

APOSTILA

DE

PASCAL I

Profa. Gilene E. S. Borges Gomes
gilene@gomeshp.com
http://www.gomeshp.com

Índice

1. INTRODUÇÃO AO PASCAL	3
1.A. EXERCÍCIOS (INTRODUÇÃO)	5
2. COMANDOS DE ATRIBUIÇÃO / ENTRADA E SAÍDA (READ / WRITE)	6
2.A. EXERCÍCIOS (READ / WRITE)	8
3. COMANDO DE ALTERNATIVA SIMPLES E COMPOSTA (IF .. THEN .. ELSE)	10
3.A. EXERCÍCIOS (IF .. THEN .. ELSE)	12
4. COMANDO DE MÚLTIPLA ESCOLHA (CASE)	14
4.A. EXERCÍCIOS (CASE)	16
5. REPETIÇÃO C/ TESTE NO INÍCIO (WHILE)	17
5.A. EXERCÍCIOS (WHILE)	20
6. REPETIÇÃO C/ TESTE NO FINAL (REPEAT .. UNTIL)	24
6.A. EXERCÍCIOS (REPEAT .. UNTIL)	27
7. REPETIÇÃO C/ VARIÁVEL DE CONTROLE (FOR .. TO / DOWNTO .. DO)	30
7.A. EXERCÍCIOS (FOR .. TO / DOWNTO .. DO)	32
8. BIBLIOGRAFIA	36
9. ANEXO A - UNIDADES PREDEFNIDAS	37
9.A. EXERCÍCIOS (ANEXO A)	41
10. ANEXO B - MENUS E ERROS	42
10.A. EXERCÍCIOS (ANEXO B)	49
11. ANEXO C - RESUMO DAS TECLAS UTILIZADAS NO TP 7.0	52
11.A. EXERCÍCIOS (ANEXO C)	53

1. Introdução ao PASCAL

PASCAL é uma linguagem de programação desenvolvida por volta de 1968 a 1970 pelo professor Niklaus Wirth.

O TURBO PASCAL é um ambiente integrado de desenvolvimento. A sua primeira versão data do início dos anos 80, nesta época implementada para os equipamentos de 8 bits. Desde então tem sido aprimorado continuamente, tanto para facilitar o seu uso pelo usuário quanto para incorporar novas técnicas e recursos na linguagem de programação.

O TURBO PASCAL não se trata apenas de um compilador, mas de um *Ambiente Integrado*, pois inclui um *Editor de Programas*, um *Compilador*, um *Link-editor*, o *Executor de Programas* e um *Depurador*, descritos a seguir:

- ✓ *Editor de Programas* – um software cuja finalidade é facilitar a criação, edição e gravação do código-fonte¹ dos programas.
- ✓ *Compilador* – um programa que lê as instruções escritas numa linguagem de programação legível pelas pessoas (Ex.: Pascal), e as traduz num programa executável que o computador consegue entender e processar diretamente. O Compilador analisa o texto do programa e fornece a descrição e localização dos erros de sintaxe porventura existentes.
- ✓ *Link-editor* – combina programas separados e padroniza as chamadas ou referências dentro delas.
- ✓ *Executor de programas* – processa as instruções de um programa.
- ✓ *Depurador* – utilitário normalmente incluído em compiladores ou interpretadores² com a finalidade de ajudar os programadores a identificar e corrigir erros de lógica que possam existir no código-fonte.

ESTRUTURA DE UM PROGRAMA PASCAL – um programa escrito em PASCAL deve sempre ser iniciado com a palavra reservada **Program**, seguida por um cabeçalho, um ponto e vírgula (;), um bloco³, e terminado com um ponto final (.):

```
Program cabeçalho ;  
declarações;  
begin           { início de um bloco }  
                seqüência de comandos;  
end.           { fim do bloco }
```

¹ Nas linguagens de programação de alto nível, o programa na forma como as pessoas o escrevem e lêem, antes que seja compilado ou interpretado.

² Tradutor de uma linguagem de programação de alto nível, que converte as suas instruções para a linguagem de máquina mas não cria uma versão executável dos programas; os interpretadores traduzem e executam os programas ao mesmo tempo.

³ Um bloco de comandos é formado por declarações, **begin**, seqüência de comandos e **end**.

Comentário – é usado para aumentar a clareza de um programa, embora não seja analisado pelo computador. Um comentário deve ser escrito entre chaves:

```
{ comentário }
```

Declarações – todas as constantes⁴, tipos e variáveis⁵ e declarados num bloco, devem estar incluídas em uma declaração da forma:

```

const      identificador : tipo = valor;      { constantes }
    . . .

type       identificador = tipo;              { tipos }
    . . .

var        lista-de-identificadores : tipo;    { variáveis }
    . . .
    
```

identificador – é formado por uma única letra ou então por uma letra seguida de letras ou dígitos, em qualquer número. Ou seja, são os nomes atribuídos às variáveis.

lista-de-identificadores - são os identificadores das variáveis, separados por vírgula (,).

tipo – é o tipo das variáveis, ou seja, é o conjunto ordenado de valores que podem ser atribuídos a estas variáveis. São eles: **integer** (inteiro), **real** (real), **boolean** (lógico), **char** (um único carácter), **string** (seqüência de caracteres).

Exemplos:

```

const zero : integer = 0;
      pi = 3.1416;

type  estacao = (Verao, Outono, Inverno, Primavera);
      diautil = (Seg, Ter, Qua, Qui, Sex);
      nota = 0..10;

var   x, y : real;
      cor : string;          { Defina uma string com 255 posições }
      mes : string [50];      { Defina uma string com 50 posições }
      dia : diautil;
      est : estacao;
    
```

UNIT – Uma unidade é uma coleção de declarações de constantes, tipos, variáveis, funções e procedimentos, compilada separadamente. Algumas destas declarações são de uso interno da unidade, enquanto outras são visíveis, isto é, podem ser usadas por programas ou outras unidades.

USES – A cláusula uses permite que uma unidade faça uso de outras unidades. Para usar uma ou mais unidades, um programa deve conter a cláusula *uses* antes das declarações de variáveis.

⁴ **Constante** é um tipo especial de variável, a qual não terá seu conteúdo modificado durante toda a execução de um programa, ou seja, seu valor é fixo.

⁵ Podemos imaginar uma **variável** como o nome de um local onde se pode colocar qualquer valor do conjunto de valores possíveis do tipo básico associado.

UNIDADES PREDEFINIDAS – O TURBO PASCAL possui diversas unidades predefinidas: System, Printer, Crt, Dos, Overlay e Graph são as principais. Os identificadores usados nestas unidades geralmente são formados por palavras compostas em inglês que lembram seu significado. Mais detalhes podem ser encontrados no anexo A.

1.A. Exercícios (Introdução)

1. O que é Turbo Pascal? O que ele possui?

2. Qual é a estrutura completa de um programa Pascal?

3. O que é uma unidade? Cite e resuma as principais unidades predefinidas pelo Turbo Pascal.

4. Qual é a diferença entre uma variável e uma constante?

5. Considerando A, B, C variáveis reais, contendo os valores 1.0, 4.5, 8.0, respectivamente; Nome e Cor, as variáveis literais, contendo as seqüências de caracteres 'Tania' e 'branco', respectivamente; e Teste, variável lógica, contendo o valor True determinar o valor das expressões lógicas:
 - a) (A = 1.0) and Teste

 - b) (Nome = 'Pedro') or (Cor <> 'branco')

 - c) not Teste and (B * 2.0 – C = 0.0)

 - d) (C < 10) or Teste and (Cor = 'preto')

 - e) Teste and not Teste

2. Comandos de atribuição / entrada e saída (read / write)

No comando de atribuição, o identificador (variável) e a expressão devem ser do mesmo tipo, exceto no caso em que a variável é real, quando a expressão pode ser inteira, sendo, então, o valor inteiro da expressão transformado em real. O comando de atribuição {:=} permite que o valor em *expressão* seja atribuído ao *identificador*.

A instrução de **atribuição** (:=) é assim definida:

identificador := expressão;

Para solicitar uma entrada de dados ao usuário, deve-se utilizar o comando **read** ou **readln**. Normalmente, o periférico utilizado é o teclado.

A instrução de **entrada** é assim definida:

read (lista-de-identificadores);
readln (lista-de-identificadores);

A diferença entre esses dois comandos é que o **readln**, após a leitura dos valores, correspondentes à lista-de-identificadores, provoca uma mudança de linha na unidade de entrada.

Para mostrar uma saída de dados ao usuário, deve-se utilizar o comando **write** ou **writeln**. Normalmente, o periférico utilizado é o vídeo.

A instrução de **saída** é assim definida:

write (lista-de-identificadores e/ou constantes e/ou expressões);
writeln (lista-de-identificadores e/ou constantes e/ou expressões);

A diferença entre esses dois comandos é que o **writeln**, após a escrita dos valores, correspondentes à lista-de-identificadores, provoca uma mudança de linha na unidade de saída.

O comando **writeln** sem parâmetros faz com que o cursor salte uma linha.

Em particular, a instrução read deverá ser mencionada com as letras ln, pois isto evita alguns pequenos problemas de entrada de dados.

Exemplo: Fazer um programa para ler três notas e mostrar a média aritmética. Coloque o fundo de tela em azul e as letras em vermelho.

```
Program Lista02a;
uses crt;
var  n1, n2, n3, md : real;
begin
  textbackground (Blue);
  clrscr;
  textcolor (Red);
  write ('Digite a 1ª nota: ');
  readln (n1);
  write ('Digite a 2ª nota: ');
  readln (n2);
  write ('Digite a 3ª nota: ');
  readln (n3);
  md := (n1 + n2 + n3) / 3;
  writeln ('Media: ', md:2:2);
  readkey;
end.
```

Mostrando variáveis do tipo real

Utilize a seguinte formatação:

```
writeln (variavel:qe:qc);
```

variavel - deve ser do tipo real;

qe - indica a quantidade de espaços a serem utilizados para mostrar o valor, inclusive vírgula;

qc - indica a quantidade de casas decimais depois da vírgula.

Exemplo:

```
vr := 36.5;   write (vr:5:2);   saída 36.50
vr := 1.2;   write (vr:7:3);   saída 1.200
```

Existirão dois espaços em branco antes do valor 1,2.

2.A. Exercícios (read / write)

1. Considerando as declarações e as atribuições abaixo:

<pre> var Num : integer; Soma, X : real; Simbolo : char; Nome, Cor, Dia : string; Cod, Teste, Tudo : boolean; </pre>	<pre> </pre>	<pre> begin Num := 5; X := 2; Cod := True; Soma := 5; Nome := 'SQL'; end. </pre>
---	------------------	--

Assinalar os comandos de atribuição não permitidos:

- | | |
|---|--|
| <input type="checkbox"/> Simbolo := 5; | <input type="checkbox"/> X := X + 1; |
| <input type="checkbox"/> Soma := Num + 2 * X; | <input type="checkbox"/> Num := '*ABC*'; |
| <input type="checkbox"/> Teste := Cod or (Sqr (X) <> Soma); | <input type="checkbox"/> Dia := 'Segunda'; |
| <input type="checkbox"/> Tudo := Soma; | <input type="checkbox"/> Soma + 2 := Sqr (X) – Sqrt (Num); |
| <input type="checkbox"/> Cor := 'preto' - Sqr (X); | <input type="checkbox"/> X := Nome >= Cod; |

2. Com as mesmas declarações do exercício 1, quais serão os valores armazenados em Soma, Nome e Tudo, supondo-se que Num, X, Dia, Teste e Cod valham, respectivamente, 5, 2.5, 'terca', false e true?

- Nome := Dia;
- Soma := Sqr (Num) / X + Round (X + 1);
- Tudo := not Teste or Cod and (Soma < X);

3. Fazer um programa para ler um valor real e mostrar:

- o valor absoluto
- a parte fracionária
- a parte inteira
- o valor arredondado

4. Fazer um programa para ler uma frase e mostrar:

- o tamanho da frase
- a 1ª posição que ocorre a letra 'A' na frase
- os dez primeiros caracteres da frase
- os cinco últimos caracteres da frase

5. Escrever um programa para ler os valores dos catetos (**b** e **c**) de um triângulo e mostrar qual é o valor da hipotenusa (**a**) desse triângulo. Regra: $a^2 = b^2 + c^2$, isole a variável A.

6. Fazer um programa para ler dois números quaisquer e mostrar a soma, a diferença, o produto e o quociente entre os dois números lidos. O fundo da tela deve estar em azul, a entrada de dados de Amarelo e a saída de dados em vermelho claro. Utilize sempre a coluna 10.

7. Fazer um programa para ler quatro valores, calcular e mostrar a média ponderada para pesos 1, 2, 3 e 4, respectivamente. Regra: multiplique cada nota pelo seu peso, some os produtos e divida o resultado pela soma dos pesos.

8. Fazer um programa para ler um número real positivo e mostrar seu quadrado e sua raiz quadrada.

9. Fazer um programa para ler o valor do lado de um quadrado e mostrar sua área ($area = lado^2$) e seu perímetro ($perimetro = 4 \cdot lado$).

10. Fazer um programa para ler os valores da base e altura de um retângulo e mostrar seu perímetro ($perimetro = 2 (base + altura)$) e sua área ($area = base \cdot altura$).

11. Fazer um programa para ler o valor do lado de um cubo e mostrar sua área ($area = 6 \cdot lado^2$) e seu volume ($volume = lado^3$).

12. Fazer um programa para ler duas notas e seus respectivos pesos. Calcular e mostrar a média ponderada.

13. **DESAFIO:** Fazer um programa para receber como entrada um número inteiro de quatro algarismos e fornecer como saída os algarismos do número dado, separando os algarismos das unidades, dezenas, centenas e milhar.
Sugestão 1: para milhar obtenha a divisão inteira do número por 1000; para centena obtenha a divisão inteira do resto da divisão inteira anterior por 100; e assim por diante.
Sugestão 2: transforme o número em *string* e obtenha cada número desejado.

Ex.: Dado o número 5834, deseja-se a saída

Número dado	: 5834
unidade	: 4
dezena	: 3
centena	: 8
milhar	: 5

3. Comando de alternativa simples e composta (if .. then .. else)

A instrução **if** tem por finalidade tomar uma decisão e efetuar um desvio no processamento, dependendo, é claro, da condição atribuída ser Verdadeira ou Falsa.

Sendo a condição Verdadeira, será executada a instrução que estiver escrita após a instrução **then**. Caso seja necessário executar mais de uma instrução para uma condição verdadeira, estas deverão estar mencionadas dentro de um bloco (definido com o uso das instruções **begin** e **end**).

Sendo a condição Falsa, será executada a instrução que estiver escrita após a instrução **else**. Se não existir a instrução **else**, será executada a instrução definida após a instrução **then**. Caso seja necessário executar mais de uma instrução para uma condição falsa, estas deverão também estar mencionadas dentro de um bloco.

Alternativa simples

```
if condição
  then C1;           { único comando }
```

```
if condição
  then begin         { mais de um comando é necessário }
    C1;
    C2;
  end;               { begin e end }
```

Alternativa composta

```
if condição
  then C1
  else Ca;
```

```
if condição
  then begin
    C1;
    C2;
  end
  else begin
    Ca;
    Cb;
  end;
```

Exemplo: Fazer um programa para ler um número e mostrar se é igual a zero, positivo ou negativo. O fundo de tela deve estar em azul e as letras em amarelo.

```
Program Lista03a;  
uses crt;  
var num : real;  
begin  
  textbackground (Blue);  
  clrscr;  
  textcolor (Yellow);  
  write ('Digite um numero: ');   readln (num);  
  if num = 0  
  then writeln ('Iguar a zero.')  
  else if num > 0  
    then writeln ('Numero positivo.')  
    else writeln ('Numero negativo.');
```

readkey;

end.

Exemplo: Fazer um programa para ler duas notas, calcular e mostrar a média e mostrar também se o aluno foi: reprovado (média inferior a 5.0); aprovado (média igual ou superior a 7.0); indicado para 3ª avaliação (média entre 5.0 e 6.9). O fundo de tela deve estar em cinza e as letras em azul.

```
Program Lista03b;  
uses crt;  
var not1, not2, med : real;  
begin  
  textbackground (LightGray);  
  clrscr;  
  textcolor (Blue);  
  writeln; writeln;  
  write ('Digite 1a. nota: '); readln (not1);  
  write ('Digite 2a. nota: '); readln (not2);  
  writeln; writeln;  
  med := (not1 + not2) / 2;  
  write ('Media: ',med:5:2);  
  if med < 5  
  then writeln (' - Aluno(a) reprovado(a).')  
  else if med >= 7  
    then writeln (' - Aluno(a) aprovado(a).')  
    else writeln (' - Aluno(a) indicado(a) para 3a. avaliação.');
```

readkey;

end.

3.A. Exercícios (if .. then .. else)

1. Elaborar um programa para ler um número inteiro e informar se ele é par ou ímpar. Regra: o número é par, se o resto da divisão do número por 2 for igual a 0. Se o número for par mostrar em azul, se for ímpar mostrar em vermelho; o fundo da tela deve estar cinza.
2. Criar um programa para ler um número e informar se ele é divisível por três ou não na cor azul.
3. Fazer um programa para ler três valores inteiros, determinar e mostrar o menor deles na linha 16, coluna 10. Suponha que os valores são diferentes.
4. Criar um programa para ler a letra inicial do sexo de uma pessoa e mostrar a descrição: Masculino, Feminino ou Inválido. Utilize a cor vermelha.
5. Escrever um programa para ler dois números quaisquer e mostrar uma mensagem indicando se são iguais ou qual deles é o maior. Mostre a mensagem na linha 20, coluna 15.
6. Fazer um programa para ler a letra inicial do estado civil de uma pessoa e mostrar a descrição de acordo com a tabela abaixo:
C – Casado(a) S – Solteiro(a) V – Viúvo(a)
Q – Desquitado(a) D – Divorciado(a) qualquer outra letra – Inválido
7. Elaborar um programa para ler um código e mostrar a descrição correspondente de acordo com a tabela apresentada a seguir:

<u>Código</u>	<u>Descrição</u>
1 até 100	Grupo 1
101 até 1000	Grupo 2
maior que 1000	Grupo 3
outro	Código inválido
8. Fazer um programa para ler três números e informe o maior e o menor. Suponha que os números são diferentes. Utilize cores diferentes. Mostre as mensagens na coluna 15.
9. Construir um programa para ler os coeficientes a, b e c de uma equação de segundo grau e calcular o delta. O programa deve mostrar o valor de *delta* e se o *delta* for negativo, a mensagem “A equação não tem solução real”. Regra: $\Delta = b^2 - 4ac$.

10. Uma faculdade atribui menções aos alunos conforme a faixa de notas que tenha atingido:
- | | | | |
|------------|---------------|------------|----------------------|
| 9,0 a 10: | SS (superior) | 7,0 a 8,9: | MS (médio superior) |
| 5,0 a 6,9: | MM (médio) | 3,0 a 4,9: | MI (médio inferior) |
| 0,1 a 2,9: | II (inferior) | 0 : | SR (sem rendimento). |
- Elaborar um programa para ler a nota e informar a menção.
11. Fazer um programa para ler três números diferentes e colocá-los em ordem crescente. Coloque o fundo de tela com azul e os números crescentes em amarelo, marrom e vermelho, respectivamente.
12. Fazer um programa para ler os valores dos três lados de um triângulo (A, B, C). Verificar e mostrar se é um triângulo ou não e qual é o tipo.
 Dica 1: Triângulo é uma forma geométrica (polígono) composta por três lados, onde cada lado é menor que a soma dos outros dois lados.
 Dica 2: Um triângulo é: a) isósceles quando possui dois lados iguais e um diferente; b) escaleno quando possui todos os lados diferentes; e c) equilátero quando possui todos os lados iguais.
13. **DESAFIO:** Construir um programa que seja capaz de concluir qual dentre os animais seguintes foi escolhido, através de perguntas e respostas. Animais possíveis: leão, cavalo, homem, macaco, morcego, baleia, avestruz, pingüim, pato, águia, tartaruga, crocodilo e cobra.

Exemplo:

É mamífero? Sim.
 É quadrúpede? Sim.
 É carnívoro? Não.
 É herbívoro? Sim.
 Então o animal escolhido foi o cavalo.

Utilize as seguintes classificações:

Mamíferos	<i>Quadrúpede</i>	<i>Carnívoro</i>	Leão
		<i>Herbívoro</i>	Cavalo
	<i>Bípede</i>	<i>Onívoro</i>	Homem
		<i>Frutífero</i>	Macaco
	<i>Voadores</i>	Morcego	
	<i>Aquáticos</i>	Baleia	
Aves	<i>Não-voadores</i>	<i>Tropical</i>	Avestruz
		<i>Polar</i>	Pingüim
	<i>Nadadoras</i>	Pato	
	<i>De rapina</i>	Águia	
Répteis	<i>Com casco</i>	Tartaruga	
	<i>Carnívoro</i>	Crocodilo	
	<i>Sem patas</i>	Cobra	

4. Comando de múltipla escolha (case)

A instrução **case** tem por finalidade tomar uma decisão baseando-se no conteúdo de uma variável ou no resultado de uma expressão. Esta instrução pode simplificar o uso da instrução **if**.

A instrução **case** é assim definida:

```
case seletor of
  lista-de-alvos : C1;
  lista-de-alvos : Cn;
  else Comando;
end;
```

```
case seletor of
  lista-de-alvos : begin
                    Ca;
                    Cb;
                  end;
  lista-de-alvos : begin
                    Ca;
                    Cb;
                  end;
  else C1;
        C2;
        Cn;
end;
```

seletor é qualquer expressão cujo valor seja de um tipo escalar (não é aceito *real* nem *string*).
lista-de-alvos é uma lista de variáveis, separadas por vírgula, do mesmo tipo do seletor.
begin .. end devem ser utilizados se existir mais de um comando para uma determinada lista-de-alvos, não sendo necessário na cláusula **else**.

Exemplo de lista-de-alvos: **1 .. 3** (um até três)
 6, 9 (seis ou nove)
 10 .. 30, 50 (dez até trinta ou cinquenta)

Exemplo: Fazer um programa para ler um número de 1 a 12 e informar o trimestre do ano correspondente, sendo janeiro, fevereiro e março, o primeiro trimestre. Se o número não corresponder a um mês válido, é mostrada uma mensagem de erro na linha 22 em vermelho. O fundo de tela deve estar em amarelo e a entrada e saída de dados em azul.

```
Program Lista04a;
uses crt;
var num : integer;

begin
  textbackground (Yellow);
  textcolor (Blue);
  clrscr;
  write ('Digite um numero: ');    readln (num);

  case num of
    1..3 : writeln (' 1° Trimestre ');
    4..6 : writeln (' 2° Trimestre ');
    7..9 : writeln (' 3° Trimestre ');
    10..12 : writeln ('4° Trimestre ');
    else    textcolor (Red);
            gotoxy (10, 22);
            writeln ('Mês inválido. ');
  end;
  readkey;
end.
```

Exemplo: Fazer um programa para ler uma letra e mostrar uma mensagem indicando se é vogal minúscula, vogal maiúscula, sinal aritmético ou outros.

```
Program Lista04b;
uses crt;
var letra : char;

begin
  textbackground (Yellow);
  textcolor (Blue);
  clrscr;
  write ('Digite uma letra: ');    readln (letra);

  case letra of
    'a', 'e', 'i', 'o', 'u' : writeln (' Vogal minúscula ');
    'A', 'E', 'I', 'O', 'U' : writeln (' Vogal maiúscula ');
    '+', '-', '/', '*' : writeln (' Sinal aritmético ');
    else writeln (' Outros. ');
  end;
  readkey;
end.
```

4.A. Exercícios (case)

1. Fazer um programa para ler um número de 1 a 7 e mostrar o dia da semana correspondente na cor azul, sendo domingo o dia de número 1. Se o número não corresponder a um dia da semana, é mostrada uma mensagem de erro na cor vermelha. O fundo deve estar na cor cinza.
2. Fazer um programa para ler a letra inicial do estado civil de uma pessoa e mostrar a descrição de acordo com a tabela abaixo:

C – Casado(a)	S – Solteiro(a)	V – Viúvo(a)
Q – Desquitado(a)	D – Divorciado(a)	qualquer outra letra – Inválido
3. Uma faculdade atribui menções aos alunos conforme a faixa de notas que tenha atingido:

90 a 100: SS (superior)	70 a 89: MS (médio superior)
50 a 69: MM (médio)	30 a 49: MI (médio inferior)
01 a 29: II (inferior)	0 : SR (sem rendimento).

Elabore um programa que lê a nota e informa a menção.
4. Elaborar um programa que, dada a idade de um nadador, classifique-o em uma das seguintes categorias:

Não aceito : 0 – 4 anos	Juvenil A : 11 – 13 anos
Infantil A : 5 – 7 anos	Juvenil B : 14 – 17 anos
Infantil B : 8 – 10 anos	Sênior : maiores de 17 anos

Mostrar uma mensagem de erro se a idade for negativa ou maior que 70 anos.
5. Construir um programa que, tendo como dados de entrada o preço de um produto e um código de origem, mostre o preço junto de sua procedência (Ex.: R\$ 500.00 – Sul). Caso o código não seja nenhum dos especificados, o produto deve ser encarado como importado.

Código de origem:

1 – Sul	5 ou 6 – Nordeste
2 – Norte	7 ou 8 ou 9 – Sudeste
3 – Leste	10 até 20 – Centro Oeste
4 – Oeste	25 até 50 – Nordeste
6. Fazer um programa para ler 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.
7. Fazer um programa para solicitar o dia, mês e ano (em números) como entrada para o usuário e mostrar a data no seguinte formato:

Exemplo: 10 12 2002 «=>» 10 de Dezembro de 2002.

5. Repetição c/ teste no início (while)

Esta estrutura de repetição, caracteriza-se por efetuar um teste lógico no início de um looping⁶, verificando se o bloco de comandos será executado.

O controle da repetição é realizado pela condição⁷. Através deste comando (while), o bloco de comandos será executado enquanto a condição for Verdadeira. No momento em que esta condição se torna Falsa, o processamento do bloco é desviado para fora do looping. Sendo a condição Falsa logo no início do looping, as instruções do bloco são ignoradas.

Para o comando while pode ser utilizado o bloco de comandos, quando se tem mais de um comando a ser repetido; ou quando se tem apenas um comando não é necessário o uso de **begin** e **end**.

A instrução **while** é assim definida:

```
while condição do  
begin  
    C1;  
    C2;  
    Cn;  
end;
```

Enquanto a condição for verdadeira, o bloco de comandos é executado.

Quando a condição for falsa, a repetição é finalizada.

Sendo a condição falsa no início, os comandos não são executados nenhuma vez.

Observações:

1. No exemplo abaixo, o ; (ponto e vírgula) após o comando **do** faz com que o programa entre em loop eterno⁸, se a condição for verdadeira.

```
while condição do;  
begin  
    C1;  
    Cn;  
end;
```

⁶ Trecho de programa executado repetidamente um número definido de vezes (*for*), ou até que uma condição seja verdadeira (*repeat*), ou até que uma condição seja falsa (*while*).

⁷ Também conhecida como condição de parada.

⁸ Loop eterno - a condição nunca se torna falsa para que a repetição seja finalizada.

2. No exemplo abaixo, a falta do **begin** após o comando **do** faz com que o programa execute o comando C_1 até que a condição se torne falsa. Geralmente, isso é um erro de lógica.

```
while condição do
    C1;
    Cn;
```

Exemplo: Fazer um programa para mostrar os números de 1 a 10, inclusive; um em cada linha.

```
Program Lista05a;
uses crt;
var x : integer;

begin
    clrscr;
    writeln ( ' Números de 1 a 10. ' );
    x := 1;
    while x <= 10 do
    begin
        writeln ( x );
        inc ( x );
    end;
    readkey;
end.
```

```
Números de 1 a 10.
1
2
3
4
5
6
7
8
9
10
```

Exemplo: Fazer um programa para mostrar os números pares de 100 a 110, inclusive, em ordem decrescente, em quatro espaços cada, lado a lado numa mesma linha.

```
Program Lista05b;
uses crt;
var cont : integer;

begin
    clrscr;
    writeln ( ' Nos. pares de 100 a 110 ' );
    writeln ( ' em ordem decrescente: ' );
    cont := 110;
    while cont >= 100 do
    begin
        write ( cont :4 );
        dec ( cont, 2 );
    end;
    readkey;
end.
```

```
Nos. pares de 100 a 110
em ordem decrescente:
 110 108 106 104 102 100
```

Exemplo: Fazer um programa para ler o sexo de dez pessoas e mostrar a quantidade de pessoas do sexo masculino e do sexo feminino, separadamente.

```

Program Lista05c;
uses crt;
var   sxm, sxf, c : integer;
        sexo : char;

begin
  clrscr;
  c := 0;      sxm := 0;   sxf := 0;
  while c < 10 do begin
    write ( 'Digite o sexo [ F/M ]: ' );
    readln (sexo);
    sexo := upcase (sexo);
    if sexo = 'F'
    then sxf := sxf + 1
    else if sexo = 'M'
    then sxm := sxm + 1;
    c := c + 1;
  end;
  writeln;
  writeln ( sxf, ' - mulheres. ' );
  writeln ( sxm, ' - homens. ' );
  readkey;
end.

```

```

Digite o sexo [ F/M ]: F
Digite o sexo [ F/M ]: f
Digite o sexo [ F/M ]: M
Digite o sexo [ F/M ]: F
Digite o sexo [ F/M ]: M
Digite o sexo [ F/M ]: m
Digite o sexo [ F/M ]: f
Digite o sexo [ F/M ]: M
Digite o sexo [ F/M ]: F
Digite o sexo [ F/M ]: f

6 - mulheres.
4 - homens.

```

Exemplo: Fazer um programa para mostrar o produto de vários valores fornecidos como entrada. O último valor será 999.

```

Program Lista05d;
uses crt;
var   prod, num : real;

begin
  clrscr;
  prod := 1;
  writeln ( ' 999 para Finalizar ' );
  write ( ' Digite um número: ' );
  readln ( num );
  while num < > 999 do begin
    prod := prod * num;
    write ( ' Digite outro número: ' );
    readln ( num );
  end;
  writeln ( ' Produto: ', prod:5:2 );
  readkey;
end.

```

```

999 para Finalizar
Digite um número: 2
Digite outro número: 3
Digite outro número: 10
Digite outro número: 2
Digite outro número: 1
Digite outro número: 999
Produto: 120.00

```

5.A. Exercícios (while)

1. Após a execução do programa abaixo, o que terá sido apresentado na tela? Faça o teste de mesa e apresente os valores das variáveis.

```
Program Ex01;
uses crt;
var  a, b, c : integer;
begin
  clrscr;
  a := 1;      b := 5;      c := 5;
  while c >= 1 do
  begin
    writeln ( a, ' - ', b );
    inc ( a );
    dec ( b );
    dec ( c );
  end;
end.
```

2. Fazer um programa individual para cada um dos itens abaixo para calcular e mostrar:
- os números inteiros de 1 a 100, inclusive.
 - os números inteiros de 1 a 20, inclusive; ordem decrescente.
 - os números pares de 100 a 200, inclusive.
 - a tabuada de multiplicação de um número fornecido como entrada.
 - o fatorial de um número inteiro fornecido como entrada (N!). Ex.: $5! = 5 \times 4 \times 3 \times 2 \times 1$.
3. Elaborar um programa individual para cada um dos itens abaixo para calcular e mostrar:
- a soma parcial de 7 números fornecidos como entrada.
 - o produto de 10 números fornecidos como entrada.
 - a média de 5 números fornecidos como entrada.
 - a quantidade de pessoas maiores de idade (≥ 18), dentre 10 idades lidas.
 - os **n** primeiros termos de uma progressão aritmética (P.A.) de elemento inicial **a** e razão **r**.
 - a média dos números pares, dentre 10 números lidos. Pode ocorrer que nem todos os números sejam pares.
4. Construir um programa individual para cada um dos itens abaixo para calcular e mostrar:
- a soma de vários valores inteiros e positivos fornecidos como entrada. O último valor será -1.
 - o soma de vários números pares fornecidos como entrada. O último valor será -1.

- c) o produto de vários valores inteiros e negativos fornecidos como entrada. O último valor será zero.
 - d) idade média das pessoas de um determinado grupo. Ler as idades até que a idade seja zero.
 - e) a soma de vários números fornecidos como entrada. Quando a soma ultrapassar 500, o programa deve ser finalizado.
 - f) o produto parcial de vários números fornecidos como entrada. Quando o produto ultrapassar 10.000, o programa deve ser finalizado.
 - g) a média de vários números fornecidos como entrada. Finalizar quando a soma for maior que 1000.
 - h) a soma dos números pares compreendidos entre dois números inteiros lidos. Suponha que o primeiro número é menor que o segundo.
 - i) o produto dos números ímpares compreendidos entre dois números inteiros lidos. Suponha que o primeiro número é menor que o segundo.
5. Fazer um programa para ler uma frase na cor branca e mostrar a quantidade de vogais na cor amarela. O fundo da tela deve estar na cor preta.
6. Fazer um programa para ler os dados (matrícula, idade, sexo, altura e concursado [S/N]) de vários funcionários de uma empresa. O último tem a matrícula igual a zero e não entra nos cálculos. Calcular e mostrar:
- a) o número de funcionárias concursadas;
 - b) o número de funcionários (somente homens);
 - c) a maior idade dos homens concursados;
 - d) a quantidade de mulheres com mais de 30 anos sem concurso;
 - e) a quantidade de concursados(as);
 - f) a média das alturas dos homens com menos de 40 anos.

Para testar o programa, utilize a tabela abaixo:

matrícula	idade	sexo	altura	concurado
01	35	F	1.80	S
02	30	M	1.70	S
03	20	F	1.65	N
04	38	F	1.73	N
05	45	M	1.80	N
06	33	F	1.75	S
07	23	M	1.90	N
08	40	M	1.30	S

Respostas

- a) 2**
- b) 4**
- c) 40**
- d) 1**
- e) 4**
- f) 1.80**

7. Uma pesquisa sobre características físicas da população de uma determinada região coletou os seguintes dados referentes a cada habitante, para serem analisados: sexo (Masculino, Feminino); idade (em anos); cor dos cabelos (louros, castanhos, pretos).

Fazer um programa que determine e escreva:

- a maior idade dos habitantes;
- a quantidade de indivíduos do sexo feminino com idade entre 15 e 35 anos;
- a quantidade de indivíduos que tenham cabelos louros.

Obs.: O último indivíduo terá idade negativa, e não corresponde a ninguém. Garanta que a cor dos cabelos será um dos três valores possíveis descritos acima.

8. Fazer um programa que calcule e escreva o valor de S. Resposta: S = 95.50.

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

9. Escrever um programa para achar quantas pessoas há num grupo; quantas tem menos de 20 anos; quantas mulheres com mais de 30 anos e quantos homens com 18 anos. O programa deve ser executado até que encontre a idade 0 como último dado. Todos os dados calculados devem ser fornecidos como saída. Garanta que o sexo digitado seja F ou M, não permita outra entrada.

10. Num frigorífico existem 8 bois. Cada boi traz preso em seu pescoço um cartão contendo seu número de identificação e seu peso. Faça um programa que escreva o número e peso do boi mais gordo e do boi mais magro.

11. Para cada uma das 10 mercadorias diferentes com que um armazém trabalha dispõe-se dos seguintes dados:

- nome da mercadoria (no máximo 25 letras);
- preço unitário;
- quantidade total vendida no mês.

Elaborar um programa para calcular e mostrar o total de vendas (*preço unitário x quantidade vendida*) de cada mercadoria realizado no mês e o faturamento total mensal do armazém.

12. Fazer um programa para ler a idade e o estado civil de 10 pessoas. Calcular e mostrar:

- a) quantidade de pessoas casadas;
- b) quantidade de pessoas solteiras;
- c) a idade da pessoa solteira mais nova;
- d) média das pessoas viúvas;
- e) porcentagem de pessoas desquitadas ou separadas dentre todas as pessoas.

13. Uma certa firma fez uma pesquisa de mercado para saber se as pessoas gostaram ou não de um novo produto lançado no mercado. Para isso, forneceu o sexo do entrevistado e sua resposta (sim ou não). Sabendo-se que foram entrevistadas 10 pessoas, fazer um programa que calcule e escreva:
- O número de pessoas que responderam Sim;
 - O número de pessoas que responderam Não;
 - O número de mulheres que responderam sim;
 - O número de homens que responderam não.

14. Uma empresa decide dar um reajuste a seus funcionários de acordo com os critérios seguintes:
- 50 % para aqueles que ganham menos de 300;
 - 20 % para aqueles que ganham entre 300 e 1000 (inclusive); e
 - 15 % para os demais.

Escreva um programa que leia o valor do salário atual para calcular e mostrar o valor do salário reajustado dos 10 funcionários da empresa. Desconsidere valores menores ou iguais a zero. Obedeça o layout definido abaixo:

	02	07	30	45
08		Salário Atual	%	Salário Reajustado
	02	10	30	50
10	1	270.00	50%	405.00
19	10	1000.00	20%	1200.00

15. **DESAFIO:** Fazer um programa que:
- Leia o número e a altura das moças inscritas em um concurso de beleza. A última linha, que não corresponde a nenhuma moça, conterá o zero no lugar do número;
 - Calcule e escreva as duas maiores alturas e quantas moças as possuem.
16. **DESAFIO:** Um comerciante deseja fazer o levantamento do lucro das mercadorias que ele comercializa. Para isto mandou digitar uma linha para cada mercadoria com o nome, preço de compra (PC) e preço de venda (PV) das mesmas. Fazer um programa que:
- determine e escreva quantas mercadorias proporcionam:
 - lucro < 10%;
 - 10% <= lucro <= 20%;
 - lucro > 20 %
 - determine e escreva o valor total de compra e de venda de todas as mercadorias, assim como o lucro total;
- Obs.: Adote um flag (uma condição) para encerrar o programa;
 Regra: $PV = PC + PC * \text{lucro} / 100$.
17. **DESAFIO:** Elaborar um programa que leia três valores e calcule e mostre a média geométrica dos números lidos. Regra: raiz cúbica do produto dos valores.
18. **DESAFIO:** Fazer um programa para ler um número e mostrar se é primo ou não. Obs.: Um número é primo se for divisível apenas por um e por ele mesmo.

6. Repetição c/ teste no final (repeat .. until)

Esta estrutura caracteriza-se por efetuar um teste lógico no final de um looping. Desta forma, os comandos indicados dentro da estrutura serão executados pelo menos uma vez antes da condição ser verificada.

Assim como no comando while, o controle da repetição é realizado pela condição. Sendo a condição Falsa, os comandos dentro da estrutura serão novamente executados. No momento em que esta condição se torna Verdadeira, o processamento do bloco é desviado para fora do looping.

Os comandos a serem repetidos pelo comando repeat não necessitam estar delimitados pelos comandos **begin** e **end**, os quais podem ser utilizados.

A instrução **repeat** é assim definida:

```
repeat
    C1;
    Cn;
until condição ;
```

Os comandos do bloco são executados pelo menos uma vez.

Se a condição for verdadeira, a repetição é finalizada e o comando seguinte ao bloco será executado. Se a condição for falsa, os comandos são executados novamente até que a condição se torne verdadeira.

Exemplo: Fazer um programa para mostrar os números inteiros de 10 a 20, inclusive; um em cada linha, em ordem decrescente.

```
Program Lista06a;
uses crt;
var x : integer;

begin
    clrscr;
    writeln ( ' Números de 20 a 10. ' );
    x := 20;
    repeat
        writeln ( x );
        dec ( x );
    until x < 10;
    readkey;
end.
```

```
Números de 20 a 10.
20
19
18
17
16
15
14
13
12
11
10
```

Exemplo: Fazer um programa para ler dois números inteiros, sendo que o primeiro deve ser menor que o segundo. Calcular e mostrar a média dos números inteiros compreendidos entre os dois números lidos.

```
Program Lista06b;
uses crt;
var   n1, n2, c, q, sm : integer;
        md : real;

begin
    md := 0;    sm := 0;    q := 0;
    repeat                                { para garantir que n1 < n2 }
        clrscr;
        writeln ('O primeiro valor deve ser menor que o segundo. ');
        write ('Digite o primeiro valor: ');    readln (n1);
        write ('Digite o segundo valor: ');    readln (n2);
    until n1 < n2;

    c := n1 + 1;
    if n1 < n2                                { Para evitar loop eterno }
    then repeat
        sm := sm + c;
        q := q + 1;
        c := c + 1;
    until c = n2;

    if q > 0                                { verifica-se para evitar divisão por zero. }
    then md := sm / q;
    writeln ('Média dos números: ', md:5:2);
    readkey;
end.
```

Exemplo: Elaborar um programa para ler o sexo de uma pessoa e garantir que é válido.

```
Program Lista06b;
uses crt;
var   sx : char;

begin
    repeat
        clrscr;
        writeln ('Digite o sexo da pessoa: ');
        readln (sx);
        sx := uppercase (sx);
    until ( sx = 'F' ) or ( sx = 'M' );
    readkey;
end.
```

Exemplo: Fazer um programa para ler a altura e o sexo de dez pessoas e mostrar a menor altura das mulheres e a maior altura dos homens. Garanta que as alturas lidas são válidas – entre 1.00m e 2.50m, inclusive. Garanta também que o sexo informado é válido: M ou F.

```
Program Lista06c;
uses crt;
var   c : integer;
      alt, menor, maior : real;
      sx : char;

begin
  menor := 3;
  clrscr;
  repeat           { executar 10 vezes }

    repeat
      write ( 'Digite a altura [1.00 - 2.50]: ' );
      readln ( alt );
    until ( alt >= 1 ) and ( alt <= 2.5 );

    repeat
      write ( 'Digite o sexo [F/M]: ' );
      readln ( sx );
      sx := upcase (sx);
    until ( sx = 'F' ) or ( sx = 'M' );

    if ( sx = 'F' ) and ( alt < menor )
    then menor := alt
    else if ( sx = 'M' ) and ( alt > maior )
    then maior := alt;

    inc ( c );
  until c = 10;

  writeln ( 'Maior altura dos homens: ', maior:5:2 );
  writeln ( 'Menor altura das mulheres: ', menor:5:2 );

  readkey;
end.
```

6.A. Exercícios (repeat .. until)

1. Construir um programa para cada um dos itens abaixo para:
 - a) calcular e mostrar a soma de 5 números fornecidos como entrada.
 - b) calcular e mostrar o produto de 10 números fornecidos como entrada.
 - c) calcular e mostrar o fatorial de um número lido.
 - d) calcular e mostrar a média de 15 números inteiros fornecidos como entrada.
 - e) ler dez números e mostrar o produto dos ímpares e a soma dos pares.
 - f) ler a idade de 8 pessoas e mostrar a menor idade.
 - g) ler a idade de 15 pessoas; mostrar:
 - quantidade de crianças (0 a 12 anos);
 - quantidade de jovens (13 a 19 anos);
 - quantidade de adultos (20 a 60 anos);
 - quantidade de idosos (maiores que 60 anos).
 - h) ler o sexo de 10 pessoas. Garantir que o sexo será (F/M). Mostrar a quantidade de mulheres e a quantidade de homens.

2. Construir um programa para cada um dos itens abaixo para:
 - a) mostrar a soma de vários números pares positivos fornecidos como entrada. O último valor será -1.
 - b) mostrar o produto de vários valores fornecidos como entrada. O último valor será 999 e não entra nos cálculos.
 - c) ler o sexo de várias pessoas e mostrar: a quantidade de homens e a quantidade de mulheres. Finalizar quando o sexo for igual a 'X'. Garantir que o sexo será: F, M ou X.
 - d) ler o estado civil de várias pessoas e mostrar a quantidade de casados, a quantidade de solteiros, a quantidade de divorciados, a quantidade de desquitados e a quantidade de viúvos. Finalizar quando o estado civil for igual a 'F'. Garantir que o estado civil será: C, S, D, Q, V ou F.
 - e) mostrar a soma de vários valores inteiros e positivos fornecidos como entrada. O último valor será -1.
 - f) mostrar o produto parcial de vários números inteiros fornecidos como entrada. Quando o produto ultrapassar 10.000 o programa deve ser finalizado.
 - g) ler vários números reais e mostrar a soma parcial dos números. Finalizar o programa quando a soma ultrapassar 500.

3. Escrever um programa para calcular e mostrar os **n** primeiros termos de uma progressão geométrica (P.G.) de elemento inicial **a** e razão **r**.

4. Fazer um programa para calcular e mostrar a soma dos números pares compreendidos entre dois números lidos, inclusive. Garanta que o primeiro número seja menor que o segundo. Mostre a seguinte mensagem: 'A soma dos pares entre *num1* e *num2* é: *soma*', onde *num1* e *num2* são os valores lidos e *soma* é o resultado.

Exemplo: A soma dos pares entre 10 e 16 é: 52

5. Elaborar um programa para sortear e mostrar dez números de 0 a 100, inclusive.
Dica: Pesquise os comandos *RANDOM* e *RANDOMIZE*.

6. A série de Fibonacci é formada pela seqüência:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ..., etc.

Escrever um programa para gerar a série de Fibonacci até o vigésimo termo. Utilize uma constante valendo 20.

7. Em uma eleição presidencial existem três candidatos. Os votos são informados através de códigos. Os dados são utilizados para a contagem dos votos e obedecem à seguinte codificação:
1, 2, 3 = voto para os respectivos candidatos;
9 = voto nulo;
0 = voto em branco.

Elaborar um programa para ler o nome dos candidatos e um conjunto de votos. A leitura deve ser finalizada quando for digitado o código igual a cinco. Ao final deve-se mostrar:

- a) Total de votos de cada candidato;
- b) Total de votos nulos;
- c) Total de votos em branco.

8. Fazer um programa que calcule e escreva a soma dos N primeiros termos da série abaixo. N deve ser fornecido como entrada.

$$S = \frac{480}{10} - \frac{470}{11} + \frac{460}{12} - \frac{450}{13} + \dots$$

Resposta: $n = 1 \rightarrow S = 48.00$ $n = 2 \rightarrow S = 5.27$
 $n = 3 \rightarrow S = 43.61$ $n = 4 \rightarrow S = 8.99$

9. Fazer um programa que calcule e escreva o valor de S. (*Resposta: S = 0.65*)

$$S = \frac{1}{1} - \frac{2}{4} + \frac{3}{9} - \frac{4}{16} + \dots - \frac{10}{100}$$

10. Sendo $H = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{N}$, fazer um programa para mostrar o valor de H.

O número N será fornecido como entrada. Mostre o resultado na tela como no exemplo a seguir:

Exemplo: Se $N = 4$;

$$H = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{N} = 2.08$$

11. **DESAFIO:** Construir um programa para ler uma frase com 30 letras (incluindo espaços em branco), calcular e mostrar:
- quantos espaços em brancos existem na frase;
 - quantas vezes aparece a letra *A*;
 - quantas vezes ocorre um mesmo par de letras (*RR*, *SS*, ...) na frase.

12. **DESAFIO:** Fazer um programa que leia dois números e imprima os números pares ou ímpares (escolha do usuário) entre os números lidos inclusive, separados por '-'.
Exemplo:

entrada: **n1** = 10 **n2** = 20 **op** = 'P'
saída: 10 - 12 - 14 - 16 - 18 - 20

13. **DESAFIO:** Uma empresa decidiu fazer um levantamento em relação aos candidatos que se apresentarem para preenchimento de vagas no seu quadro de funcionários, utilizando processamento eletrônico. Supondo que você seja o programador encarregado desse levantamento, fazer um programa que:

- Leia um conjunto de dados para cada candidato contendo: número de inscrição, idade, sexo, experiência no serviço (sim ou não); *o último conjunto contém o número de inscrição do candidato igual a zero*;
- Calcule e escreva:
 - o número de candidatos do sexo feminino,
 - o número de candidatos do sexo masculino,
 - idade média dos homens que já têm experiência no serviço,
 - porcentagem dos homens com mais de 45 anos entre o total de homens,
 - número de mulheres que têm idade inferior a 35 anos e com experiência no serviço,
 - a menor idade entre as mulheres que já têm experiência no serviço;

*Dica: $porc = num.homens.exp * 100 / num.total.hom$*

7. Repetição c/ variável de controle (for .. to / downto .. do)

Com o uso das estruturas **while** e **repeat** é possível elaborar rotinas que efetuam a execução de um looping um determinado número de vezes através da utilização de um contador (variável que controla a quantidade de repetições do bloco) ou mesmo por uma variável que aguarde a resposta do usuário. Independentemente da forma de tratamento, esta variável é denominada variável de controle.

Existe uma outra forma que visa a facilitar o uso de contadores finitos, sem fazer uso das estruturas anteriores. Desta forma, os loopings com **while** e **repeat** passam a ser utilizados em loopings onde não se conhece de antemão o número de vezes que uma determinada seqüência de instruções deverá ser executada.

A estrutura de repetição **for** é utilizada exclusivamente quando um número finito de comandos devem ser repetidos e tem seu funcionamento controlado por uma variável de controle do tipo contador, podendo ser crescente ou decrescente.

A variável-de-controle, o valor-inicial e o valor-final devem ser todas do mesmo tipo (integer ou char) e não devem ser modificados dentro do bloco⁹.

A instrução **for** é assim definida para looping crescente:

```
for variável-de-controle := valor-inicial to valor-final do
begin
    C1;
    Cn;
end;
```

A instrução **for** é assim definida para looping decrescente:

```
for variável-de-controle := valor-inicial downto valor-final do
begin
    C1;
    Cn;
end;
```

A variável-de-controle inicia seu valor em valor-inicial e finaliza com valor-final, inclusive. Não é possível haver acréscimo ou decréscimo diferente de 1.

⁹ Se isso realmente for necessário, utilize outro comando de repetição. Em Pascal, a modificação pode causar lopp eterno.

Exemplo: Fazer um programa para calcular e mostrar a soma de dez números inteiros fornecidos como entrada.

```

Program Lista07a;
uses crt;
var num, c, sm : integer;

begin
    sm := 0;
    clrscr;
    for c := 1 to 10 do
        begin
            write ( 'Digite o ', c:2, 'o. valor: ' );
            readln (num);
            inc (sm, num);
        end;
    writeln;
    writeln ( 'A soma dos números: ', sm );
    readkey;
end.
    
```

```

Digite o 1o. valor: 20
Digite o 2o. valor: 7
Digite o 3o. valor: 4
Digite o 4o. valor: 5
Digite o 5o. valor: 10
Digite o 6o. valor: 3
Digite o 7o. valor: 0
Digite o 8o. valor: 6
Digite o 9o. valor: 15
Digite o 10o. valor: 25
    
```

A soma dos números: 95

Exemplo: Fazer um programa para mostrar os números inteiros de 1 a 100 em ordem decrescente, distribuídos em 10 colunas.

```

Program Lista07b;
uses crt;
var num : integer;

begin
    clrscr;
    writeln;
    writeln ( ' Números de 1 a 100 ' );
    writeln ( ' Ordem decrescente ' );
    writeln;
    for num := 100 downto 1 do
        write (num:8);
    readkey;
end.
    
```

Observação

Se a tela do computador possui 80 colunas e serão utilizadas 10 colunas, então $80/10 = 8$; que é o espaçamento necessário para se obter 10 colunas.

```

Números de 1 a 100
Ordem decrescente

100    99    98    97    96    95    94    93    92    91
 90    89    88    87    86    85    84    83    82    81
  :
  :
 10     9     8     7     6     5     4     3     2     1
    
```

Exemplo: Fazer um programa para mostrar as letras minúsculas de **a** até **z**.

```
Program Lista07c;
uses crt;
var letra : char;

begin
  clrscr;
  writeln;
  writeln ( ' Letras minúsculas ' );
  writeln;
  for letra := 'a' to 'z' do
    write ( letra, ' ' );
  readkey;
end.
```

```
Letras minúsculas
```

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
```

7.A. Exercícios (for .. to / downto .. do)

1. Marque Verdadeiro ou Falso para as afirmações abaixo:
 - a. () Utilizando o comando **while** para fazer a repetição de dois ou mais comandos, é estritamente necessário usar os comandos **begin / end** para delimitá-los.
 - b. () O comando (**for x := 100 to 1 do;**) causa um loop eterno.
 - c. () Utilizando o comando **repeat** para fazer a repetição de dois ou mais comandos, é estritamente necessário usar os comandos **begin / end** para delimitá-los.
 - d. () O comando (**for a := 1 to 10 do;**) é executado 10 vezes.
 - e. () O comando (**while z < 30 do readln (z);**) causa um loop eterno.
 - f. () Utilizando o comando **for** para fazer a repetição de dois ou mais comandos, é estritamente necessário usar os comandos **begin / end** para delimitá-los.
 - g. () O comando (**while x < 30 do; readln (x);**) causa um loop eterno.

2. Em cada uma das opções abaixo, indique o comando de repetição mais adequado para ser utilizado na situação descrita.
- a) (_____) O laço de repetição é executado até que a condição seja falsa.
 - b) (_____) O laço de repetição é executado enquanto a condição é falsa.
 - c) (_____) O laço de repetição é executado um número fixo de vezes.
 - d) (_____) Existe a necessidade de garantir ou validar um determinado dado de entrada.
 - e) (_____) O laço de repetição é executado enquanto a condição é verdadeira.
 - f) (_____) O laço de repetição é executado até que a condição seja verdadeira.
 - g) (_____) Existe uma quantidade fixa de vezes que o laço de repetição deve ser executado.
 - h) (_____) Existe a necessidade de executar um bloco de comandos várias vezes, desde que a condição seja falsa.
 - i) (_____) O bloco de comandos é executado pelo menos uma vez somente se a condição é verdadeira.
 - j) (_____) Existe a necessidade de executar um bloco de comandos várias vezes, desde que a condição seja verdadeira.
 - k) (_____) O bloco de comandos é executado pelo menos uma vez, independente da condição.
3. Construir um programa para cada um dos itens abaixo para:
- a) mostrar os números inteiros de 10 a 20, inclusive.
 - b) mostrar a soma dos números de 1 a 10, inclusive. (*Resposta: Soma = 55*)
 - c) mostrar os números inteiros de 1 até 100, inclusive; em ordem decrescente na mesma tela.
 - d) mostrar o dobro dos números inteiros de 100 até 200, inclusive; em ordem decrescente na mesma tela.
 - e) mostrar o produto dos números ímpares inteiros de 10 a 18. (*Resposta: Produto = 36465*)
 - f) calcular e mostrar os **n** primeiros termos de uma progressão geométrica (P.G.) de elemento inicial **a** e razão **r**.
 - g) calcular e mostrar o produto final dos números inteiros de 1 a 20.
 - h) calcular e mostrar a soma parcial dos números pares de 1 a 100.
 - i) calcular e mostrar a soma dos números pares inteiros de 15 a 50. (*Resposta: S = 594*)

4. Construir um programa para cada um dos itens abaixo para:
 - a) ler 10 números reais e mostrar a média aritmética dos números.
 - b) ler 12 números reais e mostrar o produto dos números lidos.
 - c) mostrar o fatorial de um número lido. Teste $\Rightarrow 8! = 40.320$
 - d) ler 6 números e a cada número lido mostrar se é par ou ímpar.
 - e) calcular e mostrar a soma dos números pares compreendidos entre dois números lidos. Garanta que o primeiro número é menor que o segundo.
 - f) calcular e mostrar o produto dos números ímpares compreendidos entre dois números lidos. Garanta que o primeiro número é menor que o segundo.
 - g) mostrar as letras maiúsculas de A até Z em ordem decrescente.

5. Observe o exemplo a seguir:

```

Program Ex08;
uses crt;
var x, y : integer;

begin
    clrscr;
    writeln ( ' Exemplo ' );
    writeln;
    for x := 1 to 2 do
        for y := 1 to 3 do
            writeln ( x, ' ', y );
    readkey;
end.
    
```

Exemplo	
1	1
1	2
1	3
2	1
2	2
2	3

Fazer um programa independente para cada um dos itens abaixo, que utilizando duas variáveis e um comando **for** dentro de outro **for**, mostre as seguintes saídas:

a)	b)	c)	d)	e)	f)	g)	h)
3 1	6 9	2 5	4 4	1 1	7 7	10 8	10 20
3 2	6 8	2 4	4 3	1 2	8 7	10 9	10 19
3 3	6 7	2 3	4 2	1 3	8 8	10 10	10 18
2 1	5 9	2 2	4 1	2 2	9 7	10 11	11 22
2 2	5 8	3 5	3 3	2 3	9 8	9 7	11 21
2 3	5 7	3 4	3 2	3 3	9 9	9 8	11 20
1 1		3 3	3 1			9 9	12 24
1 2		3 2	2 2			9 10	12 23
1 3			2 1			8 6	12 22
			1 1			8 7	
						8 8	
						8 9	

6. Fazer um programa para ler uma frase e mostrá-la em letras maiúsculas.
7. Elaborar um programa para gerar os 10 primeiros termos da seqüência de Fibonacci, como apresentado abaixo:

1 1 2 3 5 8 13 21 34 55

8. Preparar um programa para gerar os 8 primeiros termos da seqüência:

1 2 2 4 8 32 256 8192

9. Dado um conjunto de valores, obtenha a altura e o sexo (masculino, feminino) de 5 pessoas. Fazer um programa que calcule e escreva:
- a maior e a menor altura do grupo;
 - a média de altura das mulheres;
 - o número de homens.
10. Fazer um programa para calcular e mostrar a soma dos N primeiros termos da seqüência abaixo. N deve ser fornecido como entrada.

$$\begin{array}{r} 1000 \\ \hline 1 \end{array} + \begin{array}{r} 997 \\ \hline 2 \end{array} - \begin{array}{r} 994 \\ \hline 3 \end{array} + \begin{array}{r} 991 \\ \hline 4 \end{array} - \dots$$

Resposta: $n = 1 \rightarrow 1000.00$ $n = 2 \rightarrow 1498.50$
 $n = 3 \rightarrow 1167.17$ $n = 4 \rightarrow 1414.92$

11. Dado um conjunto de valores, obtenha a idade, a altura e o sexo (masculino, feminino) de 10 pessoas. Fazer um programa que calcule e escreva:
- a maior idade;
 - a menor altura dos homens;
 - a maior altura das mulheres;
 - a média das idades maiores que 18.
12. **DESAFIO:** Capicuas são números que têm o mesmo valor, se lidos da esquerda para a direita ou da direita para a esquerda. Ex.: 44, 232, 8998, etc. Fazer um programa que determine e escreva todos os números inteiros e positivos menores que 11.000 que são quadrados perfeitos e capicuas ao mesmo tempo.

8. Bibliografia

BORLAND International, Inc. **Software Turbo Pascal - version 7.0**. Copyright (c) 1992.

ASCENCIO, Ana F. G. & CAMPOS, Edilene. A. V. **Fundamentos da Programação de Computadores**. São Paulo: Prentice Hall, 2002. 355 p.

FARRER, Harry, et. al. **Programação Estruturada de Computadores – Algoritmos Estruturados**. 2. ed. Rio de Janeiro: Guanabara, 1989. 252 p.

FARRER, Harry, et. al. **Programação Estruturada de Computadores – Pascal Estruturado**. 3. ed. Rio de Janeiro: Livros Técnicos e Científicos, 1999. 279 p.

FORBELLONE, André L. V., EBERSPÄCHER, Henri. **Lógica de Programação**. São Paulo: McGraw-Hill, 1993. 214 p.

GUIMARÃES, Ângelo, LAGES, Newton. **Algoritmos e Estruturas de Dados**. Rio de Janeiro: Livros Técnicos e Científicos, 1994. 216 p.

MANZANO, José Augusto N. G., OLIVEIRA, Jayr F. **Algoritmos: Lógica para desenvolvimento de programação de computadores**. 7. ed. São Paulo: Érica, 1996. 264 p.

MANZANO, José Augusto N. G., OLIVEIRA, Jayr F. **Estudo Dirigido de Algoritmos**. 8. ed. São Paulo: Érica, 1997. 240 p.

MANZANO, José Augusto N. G., YAMATUMI, Wilson Y. **Programando em Turbo Pascal 7.0**. 7. ed. São Paulo: Érica, 2001. 416 p.

RINALDI, Roberto. **Turbo Pascal 7.0 – Comandos e Funções**. 15. ed. São Paulo: Érica, 1998. 424 p.

SALIBA, Walter L. C. **Técnicas de Programação – Uma abordagem estruturada**. São Paulo: Makron Books, 1992. 160 p.

SALVETTI, Dirceu D., BARBOSA, Lisbete M. **Algoritmos**. São Paulo: Makron Books, 1998. 273 p.

SCHMITZ, Eber A., TELES, Antônio A. S. **Pascal e Técnicas de Programação: Incluindo Turbo Pascal**. 3. ed. Rio de Janeiro: Livros Técnicos e Científicos, 1988. 287 p.

XAVIER, Gley F. C. **Lógica de Programação**. 2.ed. São Paulo: SENAC, 2001. 378 p.

9. Anexo A - Unidades Predefinidas

As versões mais recentes do TURBO PASCAL permitem obter descrição, sintaxe, restrições e exemplos das funções e procedimentos das unidades predefinidas, posicionando-se o cursor sobre seu nome e pressionando as teclas Ctrl e F1.

- ✓ **System** – contém constantes, variáveis, funções e procedimentos predefinidos. É a única unidade que não necessita estar na cláusula **uses** dos programas ou das outras unidades que a utilizarem.

Ex.: tipos de constantes e variáveis; comentários; expressões aritméticas, lógicas e literais; funções numéricas predefinidas; funções literais predefinidas; comandos de entrada e saída; comandos de estrutura condicional e de repetição.

<i>Operações aritméticas</i>	
Nome	Resultado
A div B	Retorna o quociente inteiro da divisão de A por B (operandos inteiros)
A mod B	Retorna o resto da divisão inteira de A por B (operandos inteiros)

<i>Operadores</i>			
Operadores relacionais	Operadores lógicos	Operadores aritméticos	Prioridades
=, <=, <, <>, >=, >	and, or, not	*, /, +, -, div, mod	not *, /, div, mod, and +, -, or =, <>, <, <=, >=, >

Exemplos onde A = 10, B = 'Pascal', C = True, D = 5, E = 'Fiplac' e F = False.

- | | |
|-----------------------|---------------------------|
| a) A < D = False | d) C and F = False |
| b) D div 2 = 2 | e) not F = True |
| c) D <= A = True | f) A mod 2 = 0 |

<i>Funções numéricas predefinidas</i>		
Nome	Resultado	Tipo de resultado
ABS (EA)	Valor absoluto de EA (Expr. Aritmética)	Real ou inteiro
FRAC (EA)	Parte fracionária de EA	Real
INT (EA)	Parte inteira de EA	Real
ROUND (EA)	Arredonda o valor real de EA	Inteiro
SQR (EA)	Quadrado de EA	Real ou inteiro
SQRT (EA)	Raiz quadrada de EA	Real

Exemplos onde A = 10, B = 5.5, C = 7.8, D = 4.3 e E = -3.6:

- a) ABS (E) = 3.6
- b) FRAC (B) = 0.5
- c) INT (C) = 7
- d) ROUND (C) = 8
- e) ROUND (D) = 4
- f) SQR (A) = 100
- g) SQRT (ABS (E * A)) = 6

<i>Funções literais predefinidas¹⁰</i>		
Nome	Retorna	Resultado
CONCAT (S1, S2, ...)	Concatenação de S1, Sn ...	Literal
COPY (S, I, N)	Substring de S a partir de I com N caracteres	Literal
LENGTH (S)	No. atual de caracteres de S	Inteiro
POS (S1, S)	Posição inicial de S1 em S	Byte
UPCASE (Ch)	Retorna a letra maiúscula de Ch	Char

Exemplos onde X = 'PAIS', Y = 'CAPITAL', Z = 'BRASILIA', W = 'char'

- a) CONCAT (X, ' - ', 'BRASIL') = 'PAIS - BRASIL'
- b) COPY (Z, 1, 6) = 'BRASIL'
- c) POS ('ITA', Y) = 4
- d) UPCASE (W[1]) = 'C'
- e) LENGTH (Y) = 7
- f) POS ('CIPA', Y) = 0
- g) UPCASE (W[4]) = 'R'

<i>Procedimentos literais predefinidos¹¹</i>		
Nome	Recebe	Devolve
DELETE (S, I, N)	S, I, N	S sem N caracteres a partir de I
INSERT (S1, S, I)	S1, S, I	S com S1 a partir de I
STR (V, S)	V	S com caracteres do valor V
VAL (S, V, N)	S	V valor de S; N código de erro

Exemplos onde X = 'PAIS', Y = 'CAPITAL', Z = 'BRASILIA'.

- a) DELETE (Z, 7, 2) => Z = 'BRASIL'
- b) INSERT (Y, Z, 1) => Z = 'CAPITALBRASIL'
- c) STR (15, W) => W = '15'
- d) VAL (W, A, N) => A = 15
- e) VAL ('15x3', A, N) => N = 3 {erro}

¹⁰ S, S1 = string; I, N = integer; Ch = char;

¹¹ V = integer ou real.

<i>Funções variáveis escalares</i> ¹²	
Nome	Devolve
PRED (X)	O antecessor de X
SUCC (X)	O sucessor de X
INC (X)	Incrementa a variável X com 1. $X := X + 1$
INC (X, 2)	Incrementa a variável X com 2. $X := X + 2$
DEC (X)	Decrementa a variável X com 1. $X := X - 1$
DEC (X, 5)	Decrementa a variável X com 5. $X := X - 5$

Exemplos onde:

$x := 1;$

a) $y := \text{SUCC}(x) \Rightarrow y = 2$

b) $y := \text{PRED}(x) \Rightarrow y = 0$

c) $\text{INC}(x) \Rightarrow x = 2$

d) $\text{DEC}(x, 4) \Rightarrow x = -2$

e) $\text{INC}(x, 3) \Rightarrow x = 1$

f) $\text{DEC}(x) \Rightarrow x = 0$

- ✓ **Printer** – define um arquivo Lst do tipo texto, associado à impressora do computador, permitindo a sua utilização com os comandos **write** (Lst, ...) e **writeln** (Lst, ...).
- ✓ **Crt** – oferece diversos recursos para utilizar o vídeo no modo texto, o teclado e o som do computador.

<i>Procedimentos predefinidos</i>		
Nome	Recebe	Resultado
CLREOL		Limpa linha, do cursor até fim
CLRSCR		Limpa tela e põe cursor em 1,1
DELAY (N)	N	Espera N milissegundos
DELLINE		Apaga linha do cursor e move linhas abaixo para cima
GOTOXY (X, Y)	X, Y	Põe cursor na coluna X, linha Y
INSLINE		Insere linha vazia e move linhas abaixo para baixo
TEXTBACKGROUND (C)	C	Põe cor de fundo = C (0 – 7)
TEXTCOLOR (C)	C	Põe cor de frente = C (0 – 15)
WINDOW (X1, Y1, X2, Y2)	X1, Y1, X2, Y2	Abre janela com canto superior esquerdo em (X1, Y1), inferior direito em (X2, Y2)

Observação: quando se abre uma janela, as funções WhereX e WhereY e os procedimentos ClrEol, ClrScr, DelLine, GotoXY, InsLine, TextBackground e TextColor referem-se a esta janela. Idem para os procedimentos read, readln, write e writeln da unidade System.

Os parâmetros dos comandos TextBackground e TextColor pode ser um número ou o nome da cor em inglês, representados na tabela abaixo.

¹² X é uma variável do tipo escalar.

CORES				
TEXTBACKGROUND COR DE FUNDO		TEXTCOLOR COR DO TEXTO		
Nº	Cor obtida	Nº	Cor obtida	COR
0	Preto	0	Preto	<i>Black</i>
1	Azul	1	Azul	<i>Blue</i>
2	Verde	2	Verde	<i>Green</i>
3	Anil escuro	3	Anil	<i>Cyan</i>
4	Vermelho	4	Vermelho	<i>Red</i>
5	Magenta (rosa)	5	Magenta (rosa)	<i>Magenta</i>
6	Laranja	6	Marron	<i>Brown</i>
7	Cinza claro (branco)	7	Cinza claro (branco)	<i>LightGray</i>
		8	Cinza escuro	<i>DarkGray</i>
		9	Azul claro	<i>LightBlue</i>
		10	Verde claro	<i>LightGreen</i>
		11	Anil claro	<i>LightCyan</i>
		12	Vermelho claro	<i>LightRed</i>
		13	Magenta claro (rosa claro)	<i>LightMagenta</i>
		14	Amarelo	<i>Yellow</i>
		15	Branco	<i>White</i>

Funções predefinidas		
Nome	Retorna	Tipo
KEYPRESSED	True, se foi apertada alguma tecla	Boolean
READKEY	Caractere digitado, sem eco no vídeo	Char
WHEREX	Coluna em que se encontra o cursor	Byte
WHEREY	Linha em que se encontra o cursor	Byte

- ✓ **Dos** – contém constantes, tipos, variáveis, funções e procedimentos relacionados com o sistema operacional.

Ex.: obter data e hora; espaço no disco rígido; espaço livre no disco rígido, etc.

- ✓ **Overlay** – Normalmente, um programa deve estar totalmente na memória para poder ser executado. A unidade Overlay permite rodar programas muito grandes que de outro modo não caberiam na memória.
- ✓ **Graph** – permite utilizar o vídeo em modo gráfico, define um grande número de constantes, tipos, variáveis, funções e procedimentos.

9.A. Exercícios (Anexo A)

1. Sendo P, Q, R variáveis inteiras e S, variável real, cujos valores, num determinado momento da execução do programa, são, respectivamente, 2, 3, 12 e 4.5, quais os valores de cada uma das seguintes expressões aritméticas?
 - a) $100 * (Q \text{ div } P) + R$
 - b) $\text{Frac} (S)$
 - c) $P * (R \text{ mod } 5) - Q / 2$
 - d) $\text{Sqrt} (R - \text{Sqr} (P)) + \text{Round} (S)$
 - e) $\text{Int} (S)$
 - f) $R \text{ mod} (P + 1) - Q * R$

2. Dadas as variáveis inteiras X, Y, Z contendo os valores 2, 5, 9, respectivamente; a variável Nome do tipo literal contendo 'Maria' e a variável lógica Sim contendo o valor False, resolva as expressões lógicas abaixo:
 - a) $(X + Y > Z) \text{ and} (\text{Nome} = \text{'Maria'})$
 - b) $\text{Sim or} (Y \geq Z)$
 - c) $\text{not Sim and} (Z \text{ div } Y + 1 = X)$
 - d) $(\text{Nome} = \text{'Jorge'}) \text{ and Sim or} (\text{Sqr} (X) < Z + 10)$

3. Dadas as variáveis do tipo string R, S, T contendo os valores 'COMPUTADOR', 'FACULDADE', 'PASCAL', respectivamente; e a variável X = 10; fornecer os resultados das expressões a seguir:
 - a) $\text{Copy} (R, 8, 3)$
 - b) $\text{Pos} (\text{'A'}, R)$
 - c) $\text{Length} (R)$
 - d) $\text{Concat} (\text{Copy} (R, 1, 3), \text{' '}, \text{Copy} (R, 8, 3))$
 - e) $\text{Pos} (\text{'A'}, T)$
 - f) $\text{Concat} (\text{'I'}, \text{Copy} (S, 6, 4))$
 - g) $\text{Str} (X, C)$
 - h) $\text{Val} (\text{'1234'}, A, N)$
 - i) $\text{Length} (S)$
 - j) $\text{Val} (\text{'12x3'}, A, N)$
 - k) $\text{Insert} (\text{'S INTEGRADAS'}, S, 10)$
 - l) $\text{Delete} (S, 12, 10)$

10. Anexo B - Menus e Erros

O Turbo Pascal possui um **menu de comandos** com dez opções que possibilitam executar diversas tarefas operacionais. Cada um dos menus e suas opções são descritas a seguir:

- ✓ **File** – esta opção possibilita executar operações de controle com arquivos.
 - **New** – criar um novo programa.
 - **Open** – abrir um programa existente.
 - **Save** – salvar um programa em disco.
 - **Save as** – salvar um programa em disco com outro nome.
 - **Save all** – salvar todos os programas modificados.
 - **Change dir** – mudar o diretório de trabalho.
 - **Print** – imprimir o programa da janela ativa.
 - **Print setup** – configurar o uso de outra impressão.
 - **DOS shell** – sair temporariamente para o sistema operacional.
 - **Exit** – finalizar a execução do Turbo Pascal.

- ✓ **Edit** – esta opção possibilita executar operações do editor do programa, sendo possível remover, movimentar e copiar textos que estejam selecionados¹³.
 - **Undo** – desfazer uma operação com texto.
 - **Redo** – refazer uma operação com texto.
 - **Cut** – remover um texto previamente selecionado, enviando-o para área de transferência.
 - **Copy** – copiar um texto selecionado do editor para uma área de transferência.
 - **Paste** – copiar um texto da área de transferência para o editor.
 - **Clear** – remover o texto selecionado sem transferi-lo para a área de transferência.
 - **Show Clipboard** – apresentar o conteúdo existente na área de transferência.

- ✓ **Search** – Esta opção possibilita executar operações de busca, troca e deslocamento dentro de um programa.
 - **Find** – localizar uma seqüência de caracteres em um programa.
 - **Replace** – substituir uma seqüência de caracteres por outra.
 - **Search again** – repetir a última busca.
 - **Go to line number** – posicionar-se em uma determinada linha do programa.
 - **Show last compiler error**- mostrar o último erro de compilação, quando ocorrer.
 - **Find error** – posicionar-se na última posição de erro encontrada pelo compilador.
 - **Find procedure** – localizar uma sub-rotina dentro do programa no momento de depuração de um programa.

- ✓ **Run** – Esta opção possibilita colocar em execução o programa da janela ativa.
 - **Run** – compilar e executar o programa.
 - **Step over** – rodar o programa passo a passo com exceção das sub-rotinas existentes.
 - **Trace into** – rodar o programa passo a passo inclusive as suas sub-rotinas.

¹³ Para selecionar um texto: pressione a tecla Shift enquanto movimenta as “setinhas”; ou utilize o mouse.

- **Go to cursor** – tomar o programa até a posição em que está o cursor.
 - **Program reset** – interromper a execução de um programa durante sua depuração.
 - **Parameters** – efetuar a passagem de parâmetros.
- ✓ **Compile** – Esta opção possibilita compilar o programa.
- **Compiler** – compilar o programa da janela ativa.
 - **Make** – recompilar apenas os programas alterados.
 - **Build** – recompilar todos os programas.
 - **Destination** – determinar se o programa será compilado somente em memória ou disco.
 - **Primary file** – determinar numa lista de arquivos qual será o arquivo principal que será carregado primeiro no processo de compilação.
 - **Clear primary file** – limpar o arquivo anteriormente configurado como o arquivo primário.
 - **Information** – obter informações a respeito da última compilação executada.
- ✓ **Debug** – Esta opção possibilita depurar¹⁴ o programa, para facilitar a localização de erros lógicos.
- **Breakpoint** – colocar ou retirar um ponto de parada (exame) quando da depuração de um programa.
 - **Call stack** – apresentar uma janela com a seqüência de chamadas efetuadas de sub-rotinas.
 - **Register** – visualizar a janela de registradores da CPU.
 - **Watch** – abrir a janela de acompanhamento de valores nas variáveis do programa.
 - **Output** – abrir uma janela para apresentar as telas de saída do programa em execução ou depuração.
 - **User Screen** – exibir na sua totalidade a tela do usuário.
 - **Evaluate / modify** – permite efetuar a avaliação de expressões, constantes ou variáveis.
 - **Add watch** – possibilidade de se incluir expressões na tela de vigia.
 - **Add breakpoint** – permite a inclusão de um ponto de parada quando for executado o programa.
- ✓ **Tools** – Esta opção possibilita a utilização de ferramentas configuradas pelo usuário.
- **Messagens** – abrir uma janela para a apresentação de mensagens.
 - **Go to next** – visualizar a próxima mensagem da janela de mensagens.
 - **Go to previous** – visualizar a mensagem anterior da janela de mensagens.
 - **Grep** – efetuar a busca de seqüências de caracteres em programas gravados através do utilitário GREP.EXE.
- ✓ **Options** – Esta opção permite configurar a forma de trabalho do ambiente do Turbo Pascal.
- **Compiler** – alterar o estado das diretivas de compilação.
 - **Memory sizes** – definir o tamanho de memória.
 - **Linker** – a possibilidade de usar ou não o recurso de link.
 - **Debugger** – estabelecer critérios de depuração.
 - **Directories** – determinar os diretórios de trabalho.

¹⁴ Procurar, detectar e corrigir erros de lógica em programas.

- **Browser** – especifica configurações globais do visualizador.
 - **Tools** – efetuar a manutenção do menu de ferramentas.
 - **Environment** – efetuar mudanças no ambiente de trabalho (Preferências, Editor, Mouse, Inicialização e Cores) do Turbo Pascal conforme necessidade do usuário.
 - **Open** – a abertura do arquivo de configuração.
 - **Save** – gravar o arquivo de configuração.
 - **Save as** – gravar o arquivo de configuração com outro nome.
- ✓ **Window** – Esta opção possibilita o controle das janelas que estejam abertas.
- **Tile** – ordenar as janelas lado a lado.
 - **Cascade** – ordenar as janelas em cascata.
 - **Close all** – fechar todas as janelas.
 - **Refresh display** - redesenhar a janela ativa.
 - **Size / Move** – movimentar ou alterar o tamanho de uma janela.
 - **Zoom** – alterar o tamanho de uma janela para o tamanho máximo ou para o tamanho preestabelecido.
 - **Next** – selecionar a próxima janela como ativa.
 - **Previous** – selecionar a janela anterior como ativa.
 - **Close** – fechar a janela ativa.
 - **List** – apresentar a listagem das janelas que estejam abertas.
- ✓ **Help** – Esta opção permite executar o modo de ajuda do Turbo Pascal. O modo de ajuda poderá ser executado de qualquer parte do programa com a tecla de função F1 ou Ctrl + F1 para visualizar explicações de instruções que estejam marcadas com o posicionamento do cursor sobre elas.
- **Contents** – apresenta o sumário.
 - **Index** – apresenta todas as instruções em ordem alfabética.
 - **Error messages** – apresenta uma lista de erros de execução e de compilação.
 - **Outras formas ...**

Os programas podem conter três ***tipos de erros***:

- ✓ ***Erro de sintaxe*** – ocorre quando se viola uma regra da linguagem. Ao encontrar um destes erros, o Turbo Pascal interrompe a compilação do programa, dá uma mensagem com o número e a descrição do erro, e deixa o cursor piscando no item do programa que permitiu a detecção do erro. Deve-se observar que este item pode não estar errado, mas apenas permitiu detectar o erro. O Turbo Pascal não executa um programa enquanto houver erros de sintaxe.
- ✓ ***Erro de semântica*** – é um erro que o programa detecta durante a sua execução, interrompendo-a; por exemplo, quando o programa é levado a ler um arquivo inexistente, a tirar a raiz quadrada de um número negativo, a dividir por zero, etc. Ocorrendo um erro de semântica, o Turbo Pascal interrompe a execução do programa e mostra a descrição do erro, com o cursor piscando no início da linha em que ele foi detectado (se estiver sendo utilizado o ambiente do Turbo Pascal).

- ✓ **Erro de lógica** – aparece em programas, sintática e semanticamente corretos, que, apesar disso, não produzem os resultados desejados. São devidos a instruções corretas, para o computador, mas que não são apropriadas para a solução lógica do problema que motivou o programa. A correção dos erros de lógica é muitas vezes trabalhosa e demorada. Para ajudar a depuração de erros de lógica, o Turbo Pascal oferece diversos recursos que permitem ver e acompanhar o que ocorre durante a execução de um programa.

Erros de sintaxe e semântica não são preocupantes, pois o próprio computador os localiza e os identifica.

Lista dos erros de Execução:

- 1 - Invalid function number
- 2 - File not found
- 3 - Path not found
- 4 - Too many open files
- 5 - File access denied
- 6 - Invalid file handle
- 12 - Invalid file access code
- 15 - Invalid drive number
- 16 - Cannot remove current directory
- 17 - Cannot rename across drives
- 18 - No more files
- 100 - Disk read error
- 101 - Disk write error
- 102 - File not assigned
- 103 - File not open
- 104 - File not open for input
- 105 - File not open for output
- 106 - Invalid numeric format
- 150 - Disk is write-protected
- 151 - Bad drive request struct length
- 152 - Drive not ready
- 154 - CRC error in data
- 156 - Disk seek error
- 157 - Unknown media type
- 158 - Sector Not Found
- 159 - Printer out of paper
- 160 - Device write fault
- 161 - Device read fault
- 162 - Hardware failure
- 200 - Division by zero
- 201 - Range check error
- 202 - Stack overflow error
- 203 - Heap overflow error
- 204 - Invalid pointer operation
- 205 - Floating point overflow
- 206 - Floating point underflow

- 207 - Invalid floating point operation
- 208 - Overlay manager not installed
- 209 - Overlay file read error
- 210 - Object not initialized
- 211 - Call to abstract method
- 212 - Stream registration error
- 213 - Collection index out of range
- 214 - Collection overflow error
- 215 - Arithmetic overflow error
- 216 - General Protection fault

Lista dos erros de Compilação:

- 1 - Out of memory
- 2 - Identifier expected
- 3 - Unknown identifier
- 4 - Duplicate identifier
- 5 - Syntax error
- 6 - Error in real constant
- 7 - Error in integer constant
- 8 - String constant exceeds line
- 10 - Unexpected end of file
- 11 - Line too long
- 12 - Type identifier expected
- 13 - Too many open files
- 14 - Invalid file name
- 15 - File not found
- 16 - Disk full
- 17 - Invalid compiler directive
- 18 - Too many files
- 19 - Undefined type in pointer def
- 20 - Variable identifier expected
- 21 - Error in type
- 22 - Structure too large
- 23 - Set base type out of range
- 24 - File components may not be files or objects
- 25 - Invalid string length
- 26 - Type mismatch
- 27 - Invalid subrange base type
- 28 - Lower bound greater than upper bound
- 29 - Ordinal type expected
- 30 - Integer constant expected
- 31 - Constant expected
- 32 - Integer or real constant expected
- 33 - Pointer Type identifier expected
- 34 - Invalid function result type
- 35 - Label identifier expected
- 36 - BEGIN expected

- 37 - END expected
- 38 - Integer expression expected
- 39 - Ordinal expression expected
- 40 - Boolean expression expected
- 41 - Operand types do not match
- 42 - Error in expression
- 43 - Illegal assignment
- 44 - Field identifier expected
- 45 - Object file too large
- 46 - Undefined external
- 47 - Invalid object file record
- 48 - Code segment too large
- 49 - Data segment too large
- 50 - DO expected
- 51 - Invalid PUBLIC definition
- 52 - Invalid EXTRN definition
- 53 - Too many EXTRN definitions
- 54 - OF expected
- 55 - INTERFACE expected
- 56 - Invalid relocatable reference
- 57 - THEN expected
- 58 - TO or DOWNTO expected
- 59 - Undefined forward
- 61 - Invalid typecast
- 62 - Division by zero
- 63 - Invalid file type
- 64 - Cannot read or write variables of this type
- 65 - Pointer variable expected
- 66 - String variable expected
- 67 - String expression expected
- 68 - Circular unit reference
- 69 - Unit name mismatch
- 70 - Unit version mismatch
- 71 - Internal stack overflow
- 72 - Unit file format error
- 73 - IMPLEMENTATION expected
- 74 - Constant and case types do not match
- 75 - Record or object variable expected
- 76 - Constant out of range
- 77 - File variable expected
- 78 - Pointer expression expected
- 79 - Integer or real expression expected
- 80 - Label not within current block
- 81 - Label already defined
- 82 - Undefined label in preceding statement part
- 83 - Invalid @ argument
- 84 - UNIT expected
- 85 - ";" expected
- 86 - ":" expected

- 87 - "," expected
- 88 - "(" expected
- 89 - ")" expected
- 90 - "=" expected
- 91 - ":=" expected
- 92 - "[" or "(" expected
- 93 - "]" or ")" expected
- 94 - "." expected
- 95 - ".." expected
- 96 - Too many variables
- 97 - Invalid FOR control variable
- 98 - Integer variable expected
- 99 - Files and procedure types are not allowed here
- 100 - String length mismatch
- 101 - Invalid ordering of fields
- 102 - String constant expected
- 103 - Integer or real variable expected
- 104 - Ordinal variable expected
- 105 - INLINE error
- 106 - Character expression expected
- 107 - Too many relocation items
- 108 - Overflow in arithmetic operation
- 109 - No enclosing FOR, WHILE or REPEAT statement
- 110 - Debug information table overflow
- 112 - CASE constant out of range
- 113 - Error in statement
- 114 - Cannot call an interrupt procedure
- 116 - Must be in 8087 mode to compile this
- 117 - Target address not found
- 118 - Include files are not allowed here
- 119 - No inherited methods are accessible here
- 121 - Invalid qualifier
- 122 - Invalid variable reference
- 123 - Too many symbols
- 124 - Statement part too large
- 126 - Files must be var parameters
- 127 - Too many conditional symbols
- 128 - Misplaced conditional directive
- 129 - ENDIF directive missing
- 130 - Error in initial conditional defines
- 131 - Header does not match previous definition
- 133 - Cannot evaluate this expression
- 134 - Expression incorrectly terminated
- 135 - Invalid format specifier
- 136 - Invalid indirect reference
- 137 - Structured variables are not allowed here
- 138 - Cannot evaluate without System unit
- 139 - Cannot access this symbol
- 140 - Invalid floating point operation

- 141 - Cannot compile overlays to memory
- 142 - Pointer or procedural variable expected
- 143 - Invalid procedure or function reference
- 144 - Cannot overlay this unit
- 145 - Too many nested scopes
- 146 - File access denied
- 147 - Object type expected
- 148 - Local object types are not allowed
- 149 - VIRTUAL expected
- 150 - Method identifier expected
- 151 - Virtual constructors are not allowed
- 152 - Constructor identifier expected
- 153 - Destructor identifier expected
- 154 - Fail only allowed within constructors
- 155 - Invalid combination of opcode and operands
- 156 - Memory reference expected
- 157 - Cannot add or subtract relocatable symbols
- 158 - Invalid register combination
- 159 - 286/287 instructions are not enabled
- 160 - Invalid symbol reference
- 161 - Code generation error
- 162 - ASM expected
- 163 - Duplicate dynamic method index
- 164 - Duplicate resource identifier
- 165 - Duplicate or invalid export index
- 166 - Procedure or function identifier expected
- 167 - Cannot export this symbol
- 168 - Duplicate export name
- 169 - Executable file header too large
- 170 - Too many segments

10.A. Exercícios (Anexo B)

1. No Turbo Pascal, qual o menu e a opção que deve ser selecionada para executar os seguintes comandos:
 - a) Abrir um novo arquivo de trabalho: _____
 - b) Acionar consulta *on-line*: _____
 - c) Adicionar ou retirar o breakpoint da linha corrente do arquivo-fonte no editor: _____
 - d) Adicionar um item na Janela de Observação: _____
 - e) Alternar para o próximo arquivo: _____
 - f) Alternar para o arquivo anterior: _____
 - g) Compilar e executar um programa: _____

- h) Compilar um programa: _____
- i) Criar um arquivo executável: _____
- j) Desfazer uma operação com texto: _____
- k) Executar o programa uma linha de cada vez passando direto sobre a chamada de uma rotina, como uma única linha de código: _____
- l) Executar o programa uma linha de cada vez: _____
- m) Fechar um arquivo aberto: _____
- n) Finalizar a execução do Turbo Pascal: _____
- o) Obter o índice geral com palavras-chaves: _____
- p) Refazer uma operação com texto: _____
- q) Salvar todos os arquivos modificados: _____
- r) Salvar um arquivo no disco/disquete: _____
- s) Salvar um arquivo no disco/disquete com outro nome: _____
- t) Selecionar um arquivo já existente: _____

2. Quais os tipos de erros que podem existir em um programa? Explique resumidamente cada um.

3. O que o Turbo Pascal faz ao encontrar cada um dos tipos de erros?

4. Defina *breakpoint*.

5. O que é depuração e compilação?

6. Identifique os seguintes erros descrevendo sua indicação em inglês e português:

Execução

150 –

200 –

Compilação

02 –

03 –

04 –

05 –

26 –

36 –

37 –

42 –

50 –

54 –

57 –

58 –

62 –

64 –

85 –

86 –

87 –

88 –

89 –

90 –

91 –

94 –

95 –

97 –

113 –

11. Anexo C - Resumo das teclas utilizadas no TP 7.0

TECLA(S)	FUNÇÃO
ESC	Sair do sistema de auxílio ou do menu aberto.
F1	Apresentar o Help.
F2	Salvar o programa ativo.
F3	Localizar e abrir um programa.
F4	Executar o programa até o local que estiver o cursor.
F5	Maximizar ou restaurar o tamanho da janela ativa.
F6	Alternar para a próxima janela.
F7	Executar o programa linha por linha, inclusive sub-rotinas.
F8	Executar o programa linha por linha, sem entrar em sub-rotinas.
F9	Compilar o programa principal e todas <i>Units</i> utilizadas que foram modificadas.
F10	Ativar o menu do Turbo Pascal.
ALT + 0	Mostrar uma lista de todas as janelas abertas.
ALT + X	Finalizar a execução do Turbo Pascal.
ALT + BKSP	Desfazer a operação anterior do editor.
ALT + ENTER	Maximizar ou restaurar a janela do Turbo Pascal.
ALT + F1	Reapresentar a última tela de Help on-line visualizada.
ALT + F3	Fechar a janela ativa.
ALT + F5	Alternar entre a tela do Ambiente Integrado e a tela gerada pela execução do programa.
ALT + F9	Compilar o programa principal.
ALT + 'letra'	Apresentar opções do menu acionado.
CTRL + F1	Obter auxílio sobre um determinado item da linguagem. (O cursor deve estar posicionado sobre o item)
CTRL + F2	Parar a execução da sessão de depuração e limpar a memória alocada pelo programa.
CTRL + F3	Mostrar a seqüência de sub-rotinas chamadas pelo programa para obter a sub-rotina que está sendo executada no momento. É apresentado o nome da sub-rotina e os valores dos parâmetros passados.
CTRL + F4	Avaliar uma variável ou expressão e mostrar ou modificar o valor.
CTRL + F5	Modificar o tamanho ou posição da janela ativa.
CTRL + F7	Adicionar variáveis para visualização de seu conteúdo na depuração.
CTRL + F8	Adicionar ou retirar breakpoints para depuração.
CTRL + F9	Compilar e executar o programa.
CTRL + Pause	Finalizar a execução do programa e retornar ao Turbo Pascal.
CTRL + INS	Copiar o texto selecionado da área de transferência.
CTRL + DEL	Deletar o texto selecionado.
SHIFT + F1	Mostrar o índice do Help on-line.
SHIFT + F6	Alternar para a janela ativa anterior.
SHIFT + DEL	Remover o texto selecionado e coloca-lo na área de transferência.
SHIFT + INS	Inserir o texto selecionado da área de transferência na posição do cursor.

11.A. Exercícios (Anexo C)

1. No Turbo Pascal, qual(is) tecla(s) acionar para executar os seguintes comandos:

I – Manipulando arquivos

- a) Alternar para o arquivo anterior aberto: _____
- b) Alternar para o próximo arquivo aberto: _____
- c) Desfazer uma operação com texto: _____
- d) Fechar um arquivo aberto: _____
- e) Finalizar a execução do Turbo Pascal: _____
- f) Salvar um arquivo no disco/disquete: _____
- g) Selecionar um arquivo já existente: _____

II – Compilando e executando programas

- a) Compilar e executar um programa: _____
- b) Compilar um programa: _____
- c) Compilar um programa e suas units: _____
- d) Finalizar a execução do programa e retornar para o Turbo Pascal: _____

III – Sistema de auxílio

- a) Acionar consulta *on-line*: _____
- b) Mover página abaixo na janela de help: _____
- c) Mover página acima na janela de help: _____
- d) Obter auxílio sobre um item da linguagem: _____
- e) Obter o índice geral com palavras-chaves: _____
- f) Sair do sistema de auxílio: _____

IV – Depurar programas

- a) Adicionar ou retirar o breakpoint da linha corrente do arquivo-fonte no editor: _____
- b) Adicionar um item na Janela de Observação: _____
- c) Editar um item na Janela de Observação: _____
- d) Executar o programa uma linha de cada vez: _____
- e) Executar o programa uma linha de cada vez passando direto sobre a chamada de uma rotina, como uma única linha de código: _____
- f) Permitir a troca entre a tela do Ambiente Integrado e a tela gerada pela execução do programa: _____