

# **APOSTILA DE LÓGICA DE PROGRAMAÇÃO**

## **CAP**

### **Criação de Algoritmos e Programas**

## SUMÁRIO

<b>INTRODUÇÃO.....</b>	<b>3</b>
<b>ALGORITMO.....</b>	<b>3</b>
<b>ALGORITMO NÃO COMPUTACIONAL.....</b>	<b>3</b>
<b>PROGRAMA.....</b>	<b>3</b>
<b>LINGUAGENS DE PROGRAMAÇÃO.....</b>	<b>3</b>
<b>TÉCNICAS ATUAIS DE PROGRAMAÇÃO.....</b>	<b>3</b>
<b>ALGORITMOS EM “PORTUGOL”.....</b>	<b>3</b>
<b>OPERADORES ARITMÉTICOS.....</b>	<b>3</b>
<b>OPERADORES RELACIONAIS.....</b>	<b>3</b>
<b>LINEARIZAÇÃO DE EXPRESSÕES.....</b>	<b>3</b>
<b>MODULARIZAÇÃO DE EXPRESSÕES.....</b>	<b>3</b>
<b>OPERADORES ESPECIAIS (MOD e DIV).....</b>	<b>3</b>
<b>FUNÇÕES.....</b>	<b>3</b>
<b>BIBLIOTECAS DE FUNÇÕES.....</b>	<b>3</b>
<b>FUNÇÕES PRÉ-DEFINIDAS.....</b>	<b>3</b>
<b>OPERADORES LÓGICOS.....</b>	<b>3</b>
<b>TABELA VERDADE.....</b>	<b>3</b>
<b>EXPRESSÕES LÓGICAS.....</b>	<b>3</b>
<b>VARIÁVEIS.....</b>	<b>3</b>
<b>VARIÁVEIS DE ENTRADA E SAÍDA.....</b>	<b>3</b>
<b>CONSTANTES.....</b>	<b>3</b>
<b>IDENTIFICADORES.....</b>	<b>3</b>
<b>TIPOS DE DADOS.....</b>	<b>3</b>
<b>TIPOS PRIMITIVOS DE DADOS.....</b>	<b>3</b>
<b>COMANDOS DE I/O (INPUT/OUTPUT).....</b>	<b>3</b>
<b>SINAL DE ATRIBUIÇÃO.....</b>	<b>3</b>
<b>SINAL DE IGUALDADE.....</b>	<b>3</b>
<b>CORPO GERAL DE UM PROGRAMA.....</b>	<b>3</b>
<b>ESTRUTURAS SEQUÊNCIAIS.....</b>	<b>3</b>
<b>; PONTO E VÍRGULA ;.....</b>	<b>3</b>
<b>PRIMEIRO ALGORITMO.....</b>	<b>3</b>
<b>SEGUNDO ALGORITMO.....</b>	<b>3</b>

{LINHAS DE COMENTÁRIO}.....	3
‘ASPAS SIMPLES’.....	3
ESTRUTURAS DE DECISÃO.....	3
ALGORITMO TRÊS.....	3
ALGORITMO QUATRO.....	3
NINHOS DE SE.....	3
ALGORITMO CINCO.....	3
ESTRUTURAS DE CONDIÇÃO.....	3
ALGORITMO SEIS.....	3
ESTRUTURA DE REPETIÇÃO DETERMINADA.....	3
ALGORITMO SETE.....	3
ALGORITMO OITO.....	3
ESTRUTURA DE REPETIÇÃO INDETERMINADA COM VALIDAÇÃO INICIAL.....	3
ALGORITMO NOVE.....	3
ESTRUTURA DE REPETIÇÃO INDETERMINADA COM VALIDAÇÃO FINAL.....	3
ALGORITMO DEZ.....	3
ALGORITMO ONZE.....	3
Programas Equivalentes.....	3
EXERCÍCIOS.....	3

## INTRODUÇÃO

O trabalho a que me propus é resultado de minha experiência em ministrar a disciplina CAP (criação de Algoritmos e Programas) desde 1996, motivado pela falta de texto relacionado às condições e necessidades do curso.

O objetivo principal da Lógica de Programação é demonstrar técnicas para resolução de problemas e conseqüentemente automatização de tarefas.

O aprendizado da Lógica é essencial para formação de um bom programador, servindo como base para o aprendizado de todas as linguagens de programação, estruturadas ou não.

De um modo geral esses conhecimentos serão de supra importância pois ajudarão no cotidiano, desenvolvendo um raciocínio rápido. Partindo do princípio que "a única coisa constante no mundo é a mudança", forneço abaixo meu endereço eletrônico para que você possa me ajudar, enviando críticas, elogios ou sugestões que servirão para o eterno aprimoramento desse trabalho.

[renato@professor.mailbr.com.br](mailto:renato@professor.mailbr.com.br)

[www.renatodacosta.cjb.net](http://www.renatodacosta.cjb.net)

## **ALGORITMO**

Um Algoritmo é uma seqüência de instruções ordenadas de forma lógica para a resolução de uma determinada tarefa ou problema.

### **ALGORITMO NÃO COMPUTACIONAL**

Abaixo é apresentado um Algoritmo não computacional cujo objetivo é usar um telefone público.

Início

1. Tirar o fone do gancho;
2. Ouvir o sinal de linha;
3. Introduzir o cartão;

4. Teclar o número desejado;

DESVIO      er o sinal de chamar

5.1 Conversar;

5.2 Desligar;

5.3 Retirar o cartão;

6. Senão

6.1 Repetir;

Fim.

## PROGRAMA

Um programa é um Algoritmo escrito em uma linguagem computacional.

## LINGUAGENS DE PROGRAMAÇÃO

São Softwares que permitem o desenvolvimento de programas. Possuem um poder de criação ilimitado, desde jogos, editores de texto, sistemas empresariais até sistemas operacionais.

Existem várias linguagens de programação, cada uma com suas características próprias.

Exemplos:

- Pascal

- Clipper
- C
- Visual Basic
- Delphi e etc.

## TÉCNICAS ATUAIS DE PROGRAMAÇÃO

- Programação Seqüencial
- Programação Estruturada
- Programação Orientada a Eventos e Objetos

## ALGORITMOS EM "PORTUGOL"

Durante nosso curso iremos aprender a desenvolver nossos Algoritmos em uma pseudo-linguagem conhecida como "Portugol" ou Português Estruturado.

"Portugol" é derivado da aglutinação de Português + Algol. Algol é o nome de uma linguagem de programação estruturada usada no final da década de 50.

## OPERADORES ARITMÉTICOS

- + è Adição
- è Subtração
- \* è Multiplicação
- / è Divisão

### OPERADORES RELACIONAIS

- > è Maior que
- < è Menor que
- >= è Maior ou Igual
- <= è Menor ou Igual
- = è Igual
- <> è Diferente

### LINEARIZAÇÃO DE EXPRESSÕES

Para a construção de Algoritmos todas as expressões aritméticas devem ser linearizadas, ou seja, colocadas em linhas.

É importante também ressaltar o uso dos operadores correspondentes da aritmética tradicional para a computacional.

$$(2/3+(5-3))+1=$$

Exemplo:

Tradicional

Computacional

### MODULARIZAÇÃO DE EXPRESSÕES

A modularização é a divisão da expressão em partes, proporcionando maior compreensão e definindo prioridades para resolução da mesma.

Como pode ser observado no exemplo anterior, em expressões computacionais usamos somente parênteses "( )" para modularização.

Na informática podemos ter parênteses dentro de parênteses.

Exemplos de prioridades:

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

$$2+2/2=3$$

### OPERADORES ESPECIAIS (MOD e DIV)

**MOD** è Retorna o resto da divisão entre 2 números inteiros.

**DIV** è Retorna o valor inteiro que resulta da divisão entre 2 números inteiros.

```
13 DIV 2 = 6
```

```
13 MOD 2 = 1
```

## **FUNÇÕES**

Uma função é um instrumento (Sub-algoritmo) que tem como objetivo retornar um valor ou uma informação.

A chamada de uma função é feita através da citação do seu nome seguido opcionalmente de seu argumento inicial entre parênteses.

As funções podem ser predefinidas pela linguagem ou criadas pelo programador de acordo com o seu interesse.

Exemplos:



**Processamento**

Valor Final Y

Y=3

**BIBLIOTECAS DE FUNÇÕES**

Armazenam um conjunto de funções que podem ser usadas pelos programas.

**FUNÇÕES PRÉ-DEFINIDAS**

ABS( )	VALOR ABSOLUTO
SQRT( )	RAIZ QUADRADA
SQR( )	ELEVA AO QUADRADO
TRUNC( )	VALOR TRUNCADO
ROUND( )	VALOR ARREDONDADO
LOG( )	LOGARITMO
SIN( )	SENO
COS( )	COSENO
TAN( )	TANGENTE

As funções acima são as mais comuns e importantes para nosso desenvolvimento lógico, entretanto, cada linguagem possui suas funções próprias. As funções podem ser aritméticas, temporais, de texto e etc.

## OPERADORES LÓGICOS

Atuam sobre expressões retornando sempre valores lógicos como Falso ou Verdadeiro.

<b>E</b>	RETORNA VERDADEIRO SE AMBAS AS PARTES FOREM VERDADEIRAS.
<b>OU</b>	BASTA QUE UMA PARTE SEJA VERDADEIRA PARA RETORNAR VERDADEIRO.
<b>NÃO</b>	INVERTE O ESTADO, DE VERDADEIRO PASSA PARA FALSO E VICE-VERSA.

## TABELA VERDADE

A	B	A E B	A OU B	NÃO (A)
V	V	V	V	F
V	F	F	V	F

F	V	F	V	V
F	F	F	F	V

## EXPRESSÕES LÓGICAS

As expressões compostas de relações sempre retornam um valor lógico.

Exemplos:

$2+5>4$  è Verdadeiro

$3<>3$  è Falso

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

Exemplos:

$2+ > 5$  E  $3<>3$  è Falso

V
F
V

$2+5>4$  OU  $3<>3$  è Verdadeiro

OU

V
F

NÃO(3<>3)è Verdadeiro

NÃO

## VARIÁVEIS

Variáveis são endereços de memória destinados a armazenar informações temporariamente.

\* Todo Algoritmo ou programa deve possuir variável!

## VARIÁVEIS DE ENTRADA E SAÍDA

Variáveis de Entrada armazenam informações fornecidas por um meio externo, normalmente usuários ou discos.

Variáveis de Saída armazenam dados processados como resultados.

Exemplo:

De acordo com a figura acima A e B são Variáveis de Entrada e C é uma Variável de Saída.

## **CONSTANTES**

Constantes são endereços de memória destinados a armazenar informações fixas, inalteráveis durante a execução do programa.

Exemplo:

PI = 3.1416

## **IDENTIFICADORES**

São os nomes dados a variáveis, constantes e programas.

Regras Para construção de Identificadores:

- Não podem ter nomes de palavras reservadas (comandos da linguagem);
- Devem possuir como 1º caractere uma letra ou Underscore ( \_ );

- Ter como demais caracteres letras, números ou Underscore;
- Ter no máximo 127 caracteres;
- Não possuir espaços em branco;
- A escolha de letras maiúsculas ou minúsculas é indiferente.

Exemplos:

NOME	TELEFONE	IDADE_FILHO
NOTA1	SALARIO	PI
<p>UMNOMEMUITOCOMPRIDOEDIFICILDELER</p> <p>UM_NOME_MUITO_COMPRIDO_E_FACIL_DE_LER</p>		

## TIPOS DE DADOS

Todas as Variáveis devem assumir um determinado tipo de informação.

O tipo de dado pode ser:

- Primitivo È Pré-definido pela linguagem;
- Sub-Faixa È É uma parte de um tipo já existente;
- Escalar È Definido pelo programador.

Exemplos:

A : INTEIRO

TIPO NOTA=[1..10] DE INTEIRO

TIPO SEMANA = (Segunda-feira, Terça-feira, Quarta-feira, Quinta-feira, Sexta-feira, Sábado, Domingo)

### TIPOS PRIMITIVOS DE DADOS

<b>INTEIRO</b>	ADMITE SOMENTE NÚMEROS INTEIROS. GERALMENTE É UTILIZADO PARA REPRESENTAR UMA CONTAGEM (QUANTIDADE).
<b>REAL</b>	ADMITE NÚMEROS REAIS (COM OU SEM CASAS DECIMAIS). GERALMENTE É UTILIZADO PARA REPRESENTAR UMA MEDIÇÃO.
<b>CARACTERE</b>	ADMITE CARACTERES ALFANUMÉRICOS. OS NÚMEROS QUANDO DECLARADOS COMO CARACTERES TORNAM SE REPRESENTATIVOS E PERDEM A ATRIBUIÇÃO DE VALOR.
<b>LÓGICO</b>	ADMITE SOMENTE VALORES LÓGICOS (VERDADEIRO/FALSO).

### COMANDOS DE I/O (INPUT/OUTPUT)

**LER** è Comando de entrada que permite a leitura de Variáveis de Entrada.

**ESCREVER** è Comando de saída que exhibe uma informação na tela do monitor.

**IMPRIMIR** è Comando de saída que envia uma informação para a impressora.

### SINAL DE ATRIBUIÇÃO

Uma Variável nunca é eternamente igual a um valor, seu conteúdo pode ser alterado a qualquer momento. Portanto para atribuir valores a variáveis devemos usar o sinal de ":=".

Exemplos:

A := 2;

B := 3;

C := A + B;

### SINAL DE IGUALDADE

As constantes são eternamente iguais a determinados valores, portanto usamos o sinal de "=".

Exemplos:

PI = 3.1416;

Empresa = 'Colégio de Informática L.T.D.A.'

V = Verdadeiro

## CORPO GERAL DE UM PROGRAMA

```
PROGRAMA <<identificador>>;
```

```
CONST
```

```
    <<identificador>> = <<dado>>
```

```
VAR
```

```
    <<identificador>> : <<tipo>>;
```

```
ÍNICIO
```

```
{
```

```
    COMANDOS DE ENTRADA, PROCESSAMENTO E SAÍDA
```

```
    <<comando1>>;
```

```
    <<comandoN>>
```

```
}
```

```
FIM.
```

## ESTRUTURAS SEQUÊNCIAIS

Como pode ser analisado no tópico anterior, todo programa possui uma estrutura seqüencial determinada por um ÍNICIO e FIM.

## ; PONTO E VÍRGULA ;

O sinal de ponto e vírgula ";" indica a existência de um próximo comando (passa para o próximo).

Na estrutura ÍNICIO e no comando que antecede a estrutura FIM não se usa ";".

## PRIMEIRO ALGORITMO

Segue um Algoritmo que lê o nome e as 4 notas bimestrais de um aluno. Em seguida o Algoritmo calcula e escreve a média obtida.

```
PROGRAMA MEDIA_FINAL;
```

```
VAR
```

```
    NOTA1, NOTA2, NOTA3, NOTA4, MEDIA: INTEIRO;
```

```
    NOME : CARACTERE [35]
```

```
INICIO
```

```
    LER (NOME);
```

```
    LER (NOTA1, NOTA2, NOTA3, NOTA4);
```

```
    MEDIA := (NOTA1 + NOTA2 + NOTA3 + NOTA4) / 4;
```

```
    ESCREVER (NOME, MEDIA)
```

FIM.

## SEGUNDO ALGORITMO

Segue um Algoritmo que lê o raio de uma circunferência e calcula sua área.

```
PROGRAMA AREA_CIRCUNFERENCIA;  
  
CONST PI = 3.1416;  
  
VAR RAIIO, AREA : REAL;  
  
INICIO  
  
    LER (RAIO); {PROCESSAMENTO}  
  
    AREA := PI * SQR(RAIO); {ENTRADA}  
  
    ESCREVER ('AREA =', AREA) {SAÍDA}  
  
FIM.
```

## {LINHAS DE COMENTÁRIO}

Podemos inserir em um Algoritmo comentários para aumentar a compreensão do mesmo, para isso basta que o texto fique entre Chaves "{ }".

Exemplo:

```
LER (RAIO); {ENTRADA}
```

### 'ASPAS SIMPLES'

Quando queremos exibir uma mensagem para a tela ou impressora ela deve estar contida entre aspas simples, caso contrário, o computador irá identificar a mensagem como Variável Indefinida.

Exemplo:

```
ESCREVER ('AREA OBTIDA =', AREA) {COMANDO DE SAÍDA}
```

```
AREA OBTIDA = X.XX {RESULTADO GERADO NA TELA}
```

### ESTRUTURAS DE DECISÃO

Executa uma seqüência de comandos de acordo com o resultado de um teste.

A estrutura de decisão pode ser Simples ou Composta, baseada em um resultado lógico.

Simple:

```
SE <<CONDIÇÃO>>
    ENTÃO <<COMANDO1>>
```

Composta 1:

```
SE <<CONDIÇÃO>>
    ENTÃO <<COMANDO1>>
    SENÃO <<COMANDO1>>
```

Composta 2:

```
SE <<CONDIÇÃO>>
    ENTÃO INICIO
        <<COMANDO1>>;
        <<COMANDON>>
    FIM;
    SENÃO INICIO
        <<COMANDO1>>; <<COMANDON>>
    FIM;
```

### ALGORITMO TRÊS

Segue um Algoritmo que lê 2 números e escreve o maior.

```
PROGRAMA ACHA_MAIOR;
```

```
VAR A, B : INTEIRO;
```

INICIO

LER (A, B);

SE A>B

ENTÃO ESCREVER (A)

SENÃO ESCREVER (B)

FIM.

### ALGORITMO QUATRO

Segue um Algoritmo que lê o nome e as 4 notas bimestrais de um aluno. Em seguida o Algoritmo calcula e escreve a média obtida pelo aluno escrevendo também se o aluno foi aprovado ou reprovado.

Média para aprovação = 6

PROGRAMA MEDIA\_FINAL;

VAR

NOTA1, NOTA2, NOTA3, NOTA4, MEDIA: REAL;

NOME : CARACTERE [35]

INICIO

LER (NOME);

LER (NOTA1, NOTA2, NOTA3, NOTA4);

MEDIA := (NOTA1 + NOTA2 + NOTA3 + NOTA4) / 4;

```
SE MEDIA >= 6
```

```
    ENTÃO ESCREVER ( 'APROVADO' )
```

```
    SENÃO ESCREVER ( 'REPROVADO' )
```

```
    ESCREVER ( NOME, MEDIA )
```

```
FIM.
```

### NINHOS DE SE

Usados para tomadas de decisões para mais de 2 opções.

Forma Geral:

```
SE <<CONDIÇÃO>>
```

```
    ENTÃO <<COMANDO1>>
```

```
    SENÃO SE <<CONDIÇÃO>>
```

```
        ENTÃO <<COMANDO1>>
```

```
        SENÃO <<COMANDO1>>
```

### ALGORITMO CINCO

Segue um Algoritmo que lê 3 números e escreve o maior.

```
PROGRAMA ACHA_MAIOR;
```

```
VAR A, B, C : INTEIRO;
```

```
INICIO
```

```
    LER (A, B, C);
```

```
    SE (A>B) E (A>C)
```

```
        ENTÃO ESCREVER (A)
```

```
    SENÃO SE (B>A) E (B>C)
```

```
        ENTÃO ESCREVER (B)
```

```
        SENÃO ESCREVER (C)
```

```
FIM.
```

## ESTRUTURAS DE CONDIÇÃO

A estrutura de condição eqüivale a um ninho de SE'S.

Forma Geral:

```
FACA CASO
```

```
    CASO <<CONDIÇÃO1>>
```

```
        <<COMANDO1>>;
```

```
    CASO <<CONDIÇÃO2>>
```

```
        <<COMANDO1>>;
```

```
    OUTROS CASOS
```

<<COMANDO1>>;

FIM DE CASO

## ALGORITMO SEIS

Segue um Algoritmo que lê 3 números e escreve o maior.

PROGRAMA ACHA\_MAIOR;

VAR A, B, C : INTEIRO;

INICIO

LER (A, B, C);

FACA CASO

CASO (A>B) E (A>C)

ESCREVER (A);

CASO (B>A) E (B>C)

ESCREVER (B);

OUTROS CASOS

ESCREVER (C);

FIM DE CASO

FIM.

## ESTRUTURA DE REPETIÇÃO DETERMINADA

Quando uma seqüência de comandos deve ser executada repetidas vezes, tem-se uma estrutura de repetição.

A estrutura de repetição, assim como a de decisão, envolve sempre a avaliação de uma condição.

Na repetição determinada o algoritmo apresenta previamente a quantidade de repetições.

Forma Geral 1:

```
PARA <<VARIABLE DE TIPO INTEIRO>>:=<<VALOR INICIAL>> ATE <<VALOR  
FINAL>> FAÇA
```

```
    <<COMANDO1>>;
```

Forma Geral 2:

```
PARA <<VARIABLE DE TIPO INTEIRO>>:=<<VALOR INICIAL>> ATE <<VALOR  
FINAL>> FAÇA
```

```
    INICIO
```

```
        <<COMANDO1>>;
```

```
        <<COMANDON>>
```

```
    FIM;
```

A repetição por padrão determina o passo do valor inicial até o valor final como sendo 1. Determinadas linguagens possuem passo -1

ou permitem que o programador defina o passo.

### ALGORITMO SETE

Segue um algoritmo que escreve 10 vezes a frase "VASCO DA GAMA"

```
PROGRAMA REPETICAO;
```

```
VAR I:INTEIRO
```

```
INICIO
```

```
    PARA I :=1 ATE 10 FACA
```

```
        ESCREVER ('VASCO DA GAMA')
```

```
FIM.
```

### ALGORITMO OITO

Segue um algoritmo que escreve os 100 primeiros números pares.

```
PROGRAMA PARES;
```

```
VAR I,PAR: INTEGER;
```

```
INICIO
```

```
    PAR:=0;
```

```
    PARA I:=1 ATE 100 FACA
```

INICIO

ESCREVER (PAR);

PAR := PAR+2

FIM

FIM.

## ESTRUTURA DE REPETIÇÃO INDETERMINADA COM VALIDAÇÃO INICIAL

É usada para repetir N vezes uma ou mais instruções. Tendo como vantagem o fato de não ser necessário o conhecimento prévio do número de repetições.

Forma Geral :

ENQUANTO <<CONDIÇÃO>> FAÇA

<<COMANDO1>>;

Forma Geral 2:

ENQUANTO <<CONDIÇÃO>> FAÇA

ÍNICIO

<<COMANDO1>>;

<<COMANDON>>

FIM;

ALGORITMO NOVE

Segue um algoritmo que calcule a soma dos salários dos funcionários de uma empresa. O programa termina quando o usuário digitar um salário menor que 0.

```
PROGRAMA SOMA_SALARIOS;
```

```
VAR SOMA, SALARIO : REAL;
```

```
INICIO
```

```
    SOMA:=0;
```

```
    SALARIO:=1;
```

```
    ENQUANTO SALARIO>=0
```

```
        INICIO
```

```
            LER (SALARIO);
```

```
            SOMA:=SOMA+SALARIO
```

```
        FIM;
```

```
    ESCREVER (SOMA)
```

```
FIM.
```

**TODAS AS VARIÁVEIS QUE ACUMULAM VALORES  
DEVEM  
RECEBER UM VALOR INICIAL.**

ESTRUTURA DE REPETIÇÃO INDETERMINADA COM VALIDAÇÃO FINAL

Assim como a estrutura ENQUANTO É usada para repetir N vezes uma ou mais instruções.

Sua validação é final fazendo com que a repetição seja executada pelo menos uma vez.

Forma Geral;

REPITA

<<COMANDO1>>;

<<COMANDON>>

ATE <<CONDIÇÃO>>

### ALGORITMO DEZ

Segue um algoritmo que calcule a soma dos salários dos funcionários de uma empresa. O programa termina quando o usuário digitar um salário menor que 0.

PROGRAMA SOMA\_SALARIOS;

VAR

SOMA, SALARIO : REAL;

INICIO

**SOMA := 0;**

REPITA

LER (SALARIO);

SOMA := SOMA + SALARIO

ATE SALARIO < 0;

ESCREVER (SOMA)

FIM.

### ALGORITMO ONZE

Segue um algoritmo que escreve os 100 primeiros números pares.

PROGRAMA PARES\_2;

VAR I, PAR, CONTADOR : INTEIRO;

INICIO

CONTADOR := 0;

PAR := 0;

REPITA

ESCREVER (PAR);

PAR := PAR + 2;

CONTADOR := CONTADOR + 1;

ATE CONTADOR=100

FIM.

## Programas Equivalentes

O algoritmo onze poderia ter sido criado com qualquer estrutura de repetição. Portanto podemos ter algoritmos que são escritos de maneiras diferentes, mas, funcionam realizando o mesmo objetivo.

### EXERCÍCIOS

- 1) O QUE É UM ALGORITMO?
- 2) O QUE É UM PROGRAMA?
- 3) CRIE UM ALGORITMO NÃO COMPUTACIONAL, QUE TROQUE UM PNEU DE CARRO.
- 4) O QUE É UMA LINGUAGEM DE PROGRAMAÇÃO?
- 5) LINEARIZE AS EXPRESSÕES ABAIXO:
- 6) Complete a tabela abaixo (A e B são variáveis lógicas; V= verdadeiro e F= falso)

A	B	A ou B	A e B	não A
V	V			
V	F			
F	V			
F	F			

- 7) CRIE ALGORITMOS PARA OS SEGUINTE PROBLEMAS:

A) Dados três valores X, Y, Z, verifiquem se eles podem ser os comprimentos dos lados de um triângulo e se forem escrever uma mensagem informando se é se é um triângulo equilátero, isósceles ou escaleno.

Observações:

O comprimento de um lado do triângulo é sempre menor do que a soma dos outros dois.

Equilátero  $\Rightarrow$  Todos lados iguais

Isósceles  $\Rightarrow$  Dois lados iguais

Escaleno  $\Rightarrow$  Todos os lados diferentes

B) Recebendo quatro médias bimestrais, calcule a média do ano (ponderada), sabendo que o 1º bimestre tem peso 1, o 2º bimestre tem peso 2, o 3º bimestre tem peso 3 e o 4º bimestre tem peso 4. Sabendo que para aprovação o aluno precisa ter uma média anual maior ou igual a 7, escreva uma mensagem indicando se o aluno foi aprovado ou reprovado.

Observação:

Média anual =  $(1^\circ \text{ bimestre} * 1 + 2^\circ \text{ bimestre} * 2 + 3^\circ \text{ bimestre} * 3 + 4^\circ \text{ bimestre} * 4) / (1+2+3+4)$