } 3 > 5 \Rightarrow \text{Verdadeiro}$

$3 > 3 \text{ e } 3 > 5 \Rightarrow \text{Falso}$


F F

# Operadores e Expressões

## Prioridade de Expressões

- Na resolução das expressões aritméticas, as operações e funções matemáticas possuem prioridades na sua resolução.
- Exemplo:** sempre se resolve a multiplicação antes da soma.
- Em computação as prioridades são definidas de acordo com a linguagem escolhida. Para o Portugol, a seguinte sequência (da maior para a menor) é adotada:

*^ - exponencial (acento circumflexo)*



Sinal	Descrição	Tipo
nao	Negação	Operador lógico
<i>*, /, MOD</i>	Multiplicação, divisão e resto	Operador aritmético
<i>+ -</i>	Soma e subtração	Operador aritmético
<, >	Menor, maior	Operador relacional
<=, >=	Menor ou igual, maior ou igual	Operador relacional
=, <>	Igual, diferente	Operador relacional
e ou	e lógico, ou lógico	Operador lógico
<-	Atribuição	

# Operadores e Expressões

## Prioridade de Expressões

Do mesmo modo que na matemática, nas expressões computacionais pode-se usar os parênteses "()", para priorizar as expressões. É possível usar parênteses dentro de parênteses.

### Exemplos

$$(2 + 2) / 2 = 2$$

$$2 + 2 / 2 = 3$$

$c, b, a: \text{real}$

$a \leftarrow 2.0$

$b \leftarrow 3.0$

$$(2 + 2) / 3 =$$

$$\begin{aligned} c &\leftarrow (a + a) / b \\ c &\leftarrow 2 + 2 / 3 = 1.33 \\ c &\leftarrow 2.00 \end{aligned}$$

$$\begin{aligned} (a + a) / b &= \\ (2.0 + 2.0) / 3.0 &= \\ 4.0 / 3.0 &= 1.33 \end{aligned}$$

# Comando de I/O (*input/output*)

- leia → Comando de entrada que permite a leitura de um valor e salva em uma variável de entrada.
- escreva → Comando de saída que exibe a informação na tela do monitor.
- escreval → Comando de saída que exibe a informação na tela do monitor e insere uma quebra de linha (enter).

**Exemplo 01:** Escrever um algoritmo que lê 3 notas de um aluno e em seguida calcula e escreve a média obtida.

↓  
escreva

↓  
leia

# Comando de I/O (*input/output*)

## Atividade

Algoritmo que lê 3 notas de um aluno e em seguida calcula e escreve a média obtida

**algoritmo** “mediaAluno”

**var** nota1, nota2, nota3: **inteiro**

media: **real**

**inicio**

escreva(“Digite o valor de Nota1 = “)

leia(nota1) { Entrada }

escreva(“Digite o valor de Nota2 = “)

leia(nota2) { Entrada }

escreva(“Digite o valor de Nota3 = “)

leia(nota3) { Entrada }

media  $\leftarrow$  (nota1 + nota2 + nota3) / 3 { Processamento }

escreval(“Media = “, media) { Saida }

**fimalgoritmo**

# Comando de I/O (*input/output*)

## Atividade

Algoritmo para leitura do raio de uma circunferência e cálculo da área



# Comando de I/O (*input/output*)

## Atividade

**Exemplo:** algoritmo para leitura do raio de uma circunferência e cálculo da área

**algoritmo** areaCircunferencia

**const** PI <- 3.1416 *≡ não precisa definir*

**var** raio, area : **real**

**inicio**

**escreva**( "Digite o valor de Raio = ") { Saída }

**leia**(raio) { Entrada }

    area <- PI \* raio \* raio { Processamento }

**escreva**( "Area = ", area) { Saída }

**fimalgoritmo**