

# ALGORITMOS

## 1 Algoritmo

### 1.1 Definição

- É a descrição dos passos necessários para a resolução de um problema.
- É uma receita que qualquer um entenda.
- *A Definição Melhor* → Sequência finita de passos que se corretamente seguidos, nos levam a resultados previsíveis.

**Exemplo:** Receita em geral (para cozinha); Modo de usar de vários produtos; Instalação de Aparelhos em Geral, etc...

**Outro Exemplo:** O exemplo demonstrado abaixo, apresenta um Algoritmo para Lavar a Cabeça:

ALGORITMO para Lavar a Cabeça

- 1 – Início
- 2 – Molhe o cabelo
- 3 – Coloque Shampoo
- 4 – Faça Massagem
- 5 – Enxágüe
- 6 – Repita o Processo
- 7 – Fim

**Observações do Algoritmo apresentado acima:**

- 1) É a descrição de um procedimento rotineiro;
- 2) Tem um INÍCIO e um FIM claros;
- 3) A descrição é feita passo a passo, de maneira bem definida;
- 4) Há imperfeições:
  - 4.1) Não especifica a quantidade de shampoo;
  - 4.2) Não especifica quantas vezes o processo deve ser repetido;
  - 4.3) Não especifica qual o processo ou qual passo que deve ser repetido.

**Conclusão:** Com isso, verificamos que enquanto houverem imperfeições e dúvidas, o Algoritmo deve ser MELHORADO.

**Melhorando o Algoritmo para Lavar a Cabeça:**

ALGORITMO para Lavar a Cabeça

1 – Início

2 – Molhe o Cabelo

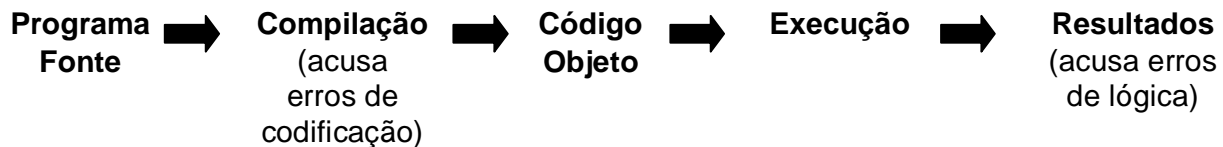
3 – Repita 2 (duas) vezes:

3.1 – Coloque a quantidade correspondente a uma tampa de shampoo

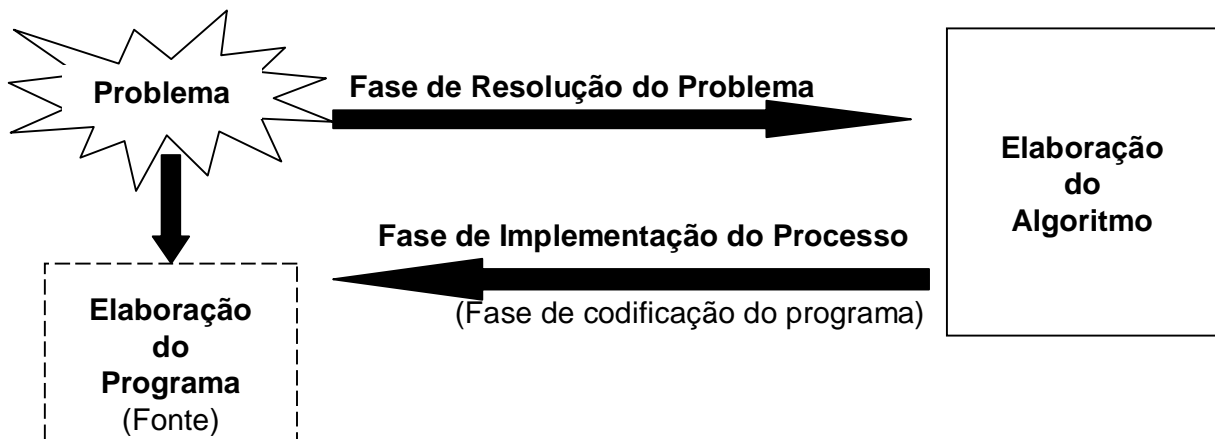
3.2 – Faça massagem durante 1 minuto

3.3 – Enxágüe

4 – Fim

**1.2 Características****→ Para que servem os Algoritmos em Computação?**

- Servem para a elaboração do programa fonte. Está antes do fluxograma acima.
- Serve para sairmos do problema e chegarmos ao programa.

**→ Qualidades de um bom Algoritmo:**

- 1) **Definição Perfeita** → Deve descrever exatamente quais são as instruções que devem ser executadas e em que seqüência. Deve ser tornado explícito o maior número possível de informações, pois a falta de alguma informação pode levar a uma interpretação errada do algoritmo;
- 2) **Ausência de Ambigüidade** → Não deve deixar dúvidas sobre o que deve ser feito. A ambigüidade acerca do que deve ser feito também pode levar a uma interpretação errada do algoritmo;

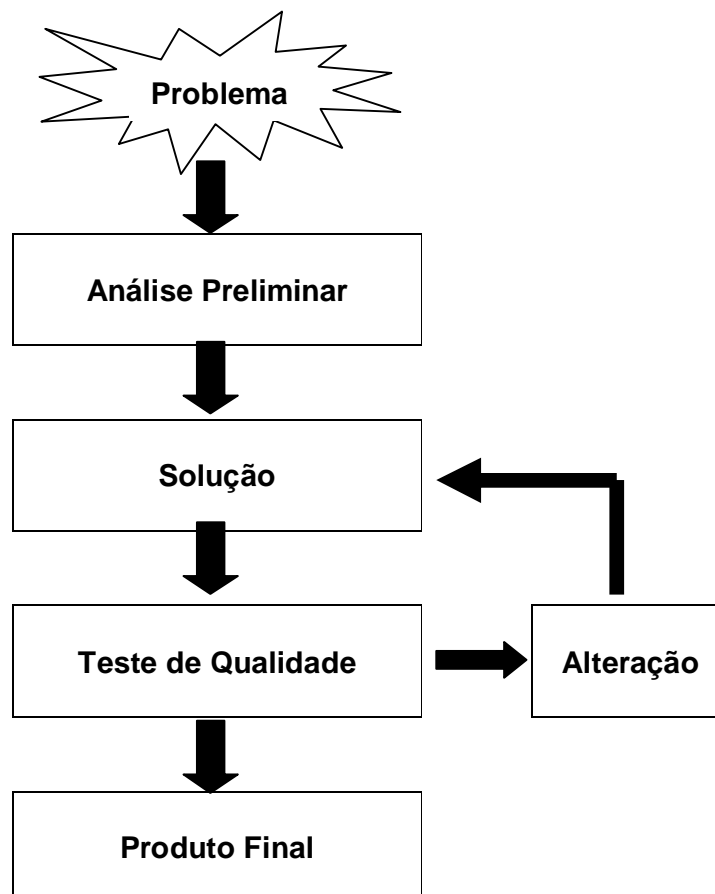
- 3) **Eficácia** → Conseguir resolver o problema em qualquer situação. Todas as situações de exceção que possam alterar o comportamento do algoritmo devem ser especificadas e tratadas;
- 4) **Eficiência** → Resolver o problema com o mínimo de recursos. Sempre se deve buscar aquele algoritmo que, dentre os diversos algoritmos que resolvam um mesmo problema, utilize a menor quantidade de recursos. No caso de algoritmos para processamento de dados, os recursos a serem considerados são espaços de memória (principal e auxiliar) e tempo de processamento (economia de C.P.U.), entre outros.

→ **Estratégias na Construção de Algoritmos:**

- Especifique o problema claramente e entenda-o completamente;
- Explícite todos os detalhes supérfluos;
- Entre no problema (envolva-se totalmente com o problema);
- Use todas as informações disponíveis;
- Decomponha o problema (Top-Down);
- Use o sentido inverso, se necessário (Bottom-Up).

→ **Como Construir Algoritmos?**

→ **Visão Esquemática da Construção de Algoritmos:**



- **Análise Preliminar** → Entenda o problema com a maior precisão possível, identifique os dados; identifique os resultados desejados.
- **Solução** → Desenvolva um algoritmo para resolver o problema.
- **Teste de Qualidade** → Execute o algoritmo desenvolvido com dados para os quais o resultado seja conhecido. O ideal é que o universo dos dados tenha todas as combinações possíveis.

Note que a qualidade de um algoritmo pode ser limitada por fatores como tempo para a sua confecção e recursos disponíveis.

- **Alteração** → Se o resultado do teste de qualidade não for satisfatório, altere o algoritmo e submeta-o a um novo teste de qualidade.
- **Produto Final** → O algoritmo concluído e testado, pronto para ser aplicado.

## 2 Portugal ou Pseudo-Código (Linguagem Estruturada)

### 2.1 Conceito

A Linguagem Estruturada é a forma que tem sido mais utilizada para a elaboração de algoritmos. É a forma que mais se assemelha com a forma em que os programas são escritos nas linguagens de programação.

A Linguagem Estruturada, ou algorítmica, é suficientemente geral para que a passagem do algoritmo ao programa deva ser quase que uma operação direta, não importando a linguagem de programação a ser utilizada.

### 2.2 Operadores

**2.2.1 Operadores Aritméticos:** +; -; \* (*multiplicação*); / (*divisão real*); **DIV** (*divisão de inteiros*); **MOD** (*resto da divisão inteira*);  $\uparrow$  (*potenciação*).

Exemplos:  $1.0/0.5=2.0$ ;  $7/2=3.5$ ;  $7DIV2=3$ ;  $7MOD2=1$ ;  $2^2=2\uparrow 2$ .

**2.2.2 Operadores Relacionais:** < (*menor*); ≤ (*menor ou igual*); = (*igual*); ≠ (*diferente*); ≥ (*maior ou igual*); > (*maior*).

**2.2.3 Operadores Lógicos:** E; OU; NÃO

→ Operador Lógico E → só resultará em VERDADEIRO, se todas as condições forem verdadeiras.

→ Operador Lógico OU → só resultará em FALSO, se todas as condições forem falsas.

→ Operador Lógico NÃO → nega uma condição. Então, se vier antes de uma condição verdadeira, o resultado será FALSO. E, se vier antes de uma condição falsa, o resultado será VERDADEIRO.

Veja o quadro a seguir:

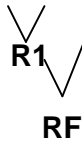
X	Y	X E Y	X OU Y	NÃO X	NÃO Y
V	V	V	V	F	F
F	V	F	V	V	F
V	F	F	V	F	V
F	F	F	F	V	V

## 2.2.4 Hierarquia de Operadores:

1. Parênteses (com vários níveis)
2. Funções
3. Potenciação ( $\uparrow$ )
4.  $*$ ;  $/$ ; **DIV**; **MOD**
5.  $+$ ;  $-$
6. Operadores Relacionais

### Exemplos:

1)  $X + Y - Z$

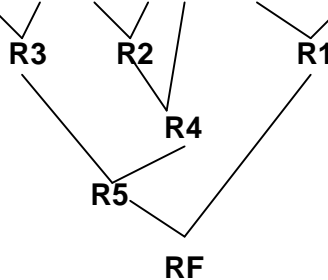


**Observação:** Com prioridades iguais, como no exemplo ao lado os sinais  $+$  e  $-$ , resolve-se primeiro a operação mais à esquerda.

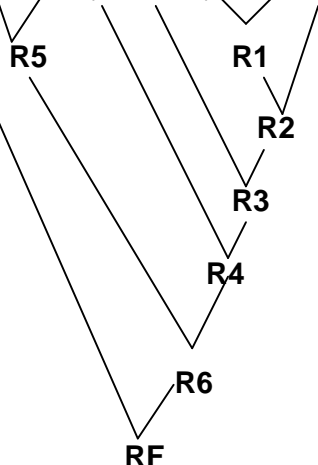
2)  $X * (Y - Z)$



3)  $A = B * C - E \uparrow F / G + (A \text{ DIV } B)$   $\Rightarrow A \leftarrow RF$



4)  $X = (A - B / C * (E + A * (C \uparrow B - E)))$   $\Rightarrow X \leftarrow RF$



### 3 Regras para Construção dos Algoritmos

#### 3.1 Declaração de Constantes

- Constantes são dados que durante a execução do programa, permanecem com os seus valores inalterados;
- É importante tratá-las pelo nome.

#### 3.2 Declaração de Variáveis

- Variáveis são dados cujos valores variam durante a execução do programa;
- São entidades que representam dados do programa;
- Possuem Nome e Valor;
- Representa uma posição de memória do computador. Exs.: A=3; X=0; Y=10; K=0.01 {  
*Num dado momento, pois depois todos os valores podem estar alterados.*
- Tipos de Variáveis:
  - Numéricas:
    - ➔ Inteiras. Ex.: MÍNIMO = 0
    - ➔ Reais. Ex.: PI = 3.1416...
  - Literais. Exs.: CIDADE = "Volta Redonda"; ESTADO = "R.J.";
  - Caracter. Exs.: RESP = "S"; SEXO = "M"; LETRA = "A".
  - Lógicas. Exs.: ACHEI = Verdadeiro; CHAVE = Falso.

#### ➔ Formação do Identificador (NOME das Variáveis e Constantes)

- Começa sempre com uma letra que pode ser seguida de mais letras e/ou números;
- Deve ter no máximo 8 caracteres;
- Não são permitidos caracteres especiais (só letras e números);
- O nome deve refletir o significado da variável ou constante. Exs.: X=16; Y=02; Z=1998 (*não são bons nomes*), mas DIA=10; MÊS=02; ANO=2012 (*são bons nomes*).

### 3.3 Comandos Básicos

#### 3.3.1 Comando de Atribuição (←)

A instrução de ATRIBUIÇÃO permite que o conteúdo de uma variável seja alterado.

**Sintaxe:**

**Nome ← valor**

**Onde:** Um valor ou o resultado de uma expressão será armazenado sob um nome simbólico que está a esquerda do sinal de atribuição “  $\leftarrow$  ”.

Tal expressão poderá ser um simples valor atribuído à uma constante ou variável, ou ainda poderá ser uma expressão aritmética, envolvendo outras variáveis previamente definidas. Contudo, o tipo do valor do resultado obtido através do cálculo da expressão, deve ser do mesmo tipo da variável que irá receber este valor.

**Exemplos:**  $X \leftarrow 0$ ;  $Y \leftarrow X$ ;  $A \leftarrow X+Y$  (expressão aritmética);  $DIA \leftarrow 30$ .

### 3.3.2 Comando de Leitura (leia)

Transporta informações de um periférico de entrada para a memória principal do computador.

As informações são lidas de um dispositivo de entrada, geralmente do teclado.

**Sintaxe:**

**leia (*lista-de-variáveis*)**

**Onde:** A execução da instrução de leitura pressupõe que os dados serão fornecidos do meio externo (dispositivo de entrada - teclado) e serão armazenados na memória sob os nomes simbólicos explicitados na *lista-de-variáveis*, na ordem dada.

**Exemplos:**

leia (NOME, TELEFONE)

leia (DIA, MES, ANO)

leia (NUMERO)

leia (PRODUTO, PRECO)

### 3.3.3 Comando de Impressão (escreva)

Transporta informações da memória principal do computador para um periférico de saída.

As informações são exibidas em um dispositivo de saída, geralmente em impressora ou vídeo.

**Sintaxe:**

**escreva (*lista-de-variáveis/constantes ou “texto”*)**

**Onde:** A execução da instrução de impressão pressupõe que os dados estejam armazenados na memória e serão colocados disponíveis no meio externo (dispositivo de saída – impressora ou vídeo) através dos nomes simbólicos atribuídos às variáveis ou às constantes.

A opção “*texto*” prevista no formato da instrução permite também que sejam explicitados textos para a exibição de mensagens.

**Exemplos:**

escreva ("UniFoa – Centro Universitário de Volta Redonda") → imprime  
texto

escreva (DIA, MES, ANO) → imprime os valores (conteúdo) das variáveis  
DIA, MES e ANO

escreva ("Dia = ", DIA, "Mês = ", MES, "Ano = ", ANO) → imprime textos e os  
valores (conteúdos) das variáveis.

escreva ("O resultado da expressão  $5 * 3 - 2 + 4 / 2$  é = ",  $5*3-2+4/2$ )  
↳ imprime texto e o resultado da expressão

**3.4 Detalhamento das Regras**

- 1) ALGORITMO é sempre a primeira palavra, seguida do título (minúsculo);
- 2) Os passos são numerados;
- 3) Início e Fim claros;
- 4) Os passos (comentários) são escritos entre colchetes [ ];
- 5) Estruturas serão grifadas;
u>
- 6) Usar "identação", que é um deslocamento para direita das instruções subordinadas.

Exemplo de Algoritmo usando a *Estrutura de Controle Condicional Composto* (*Se-Então-Senão*), com aplicação da **Metodologia de Desenvolvimento de Algoritmos**:

ALGORITMO exemplo do Se-Então-Senão

```

1 [Início]
2 [Declaração de Variáveis]
   IDADE: inteiro
3 [Leitura da Idade]
   escreva ("entre com a Idade: ")
   leia(IDADE)
4 [Verificação da idade]
   se idade ≥ 18
       então escreva ("Maior de Idade !")
       senão escreva ("Menor de Idade !")
       fim-se
5 [Impressão da idade lida]
   escreva ("A idade lida foi: ",IDADE)
6 [Fim]
```

**4 Estruturas Básicas****4.1 Seqüência Simples**



As instruções do algoritmo são executadas uma após a outra, sem que haja desvios na seqüência das instruções. Cada instrução é executada uma e somente uma vez.

**Exemplo1:** Faça um algoritmo para ler um valor numérico qualquer e imprimir o valor lido.

ALGORITMO exemplo da Estrutura Seqüencial

- 1 [Início]
- 2 [Declaração de Variáveis]  
    NUMERO: real
- 3 [Leitura do Valor Numérico]  
    escreva ("Entre com um valor numérico: ")  
    leia (NUMERO)
- 4 [Impressão do número lido]  
    escreva ("O número lido foi: ", NUMERO)
- 5 [Fim]

**Exemplo2:** Refaça o algoritmo do *Exemplo1*, fazendo com que o número lido seja impresso ao quadrado.

ALGORITMO número ao quadrado

- 1 [Início]
- 2 [Declaração de Constantes]  
    ELEV2  $\leftarrow$  2
- 3 [Declaração de Variáveis]  
    NUMERO, NUMQUAD: real
- 4 [Leitura do Valor Numérico]  
    escreva ("Entre com um valor numérico: ")  
    leia (NUMERO)
- 5 [Cálculo do quadrado de um número]  
    NUMQUAD  $\leftarrow$  NUMERO  $\uparrow$  ELEV2
- 6 [Impressão do número lido]  
    escreva ("O número lido foi: ", NUMERO)
- 7 [Impressão do Resultado]  
    escreva ("O quadrado do número lido é : ", NUMQUAD)
- 8 [Fim]

**IMPORTANTE:** TUDO O QUE É LIDO DEVE SER IMPRESSO.

**Exemplo3:** Faça um algoritmo para calcular  $X = \frac{A + B}{A - B}$ .

ALGORITMO cálculo da expressão

- 1 [Início]
- 2 [Declaração das Variáveis]  
    A, B, X: real
- 3 [Leitura dos valores de A e B]  
    escreva ("Entre com um valor para A: ")  
    leia (A)

```

        escreva ("Entre com um valor para B: ")
        leia (B)
4 [Cálculo da Expressão]
     $X \leftarrow (A+B) / (A-B)$ 
5 [Impressão dos valores A e B lidos]
    escreva ("O valor de A lido foi: ", A, " e o valor de B lido foi: ", B)
6 [Impressão do resultado do cálculo da expressão]
    escreva ("Para os valores de A e B lidos, o resultado da expressão
         $X=(A+B)/(A-B)$ , será: ", X)
7 [Fim]

```

## 4.2 Condicional Simples

Nesta estrutura, a seleção de uma ação é feita a partir da especificação de uma alternativa e é dirigida por uma condição. Quando o resultado da condição é "VERDADEIRO", executa-se os comandos do **então**. Caso contrário (resultado da condição = "FALSO"), vai para o final da estrutura. Desta forma, o fluxo de execução é condicionado por uma tomada de decisão (teste de uma condição) e alguns passos (comandos) podem não ser processados, devido a um desvio (salto).

### Sintaxe:

```

    se condição
        então
            comando(s)
    fim-se

```

**Exemplo:** Faça um algoritmo para ler dois valores e dizer se são iguais ou não. Se não forem iguais, dizer qual deles é maior que o outro.

ALGORITMO exemplo da estrutura Condicional Simples

```

1 [Início]
2 [Declaração de Variáveis]
    VAL1, VAL2: real
3 [Leitura dos valores]
    escreva ("Entre com o primeiro valor: ")
    leia (VAL1)
    escreva ("Entre com o segundo valor: ")
    leia (VAL2)
4 [Processamento]
    se VAL1 = VAL2
        então
            escreva ("Os valores lidos são iguais !")
    fim-se
    se VAL1 > VAL2
        então
            escreva ("O primeiro valor lido é maior que o segundo !")
    fim-se

```

```

    se VAL1 < VAL2
        então
            escreva ("O segundo valor lido é maior que o primeiro ! ")
    fim-se
5 [Impressão dos valores lidos]
    escreva ("O primeiro valor lido foi: ", VAL1, " e o segundo valor lido foi: ",
        VAL2)
6 [Fim]

```

**Rastreo:**

**VAL1**  
-2

**VAL2**  
-2

**Mensagem:**

Entre com o primeiro valor: 2  
Entre com o segundo valor: 2  
Os valores lidos são iguais !  
O primeiro valor lido foi: 2 e o segundo valor lido foi: 2

1

2

Entre com o primeiro valor: 1  
Entre com o segundo valor: 2  
O segundo valor lido é maior que o primeiro !  
O primeiro valor lido foi: 1 e o segundo valor lido foi: 2

3

1

Entre com o primeiro valor: 3  
Entre com o segundo valor: 1  
O primeiro valor lido é maior que o segundo !  
O primeiro valor lido foi: 3 e o segundo valor lido foi: 1

### 4.3 Condicional Composto

Nesta estrutura, a seleção de uma ação é feita a partir da especificação de duas alternativas e é dirigida por uma condição. Se a condição for verdadeira, um curso de ação é tomado; caso contrário, um outro curso de ação é que será tomado. Desta forma, o fluxo de execução é condicionado por uma tomada de uma decisão (teste de uma condição) e alguns passos (comandos) podem não ser processados, devido a um desvio (salto).

**Sintaxe:**

```

    se condição
        então
            comando(s)_1
        senão
            comando(s)_2
    fim-se

```

**Onde:** A entrada da estrutura de controle está no ponto onde a condição deve ser testada. Após a avaliação da condição, uma das alternativas (*VERDADEIRA* ou *FALSA*) será executada. Se for verdadeira, o(s) comando(s)\_1 do **então** será(ão) executado(s); caso contrário, o(s) comando(s)\_2 do **senão** será(ão) executado(s). Vale ressaltar o seguinte: que apenas um dos comandos será executado, pois uma condição só pode ter um dos dois possíveis valores: Verdadeiro ou Falso.

**Exemplo1:** Refaça o algoritmo que lê dois valores, diz se são iguais ou não e qual deles é maior que o outro, se não forem iguais (exemplo da estrutura Condicional Simples), usando a estrutura **Condicional Composto**:

ALGORITMO exemplo da estrutura Condicional Composto

```

1 [Início]
2 [Declaração de Variáveis]
    VAL1, VAL2: real
3 [Leitura dos valores]
    escreva ("Entre com o primeiro valor: ")
    leia (VAL1)
    escreva ("Entre com o segundo valor: ")
    leia (VAL2)
4 [Processamento]
    se VAL1 = VAL2
        então
            escreva ("Os valores lidos são iguais !")
        senão
            se VAL1 > VAL2
                então
                    escreva ("O primeiro valor lido é maior que o
                        segundo ! ")
                senão
                    escreva ("O segundo valor lido é maior que o
                        primeiro ! ")
            fim-se
        fim-se
5 [Impressão dos valores lidos]
    escreva ("O primeiro valor lido foi: ", VAL1, " e o segundo valor lido foi: ",
        VAL2)
6 [Fim]
    
```

**Exemplo2:** Refaça o algoritmo que calcula  $X = (A + B) / (A - B)$ , eliminando o problema criado quando o denominador for igual a 0 (zero).

ALGORITMO cálculo da expressão

```

1 [Início]
2 [Declaração das Variáveis]
    A, B, X: real
3 [Leitura dos valores de A e B]
    escreva ("Entre com um valor para A: ")
    
```

```

leia (A)
escreva ("Entre com um valor para B: ")
leia (B)
4 [Cálculo da Expressão]
  se A = B
    então
      escreva ("Denominador Nulo !")
    senão
       $X \leftarrow (A+B) / (A-B)$ 
      escreva ("Para os valores de A e B lidos, o resultado da
        expressão  $X=(A+B)/(A-B)$ , será: ", X)
  fim-se
5 [Impressão dos valores A e B lidos]
  escreva ("O valor de A lido foi: ", A, " e o valor de B lido foi: ", B)
6 [Fim]

```

**Rastreio:**

A  
~~2~~

B  
~~2~~

4

2

**Mensagem:**

Entre com um valor para A: 2  
 Entre com um valor para B: 2  
 Denominador Nulo !  
 O valor de A lido foi: 2 e o valor de B lido foi: 2

Entre com um valor para A: 4  
 Entre com um valor para B: 2  
 Para os valores de A e B lidos, o resultado da  
 expressão  $X=(A+B)/(A-B)$ , será: 3  
 O valor de A lido foi: 4 e o valor de B lido foi: 2

### 4.3.1 Aninhamento de Se's

Em algumas aplicações, uma das alternativas de uma estrutura Se-então-senão pode envolver outras decisões. Quando isso ocorre, dizemos que houve aninhamento de Se's. O algoritmo abaixo imprime o maior de três valores lidos, sem levar em consideração a leitura/entrada de valores iguais:

ALGORITMO valor máximo entre três números lidos

```

1 [Início]
2 [Declaração de Variáveis]
  VAL1, VAL2, VAL3, MAX: real
3 [Leitura dos números]
  escreva ("Entre com o primeiro valor: ")
  leia (VAL1)
  escreva ("Entre com o segundo valor: ")
  leia (VAL2)
  escreva ("Entre com o terceiro valor: ")
  leia (VAL3)
4 [Verificação do maior valor lido]

```

```

    se VAL1 > VAL2
        então
            se VAL1 > VAL3
                então
                    MAX ← VAL1
                senão
                    MAX ← VAL3
            fim-se
        senão
            se VAL2 > VAL3
                então
                    MAX ← VAL2
                senão
                    MAX ← VAL3
            fim-se
        fim-se
5 [Impressão do maior valor lido]
  escreva ("Dos valores ", VAL1, " , ", VAL2, " , ", VAL3, " , ", o maior valor
    lido foi: ", MAX)
6 [Fim]

```

#### 4.3.2 Utilização de Condições Compostas

Para alguns problemas, as relações simples são inadequadas para descrever as condições requeridas. O resultado poderia usualmente ser obtido com embutimento, entretanto isto pode tornar os algoritmos desnecessariamente complicados e difíceis de entender.

Um método alternativo é utilizar condições compostas. Estas condições são obtidas das relações simples, utilizando os operadores lógicos “e”, “ou” e “não”. Veja o exemplo abaixo:

##### a) Sem o uso de condição composta

```

    se SALARIO < 400
        então
            se NUMFALTA = 0
                então
                    SALARIO ← SALARIO + GRATIFIC
            fim-se
        fim-se

```

##### b) Com o uso de condição composta

```

    se SALARIO < 400 e NUMFALTA = 0
        então
            SALARIO ← SALARIO + GRATIFIC
        fim-se

```

→ Refazendo o algoritmo que imprime o maior de três valores lidos (exemplo de aninhamento de Se's), usando a condição composta, o algoritmo ficaria da seguinte forma:

ALGORITMO valor máximo entre três números lidos

1 [Início]

2 [Declaração de Variáveis]

VAL1, VAL2, VAL3, MAX: real

3 [Leitura dos números]

escreva ("Entre com o primeiro valor: ")

leia (VAL1)

escreva ("Entre com o segundo valor: ")

leia (VAL2)

escreva ("Entre com o terceiro valor: ")

leia (VAL3)

4 [Verificação do maior valor lido]

se VAL1 > VAL2 e VAL1 > VAL3

então

MAX ← VAL1

senão

se VAL2 > VAL3

então

MAX ← VAL2

senão

MAX ← VAL3

fim-se

fim-se

5 [Impressão do maior valor lido]

escreva ("Dos valores ", VAL1, " , ", VAL2, " , ", VAL3, " , ", o maior valor lido foi: ", MAX)

6 [Fim]

#### 4.4 Alternativa de Múltipla Escolha

Certas aplicações envolvem um grande número de testes lógicos, e para tratá-los é preciso aninhamento de SE's. Existe uma estrutura que aplicada a certos casos de aninhamento produz o mesmo resultado e torna o algoritmo mais inteligível e com menos vulnerabilidade a ambigüidades. Esta estrutura é chamada de Escolha Condicional.

##### Sintaxe:

##### escolha

caso Condição1  
Comando(s)\_1

caso Condição2  
Comando(s)\_2

**caso** Condição  
Comando(s)\_n  
[ **caso contrário** / **senão**  
Comando(s)\_s ]  
**fim-escolha**

**Onde:** Nesta estrutura testa-se, começando pela primeira, uma a uma as condições até que se encontre uma verdadeira. Quando isso ocorre, é executado o(s) comando(s) associado(s) à condição verdadeira. Após encontrar uma condição verdadeira, o fluxo de execução é passado diretamente para o comando abaixo do **fim-escolha**, pois somente uma alternativa é executada.

Caso nenhuma das condições seja verdadeira, pode-se incluir a cláusula “**caso contrário**” ou “**senão**”, para que quando isso ocorra, seja(m) executado(s) o(s) comando(s) associado(s) ao “**caso contrário**” ou “**senão**”.

**Exemplo1:**

## ALGORITMO Exemplo1 de Caso

1 [Início]

## 2 [Declaração de Variáveis]

 $X, Y, K, Z, N, A, B : \text{real}$ 

### 3 [Processamento]

$$X \leftarrow 5$$
$$Y \leftarrow 3$$
$$Z \leftarrow 2$$
$$K \leftarrow (X+Y) / Z$$
escolha

### caso K=1

$$N \leftarrow X * Y \uparrow Z$$
$$A \leftarrow N \uparrow K$$

caso K=2

$$X \leftarrow 0$$
$$Y \leftarrow 5$$
$$N \leftarrow X + Y$$

caso  $K=3$

$$A \leftarrow X - Z$$
$$B \leftarrow X + Z$$

caso  $K=4$

$$N \leftarrow X - Y$$
$$A \leftarrow Y + K$$
$$B \leftarrow K^* Z$$

fim-escolha

4 [Fim]



**Exemplo2:** Faça um algoritmo Menu de Opções, que para cada uma das opções abaixo lidas, imprima as seguintes mensagens:

Opções	Mensagens
1	Executa a rotina de Inclusão de Professores
2	Executa a rotina de Alteração de Professores
3	Executa a rotina de Exclusão de Professores
4	Executa a rotina de Consulta de Professores

ALGORITMO Exemplo2 de Caso

1 [Início]

2 [Declaração de Constantes]

OP1 ← 1

OP2 ← 2

OP3 ← 3

OP4 ← 4

2 [Declaração de Variáveis]

OPCAO : inteiro

3 [Leitura, Processamento e Impressão]

escreva ("Entre com uma opção de 1 a 4: ")

leia (OPCAO)

escolha

caso OPCAO = OP1

escreva ("Executa a rotina de Inclusão de Professores")

caso OPCAO = OP2

escreva ("Executa a rotina de Alteração de Professores")

caso OPCAO = OP3

escreva ("Executa a rotina de Exclusão de Professores")

caso OPCAO = OP4

escreva ("Executa a rotina de Consulta de Professores")

caso contrário (senão)

escreva ("Opção Inválida ! As opções válidas são de 1 a 4.")

fim-escolha

escreva ("A opção lida foi: ", OPCAO)

4 [Fim]

## 4.5 Condicional Repetitivo

A estrutura de repetição, ou comumente LAÇO, é a estrutura que possibilita a execução automática de um conjunto de instruções repetidas vezes.

### 4.5.1 Laços Condicionais

Nesta estrutura a execução automática de um conjunto de instruções repetidas vezes, é controlada por uma condição, ou seja, a execução automática do conjunto de instruções vai depender de um teste lógico.

**Sintaxe:**

**enquanto** <condição> **faça**  
 comando(s)  
**fim-enquanto**

**Onde:** O(s) comando(s) delimitado pelas palavras “enquanto” e “fim-enquanto” será(ão) executado(s) repetidas vezes, enquanto a condição testada for verdadeira. Caso a instrução seja falsa, o fluxo de execução passará para a instrução seguinte que deverá estar abaixo do “fim-enquanto”.

**Exemplo1:**

```

ALGORITMO exemplo1 da estrutura enquanto
1 [Início]
2 [Declaração de Variáveis]
  A,B,C: inteiro
3 [Inicialização de Variáveis]
  A ← 10
  B ← A/10
  C ← B*5
4 [Processamento]
  enquanto C>B faça
    B ← B+1
    A ← A -1
  fim-enquanto
5 [Impressão]
  escreva (A, ", ", "B," e ",C)
6 [Fim]
  
```

**Rastreio:**

<u>A</u>	<u>B</u>	<u>C</u>	<u>Impressão:</u>
10	1	5	6,5 e 5
9	2		
8	3		
7	4		
6	5		

**Exemplo2:** Faça um algoritmo para multiplicar uma quantidade qualquer de valores numéricos lidos.

**FLAG** → colocação de um valor conhecido, estranho ao conjunto de valores lidos, para que alguma providência seja tomada quanto ao processamento.

```

ALGORITMO exemplo2 da estrutura enquanto
1 [Início]
2 [Declaração de Constante]
  FLAG ← 0
  
```

```

3 [Declaração de Variáveis]
    MULT, NUM: real
    CONT : inteiro
4 [ Inicialização de Variáveis]
    MULT ← 1
    CONT ← 0
5 [Leitura e multiplicação dos números]
    escreva ("Entre com um número (0 - sai): ")
    leia (NUM)
    enquanto NUM ≠ FLAG faça
        MULT ← MULT*NUM
        CONT ← CONT + 1
        escreva ("O número lido foi: ", NUM)
        escreva ("Entre com um número (0 - sai): ")
        leia (NUM)
    fim-enquanto
6 [Impressão]
    se CONT = 0
        então
            escreva ("Para o valor ", FLAG, " lido, não permite continuar o
                processamento.")

        senão
            escreva ("O produto dos valores lidos é: ", MULT)
    fim-se
7 [Fim]

```

**Rastreio:**

	<u>FLAG</u>	<u>MULT</u>	<u>NUM</u>	<u>CONT</u>
1)	0	1	0	0
2)	0	<del>1</del>	<del>2</del>	<del>0</del>
		2	3	1
		<del>6</del>		2
		0		

- Mensagens:**
- 1) Entre com um número (0 - saí): 0  
Para o valor 0 lido, não permite continuar o processamento.
  - 2) Entre com um número (0 - saí): 2  
O número lido foi: 2  
Entre com um número (0 - saí): 3  
O número lido foi: 3  
Entre com um número (0 - saí): 0  
O produto dos valores lidos é: 6

### 4.5.2 Laços Repetidos

O controle de repetição permite que uma sequência de comandos seja executada repetidamente até que uma determinada condição de interrupção seja satisfeita.

**Sintaxe:**

- *Interrupção no início:*

```
repita
    se condição
        então
            abandone
    fim-se
    sequência B de comandos
fim-repita
```

**Onde:** A sequência B de comandos será repetida até que a condição seja satisfeita. Quando isto ocorrer, a repetição é interrompida e a sequência de comandos que vier logo após a expressão "fim-repita" passa a ser executada.

- *Interrupção no interior*

```
repita
    sequência A de comandos
    se condição
        então
            abandone
    fim-se
    sequência B de comandos
fim-repita
```

**Onde:** As sequências A e B de comandos serão repetidas até que a condição seja satisfeita. Quando isto ocorrer, a repetição é interrompida e a sequência de comandos que vier logo após a expressão "fim-repita" passa a ser executada.

**Exemplo:**

```
ALGORITMO exemplo da estrutura repita - fim-repita
1 [Início]
2 [Declaração de Variáveis]
   A,B,C: inteiro
3 [ Inicialização de Variáveis]
   A ← 10
   B ← A/10
   C ← B*5
```

```

4 [Processamento]
    repita
        B ← B+1
        se C ≤ B
            então
                abandone
        fim-se
        A ← A -1
    fim-repita
5 [Impressão]
    escreva (A," ", "B," e ",C)
6 [Fim]

```

**Rastreio:**

<u>A</u>	<u>B</u>	<u>C</u>	<u>Impressão:</u>
10	1	5	7,5 e 5
9	2		
8	3		
7	4		
	5		

### 4.5.3 Laços Repetidos com Teste no Final

**Sintaxe:**

```

repita
    comando(s)
até <condição>

```

**Onde:** O teste é feito depois de executar o(s) comando(s). O(s) comando(s) serão executado(s) pelo menos uma vez. Quando a condição é verdadeira, a seqüência de comandos que vier logo após a expressão “**até** <condição>” passa a ser executada. Se a condição for falsa, o(s) comando(s) serão executados novamente, até que a condição se torne verdadeira.

**Exemplo:**

```

ALGORITMO exemplo da estrutura repita - até
1 [Início]
2 [Declaração de Variáveis]
    X,Y,Z: real
3 [ Inicialização de Variáveis]
    X ← 40
    Y ← 10
    Z ← Y

```

```

4 [Processamento]
    repita
        Z ← Z*2
        Y ← Y*Z
    até Z ≥ X
5 [Impressão]
    escreva (X, ", ", Y, " e ", Z)
6 [Fim]

```

**Rastreio:**

<u>X</u>	<u>Y</u>	<u>Z</u>	<u>Impressão:</u>
40	10	10	40,8000 e 40
	200	20	
	8000	40	

#### 4.5.4 Laços Contados

Para algumas aplicações, o controle condicional de laços imposto pela construção "enquanto", é desnecessariamente complicado. Em muitos casos, pode-se desejar executar um laço, um número fixo de vezes, onde este número já é conhecido. Para aplicações como estas, existe uma forma modificada de construção repetitiva. Esta construção está descrita abaixo:

**Sintaxe:**

**para** variável = valor-inicial **até** valor-final [**de** <inc/dec> **em** <inc/dec>] **faça**  
comando(s)

**fim-para**

**Onde:** Esta estrutura é usada quando se deseja repetir um trecho de programa um número conhecido de vezes. O conjunto de comandos delimitados pelas cláusulas "**para**" e "**fim-para**" será executado cada vez que a variável receber um valor, começando com o <valor-inicial> e indo até o <valor-final>, sendo incrementada de <inc> ou decrementada de <dec>. Caso se omita o incremento, pois o mesmo é opcional, a variável será incrementada de um em um.

**Obs.:** Incremento de um em um não precisa constar na estrutura. Para qualquer incremento diferente de um, é obrigatório constar.

**Exemplo1:**

```

ALGORITMO exemplo1 da estrutura para
1 [Início]
2 [ Declaração de Constantes]
    TOTALREP ← 5
3 [Declaração de Variáveis]

```

```

        NOME : literal
        I : inteiro
4 [Processamento]
    para I=1 até TOTALREP faça
        escreva ("Entre com o nome: ")
        leia(NOME)
        escreva (I, " - ", NOME)
    fim-para
5 [Fim]

```

**Rastreio:**

<u>I</u>	<u>NOME</u>	<u>Impressão:</u>
1	José	1 - José
2	Antônio	2 - Antônio
3	João	3 - João
4	Maria	4 - Maria
5	Júlia	5 - Júlia

**Exemplo2:** Faça um algoritmo para calcular a seguinte série:

$$S = \frac{1}{1} + \frac{3}{2} + \frac{5}{3} + \frac{7}{4} + \dots + \frac{99}{50}$$

ALGORITMO exemplo2 da estrutura para

```

1 [Início]
2 [ Declaração de Constantes]
    PASSO ← 2
    MINDEN ← 1
    MAXDEN ← 50
3 [Declaração de Variáveis]
    NUM, I, S, TERMO : real
4 [Inicialização das Variáveis]
    NUM ← -1
    S ← 0
5 [Cálculo da Série]
    para I=MINDEN até MAXDEN faça
        NUM ← NUM + PASSO
        TERMO ← NUM / I
        S ← S + TERMO
    fim-para
6 [Impressão do Resultado]
    escreva ("O resultado do cálculo da série é: ", S)
7 [Fim]

```

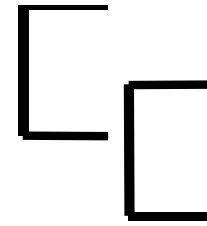
#### 4.5.5 Laços Aninhados

Assim como era possível ter uma construção "**se - então - senão**" dentro de outra construção "**se - então - senão**", também é possível ter um laço dentro do intervalo do

outro. As regras de embutimento são similares em ambos os casos. A construção interna deve estar completamente embutida na construção externa. Não pode haver sobreposição. As figuras abaixo mostram exemplos de embutimentos válidos e inválidos, respectivamente.



EMBUTIMENTO VÁLIDO



EMBUTIMENTO INVÁLIDO

## 5 Estruturas de Dados

### 5.1 Conceito

Para trabalharmos com informações no computador é preciso representá-la de forma adequada. Todos os computadores possuem formas fundamentais de representação de informações: as variáveis. Estas estruturas primitivas conseguem armazenar grande variedade de informações, porém esta variedade é limitada.

As estruturas de dados existem de forma a permitir que problemas que exigem formas complexas de representação de dados possam ser implementados.

Neste ponto, podemos citar como as principais estruturas de dados:

- a) Vetor (array);
- b) Registro (record);
- c) Pilha (stack);
- d) Fila (queue);
- e) Lista.

### 5.2 Vetor

#### 5.2.1 Definição

Até agora trabalhamos com variáveis simples, que nos permitem armazenar um valor a cada momento.

Muitas vezes, porém, precisamos usar um grupo de variáveis do mesmo tipo e que possam ser referenciadas pelo mesmo nome. Um exemplo dessa necessidade é mostrado abaixo, em um algoritmo que soma 10 números lidos e depois de imprimir o resultado da soma, imprime também os números lidos ( *sem o uso das estruturas de repetição*):

```
ALGORITMO soma 10 números
1 [início]
2 [Declaração de Variáveis]
   A,B,C,D,E,F,G,H,I,J,SOMA : real
3 [Inicialização da Variável]
```



```

SOMA ← 0
4 [Leitura dos números]
  escreva ("Entre com 10 números:")
  leia(A,B,C,D,E,F,G,H,I,J)
5 [Cálculo da Soma]
  SOMA ← A+B+C+D+E+F+G+H+I+J
6 [Impressão do resultado da soma]
  escreva("A soma dos 10 números lidos foi: ", SOMA)
7 [Impressão dos números lidos]
  escreva("Os números lidos foram: ", A, ", ", B, ", ", C, ", ", D, ", ", E, ", ", F, ", ", G,
    ", ", H, ", ", I, " e ", J)
8 [Fim]

```

Agora, imaginem se fosse 100, em vez de 10 números?

Por sorte, também podemos trabalhar com uma **VARIÁVEL INDEXADA**, mais conhecida como **VETOR**. Um vetor é um agregado homogêneo, ou seja, em processamento de dados, é um grupo de variáveis de mesmo tipo, que recebem o mesmo nome e são individualizadas por um **ÍNDICE**.

### 5.2.2 Sintaxe

A declaração do vetor se faz da seguinte forma:

**Nome do vetor: Vetor (dimensão) de tipos dos elementos**

A referência a um vetor no algoritmo se faz da seguinte forma:

**Nome do vetor (posição)**

### 5.2.3 Representação Gráfica

A representação gráfica desses vetores unidimensionais é:

	1	2	3	4	5	← índice (posição)
Nome do Vetor						← elementos

**Exemplo:** Agora, podemos rescrever o algoritmo anterior, que soma 10 números, estipulando esta quantidade na dimensão do vetor:

```

ALGORITMO soma 10 números
1 [Início]
2 [Declaração de Variáveis]
  VET : vetor (10) de real
  SOMA : real
  I : inteiro

```

```

3 [Inicialização da Variável]
  SOMA ← 0
4 [Leitura dos números e Cálculo da Soma]
  escreva ("Entre com 10 números: ")
  para I = 1 até 10 faça
    leia (VET(I))
    SOMA ← SOMA + VET(I)
  fim-para
5 [Impressão do Resultado da Soma]
  escreva ("A soma dos 10 números lidos foi: ", SOMA)
6 [Impressão dos Números Lidos]
  escreva ("Os números lidos foram: ")

  para I = 1 até 10 faça
    escreva (VET(I), " , ")
  fim-para
7 [Fim]

```

**Rastreio:**

	<u>VET</u>	<u>SOMA</u>	<u>I</u>	
1	2	0	1	1
2	3	110	2	2
3	4		3	3
4	1		4	4
5	10		5	5
6	20		6	6
7	45		7	7
8	15		8	8
9	5		9	9
10	5		10	10

**Mensagens:**

A soma dos 10 números lidos foi: 110  
 Os números lidos foram: 2,3,4,1,10,20,45,15,5,5

**5.2.4 Operações sobre Vetores**

Quando queremos inserir algum valor no vetor, basta atribuir a um de seus elementos este valor. A escolha do elemento que receberá o valor é feita por quem está construindo o algoritmo (obedecendo a algum critério) e a referência a esse elemento é feita através de um índice.

Suponha que tivéssemos um vetor com cinco elementos, todos sem valor, e que se chamasse NÚMEROS:

NUMEROS	
1	
2	
3	
4	
5	

Se desejarmos inserir o valor 48 no terceiro elemento do vetor e o valor 74 no quarto elemento, bastaria fazer:

NUMEROS (3)  $\leftarrow$  48

NUMEROS (4)  $\leftarrow$  74

Agora, o vetor NUMEROS ficaria assim:

NUMEROS	
1	
2	
3	48
4	74
5	

Suponha agora, que desejássemos imprimir o terceiro elemento deste mesmo vetor. Isso é feito, também, por meio de um índice como mostra o comando abaixo:

escreva (NUMEROS(3))

**Exemplo1:** Faça um algoritmo para calcular a média aritmética de uma quantidade N de valores numéricos lidos.

**1.ª Solução:**

ALGORITMO média

1 [Início]

2 [Declaração de Constante]

MAX  $\leftarrow$  10

3 [Declaração de Variáveis]

MEDIA, SOMAT : real

I, N : inteiro

VETNUM : vetor(MAX) de real

4 [Inicialização das Variáveis]

SOMAT  $\leftarrow$  0

5 [Leitura da Quantidade de Valores Numéricos]

escreva ("Entre com a quantidade de valores numéricos que serão lidos (a quantidade máxima permitida é 10): ")

repita

leia (N)

```

    até (N > 0) e (N ≤ MAX)
6 [Leitura do Vetor VETNUM]
    para I = 1 até N faça
        escreva ("Entre com o ", I, ".º valor numérico: ")
        leia ( VETNUM(I) )
    fim-para
7 [Cálculo da Média]
    para I = 1 até N faça
        SOMAT ← SOMAT + VETNUM(I)
    fim-para
    MEDIA ← SOMAT / N
8 [Impressão dos Valores Lidos]
    escreva ("Os valores lidos foram: ")

    para I = 1 até N faça
        escreva ( VETNUM(I) )
    fim-para
9 [Impressão do Resultado]
    escreva ("A Média dos valores lidos é: ", MEDIA)
10 [Fim]

```

## **2.ª Solução:**

ALGORITMO média

```

1 [Início]
2 [Declaração de Constantes]
    MAX ← 10
    FLAG ← 0
3 [Declaração de Variáveis]
    MEDIA, SOMAT, NUM      : real
    I, N                    : inteiro
    VETNUM                  : vetor (MAX) de real
4 [Inicializações]
    SOMAT ← 0
    I ← 0
5 [Leitura dos Valores]
    escreva ("Entre com os valores numéricos (0 – Sai): ")
    leia (NUM)
    enquanto (NUM ≠ FLAG) e (I < MAX) faça
        I ← I + 1
        VETNUM (I) ← NUM
        leia (NUM)
    fim-enquanto
    N ← I
6 [Verificação de Vetor Cheio]
    se (N = MAX) e (NUM ≠ FLAG)
        então
            escreva ("O valor ", NUM, "não foi armazenado, pois o Vetor de
                Números está cheio !")

```

```

    fim-se
7 [Cálculo da Média e Impressão]
    se N > 0
        então
            para I = 1 até N faça
                SOMAT ← SOMAT + VETNUM(I)
            fim-para
            MEDIA ← SOMAT / N
            escreva ("Os valores lidos foram: ")
            para I = 1 até N faça
                escreva (VETNUM(I))
            fim-para
            escreva ("A Média Aritmética dos valores lidos é: ", MEDIA)
        senão
            escreva ("Não há valores para cálculo da Média !")
    fim-se
8 [Fim]

```

**Exemplo2:** Dados os vetores **A** e **B**, com **3** elementos numéricos cada, faça um algoritmo para obter o vetor **C**, através da soma dos vetores **A** e **B**.

```

ALGORITMO soma de vetores
1 [Início]
2 [Declaração de Constantes]
    MAX ← 3
    FLAG ← 0
3 [Declaração de Variáveis]
    VETA, VETB, VETC: vetor (MAX) de real
    I : inteiro
4 [Leitura dos Valores]
    escreva ("Entre com os valores que irão compor o Vetor A (0 – Sai): ")
    para I = 1 até MAX faça
        leia (VETA(I))
    fim-para
5 [Leitura de VETB]
    escreva ("Entre com os elementos que irão compor o Vetor B: ")
    para I = 1 até MAX faça
        leia (VETB(I))
    fim-para
6 [Obtenção do VETC]
    para I = 1 até MAX faça
        VETC(I) ← VETA(I) + VETB(I)
    fim-para
7 [Impressão dos Valores Lidos]
    escreva ("Os Vetores lidos foram:")
    para I = 1 até MAX faça
        escreva ("Vetor A ( ", I, " ) : ", VETA(I), " e Vetor B ( ", I, " ) : ", VETB(I))
    fim-para
8 [Impressão do Vetor Resultante]

```

```

    escreva ("O Vetor C resultante é:")
    para I = 1 até MAX faça
        escreva ("Vetor C ( ", I, " ) : ", VETC(I))
    fim-para
9 [Fim]

```

**Exemplo3:** Um armazém trabalha com 50 mercadorias diferentes. O dono do armazém anota a descrição, a quantidade de cada mercadoria vendida durante o mês e o seu preço unitário. Faça um algoritmo para calcular o faturamento mensal do armazém.

```

ALGORITMO faturamento mensal
1 [Início]
2 [Declaração de Constantes]
    MAX ← 50
3 [Declaração de Variáveis]
    I          : inteiro
    DESCR1     : vetor (MAX) de literal
    QUANT      : vetor (MAX) de inteiro
    PRECO      : vetor (MAX) de real
    FATURA    : real
4 [Inicialização da Variável]
    FATURA ← 0
5 [Leitura dos Dados e Cálculo do Faturamento]
    para I = 1 até MAX faça
        escreva ("Entre com a descrição da mercadoria: ")
        repita
            leia (DESCR1(I))
        até DESCR1(I) ≠ " "
        escreva ("Entre com o preço unitário da mercadoria: ")
        repita
            leia (PRECO(I))
        até PRECO(I) > 0
        escreva ("Entre com a quantidade vendida no mês da mercadoria: ")
        repita
            leia (QUANT(I))
        até (QUANT(I) ≥ 0)
        FATURA ← FATURA + (QUANT(I) * PRECO(I))
    fim-para
6 [Impressão dos dados lidos]
    para I = 1 até MAX faça
        escreva ("A mercadoria ", DESCR1(I), " lida, tem preço unitário = ",
            PRECO(I), " e teve uma quantidade vendida no mês = ",
            QUANT(I))
    fim-para
7 [Impressão do Faturamento Mensal do Armazém]
    escreva ("O Faturamento Mensal do Armazém é: ", FATURA)
8 [Fim]

```

**Exemplo4:** Faça um algoritmo para ler um vetor de 10 elementos numéricos e verificar se existem elementos iguais a 30. Se existirem, escreva as posições em que estão armazenados.

```

ALGORITMO localiza 30
1 [Início]
2 [Declaração de Constantes]
    MAX ← 10
    FIXO ← 30
3 [Declaração de Variáveis]
    I, J      : inteiro
    NUMLIDOS  : vetor (MAX) de real
    POSIC30   : vetor (MAX) de inteiro
4 [Inicialização da Variável]
    J ← 0
5 [Leitura do Vetor NUMLIDOS]
    escreva ("Entre com 10 elementos numéricos: ")
    para I = 1 até MAX faça
        leia (NUMLIDOS(I))
    fim-para
6 [Identificação de posições com elementos iguais a 30]
    para I = 1 até MAX faça
        se NUMLIDOS(I) = FIXO
            então
                J ← J + 1
                POSIC30(J) ← I
        fim-se
    fim-para
7 [Impressão dos números lidos]
    para I = 1 até MAX faça
        escreva ("O número lido na posição ", I, " foi: ", NUMLIDOS(I))
    fim-para
8 [Impressão do Resultado]
    se J > 0
        então
            escreva ("Dos números lidos acima, os das posições ")
            para I = 1 até J faça
                escreva (POSIC30(I))
            fim-para
            escreva (" são iguais a ", FIXO)
        senão
            escreva ("Não houve a leitura de elementos iguais a ", FIXO)
    fim-se
9 [Fim]

```

## 5.3 Matriz

### 5.3.1 Definição

Considere o seguinte exemplo: deseja-se ler quatro notas de cada aluno de uma classe de 20 alunos. Depois, deseja-se imprimir um relatório onde cada linha contém os seguintes dados de cada aluno: as quatro notas e a média obtida.

Neste exemplo, teríamos que usar cinco vetores: um para guardar as médias e mais quatro vetores para guardar cada uma das notas dos alunos. Para representar estruturas desta forma, introduzimos uma nova estrutura de dados, a qual chamaremos de **matriz**. Como um vetor, uma matriz é uma variável indexada; entretanto, são estruturas bidimensionais, pois utilizam dois índices, enquanto que um vetor, é uma estrutura unidimensional por utilizar apenas um índice. Com isso, os dados dos alunos serão guardados em uma variável indexada de dois índices: o primeiro especificando a linha (dados de um aluno) e o segundo especificando a coluna (nota1, nota2, nota3, nota4 e média). Os índices são separados por vírgulas. Se o relatório mostrado é implementado como uma matriz, chamada MAT\_NOTAS, a segunda nota do quarto aluno será armazenada em MAT\_NOTAS(4,2), como mostra a figura abaixo:

MAT\_NOTAS

	NOTA1	NOTA2	NOTA3	NOTA4	MÉDIA
1					
2					
3					
4		X			
5					
:					
:					
20					

### 5.3.2 Sintaxe

Você deve ter notado que um vetor é um caso especial de matriz, tendo apenas uma dimensão. Realmente este é o caso, e sugere que certas restrições sejam as mesmas, tanto para matrizes, como para vetores. Por exemplo, como no caso dos vetores, insistiremos em que todos os elementos de uma matriz devam ser do mesmo tipo.

De um modo geral, podemos declarar uma matriz da seguinte forma:

**Nome da matriz: matriz(n,m) de tipos dos elementos**

**Onde:**

**Nome da matriz** → É o nome dado à variável indexada bidimensional.

**n, m** → São as dimensões da matriz, onde **n** se refere ao número de linhas que contém a matriz e **m** se refere ao número de colunas que contém a matriz.

**tipo dos elementos** → É o tipo dos elementos que compõem a matriz.



	1	2	3	...	m
1					
2					
3					
:					
:					
n					

**Exemplo1:** Faça um algoritmo para ler e imprimir uma matriz de N linhas e N colunas.

ALGORITMO matriz

1 [Início]

2 [Declaração de Constantes]

MAX  $\leftarrow$  10

3 [Declaração de Variáveis]

MATRIZ : matriz (MAX,MAX) de real

I, J, L, C : inteiro

4 [Leitura do Dimensionamento da Matriz]

escreva ("Entre com o número de linhas que conterà a matriz: ")

repita

leia (L)

até (L > 1) e (L  $\leq$  MAX)

escreva ("Entre com o número de colunas que conterà a matriz: ")

repita

leia (C)

até (C > 1) e (C  $\leq$  MAX)

5 [Leitura da Matriz]

escreva ("Entre com os elementos numéricos que irão compor a matriz: ")

para I = 1 até L faça

para J = 1 até C faça

leia (MATRIZ(I,J))

fim-para

fim-para

6 [Impressão da Matriz lida]

escreva ("Os elementos numéricos lidos foram: ")

para I = 1 até L faça

para J = 1 até C faça

escreva (MATRIZ(I,J))

fim-para

fim-para

7 [Fim]

**Exemplo2:** Faça um algoritmo que leia duas matrizes inteiras e positivas de dimensão 3 X 3, calcule e imprima a soma das matrizes.

ALGORITMO soma de matrizes

1 [Início]

2 [Declaração de Constantes]

MAX  $\leftarrow$  3

3 [Declaração de Variáveis]

MAT1, MAT2, MAT3: matriz (MAX,MAX) de inteiro

I, J : inteiro

4 [Leitura da Primeira Matriz]

escreva ("Entre com os elementos numéricos que irão compor a primeira matriz: ")

para I = 1 até MAX faça

para J = 1 até MAX faça

repita

leia (MAT1(I,J))

até (MAT1(I,J) > 0)

fim-para

fim-para

5 [Leitura da Segunda Matriz]

escreva ("Entre com os elementos numéricos que irão compor a segunda matriz: ")

para I = 1 até MAX faça

para J = 1 até MAX faça

repita

leia (MAT2(I,J))

até (MAT2(I,J) > 0)

fim-para

fim-para

6 [Soma das Matrizes lidas]

para I = 1 até MAX faça

para J = 1 até MAX faça

MAT3(I,J)  $\leftarrow$  MAT1(I,J) + MAT2(I,J)

fim-para

fim-para

7 [Impressão das Matrizes lidas]

escreva ("A primeira matriz lida foi: ")

para I = 1 até MAX faça

para J = 1 até MAX faça

escreva (MAT1(I,J))

fim-para

fim-para

escreva ("A segunda matriz lida foi: ")

para I = 1 até MAX faça

para J = 1 até MAX faça

escreva (MAT2(I,J))

fim-para

fim-para

8 [Impressão da Matriz resultante]

escreva ("A matriz resultante da soma das matrizes, anteriormente lidas, será: ")

```

    para I = 1 até MAX faça
        para J = 1 até MAX faça
            escreva (MAT3(I,J))
        fim-para
    fim-para
9 [Fim]

```

**Exemplo3:** Dada uma matriz MAT de 4 X 5 elementos, faça um algoritmo para somar os elementos de cada linha gerando o vetor SOMALIN. Em seguida, somar os elementos do vetor SOMALIN na variável TOTAL que deverá ser impressa no final, da seguinte forma:

	1	2	3	4	5		Somalin	
1	1	0	2	-1	3	→	5	1
2	4	3	2	1	0	→	10	2
3	1	-2	3	4	5	→	11	3
4	8	5	1	3	2	→	19	4
							↓	
							45	
							Total	

```

ALGORITMO matriz_vetor
1 [Início]
2 [Declaração de Constantes]
    LIN ← 4
    COL ← 5
3 [Declaração de Variáveis]
    MAT      : matriz (LIN,COL) de real
    SOMALIN   : vetor (LIN) de real
    TOTAL     : real
    I, J      : inteiro
4 [Inicialização da Variável]
    TOTAL ← 0
5 [Leitura da Matriz]
    escreva ("Entre com os elementos numéricos que irão compor a matriz: ")
    para I = 1 até LIN faça
        para J = 1 até COL faça
            leia (MAT(I,J))
        fim-para
    fim-para
6 [Formação do vetor SOMALIN]
    para I = 1 até LIN faça
        SOMALIN(I) ← 0
        para J = 1 até COL faça
            SOMALIN(I) ← MAT(I,J) + SOMALIN(I)
        fim-para
        TOTAL ← TOTAL + SOMALIN(I)
    fim-para

```

```
7 [Impressão da Matriz lida]
  escreva ("A matriz lida foi: ")
  para I = 1 até LIN faça
    para J = 1 até COL faça
      escreva (MAT(I,J))
    fim-para
  fim-para

8 [Impressão dos resultados]
  para I = 1 até LIN faça
    escreva ("A soma dos elementos contidos na ", I, ".ª linha da matriz
    é:", SOMALIN(I))
  fim-para
  escreva ("A soma de todos os valores armazenados na matriz é: ", TOTAL)
9 [Fim]
```

## 5.4 Registro

### 5.4.1 Definição

Um registro é um grupo de informações que em geral (e não necessariamente) são de tipos diferentes. Por poder ser composto de variável de tipos diferentes, um registro também é conhecido como “variável composta heterogêneas”.

### 5.4.2 Sintaxe

```
Nome_do_Registro = Registro
  Nome_da_Variável_1 : tipo
  Nome_da_Variável_2 : tipo
  Nome_da_Variável_3 : tipo
  .
  Nome_da_Variável_n : tipo
Fim_Registro
```

**Onde:**

**Registro** e **Fim-Registro** → São palavras-chave.

**Nome\_da\_Variável** e **tipo** → São nomes dos campos (variáveis) que compõem o registro, cada um com o seu respectivo tipo.

**Os campos (variáveis) que compõem o registro são acessadas da seguinte forma:**

Nome\_do\_Registro.Nome\_da\_Variável

**Exemplo:** Seja REG um registro que possui as seguintes informações: matrícula, nome, curso e endereço, o nome completo das variáveis de REG será:

REG.MATRICULA  
REG.NOME  
REG.CURSO  
REG.ENDERECO

O endereço de uma pessoa, geralmente é composto de bairro, logradouro, número e CEP. Com isso, a estrutura de endereço será:

ENDERECO.BAIRRO  
ENDERECO.LOGRADOURO  
ENDERECO.NUMERO  
ENDERECO.CEP

Observe que uma estrutura do tipo registro pode conter outra estrutura do tipo registro. ENDERECO é um registro e faz parte de REG, que também é um registro.

Uma definição completa do registro REG, de forma gráfica, ficará da seguinte forma representado:

**REG**  
    **MATRICULA**  
    **NOME**  
    **CURSO**  
    **ENDERECO**  
        **BAIRRO**  
        **LOGRADOURO**  
        **NUMERO**  
        **CEP**

### 5.4.3 Exemplos

a) Declaração:

FICHA = REGISTRO  
    NOME : literal  
    FONE : literal  
    CPF : literal  
    DVCPF : inteiro  
    MATRIC : inteiro  
    SALARIO : real  
    CODIGO : caracter  
FIM-REGISTRO

b) Algoritmo de demonstração do uso de Registro:

## ALGORITMO Ex2\_REG

```

1 [Início]
2 [Declaração de Variáveis]
    PESSOAS = Registro
        NOME      : literal
        IDADE     : inteiro
        SEXO      : caracter
        ALTURA   : real
    Fim-Registro
3 [Leitura]
    escreva ("Entre com o Nome → ")
    leia(PESSOAS.NOME)
    escreva ("Entre com a Idade → ")
    leia(PESSOAS.IDADE)
    escreva ("Entre com o Sexo → ")
    leia(PESSOAS.SEXO)
    escreva ("Entre com a Altura → ")
    leia(PESSOAS.ALTURA)
4 [Impressão]
    escreva ("Você digitou os seguintes dados : ")
    escreva(PESSOAS.NOME)
    escreva(PESSOAS.IDADE)
    escreva(PESSOAS.SEXO)
    escreva(PESSOAS.ALTURA)
5 [Fim]

```

#### 5.4.4 Declaração COM

Serve para tornar menos cansativa a notação exigida pela estrutura registro. O uso desta declaração abrevia as referências aos campos do registro, permitindo que se cite somente os nomes dos campos.

**Sintaxe:**

**Com** variavel\_a\_ser\_declarada **faça**  
COMANDO(s)

**Onde:**

**Com e faça** → São palavras-chave.

**Variável\_a\_ser\_declarada** → É o identificador da variável do tipo Registro.

**COMANDO(s)** → Pode ser um comando simples ou composto.

**Exemplo:**

```

      .....
X = Registro

```

```

NOTAS      : vetor[10] de real
CONCEITO   : caracter
MEDIA      : real
Fim-Registro

```

```

X.CONCEITO := 'A'
X.NOTAS[1] := 10.0
X.NOTAS[5] := 8.0
X.MEDIA := 7.0

```

**Refazendo a área de comandos com o uso do Com :**

```

Com X faça
  CONCEITO := 'A'
  NOTAS[1] := 10.0
  NOTAS[5] := 8.0
  MEDIA := 7.0

```

### 5.4.5 Vetores de Registros

Como sabemos, os elementos que compõem os vetores podem ser de tipo simples ou estruturados. Portanto, é comum o uso de “**vetores de registros**”, cuja declaração e representação é feita a seguir:

Seja o registro MUSICA, cujos campos são NOME, INTERPRETE, GRAVADORA; deseja-se guardar em um vetor VET, estas informações sobre N músicas.

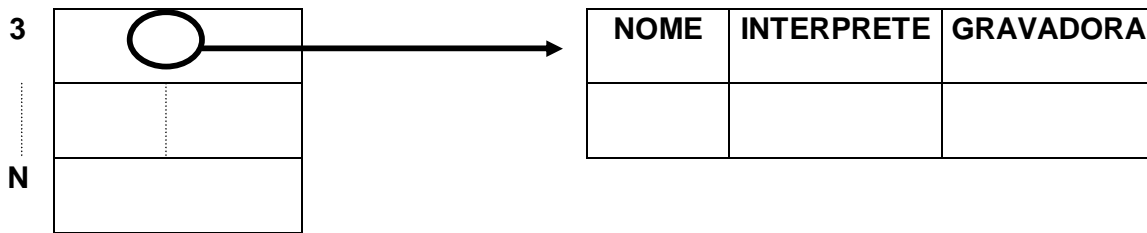
```

MUSICA = Registro
  NOME      : literal
  INTERPRETE : literal
  GRAVADORA : literal
Fim-Registro
VET = vetor(N) de MUSICA

```

A figura, a seguir, mostra a representação gráfica do vetor de registros “**VET**”:



**Designação dos campos:**

leia (VET( I ).NOME)  
 leia (VET( I ).INTERPRETE)  
 leia (VET( I ).GRAVADORA)

OU

**Com** VET( I ) **faça**  
 leia (NOME)  
 leia (INTERPRETE)  
 leia (GRAVADORA)

**5.4.6 Registros de Vetores**

Assim como temos os Vetores de Registros, onde em cada posição do vetor pode ser colocado o conteúdo de uma variável do tipo estruturado Registro, podemos ter o caso inverso, isto é, um registro onde um ou mais campos podem ser representados por variáveis do tipo Vetor.

**Exemplo:**

Como deve ser representado em Estrutura de Dados a situação em que há 20 shows a serem apresentados, sendo que cada um deles será repetido em 5 datas e locais diferentes?

Veja a seguir a declaração da variável que chamaremos de SHOWS:

APRESENT = **Registro**

NOME : literal(30)  
 LOCAL : vetor(5) de literal  
 DATA : vetor(5) de literal

**Fim-Registro**

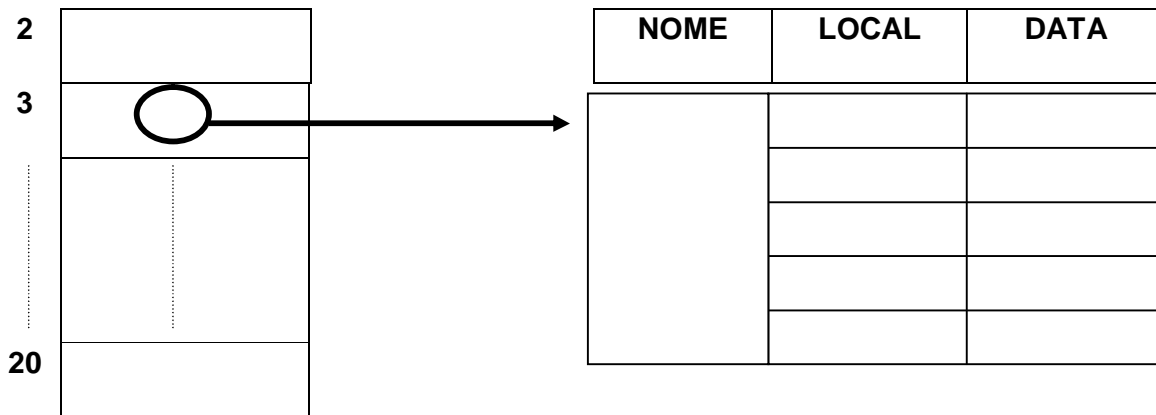
NOMESHOW = vetor(20) de APRESENT

SHOWS : NOMESHOW

A figura, a seguir, mostra a representação gráfica do vetor “**SHOWS**” que contém Registro de Vetores:







*Acesso aos campos do registro:*

```

SHOWS( I ).NOME ← "O LADO BOM DO MAL"
SHOWS( I ).LOCAL( J ) ← "BRASÍLIA"
SHOWS( I ).DATA( J ) ← "23/02/99"
SHOWS( I ).LOCAL( J+1 ) ← "SALVADOR"
SHOWS( I ).DATA( J+1 ) ← "28/02/99"
SHOWS( I+1 ).NOME ← "O MISTÉRIO DA MENTE HUMANA"
SHOWS( I+1 ).LOCAL( J ) ← "SÃO PAULO"
SHOWS( I+1 ).DATA( J ) ← "09/03/99"
SHOWS( I+1 ).LOCAL( J+1 ) ← "CAMPINAS"
SHOWS( I+1 ).DATA( J+1 ) ← "29/04/99"

```

### 5.4.7 Registros Embutidos

O campo de um REGISTRO também pode ser uma variável de tipo primitivo ou estruturado, inclusive outro REGISTRO.

Um exemplo clássico é o da FICHA que cadastra uma pessoa no consultório médico, na escola, na empresa, etc., cujo conteúdo, entre outros campos tem: NOME, FILIAÇÃO, ENDEREÇO, IDADE, ESTADO CIVIL. Sabe-se que FILIAÇÃO é composta de PAI e MÃE, e que endereço contém RUA, NÚMERO, COMPLEMENTO, BAIRRO, CIDADE, CEP e UF.

#### Exemplo:

```

[ Declaração de Variáveis ]
  PAIS = Registro
    PAI, MAE : literal
  Fim-Registro
  ENDER = Registro
    RUA, BAIRRO, CIDADE : literal
    NUMERO                : literal
    COMPLEM                : literal
    CEP                    : literal
    UF                     : literal
  Fim-Registro

```

CADASTRO = **Registro**

NOME : literal  
 FILIAÇÃO : PAIS  
 ENDEREÇO : ENDER  
 IDADE : inteiro  
 ESTCIVIL : literal

**Fim-Registro**

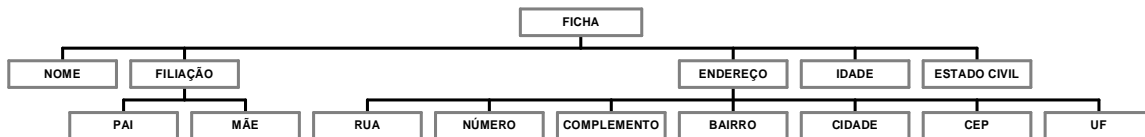
FICHA : CADASTRO;

*Referência aos campos MAE, IDADE e CIDADE, por exemplo:*

a) FICHA.FILIAÇÃO.MAE ← “ANA MARIA SILVA”  
 FICHA.IDADE ← 15  
 FICHA.ENDEREÇO.CIDADE ← “SÃO PAULO”

b) **Com** FICHA, FILIAÇÃO, ENDEREÇO **faça**  
 MAE ← “ANA MARIA SILVA”  
 IDADE ← 15  
 CIDADE ← “SÃO PAULO”

*A representação hierárquica do registro FICHA é mostrada pela figura abaixo:*



## 6 Sub-Algoritmos: Funções e Procedimentos

### 6.1 Definição (Importância e Utilidade)

Freqüentemente, uma certa seqüência de comandos deveria ser repetida em vários lugares de um algoritmo. Isto acarretaria: perda de tempo, perda de trabalho e aumento do texto do algoritmo (o que o deixaria muito confuso).

A fim de resolver isso, vamos introduzir o conceito de sub-algoritmo. Podemos dizer que, informalmente, que sub-algoritmo é um nome que serve para abreviar uma certa seqüência de comandos, que é usada várias vezes no algoritmo.

Sub-algoritmo é um dos poucos conceitos, realmente fundamentais, da arte e da técnica da programação estruturada, cujo domínio exerce uma importância no Estilo e na Qualidade do trabalho de um programador. Sub-Algoritmo serve, também, para melhor dividir o algoritmo, estruturá-lo e modularizá-lo.

Divisão é essencial para entender um algoritmo, principalmente quando ele é tão complexo que seu texto não possa ser compreendido todo de uma só vez.

Estruturação por sub-algoritmos é indispensável tanto para a boa documentação como para a verificação de algoritmos.

A modularização possibilita um aumento na produção de algoritmos pelo uso de componentes de algoritmos escritos previamente.

Portanto, muitas vezes é conveniente formular como sub-algoritmos, uma sequência de comandos, mesmo quando ela ocorre em só lugar no algoritmo e o texto não será encurtado.

## 6.2 Declaração de Sub-Algoritmos

Uma declaração de sub-algoritmo consiste em duas partes:

- a) O **cabeçalho**, onde são definidos o nome do sub-algoritmo, seu tipo (função ou procedimento), os parâmetros utilizados, bem como a forma como estes parâmetros se comunicarão com o algoritmo “chamador”;
- b) O **corpo**, onde são colocadas as instruções que serão executadas pelo sub-algoritmo.

## 6.3 Funções

### 6.3.1 Definição

Função é um sub-algoritmo que, necessariamente, retorna um valor, podendo ser usada dentro de expressões.

A utilização de um sub-algoritmo (neste caso, uma função) afeta o fluxo de controle no algoritmo. Quando uma função é chamada ou ativada, o controle é passado para o trecho do algoritmo onde a função está declarada. Após a execução da função, o controle retorna ao ponto de chamada.

### 6.3.2 Declaração

A declaração de uma função, como de todo sub-algoritmo, consiste de um cabeçalho e de um corpo.

#### Sintaxe:

```
Função nome-da-função : tipo-valor-retornado  
1 [Início]  
    Comando(s)  
n [Fim]
```

#### **Onde:**

A palavra “**Função**” indica que será declarada uma função, cujo nome vem logo em seguida (*nome-da-função*). Como toda função retorna um valor, é necessário indicar na declaração da função o tipo do valor retornado. Isto é feito com o uso do caracter especial “ : ” seguido do “*tipo-valor-retornado*”. O corpo da função é delimitado pelas palavras “*Início*” e “*Fim*”.

#### Exemplo:

```
ALGORITMO Hipotenusa
1 [Declaração de Variáveis]
  A, B, C : real
2 [Declaração da Função]
  Função HIPO : real
  2.1 [Início]
  2.2 [Processamento]
     $HIPO \leftarrow (B^2 + C^2)^{1/2}$ 
  2.3 [Fim]
3 [Corpo do Algoritmo Principal]
  3.1 [Início]
  3.2 [Leitura]
    escreva ("Entre com os valores de B e C: ")
    leia (B,C)
  3.3 [Processamento]
     $A \leftarrow HIPO$   [ Ativação da Função HIPO ]
  3.4 [Impressão]
    escreva ("O valor de A = ", A)
  3.5 [Fim]
```

## 6.4 Procedimentos

### 6.4.1 Definição

Embora a função seja uma ferramenta muito útil na construção de algoritmos, sua capacidade é restrita. Em algumas situações desejamos especificar uma operação que não é convenientemente determinada como parte de uma expressão. Tais operações, como cálculos que resultem em vários valores, tem um alcance muito maior, e o retorno de um valor simples para uso numa expressão seria muito artificial.

Por essas razões, introduzimos uma segunda forma de sub-algoritmo, que chamaremos de **Procedimento**. Embora a função e o procedimento sejam similares em muitos aspectos, existem duas importantes diferenças:

- a) A ativação de um procedimento não pode ser feita dentro de uma expressão. Após a execução do procedimento, o controle retorna ao ponto imediatamente após ao da ativação do procedimento.
- b) Não existe retorno de um único valor, como no caso da função. Qualquer quantidade de valores pode ser retornada.

### 6.4.2 Declaração

Apresentamos, abaixo, a declaração de procedimentos:

#### Sintaxe:

```
Procedimento nome-do-procedimento
1 [Início]
```

Comando(s)  
n [Fim]

**Onde:**

A palavra “**Procedimento**” indica que será declarado um procedimento, cujo nome está especificado em “nome-do-procedimento”. O corpo do procedimento é delimitado pelas palavras “*Início*” e “*Fim*”.

**Exemplo1:**

```

ALGORITMO Incrementa_Índices
1 [Declaração de Variáveis]
    IND1, IND2, IND3 : inteiro
2 [Declaração do Procedimento]
    Procedimento INC_INDICES
        2.1 [Início]
        2.2 [Processamento]
            IND1  $\leftarrow$  IND1 + 1
            IND2  $\leftarrow$  IND2 + 1
            IND3  $\leftarrow$  IND3 + 1
        2.3 [Fim]
3 [Corpo do Algoritmo Principal]
    3.1 [Início]
    3.2 [Inicialização das Variáveis]
        IND1  $\leftarrow$  0
        IND2  $\leftarrow$  0
        IND3  $\leftarrow$  0
    3.3 [Processamento]
        INC_INDICES      [ Ativação do Procedimento ]
    3.4 [Impressão]
        escreva (“Índice 1 = ”, IND1, “ , Índice 2 = ”, IND2, “ e Índice 3
                = ”, IND3)
    3.5 [Fim]

```

**Exemplo2:**

```

Procedimento TROCAXY
1 [Início]
2 [Processamento]
    AUX  $\leftarrow$  X
    X  $\leftarrow$  Y
    Y  $\leftarrow$  AUX
3 [Fim]

```

**6.5 Escopo de Variáveis**

Se um objeto, seja uma variável ou sub-algoritmo, só tem significado em um certo trecho do programa, ele é chamado de objeto **local**. Se um objeto tem que ser usado em qualquer trecho do algoritmo, então ele é chamado de objeto **global**.

O sub-algoritmo anterior (*TROCAXY*) utiliza uma variável auxiliar (*AUX*) que não foi declarada no procedimento. Neste caso, esta variável não é *local* ao procedimento *TROCAXY*.

Se a variável *AUX* só for usada neste procedimento, então seria conveniente torná-la **local**. A nova versão do sub-algoritmo *TROCAXY* ficará então da seguinte forma:

```
Procedimento TROCAXY
1 [Início]
2 [Declaração de Variáveis]
    AUX : real
3 [Processamento]
    AUX  $\leftarrow$  X
    X  $\leftarrow$  Y
    Y  $\leftarrow$  AUX
4 [Fim]
```

As variáveis declaradas no corpo do algoritmo principal são **globais** a todo o algoritmo, podendo ser usadas em qualquer parte do mesmo.

## 6.6 Parâmetros de Sub-Algoritmos

Se uma determinada seqüência de operações é aplicada a operandos diferentes em várias partes do algoritmo, esta seqüência poderá ser formulada como um sub-algoritmo, em que os operandos serão passados ao sub-algoritmo como parâmetros.

Com isso, o cabeçalho do sub-algoritmo será acrescido dos parâmetros.

### Exemplo:

```
Função MEDIA (X,Y) : real
1 [Início]
2 [Processamento]
    MEDIA  $\leftarrow$  (X + Y) / 2
3 [Fim]
```

Neste exemplo, X e Y são parâmetros que a função recebe.

As variáveis introduzidas no cabeçalho do sub-algoritmo são chamadas de parâmetros formais.

Os objetos que substituirão os parâmetros formais são denominados parâmetros reais, sendo especificados em cada chamada do sub-algoritmo. No caso da função acima, poderíamos ativá-la da seguinte maneira:

Z  $\leftarrow$  MEDIA (8,7)

Os parâmetros reais podem ser passados aos parâmetros formais de um sub-algoritmo de duas formas diferentes: **passagem de parâmetro por valor** e **passagem de parâmetro por referência**.

#### 6.6.1 Passagem de Parâmetro por Valor

Os parâmetros passados por **valor**, ao retornar do sub-algoritmo mantém os seus valores originais (*NÃO SÃO ALTERADOS DENTRO DO SUB-ALGORITMO*).

**Exemplo:**

```
PROCEDIMENTO IMPREMAT(LIN,COL,MAT)
1 [Início]
2 [Declaração de Variáveis]
   I, J : inteiro
3 [Impressão]
   para I = 1 ate LIN faça
       para J = 1 ate COL faça
           escreva(MAT(I,J))
       fim-para
   fim-para
4 [Fim]
```

O procedimento demonstrado acima tem o objetivo de imprimir uma matriz, anteriormente lida. Com isso, os parâmetros *LIN*, *COL* e *MAT* passados no cabeçalho do procedimento, conterão informações necessárias para efetuar tal operação e não terão os seus conteúdos alterados ao retornarem à execução do Algoritmo Principal.

#### 6.6.2 Passagem de Parâmetro por Referência (Endereço)

Os parâmetros passados por **referência** ou **endereço**, são modificados no sub-algoritmo e mantém tal modificação após o encerramento da execução do sub-algoritmo.

**Exemplo:**

```
PROCEDIMENTO LEDIM(LIN,COL)
1 [Início]
2 [Leitura]
   repita
       leia(LIN)
   até (LIN>1) e (LIN ≤ 50)
   repita
       leia(COL)
   até (COL>1) e (COL ≤ 50)
3 [Fim]
```

O procedimento demonstrado acima, tem o objetivo de ler a dimensão da matriz, que ainda não é conhecida. Com isso, os parâmetros *LIN* e *COL* passados no cabeçalho do procedimento, terão os seus conteúdos alterados ao retornar à execução do Algoritmo Principal.

**Exemplo:** O programa abaixo soma duas matrizes quaisquer, usando sub-rotinas.

*EM ALGORITMO:*

ALGORITMO MATRIZES

1 [Declaração de Constantes]

MIN  $\leftarrow$  1

MAX  $\leftarrow$  50

2 [Declaração de Variáveis]

LINHAS, COLUNAS : inteiro

MATA, MATB, MATC : matriz(MAX,MAX) de real

3 [Declaração do Procedimento LEDIM]

PROCEDIMENTO LEDIM(DIM)

3.1 [Início]

3.2 [Leitura]

repita

leia(DIM)

até (DIM>MIN) e (DIM  $\leq$  MAX)

3.3 [Fim]

4 [Declaração do Procedimento LEMAT]

PROCEDIMENTO LEMAT(LIN,COL,MAT)

4.1 [Início]

4.2 [Declaração de Variáveis]

I, J : inteiro

4.3 [Leitura da matriz]

para I = MIN ate LIN faça

para J = MIN ate COL faça

leia(MAT(I,J))

fim-para

fim-para

4.4 [Fim]

5 [Declaração do Procedimento SOMAT]

PROCEDIMENTO SOMAT(LIN,COL,MAT1,MAT2,MAT3)

5.1 [Início]

5.2 [Declaração de Variáveis]

I, J : inteiro

5.3 [Realização da Soma]

para I = MIN ate LIN faça



para J = MIN ate COL faça  
        MAT3(I,J)  $\leftarrow$  MAT1(I,J) + MAT2(I,J)  
    fim-para

fim-para  
5.4 [Fim]

6 [Declaração do Procedimento IMPREMAT]  
  PROCEDIMENTO IMPREMAT(LIN,COL,MAT)

  6.1 [Início]

  6.2 [Declaração de Variáveis]  
    I, J : inteiro

  6.3 [Impressão]  
    para I = MIN ate LIN faça  
      para J = MIN ate COL faça  
        escreva(MAT(I,J))  
      fim-para  
    fim-para

  6.4 [Fim]

7 [Corpo do ALGORITMO PRINCIPAL]

  7.1 [Início]

  7.2 [Leitura do dimensionamento das matrizes]  
    escreva ("Entre com o número de Linhas que conterà a Matriz:")  
    LEDIM(LINHAS)  
    escreva ("Entre com o número de Colunas que conterà a Matriz:")  
    LEDIM(COLUNAS)

  7.3 [Leitura das matrizes]  
    escreva ("Entre com os elementos que comporão a 1.<sup>a</sup> Matriz:")  
    LEMAT(LINHAS,COLUNAS,MATA)  
    escreva ("Entre com os elementos que comporão a 2.<sup>a</sup> Matriz:")  
    LEMAT(LINHAS,COLUNAS,MATB)

  7.4 [Soma das matrizes]  
    SOMAT(LINHAS,COLUNAS,MATA,MATB,MATC)

  7.5 [Impressão das matrizes lidas]  
    escreva ("Os elementos lidos na 1.<sup>a</sup> Matriz foram:")  
    IMPREMAT(LINHAS,COLUNAS,MATA)  
    escreva ("Os elementos lidos na 2.<sup>a</sup> Matriz foram:")  
    IMPREMAT(LINHAS,COLUNAS,MATB)

  7.6 [Impressão da matriz resultante]  
    escreva ("A Matriz resultante da soma dos elementos lidos,  
    anteriormente nas duas matrizes, é:")  
    IMPREMAT(LINHAS,COLUNAS,MATC)

  7.7 [Fim]