Curso Técnico em Informática
ADVICE - 2009

Programação Estruturada
Programação em Ambiente Visual

Apostila de Conceitos Gerais

Prof. Oscar Santana

Definição de Lógica

A lógica trata da correção do pensamento. Como filosofia, procura saber por que pensamos assim e não do outro jeito. Com arte ou técnica, nos ensina a usar corretamente as leis do pensamento.
Poderíamos dizer também que a lógica é a arte de pensar corretamente e, visto que a forma mais complexa do pensamento é o raciocínio, a lógica estuda ou tem em vista a "correção do raciocínio". Podemos ainda dizer que a lógica tem em vista a "ordem da razão". Isto dá a entender que a nossa razão pode funcionar desordenadamente. Por isso a lógica ensina a colocar **Ordem no Pensamento**.

1.1. Algoritimizando a Lógica

Construir algoritmos é o objetivo fundamental de toda a programação, mas afinal o que é algoritmo?
"Algoritmo é uma seqüência de passos que visam atingir um objetivo bem definido."
"Algoritmo é a descrição de um conjunto de ações que obedecidas, resultam numa sucessão finita de passos, atingindo o objetivo."
Em geral, um algoritmo destina-se a resolver um problema: fixa um padrão de comportamento a ser seguido, uma norma de execução a ser trilhada, com vista a alcançar, como resultado final, a solução de um problema.

1.2. Exemplo de um algoritmo (não computacional)

Objetivo: 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;
5. se der o sinal de chamar
    5.1 conversar
    5.2 desligar
    5.3 retirar o cartão
6. senão
    6.1 repetir

Fim

Obs: um programa é um algoritmo escrito em linguagem computacional.

2. Tópicos Preliminares

2.1 Tipos Primitivos

Aproximando-nos da maneira pela qual o computador manipula as informações, vamos dividi-las em 4 tipos primitivos:

2.1.1 **Inteiro:** toda e qualquer informação numérica que pertença ao conjunto dos números inteiros relativos (negativa, nula ou positiva).

Ex:

- Ele tem *15* irmãos.
- A temperatura desta noite será de *-2* graus.
- Outros exemplos: idade, numero_dependentes, numero_de_filhos

2.1.2 **Real:** toda e qualquer informação numérica que pertença ao conjunto dos números reais (negativa, nula ou positiva).

Ex:

- Ela tem 1,73 metros de altura.
- Meu saldo bancário é de R$ 120,96
- Outros exemplos: altura, peso, comprimento

2.1.3 **Caractere:** toda e qualquer informação composta por um conjunto de caracteres alfanuméricos (0..9) e o/ou especiais.

Ex: #, $, %, &, *

- Outros exemplos: e-mail, data_nascimento, telefone, cidade

2.1.4 **Lógico:** toda e qualquer informação que pode assumir apenas duas situações.

Ex:

- verdadeiro ou falso
- ligado ou desligado

2.2 Constantes

Entendemos que uma informação é constante quando não sofrem nenhuma variação no decorrer do tempo.

Ex: π 3,1416
Salário mínimo

2.3 Variável

Uma informação é classificada como variável quando tem a probabilidade de ser alterada em algum instante no decorrer do tempo.
Ex: temperatura, peso

2.3.1 Formação de Identificadores – Regras básicas

- Devem começar por um caractere alfabético

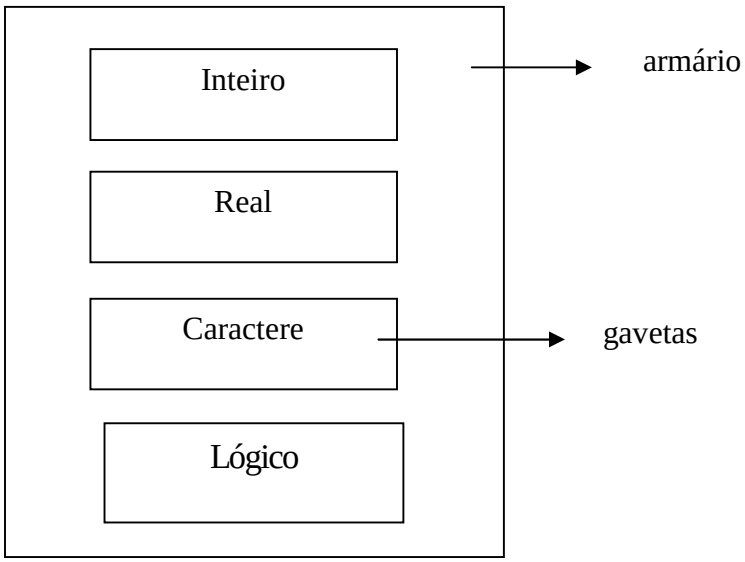
- Podem ser seguidos por mais caracteres alfabéticos e/ou numéricos
- Não é permitido o uso de caracteres especiais
- O pascal não é sensitive, não faz diferença entre maiúsculo e minúsculo

Exemplos

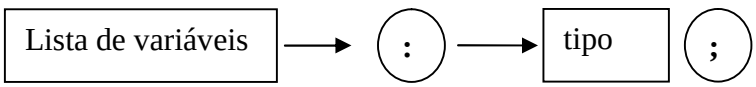
Válidos	Inválidos
Notas; x; k7; bj153, fgts	5x ; e(13) ; a :B ; x-y ; nota/2

2.3.1 Declaração de variáveis

No ambiente computacional as informações variáveis são gravadas em disp ositivos eletrônicos analogicamente chamados de memória.
Memória = armário
Variáveis = gavetas



Portanto precisamos definir nomes para determinadas gavetas especificando qual o "material dos objetos" que lá podem ser armazenados. Regra si ntática:



Exemplos:
X: inteiro;
Nome, endereço, data: caractere;
ABC, peso, dólar: real;

Operadores aritméticos
Conjunto de símbolos que representa as operações básicas da matemática.

+	Adição
*	multiplicação
**	Potenciação
-	Subtração
/	Divisão
//	Radiciação

Usaremos outras operações matemáticas não convencionais cujos nomes dos operadores são:
Mod – resto da divisão
Div – quociente da divisão inteira
Estes operadores só podem ser aplicados com números inteiros.
Ex: 9 mod 4 = 1
9 div 4 = 2

Exercícios
Utilizando os operadores especiais MOD e DIV resolva as expressões abaixo:

11 div 4	9 div 4
11 mod 4	10 mod 2,5
10 div 3	15 mod 6
9 mod 4	19 mod 6
2,5 mod 2	3,5999 div 2

Operadores relacionais
Conjunto de símbolos que representa as operações básicas da matemática.

>	Maior que
<	Menor que
>=	Maior ou igual
<=	Menor ou igual
=	Igual
< >	Diferente

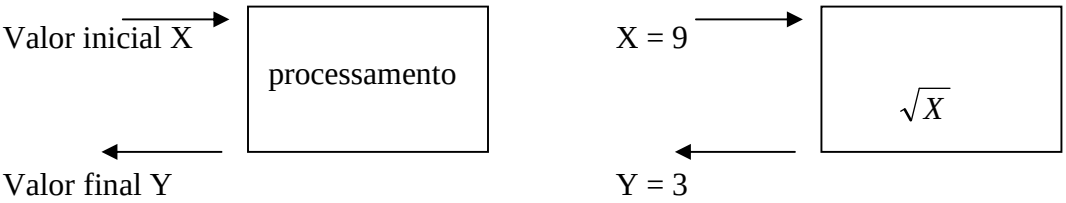
Linearização de Expressões
Para a construção de algoritmos todas as expressões aritméticas devem s er linearizadas, ou seja, colocadas em linhas. É importante também resolver o uso dos operadores correspondentes da aritmética tradicional para computadores (computacional).

[2 + (5 -3) + 1]	(2/3 + (5 -3) + 1)
3 **tradicional**	**computacional**

Modularização de Expressões

A modularização é a divisão da expressão em partes, proporcionando a resolução da mesma.
Como pode ser observado no exemplo usamos somente parênteses "( )" para a modularização. Na informática podemos ter parênteses dentro de parênteses.
Ex de prioridades:
(2+2)/2 = 2
2+2 / 2 = 3

Funções
Uma função é um instrumento 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 pré-definidas para linguagem ou criadas pelo programador de acordo com seu interesse.
Exemplos:



Bibliotecas de Funções
Armazenar um conjunto de funções que podem ser usados pelos programas.

Funções Pré-definidas

ABS ( )	Valor absoluto
SQRT ( )	Raiz quadrada
SQR ( )	Elevar ao quadrado
ROUND ( )	Valor arredondado
LOG ( )	Logaritmo
SIN ( )	Seno
COS ( )	Co-seno
TAN ( )	Tangente
TRUNC ( )	Valor truncado

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

OBS:
Função Round – arredonda o número fracionário. Se o valor decimal for de 0,5 ou maior o número é arredondado para cima, caso contrário para baixo.

Função Trunc – trunca um número fracionário, retornado somente uma parte inteira.

Operadores Lógicos
Atuam sobre expressões retornando sempre valores lógicos como fals o ou verdadeiro.
E (AND): retorna verdadeiro se ambas as partes forem verdadeiras
OU (OR): basta que uma parte seja verdadeira para retornar verdadeiro.
NÃO (NOT): inverte o estado, de verdadeiro p/ falso e vice-versa.

Tabela de Decisão ou Verdade – Operador Lógico E

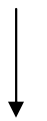
Condição 1	Condição 2	Resultado
Falsa	Falsa	Falso
Verdadeira	Falsa	Falso
Falsa	Verdadeira	Falso
Verdadeira	Verdadeira	Verdadeiro

Tabela de Decisão ou Verdade – Operador Lógico OU

Condição 1	Condição 2	Resultado
Falsa	Falsa	Falso
Verdadeira	Falsa	Verdadeiro
Falsa	Verdadeira	Verdadeiro
Verdadeira	Verdadeira	Verdadeiro

Tabela de Decisão ou Verdade – Operador Lógico NÃO

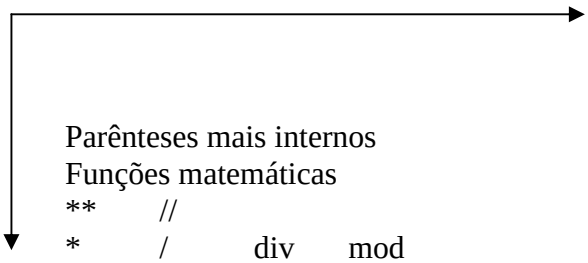
Condição	Resultado
Verdadeira	Falso
Falsa	Verdadeiro

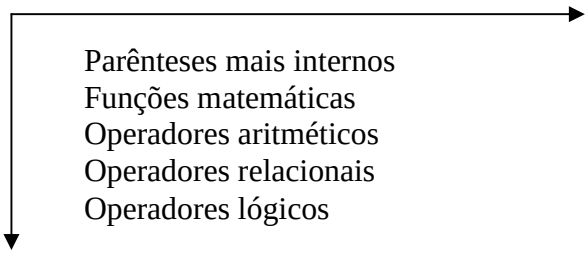


NÃO
E
OU

Prioridades

Na resolução das expressões aritmétic as, as seguintes operações e funções matemáticas guardam entre si uma hierarquia.





Exercício

Com base nas tabelas de decisão indicadas a seguir, determine o resultado lógico das expressões mencionadas, assinalando se não ver dadeiras ou falsas. Considere para as respostas os seguintes valores:

X =1 A=3 B=5 C=8 D=7

a) .não. (X > 3)
b) (X < 1) .e. .não. (B > D)
c) .não. (D < 0) .e. (C > 5)
d) .não. (X > 3) .ou. (C < 7)
e) (A > B) .ou. (C > B)
f) (C > = 2)
g) (X < 1) .e. (B > = D)
h) (D < 0) .ou. (C < 5)
i) .não. (D > 3) .ou..não. (B < 7)
j) (A > B) .ou..não. (C > B)

Comando de Atribuição

Permite fornecer um valor a uma certa variável (guardar um objeto numa certa gaveta) onde o tipo dessa informação deve ser compatível com o tipo da variável, isto é, somente podemos atribuir um valor lógico a uma variável capaz de comportá -lo, ou seja, uma variável declarada do tipo lógico.

Alguns exemplos:

A ← verdadeiro
X ← 8 + 13 div 5
B ← 5 = 3

Esses comandos atribuem às variáveis A, X e B os valores fornecidos à direita do símbolo de atribuição.
Na linguagem de programação que utilizaremos (Pascal) o símbolo " ←" é substituído por ":="

Exercícios

1. **Sendo P, Q e R variáveis inteiras e S variável real, cujo valores, num determinado momento da execução do programa, são respectivamente 2, 3, 12 e 4,5, quais os valores de cada uma das expressões aritméticas.**

a) 100* (Q DIV P) + R
b) P * (R MOD 5) – Q/2
c) TRUNC (5 – R) – ROUND (SQR(Q) – R/4 * P – 3)
d) SQRT (R – SQR (P)) + ROUND (S)
e) R MOD (P + 1) – Q * R
f) 1 + ROUND (P * P * P – 2 * R)/5 – TRUNC (S – 1)
g) 1 + (R + P) DIV (Q * Q) * TRUNC (2 * P * Q * S)
h) P + ROUND (2,9 + TRUNC (0,3 + S) * 2

2. **Preencha a tabela abaixo de acordo com as variáveis especificadas nas tabelas:**

VARIÁVEIS				RELAÇÕES			
X	Y	Z	COR	NOME	X*X+Y>Z	COR = AZUL	JOSE < > NOME
1	2	5	'AZUL'	PAULO			
4	3	1	'VERDE'	JOSÉ			
1	1	2	'BRANCO'	PEDRO			
1	2	1	'AZUL'	JOSÉ			

3. **Complete o quadro, a seguir com o valor das relações indicadas, tendo -se em vista os valores atribuídos às variáveis.**

VARIÁVEIS				**RELAÇÕES**		
A	B	NOME	COR	A + 1 >= SQRT (B)	NOME < > ANA	PROFISSÃO MÉDICO
3,0	16,0	Mirian	‘Advogado			
5,0	64,0	Pedro	‘Médico			
2,5	9,0	Ana	‘Professor’			

4. **P = 2, Q = 3, R = 12, S = 4,5**

a) 100 * (3 DIV 2) + 12
b) 2 * (R MOD 5) – Q /2
c) TRUNC (S – R) + ROUND (Q]SQR (Q) – R/4 * P – 3)
d) SQRT(R – SQR (P)) + ROUND S
e) 12 MOD (2 + 1) – 3 * 12
f) 1 + ROUND (2 * 2 * 2 – 2 * 12) / 5 TRUNC (4,5 – 1)
g) 1 + (12 + 2) DIV (3 * 3) + TRUNC (2*2*3 – 4,5)
h) 2 + ROUND (2,9 + TRUNC (0,3 + 4,5) * 2)

Comandos de Entrada e Saída

a) LER: comando de entrada que permite a leitura de variáveis de entrada.
b) ESCREVER: comando de saída que exibe uma informação na tela do monitor.
c) IMPRIMIR: comando de saída que envia uma informação para a impressora.

Estruturas Chaves na Construção de Algoritmos

Existem 3 estruturas básicas de controle nas quais se baseiam os algoritmos: seqüenciação, decisão e repetição.

Seqüenciação

Os comandos de algoritmos fazem parte de uma seqüência onde é relevante a ordem na qual se encontram os mesmos, pois serão executados um de cada vez, estritamente de acordo com essa ordem:

Comando 1
Comando 2
Comando 3
.
.
.
Comando n

Blocos

Um bloco pode ser definido como um conjunto de ações com uma freq üência definida. Serve para definir limites nos quais as variáveis declaradas em seu "interior" são conhecidas.

Ex:

```
Início          {início do algoritmo}
.
.               Seqüência de ações
.
Fim.            {fim do algoritmo}
```

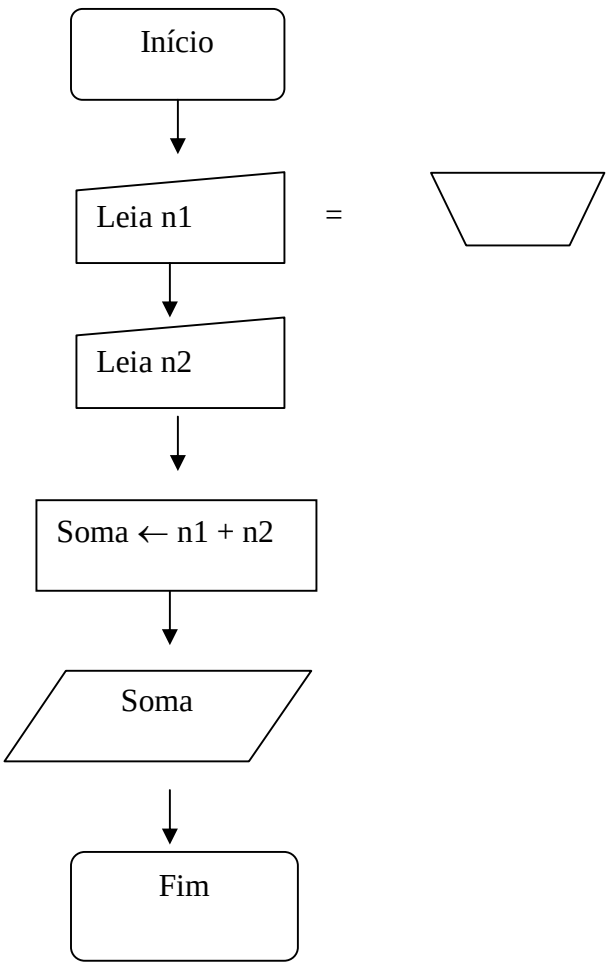
Obs: Inicio e fim são delimitadores obrigatórios.

1. Algoritmo que calcula a soma de dois números inteiros.

```
Programa somar;
Variáveis
        n1, n2, soma: inteiro;
Início
```

Leia n1;
Leia n2;
Soma ← n1+n2;
Escreva soma;
Fim.

1.1 Diagrama de Blocos (Algoritmo acima)



2. Algoritmo que calcula a média de 3 notas.

3. Algoritmo que implementa o recibo de contas de luz considerando os seguintes dados:

- Leitura do mês anteiror (em Kwh)
- Leitura do mês atual (em Kwh)
- Valor do Kwh

- Taxa de ICMS de 17%
- Consumo de energia
- Valor Total a ser pago

4. **Faça um algoritmo que calcule a área de uma circunferência. A fórmula que executa este cálculo é $A = \pi R^2$, sendo π uma constante na fórmula (3,14159) e R o valor do raio.**

```
Programa area_circulo;
Variáveis
        A, R: real;
Início
        Leia R;
        A ← 3,14159 * (R*R);
        Escreva A;
Fim.
```

Variações:

```
A ← 3,14159 * SQR (R);

A ← 3,14159 * R ↑ 2;

A ← 3,14159 * (R**2);
```

5. **Construir um algoritmo que efetue o cálculo do salário líquido de um professor. Para fazer este algoritmo você deverá possuir alguns dados, tais como, valor da hora aula, número de horas trabalhadas no mês e percentual de desconto do INSS. Em primeiro lugar, estabelecer qual será o salário bruto para efetuar o desconto e ter o valor do salário líquido.**
   HT – horas trabalhadas
   VH – valor hora aula
   PD – percentual de desconto
   SB – salário bruto
   TD – total de desconto
   SL – salário líquido

6. **Fazer um algoritmo que efetue o cálculo da quantidade de litros de combustível gastos em uma viagem, utilizando um automóvel que faz 12 km por litro. Para obter o cálculo, o usuário deve fornecer o tempo gasto (tempo) e a velocidade média (velocidade) durante a viagem. Desta fórmula será possível obter a distância percorrida com a fórmula (distância ← tempo * velocidade). Possuindo o valor da distância, basta calcular a quantidade de litros de combustível utilizada na viagem utilizando a fórmula (litros_usados ← distância/12). Ao final, o algoritmo deve apresentar os valores da velocidade média (velocidade), tempo gasto na viagem (tempo), a distância percorrida (distância) e a quantidade de litros (litros usados) utilizados na viagem.**

7. **Ler 2 valores (podem ser reais, inteiros ou caracteres) para as variáveis A e B e efetuar a troca dos valores de forma que a variável A passe a possuir o valor da variável B e a variável B passe a possuir o valor da variável A. Apresentar os valores trocados.**

8. **Elaborar um algoritmo que calcule e apresente o volume de uma caixa retangular por meio da fórmula (volume ← comprimento * largura * altura)**

9. **Efetuar a leitura de um número inteiro e apresentar o resultado do quadrado desse número.**

10. **Ler uma temperatura em graus Celsius e apresentá-la convertida em Fahrenheit a fórmula de conversão é F ← (9*C=160)/5, sendo F a temperatura em Fahrenheit e C a temperatura em Celsius.**

LABORATÓRIO – Palavras Utilizadas em Pascal

Até	Until
Booleano	Boolean
Caractere (1)	Char
Caractere (acima 1)	String
Caso	Case
Decrescente	Downto
E	And
Enquanto	While
Então	Then
Escreva	Write
Faça	Do
Fim	End
Início	Begin
Inteiro	Integer
Leia	Read
Ou	Or
Para	For
Para	To
Programa	Program
Real	Real
Repita	Repeat
Se	If
Senão	Else
Vá para	Goto
Variáveis	Var
Vetor	Array

TURBO	ZIM
Program somar; Uses CRT; Var Soma, n1, n2: integer; Begin CLRSCR; Write (‘Entre com o 1º número: ‘); Readln (n1); Write (‘Entre com o 2º número: ‘); Readln (n2); Soma := n1 + n2; Write (‘A soma corresponde a: ‘, soma); Readkey; End.	Program somar; - - - Var Soma, n1, n2: integer; Begin - - - Write (‘Entre com o 1º número: ‘); Readln (n1); Write (‘Entre com o 2º número: ‘); Readln (n2); Soma := n1 + n2; Write (‘A soma corresponde a: ‘, soma); - - - End.

Estruturas de Controle – Desvio Condicional Simples

Imagine a seguinte situação: um programa que apresente a média escolar de um aluno. Até aqui, simples, mas além de calcular a média, o pr ograma deve apresentar se ele está aprovado ou reprovado, segundo a análise de sua média.
Observe que aqui será necessário verificar a média do aluno para então tornar uma decisão no sentido de apresentar sua real situação: aprovado ou reprovado.
Português Estruturado

Se <condição > então
<instruções para a condição verdadeira>
fim_se
<instruções para a condição falsa ou após ser verdadeira>

Exemplo:

Programa somar;
Variáveis
X, A, B: inteiro;
Início
Leia A;
Leia B;
X ← A + B
Se (X > 10) então
Escreva X;

```
        Fim_se;
Fim.
```

```
Se  <condição > então
    <instruções para a condição verdadeira>
Senão
    <instruções para a condição falsa >
fim_se
Fim.
```

Exemplo:

```
Programa somar;
Variáveis
        X, A, B, R: inteiro;
Início
        Leia A;
        Leia B;
        X ← A + B
        Se (X > = 10) então
                R ← X + 5
        Senão
                R ← X - 7
        Fim_se;
                Escreva R;
Fim.
```

Desvio Condicional Encadeados

```
Se  <condição > então
    <instruções para a condição1 verdadeira>
Senão
    Se <condição 2> então
            <instruções para a condição2 verdadeira porém condição 1 falsa >
    Senão
            <instruções para a condição1 e condição2 falsa>

     fim_se
fim_se
Fim.
```

Estruturas de Controle – Tomada de Decisão

Desvio Condicional Simples

A instrução **se...então...fim_se** tem por finalidade tomar uma decisão. Sendo a condição verdadeira, serão executadas todas as instruções que estejam entre a instrução **se...então** e a instrução **fim_se**. Sendo a condição falsa, serão executadas as instruções que estejam após o comando **fim-se**.

se (<condição>) **então**
    <instruções para condição verdadeira>
fim_se
    <instruções para condição falsa ou após ser verdadeira>

1. Ler dois valores numéricos, efetuar a adição e apresentar o seu resultado caso o valor somado seja maior que 10.

```
programa soma_numeros;
Var
X, A, B: inteiro;
Início
Leia A;
Leia B;
X ←A + B;
se (X > 10) então
        escreva X;
fim_se;
fim.
```

Desvio Condicional Composto

Sendo a condição verdadeira, serão executadas todas as instruções que estejam posicionadas entre o **se...então** e a instrução **senão.** Sendo a condição falsa, serão executadas as instruções que estejam entre o **senão** e a instrução **fim_se.**

se (<condição>) **então**
    <instruções para condição verdadeira>
senão
    <instruções para condição falsa >
fim_se

2. Ler dois valores numéricos e efetuar a adição. Caso o valor somado seja maior ou igual a 10, deverá ser apresentado somando a ele mais 5; caso o valor somado não seja maior ou igual a 10, este deverá ser apresentado subtraindo 7.

```
programa soma_numeros;
Var
```

```
A, B, X, R: inteiro;
Início
Leia A;
Leia B;
X ←A + B;
se (X > = 10) então
        R ←X + 5
senão
        R ←X - 7;
fim_se
escreva R;
fim.
```

Desvio Condicional Encadeado

```
se (<condição1>) então
   <instruções para condição1 verdadeira>
senão
        se (<condição2>) então
                <instruções para condição2 verdadeira, porém 1 falsa>
        senão
                <instruções para condição1 e condição 2 falsa>
        fim_se
fim_se
```

Algoritmo que efetua o cálculo do reajuste de 15% caso seu salário seja menor que 500. Se o salário for maior ou igual a R$ 500,00, mas menor ou igual a R$ 1000,00, seu reajuste será de 10%, caso seja ainda maior que R$ 1 000,00, o reajuste deverá ser de 5%.

```
programa reajusta_salario;
Var
novo_salario, salario: real;
Início
Leia salario;
se (salario < 500) então
        novo_salario ← salário *1.15
senão
        se (salário < = 1000) então
                novo_salario ← salário *1.10
        senão
                novo_salario ← salário *1.05
        fim_se
fim_se
escreva novo_salário;
fim.
```

Dado 3 valores A, B, C verificar:
- Se os comprimentos não são zero
- Triângulo: (A<B+C) e (B<A+C) e (C<A+B)
- Se compõe um triângulo eqüilátero, isósceles ou escaleno :
    - Eqüilátero: (A = B) e (B = C)
    - Isósceles: (A = B) ou (A = C) ou (B = C)
    - Escaleno: (A < > B) ou (B < > C)

Exercícios - Estrutura de Decisão

1. Fazer um programa para ler um número e mostrar se é igual a zero, positivo ou negativo.

2. Fazer um programa para ler o nome, 2 notas, mostrar a média com a mensagem:
   Inferior a 5,0 – "Reprovado"
   De 5,1 a 6,9 – "Recuperação"
   De 7,0 a 10 – "Aprovado"
   Obs: mostrar o nome.

3. Fazer um programa que calcule quanto você gastou em reais e quantos litors você consumiu de combustível, sendo que têm 3 tipos de carro: gol 12 km/l, vecta 8 km/l e Palio 10 km/l. Obs: o preço por litro é R$ 2,65.

4. Faça um programa que calcule os juros de um determinado produto. Se o produto for pago em 30 dias (10%), em 60 (20%) e em 90 (30%). O progr ama deve fazer a leitura do nome do produto, do valor e da condição do pagamento e depois faça o cálculo.

5. Faça um programa que calcule o valor de uma ligação telefônica (São Carlos = 0,30, São Paulo = 0,70, RJ = 1,20). Os dados de entrada serão: tempo da ligação e cidade.

6. Faça um programa que calcule por meio da idade sua categoria na natação:

Idade	Categoria
> = 5 e < = 7	Infantil
> = 8 e < = 10	Infantil B
> = 11 e < = 13	Juvenil A
> = 14 e < = 17	Juvenil B
> = 18	Sênior

7. Fazer um programa que mostre o maior e o menor número > 10.

Estrutura de Controle – Laços ou Malhas de Repetição

Existem ocasiões em que é necessário efetuar a repetição de um determinado número de vezes. Neste caso, poderá ser criado um looping que efetue o processamento de um determinado trecho, tantas vezes forem necessárias. Os loopings também são chamados de laços de repetição ou malhas de repetição.
A principal vantagem deste recurso é que o programa passa a ter um tamanho menor, podendo sua amplitude de processamento ser aumentada sem alterar o tamanho do código de programação.

a) Repetição do tipo teste lógico no início do looping

A estrutura **enquanto...faça...fim_enquanto** tem seu funcionamento controlado por decisão. Sendo assim, poderá executar um determinado conjunto de instruções **enquanto** a condição verificada for verdadeira. No momento que esta condição se torna falsa, o processamento da rotina é desviado para fora do looping. Se a condição for falsa logo de início, as instruções contidas no looping são ignoradas.

Exemplo:

```
Programa looping_1A;
Variáveis
        X, R, cont: inteiro;
Início
        Cont ← 1;
        Enquanto (cont < = 5) faça
                Leia x;
                R ← X * 3;
                Escreva R;
                Cont ← cont +1;
        Fim_enquanto;
Fim.
```

Cont = controla o número de vezes que o programa deverá ser executado.

Para ilustrar de forma um pouco diferente imagine que o problema anterior deverá ser executado enquanto o usuário queira. Desta forma em vez de possuir dentro da rotina um contador de vezes, pode-se possuir uma instrução pedindo que o usuário informe se deseja continuar ou não.

Exemplo:

```
Programa looping_1B;
Variáveis
        X, R: inteiro;
        Resp: caractere;
Início
        Resp ← “sim”;
```

```
        Enquanto (Resp = “sim”) faça
                Leia x;
                R ← X * 3;
                Escreva  (‘Deseja continuar? ‘);
                Leia Resp;
        Fim_enquanto;
Fim.
```

b) Repetição do tipo teste lógico no fim do looping (parecida com o enquanto)
Sua estrutura repita...até que. Executa um conjunto de instruções pelo menos uma vez antes de verificar a validade da condição estabelecida.
Referente da estrutura enquanto que executa somente um co njunto de instruções, enquanto a condição é verdadeira.
Exemplo:

```
Programa looping_2A;
Variáveis
        X, R, cont: inteiro;
Início
        Cont ← 1;
        Repita
                Leia x;
                R ← X * 3;
                Escreva R;
                Cont ← cont +1;
        Até_que (cont > 5);
Fim.
```

Exemplo:

```
Programa looping_2B;
Variáveis
        X, R: inteiro;
        Resp: caractere;
Início
        Resp ← “sim”;
        Repita X;
        R ← X * 3;
        Escreva R;
        Escreva  (‘Deseja continuar?‘);
        Leia Resp;
        Até_que Resp < > “Sim”;
Fim.
```

Fazer um programa para um estacionamento...

Estrutura de Repetição – Para...de...até...passo...faça...fim_para

Existe uma possibilidade de facilitar o uso dos contadores finitos sem fazer uso das duas estruturas de repetição vistas anteriormente (repita e enquanto), deixando -as para utilização de loopings em que não se conhece de antemão o número de vezes que uma determinada seqüência de instruções deverá ser executada. Os loopings que possuem um número finito de execuções poderão ser processados por meio de estrutura de laços contados do tipo **para**, sendo conseguida com a utilização do conjunto de instruções **para...de...até...passo...faça...fim_para.**
A estrutura **para...de...até...passo...faça...fim_para** tem seu funcionamento controlado por uma variável denominada contador. Sendo assim, poderá executar um determinado conjunto de instruções um determinado número de vezes.

Sintaxe:

para <variável> **de** <inicio> **até** <fim> **passo** <incremento> **faça**
    <instruções>
fim_para

Exemplo - Pedir a leitura de um valor para a variável X, multiplicar esse valor por 3, implicando-o à variável de resposta R, e apresentar o valor obtido, repetindo esta seqüência por cinco vezes.

Português Estruturado

```
programa looping_3;
var
        x, r: inteiro;
        cont: inteiro;
Início
        para cont de 1 até 5  passo 1 faça
                leia x;
                r ← x * 3;
                escreva r;
        fim_para;
Fim.
```

Será executado o conjunto de instruções entre a instrução **para** e a instrução **fim_para** sendo a variável cont (variável de controle) inicializada com valor 1 e incrementada de mais 1 por meio da instrução passo até o valor 5. Este tipo de estrutura de repetição poderá ser utilizado todas as vezes que houver a necessidade de repetir trechos finitos, em que se conhecem os valores inicial e final.

Automóvel R$ 1,00 por hora
Caminhonete R$ 1,50 por hora
O programa para quando atingir um total de 100 ca rros e então informará quanto tem em caixa.

Fazer um programa que emite o resultado dos alunos de uma determinada disciplina, considerando que a média é calculada a partir de 3 notas. O programa deverá exibir o nome do aluno e ao lado o resultado ‘aprovad o’ ou ‘reprovado’. A média de aprovação é 6,0 e freqüência mínima é de 105 aulas. A turma tem 55 alunos.

Programa que lê as idades de 10 pessoas e exibe a maior, a menor idade e a média das idades.

Comandos:
Textbackground (cor)
Textcolor (cor)
Blink

Cores Pascal

Blue+blink
Green
Cyan
Red
Magenta
Brown
Lightgray
Darkgray
Lightblue
Lightcyan
Lightred
Lightmagenta
Yellow
Write
black

Fazer um programa para ler um caractere e mostrar uma mensagem indicando se é vogal maiúsculo, vogal minúsculo, sinal aritmético ou outros.

Exercícios

1. Fazer um programa para ler o nome e a letra inicial do estado civil de uma pessoa e mostrar a descrição de acordo com a tabela abaixo:

C	Casado (a)
Q	Desquitado (a)
S	Solteiro (a)
D	Divorciado (a)
V	Viúvo (a)
QQ Letra	Inválido

2. Fazer um programa para um número inteiro de 1 a 12 em vermelho e informar o mês correspondente em verde, sendo janeiro o mês de número 1. Se o número não corresponder a um mês válido, é mostrada uma mensagem de erro em azul.

3. Uma universidade atribui menções aos alunos conforme a faixa de notas que tinha atingido 90 a 100:

90 - 100	SS – Superior
70 - 89	MS – Médio Superior
50 - 69	MM – Médio
30 - 49	MI – Médio Inferior
01 - 29	II – Inferior
0	SR – Sem Rendimento

Mostrar as notas e informar a menção.

4. Fazer um programa para mostrar os números inteiros de 1 a 100 em ordem decrescente distribuídos em 10 alunos.
5. Fazer um programa para mostrar os números de 1 a 10 inclusive; 1 em cada linha.
6. Programa que recebe dois números e a operação a ser feita (+,-,*,/) e desenvolve o resultado da operação.
7. Fazer um programa que tenha como entrada o sexo M ou F e faça a contagem de quantos são F e M.