Guia do Operador

TURBO PASCAL

COMANDOS BÁSICOS

McGraw Hill

Roberto Bertini Renzetti

85.00
Em

TURBO PASCAL – GUIA DO OPERADOR

Comandos Básicos

Dados de Catalogação na Publicação
(CIP) Internacional
(Câmara Brasileira do Livro, SP, Brasil)

Renzetti, Roberto Bertini.
R334t Turbo Pascal : guia do operador, comandos básicos / Roberto Bertini Renzetti. São Paulo : McGraw-Hill, 1987.

1. Turbo Pascal (Linguagem de programação para computadores) I. Título.

86-2042 CDD-001.6424

Índices para catálago sistemático:

1. Linguagem de programação : Computadores : Processamento de dados 001.6424
2. Turbo Pascal : Linguagem de programação : Processamento de dados 001.6424

TURBO PASCAL – GUIA DO OPERADOR
COMANDOS BÁSICOS

Roberto Bertini Renzetti
Consultor e Docente de Informática

McGraw-Hill
São Paulo
Rua Tabapuã, 1.105, Itaim-Bibi
CEP 04533
(011) 881-8604 e (011) 881-8528

Rio de Janeiro • *Lisboa* • *Porto* • *Bogotá* •
Buenos Aires • *Guatemala* • *Madrid* • *México* •
New York • *Panamá* • *San Juan* • *Santiago*

Auckland • Hamburg • Kuala Lumpur • London •
Milan • Montreal • New Delhi • Paris • Singapore •
Sydney • Tokyo • Toronto

Turbo Pascal – Guia do Operador*

Editor: Milton Mira de Assumpção Filho
Coordenadora de Revisão: Daisy Pereira Daniel
Supervisor de Produção: José Roberto Petroni
Capa: Neo Comunicação

* Baseado na obra *Using Turbo Pascal*, de Steve Wood.

AGRADECIMENTOS

À minha família pelo amor e companhia.

Aos meus avós, em especial, pelo carinho.

Ao grande amigo Luciano, pela amizade, ensinamentos, incentivo e oportunidades.

Ao Pierluigi, pela "primeira chance" na Informática.

Ao Eduardo, pelas "dicas" sobre IBM's.

E, finalmente, a todos meus amigos e amigas das turmas do Mackenzie, do Senac Informática e da MicroHobby.

SUMÁRIO

INTRODUÇÃO

Adotamos a versão Turbo Pascal 3.0, da empresa Borland International, por ser uma das mais vendidas e conhecidas mundialmente, além de respeitar o padrão ISO para o Pascal e conter algumas implementações que auxiliarão o programador.

O Turbo Pascal consiste, basicamente, em três módulos: um editor, um compilador e um "depurador". Graças ao inter-relacionamento dos módulos e à interação amigável como o programador, você será capaz de criar aplicativos e programas do mais alto nível.

Este guia é destinado a auxiliar o programador novato ou veterano, que necessite de um esclarecimento rápido a alguma dúvida. Divide-se em duas partes: a primeira, que contém uma introdução aos comandos e conceitos básicos e a segunda, com tabelas de consulta rápida, agrupadas por assuntos.

Outras informações como mensagens de erro, diretivas de compilação e tabela ASCII encontram-se em apêndices no fim do guia.

NOTAÇÃO ADOTADA NESTE GUIA

[ ]	Toda sintaxe que estiver entre colchetes é opcional.
carac	Indica uma variável tipo caractere.
ident	Identificador.
str	Indica uma variável tipo string ("cadeia de caracteres alfa-numéricos").
var	Indica uma variável real ou inteira.
var_arq	Indica uma variável tipo arquivo.
var_arqtxt	Indica uma variável tipo arquivo-texto.
var_escalar	Indica uma variável escalar.
var_int	Indica uma variável inteira.
var_pntr	Indica uma variável tipo ponteiro (pointer)
var_real	Indica uma variável real.

Observação:

1) As exceções estão comentadas.
2) Esta notação somente não foi usada nos comandos do sistema para IBM PC.

PARTE 1

CONCEITOS BÁSICOS DO TURBO PASCAL

CARACTERÍSTICAS DA LINGUAGEM PASCAL

– Linguagem de uso geral, criada por Niklaus Wirth.

– Possui sintaxe rígida, mas que favorece o aprendizado de uma metodologia de programação.

– É uma linguagem estruturada, ou seja, composta basicamente de funções e procedimentos que podem ser combinados criando "subprogramas". Os subprogramas também são combinados criando, assim, um programa.

CARACTERÍSTICAS DO TURBO PASCAL

– Mantém o padrão estabelecido mundialmente para Pascal, além de fornecer algumas extensões.

– Possui um editor, um compilador e um "depurador" (programa para achar erros).

– O editor possui o mesmo potencial e comandos do processador de textos WORDSTAR (mas as teclas de controle podem ser alteradas, se desejado).

– O compilador traduz o programa-fonte (programa em Pascal) num compacto e eficiente programa-objeto (programa em linguagem de máquina).

– O depurador emite mensagens de erros claras e elucidativas.

– O programa-fonte é quase inteiramente compatível com outros micros. Assim, se você quiser passar um programa escrito num micro da linha APPLE para um micro IBM PC é só redigitar o programa-fonte no outro micro (observando somente se não são usadas rotinas e comandos intrínsecos à máquina e/ou sistema operacional).

– Você pode gravar uma determinada rotina para a utilização em outros programas-fonte de Turbo Pascal. Com isso, criará uma "biblioteca" de rotinas.

CARREGANDO O TURBO PASCAL

Antes de usar o Turbo Pascal pela primeira vez, faça uma cópia de segurança (backup) e guarde o original em um lugar seguro. Use sempre o backup.

Se o seu disco-mestre não estiver devidamente instalado, use o instalador do Turbo Pascal, o TINST, antes de continuar.

– Ligue o computador com a cópia no drive A.

– Digite TURBO e tecle enter

– Aparecerá, no vídeo, a pergunta "Load Error Messages?" (Carregar mensagens de erro?). Responda com Y (sim) ou N (não): Y mostrará código de erro e mensagem escrita; N indicará somente o código do erro.

– O menu principal do Turbo Pascal será apresentado.

OPÇÕES DO MENU PRINCIPAL DO TURBO PASCAL

Logged drive: (drive corrente)

Indica qual drive está sendo usado. Digitando-se L, podemos trocar o drive em uso. Responda o novo drive com a letra desejada seguida de dois pontos (:).

Work file: (arquivo de trabalho)

Seleciona o arquivo de trabalho a ser usado no Edit, Compile, Run, Execute e Save. Ao digitar W, você deverá responder o nome do arquivo seguindo a regra. nome com no máximo 8 caracteres (sendo

o primeiro obrigatoriamente letra) seguido, opcionalmente, por um ponto e um tipo de arquivo (com 3 letras no máximo). Turbo Pascal assumirá a terminação.PAS caso você não coloque o ponto e o tipo de arquivo.

Main file: (arquivo principal)

Você deve digitar M para definir o arquivo principal quando estiver trabalhando com a diretiva de compilação $I. O arquivo principal deve ser aquele que contém esta diretiva.

Edit (edição)

Ao digitar E, você entra no editor para manusear o arquivo definido pela opção W.

Compile (compilar)

Este comando (C) ativa o compilador. O arquivo de trabalho será compilado se não existir um arquivo principal. O modo de compilação é especificado pela opção O.

Run (rode, corra)

Ativa (roda) um programa na memória ou um programa-objeto do disco (se definido pela opção 0).

Save (salvar)

Grava o arquivo de trabalho em disco. Cria um arquivo .BAK, se existir uma versão antiga já gravada.

Execute (execute)

Permite rodar outros programas, sem sair do Turbo Pascal. Não existe na versão para IBM PC. Deve-se digitar X.

Dir (diretório)

Ao digitar D, mostra o diretório do disco. Você pode especificar uma máscara (ou seja, usar um caractere que substitua vários caracteres de uma vez). Exemplo: *.PAS vai mostrar somente os arquivos-fonte em pascal do seu disco.

Quit (abandona)

Abandona o Turbo Pascal voltando ao sistema.

Compiler Options (opções do compilador)

Permite alterar as opções de compilação. Ao digitar-se O, passamos para este submenu:

Memory (memória)

O programa-objeto será gerado na própria memória. Digite M.

Com-file (arquivo tipo .COM)

O programa-objeto será gerado e gravado no disco com terminação .COM. Digite C.

cHn-file (arquivo tipo .CHN)

O programa-objeto será gerado no disco com terminação .CHN. Não pode ser executado diretamente do sistema operacional. Deve ser executado a partir de um programa pascal .COM ou .CHN. Digite H.

Find run-time errors (achar erros na execução)

Serve para achar erros na execução de programas com terminação .COM ou .CHN. Digite F.

Quit (abandona)

Volta ao menu principal ao digitar-se Q.

COMANDOS DE EDIÇÃO

Estes são os comandos-padrão quando editamos um programa-fonte (opção E do menu principal). Qualquer um destes comandos pode ser alterado com o TINST.

Controle do cursor	Teclas	IBM PC (opcional)
um caractere à esq.	CTRL-S	seta esquerda
um caractere à dir.	CTRL-D	seta direita
uma linha para cima	CTRL-E	seta para cima
uma linha para baixo	CTRL-X	seta para baixo
uma tela para cima	CTRL-R	PgUp
uma tela para baixo	CTRL-C	PgDn
uma palavra à esquerda	CTRL-A	Ctrl-seta esq.
uma palavra à direita	CTRL-F	Ctrl-seta dir.
scroll para cima	CTRL-W	----
scroll para baixo	CTRL-Z	----
começo da linha à esquerda	CTRL-QS	Home
fim da linha à direita	CTRL-QD	End
topo da tela	CTRL-QE	Ctrl-Home
fim da tela	CTRL-QX	Ctrl-End
topo do arquivo	CTRL-QR	Ctrl-PgUp
fim do arquivo	CTRL-QC	Ctrl-PgDn
começo do bloco	CTRL-QB	----

Controle do cursor	**Teclas**	**IBM PC (opcional)**
fim do bloco	CTRL-QK	----
posição anterior	CTRL-QP	----
tabulação	CTRL-I ou Tab	Tab
tabulação automática (liga/desliga)	CTRL-QI	----

Inserção ou deleção	**Teclas**	**IBM PC (opcional)**
inserção (lig./desl.)	CTRL-V	INS
deleção ou backspace (apaga caracteres à esquerda do cursor)	DEL ou Backspace	Backspace
apaga caractere sob cursor	CTRL-G	DEL
apaga palavra à direita	CTRL-T	----
insere linha	CTRL-N	----
apaga linha	CTRL-Y	----
apaga até fim da linha	CTRL-QY	----

Manuseio de blocos	**Teclas**	**IBM PC (opcional)**
marcar começo de bloco	CTRL-KB	F7
marcar fim do bloco	CTRL-KK	F8

Manuseio de blocos	**Teclas**	**IBM PC (opcional)**
marcar uma palavra	CTRL-KT	----
mostrar/esconder bloco	CTRL-KH	----
copiar bloco marcado	CTRL-KC	----
mover bloco marcado	CTRL-KV	----
ler bloco de um arquivo	CTRL-KR	----
escrever bloco num arquivo	CTRL-KW	----

Procura e troca (ver nota p/ opções)	**Teclas**	**IBM PC (opcional)**
procurar string	CTRL-QF	----
procurar/trocar string	CTRL-QA	----
repetir procura/troca	CTRL-L	----

Outros comandos	**Teclas**	**IBM PC (opcional)**
inserir caractere de controle	CTRL-P, CTRL-X	----
abandonar	CTRL-U	----
restaurar linha	CTRL-QL	----
terminar edição	CTRL-KD	----

Observações

– Para restaurar uma linha apagada, você não pode ter tirado o cursor desta linha.

– Comandos para marcar uma palavra, tabulação automática e a seção "outros comandos" não pertencem ao padrão WORDSTAR.

– Opções para procura e troca:

n Procura e troca **n** vezes no texto.

B Procura e troca da posição do cursor até o começo do texto.

G Procura e troca em todo o texto, com pausa a cada ocorrência.

N Procura e troca em todo texto, sem pausa a cada ocorrência.

U Não diferencia caracteres maiúsculos ou minúsculos.

W Procura e troca apenas as palavras iguais à escolhida.

PARTES DE UM PROGRAMA PASCAL

Um programa (ou subprograma) em Pascal pode ser dividido em três áreas básicas:

– O cabeçalho (opcional): deve indicar o nome do programa.

- As definições e declarações (opcionais), divididas em:
  - declarações de etiquetas (labels).
  - definições de constantes.
  - definições de tipo.
  - declarações de variáveis.
  - definições de subprogramas (procedimentos e funções).
- Comandos: definem o programa principal, devendo sempre começar com a palavra reservada BEGIN e terminar com END seguido de ponto.

IDENTIFICADORES DE BLOCOS

program [ident [(características de I/0)];]

Marca o começo de um bloco de programa (opcional no Turbo).

ex.: Program Cadastro (Input_Output);

label [ident1] [, ident2, . . . ;]

Marca o começo da área de declaração de etiquetas (labels).

ex.: Label saída, tiro, erro;

const [ident1 = constante1 ;] [ident2 = constante2;]

Marca o início da área de definições de constantes.

ex.: Const

```
PREMIO              = 200;
DIAFINAL : Integer  = 15;
E        : Real     = 2.718282;
```

type [ident1 = tipo1;] [ident2 = tipo2;]

Marca o início da área de definições de tipo.

ex.: Type

```
PALAVRA  =  string [10];
SEMESTRE =  (JAN, FEV, MAR,
             ABR, MAI, JUN);
```

var [ident1 : tipo1;] [ident2 : tipo2;]

Marca o começo da área de declaração de variáveis.

ex.: Var

```
LETRA   : Char;
L, X, Y : Integer;
R       : Real;
INFO    : PALAVRA;
MES     : SEMESTRE;
```

procedure [ident [(parâmetros)];]

```
begin
```

```
        [subprograma]
        end;
```

Marca definições de procedimentos.

ex.:

```
Procedure QUADRADO (L: INTEGER);
   Var
      N : Integer;
   Begin
      ClrScr;
      For  N : = 1 to L do Begin
                              GotoXY (N, 1);
                              Write ('*');
                              GotoXY (1, N);
                              Write ('*');
                              GotoXY (N, L);
                              Write ('*');
                              GotoXY (L, N);
                              Write ('*');
                           End;
       WriteLn;
    End;
```

Observação: para chamar este procedimento no seu programa use: QUADRADO(L), onde L deve ter um valor entre 1 e 23.

function [ident [(parâmetros)]: tipo;]

```
begin
[subprograma]
end;
```

Marca definições de funções.

ex.:

```
Function DOBRO (L : Integer): Integer;
  Begin
    DOBRO: = L*2;
  End;
```

Observação: para chamar esta função no seu programa use: DOBRO(L), onde L deve ter um valor qualquer.

begin

[comandos;]

end

Marca a área de comandos de um programa ('.' depois do end) ou subprograma (';' depois do end). Pode ainda marcar um comando múltiplo (';' depois do end).

Nota

Os exemplos apresentados não abrangem todas as possibilidades de sintaxe dos comandos. Para mais detalhes, consulte o livro **Turbo Pascal – Guia do Usuário**, de Steve Wood, da Ed. McGraw-Hill.

REGRA PARA NOMES DE IDENTIFICADORES DEFINIDOS PELO USUÁRIO

Devem começar com uma letra ou sublinhado. Pode ter no máximo 126 caracteres, tanto letras como números e/ou sublinhados. Não há diferença entre maiúsculas e minúsculas. Exemplos:

Turbo	Dia_pag
a3	nome_func

PALAVRAS RESERVADAS DO TURBO PASCAL

As palavras reservadas com um asterisco não pertecem ao Pascal Padrão. Nenhuma palavra reservada deve ser redefinida pelo usuário.

absolute*	else	in
and	end	inline*
array	external*	label
begin	file	mod
case	for	nil
const	forward	not
div	function	of
do	goto	or
downto	if	overlay*

packed	shl*	until
procedure	shr*	var
program	string*	while
record	then	with
repeat	to	xor*
set	type	

IDENTIFICADORES PADRÕES

Podem ser redefinidos, mas não é aconselhável.

Addr	Chain	CrtInit
ArcTan	Char	Cyan
Assign	Chr	DarkGray
Aux	Close	DelLine
AuxInPtr	ClrEol	Delay
AuxOutPtr	ClrScr	Delete
Black	Con	Dispose
Blue	ConInPtr	Draw
BlockRead	ConOutPtr	EOF
BlockWrite	Concat	EOLN
Boolean	ConStPtr	Erase
Brown	Copy	Execute
BufLen	Cos	Exit
Byte	CrtExit	Exp

False
FilePos
FileSize
FillChar
Flush
Frac
FreeMem
GetMem
GotoXY
GraphBackGround
GraphColorMode
GraphMode
GraphWindow
Green
Halt
HeapPtr
Hi
HiRes
HiResColor
IOresult
Input
InsLine
Insert
Int
Integer
Kbd
KeyPressed
Length
LightBlue
LightCyan
LightGray
LightGreen
LightMagenta
LightRed
Ln
Lo
LowVideo
Lst
LstOutPtr
Mark
MaxInt
Mem
MemAvail
Move
New
NormVideo
NoSound
Odd
Ord
Output
Palette
Pi
Plot
Port
Pos
Pred
Ptr
Random
Randomize
Read
Readln
Real
Release
Rename
Reset
Rewrite
Round
Seek
Sin
SizeOf
SeekEof
SeekEoln
Sqr
Sqrt
Str
Succ
Swap
Text

TextBackGround	Upcase	WhereY
TextColor	Usr	White
TextMode	UsrInPtr	Window
Trm	UsrOutPtr	Write
True	Val	WriteLn
Trunc	WhereX	Yellow

DIRETIVAS DE DEFINIÇÃO DE ROTINAS
DIRETIVAS DE DECLARAÇÃO DE VARIÁVEIS

absolute endereço

Variável declarada em local específico da memória. Este comando deve ser usado na área de declarações de variáveis, indicando sempre um endereço. Em 16 bits, este endereço deve estar na forma **segmento: offset.**

ex.: em CP/M:

```
Var
STR_COMANDO: string [128] absolute $80;
```

ex.: em MS DOS e/ou CP/M 86:

```
Var
STR_COMANDO: string [128] absolute Cseg: $80;
```

external 'Str1'

Carrega um arquivo em disco, indicado pela string '**str1**', que contém o código-objeto para um subprograma (procedimento ou função), a terminação padrão é.**COM**. Podem-se passar parâmetros para este subprograma.

ex.:

```
Procedure IDENT (X: Integer);
   external 'ARQUIVO';
Function IDENT_DOIS (L: Integer): Integer;
   external 'QUALQUER';
```

forward

O cabeçalho do subprograma (procedimento ou função) termina com uma declaração **forward** e está separado da sua área de declaração de variáveis e comandos. É usado quando existe referência a um subprograma, antes deste estar definido. Podem-se passar parâmetros. Depois, é só definir o subprograma (sem indicar os parâmetros).

ex.:

```
Function IDENT (R: Real): Real; Forward;
{aqui viria a definição de uma outra função
   qualquer que fizesse referência a IDENT}
{aqui deve ser definida a função IDENT}
```

overlay

Compila automaticamente o subprograma

para um arquivo tipo overlay. Não é preciso fechar um arquivo overlay. Versão Turbo 2.0 em diante.

ex.: Overlay Procedure XPTO;

{definição normal de um subprograma}

Nota:

Os exemplos apresentados não abrangem todas as possibilidades de sintaxe dos comandos. Para mais detalhes, consulte o livro "**Turbo Pascal – Guia do Usuário**, de Steve Wood, Ed. McGraw-Hill.

PARTE 2

TABELAS

SÍMBOLOS ESPECIAIS

Identificador	Pascal padrão	Descrição
#	N	Precede um inteiro entre 0 e 255, representando um caractere ASCII ou um caractere especial.
$	N	Precede um valor hexadecimal.
'	S	O apóstrofe delimita uma constante tipo caractere ou uma string.
( )	S	Envolve parâmetros dos subprogramas, envolve valores nas definições de tipos, assinala valores para constantes "tipadas" estruturadas e modifica a precedência dos operadores.
,	S	A vírgula separa identificadores em vários contextos.
.	S	Marca o fim do programa principal.
..	S	Indica uma faixa de valores.

:	S	Precede o identificador de tipo nas declarações de variáveis, listas de parâmetros e definições de funções.
:=	S	Sinal de atribuição.
;	S	Separa comandos e termina declarações, definições e cabeçalhos.
=	S	Liga definições de tipo e constantes aos respectivos identificadores. (Veja também a tabela de operadores).
[ ] ou (. .)	S	Delimita identificadores de constantes do tipo conjunto (set) ou indicam referência a um elemento de matriz.
^	S	Define e identifica dados tipo ponteiro (POINTER) e

		precede caracteres 'A' a 'Z', [,\,], ^, ou_para indicar um caractere de controle.
{ } ou (* *)	S	Envolve um comentário.

TIPOS PREDEFINIDOS

Identificador	Pascal padrão	Descrição
Boolean	S	Assume valor TRUE ou FALSE. Ocupa um byte de memória.
Byte	N	Valores inteiros entre 0 e 255. Compatível com variáveis inteiras. Ocupa um byte de memória.
Char	S	Conjunto de caracteres ASCII mais outros caracteres de códigos entre #128 e #255. Ocupa um byte de memória.
Integer	S	Valores inteiros entre –32768 e 32767. Ocupa 2 bytes de memória.

Real	S	Números reais entre 1E-38 e 1E+38. Ocupa 6 bytes de memória.
string[n]	N	Define strings ("cadeia de caracteres alfanuméricos"), onde **n** é o comprimento máximo. **N** pode ser uma constante ou um número.
Text	S	Arquivo externo com estrutura tipo Char.
Text[nn]	N	O mesmo que TEXT, com buffer de tamanho **nn** blocos de 128 bytes. Só para sistemas MS DOS e CP/M 86 e versão Turbo 3.0.

CONSTANTES PREDEFINIDAS

Identificador	Pascal padrão	Descrição
MAXINT	S	Contém o valor inteiro 32767.

NIL	S	Valor tipo ponteiro que não aponta para lugar nenhum. Ocupa 4 bytes.
PI	N	Valor real 3.1415926536.
números	S	Constantes para tipos inteiros ou reais com representação decimal ou hexadecimal.
Caracteres	S	Constantes para o conjunto de caracteres ASCII mais alguns caracteres especiais que não fazem parte do padrão ASCII.
TRUE e FALSE	S	Constantes para o tipo Booleano.

CONSTANTES "TIPADAS"

São constantes em que o próprio usuário define o valor e o tipo, na área de definição de constantes. Não existe no Pascal Padrão. Exemplos:

```
const BONUS      = 2000; {constante normal}
```

```
PREMIO_MAX  :  integer = bonus; {constante tipada}
LETRA       :  char = 'X'; {constante tipada}
E           :  real = 2.718282; {constante tipada}
```

COMANDOS DE DEFINIÇÃO DE TIPO

array [ ] of

Usado para definir matrizes, onde a quantidade total de elementos é definida por uma ou mais **faixas.**

ex.:

```
Var
   M : Array [1 . . 10] of Integer;
         {matriz unidimensional}
   M2: Array [1 . . 20, 1 . . 30] of Real;
         {matriz bidimensional}
```

file of

Usado para definir arquivos.

ex.:

```
Var
   XPTO      : File of Byte;
   NÚMEROS   : File of Integer;
   DEFIN     : File of REGISTRO;
                  {deve-se definir REGIS-
                  TRO na área do TYPE
                  usando-se Record e End }
```

packed

Não usado no Turbo Pascal.

record end

Usado para definir estruturas de registros.

ex.:

```
Type
   REGISTRO = Record
                 NOME : String[20];
                 TEL  : String[13];
                 IDADE: 1 . . 99;
              End;

   FICHA    = Record
                 CAMPO1 : Char;
                 CAMPO2 : Real;
                 DADOS  : REGISTRO;
              End;
```

set of

Usado para definir conjuntos.

ex.:

```
Type
   LETRAS :  Set of 'A' . . 'Z';
   VOGAIS :  Set of (A, E, I, O, U)
   DIAS   :  Set of 1 . . 30;
```

Observações

1) **Faixa** indica uma seqüência de tipos separados por dois pontos ('. .'), começando em elemento 1 e terminando em elemento n. Exemplo:

1 . . 10 (números inteiros entre 1 e 10)

2) Os exemplos apresentados não abrangem todas as possibilidades de sintaxe dos comandos. Para mais detalhes, consulte o livro **Turbo Pascal – Guia do Usuário**, de Steve Wood, da Ed. McGraw-Hill.

COMANDOS DE CONTROLE

begin comando1;comando2;[comando3;...] **end**

Marca um comando composto quando usar ';' depois do **end** ou marca o programa principal (se usar '.' depois do **end**).

case seletor **of**

```
    constante1   : comando1;
    constante2   : comando2;
        ''              ''
        ''              ''
    [constante n:  comando n;]
[else comando;]
end; {case}
```

Decisão múltipla. ELSE é opcional. Se o **seletor** for igual a uma das constantes indicadas, será executado somente o comando correspondente. Caso contrário, será executado o comando após o ELSE (se existir).

ex.:

```
Case SELETOR of
        1    : WriteLn ('O seletor está valendo 1');
        2    : WriteLn ('O seletor está valendo 2');
        3, 4: WriteLn ('O valor e '' 3 ou 4');
Else
        WriteLn ('O valor e '' diferente de 1, 2,
            3, 4');
End; {Case}
```

exit

Força retorno imediato para a rotina de origem, sai de um bloco de comandos ou termina a

execução do programa. Só versão Turbo 3.0.

for Var_escalar: = valor_inicial **to** valor_final **do** comando;

for Var _escalar: = valor_inicial **downto**_valor final **do** comando;

Laço (loop) predeterminado com teste no início. Aumenta ou decrementa o valor da variável indicada (dependendo se você usar **to** ou **downto**) até que ela assuma o valor final, executando assim **n** vezes o comando, onde **n** é o resultado (em módulo) de **valor_inicial – valor_final**. As variáveis e valores devem ser todas do mesmo tipo (mas nunca tipo Real).

Ex.:

```
For N: = 1 to 20 Do WriteLn ('A variável
         vale', N);
For X: = 10 downto 0 Do Begin
           WriteLn (X);
           WriteLn ('----------');
         End;
```

goto etiqueta;

Transfere controle para um comando com etiqueta (label – já devidamente declarada).

ex.: Goto ROTINA_PRINCIPAL;

if condição **then** comando1 **else** comando2;

Decisão com duas alternativas, onde ELSE é opcional. Executará **comando1** se **condição** for verdadeira, caso contrário será executado **comando2** (se existir). Podemos ter várias condições, se usarmos os operadores lógicos.

ex.:

```
If A = 20  Then
                WriteLn ('Valor e " 20');
           Else
                WriteLn ('Valor diferente de 20');
```

repeat comando; **until** condição;

Laço (loop) condicional com teste posterior. Executa comando até que a condição seja verdadeira.

```
ex.: X:= 0;
     Repeat
          WriteLn ('X esta valendo', X);
          X:= X+1;
     Until X >= 10;
```

while condição do comando;

Laço (loop) condicional com teste anterior.

Executa comando enquanto a condição for verdadeira.

```
ex.: X:= 0;
     While X <= 10 do Begin
          WriteLn ('X esta valendo', X);
          X:= X+1;
        End;
```

with registro **do** comando;

Permite acesso a identificadores de campos de um registro sem precisar especificar o identificador do registro.

```
ex.:
With VAR_REGISTRO do Begin
   NOME  := 'Ricardo';
   TEL     := '222-2222';
   Write ('Digite uma idade qualquer');
   Readln(I);
   IDADE := I;
End;
```

Observações:

1) Entenda que comando pode ser simples ou composto.

2) Os exemplos apresentados não abrangem

todas as possibilidades de sintaxe dos comandos. Para mais detalhes consulte o livro **Turbo Pascal – Guia do Usuário**, de Steve Wood, da Ed. McGraw-Hill.

PARÂMETROS PARA O WRITE

Tipo de dado	Parâ-metro	Exemplo	Saída
Booleano	———	Write ('*', TRUE, '*');	*TRUE*
Booleano	:n	Write ('*', TRUE: 6, '*');	* TRUE*
Caractere	———	Write ('*', 'A', '*');	*A*
Caractere	:n	Write ('*', 'A': 3, '*');	* A*
Inteiro	———	Write ('*', 99, '*');	* 99*
Inteiro	:n	Write ('*', 99:5, '*');	* 99*
Real	———	Write ('*', 33.6, '*');	* 3.3600000000E+01*
Real	:n	Write ('*', 33.6:10, '*');	* 3.36E+01*
Real	:n : m	Write ('*', 33.6:6:2, '*');	* 33.60*
String	———	Write ('*', 'Turbo', '*');	*Turbo*
String	:n	Write ('*', 'Turbo':7, '*');	* Turbo*

Observação:

n e **m** equivalem a números ou expressões inteiras com valores entre 0 e 255 (para n) e 0 e 24 (para m).

Observação:

n e **m** equivalem a números ou expressões inteiras com valores entre 0 e 255 (para n) e 0 e 24 (para m).

OPERADORES RELACIONAIS, ARITMÉTICOS E DE CONJUNTO (SET)

Identificador	Pascal padrão	Descrição
		Operadores aritméticos
– (núm.)	S	Inverte o sinal de números reais ou inteiros.
not	S	Operação booleana **not** e operações com inteiros.
*	S	Multiplicação com números inteiros ou reais.
/	S	Divisão de números reais.
div	S	Divisão inteira.
mod	S	Resto de divisão inteira.
and	S	Operação booleana **and** e operações com inteiros.
shl	N	Deslocamento de bits à esquerda (só valores inteiros).
shr	N	Deslocamento de bits à direita (só valores inteiros).

Identificador	Pascal padrão	Descrição
+	S	Adição de números inteiros ou reais.
-	S	Subtração de números inteiros ou reais.
or	S	Operação booleana **or** e operações com inteiros.
xor	S	Operação booleana **or exclusivo** e operações com inteiros.
		Operadores relacionais
val1 = val2	S	Verdadeiro se valores forem iguais.
val1 <> val2	S	Verdadeiro se valores diferentes.
val1 < val2	S	Verdadeiro se vall for menor que val2.
val1 <= val2	S	Verdadeiro se val1 for menor ou igual a val2.
val1 > val2	S	Verdadeiro se val1 for maior que val2.

Identificador	Pascal padrão	Descrição
val1 >= val2	S	Verdadeiro se val1 for maior ou igual a val2.
		Operadores de conjunto
conj1 * conj2	S	Intersecção. O conjunto resultante contém membros comuns a ambos.
conj1 + conj2	S	União. Conterá todos os elementos dos dois conjuntos.
conj1 – conj2	S	Diferença. O resultado conterá elementos do **conj1** que não estão em **conj2**.
conj1 = conj2	S	Igualdade – TRUE se conjuntos forem iguais.
conj1<>conj2	S	Diferença – TRUE se conjuntos diferentes.
conj1<=conj2	S	TRUE se **conj1** for subconjunto de **conj2**.

Identificador	Pascal padrão	Descrição
conj1 >= conj2	S	TRUE se **conj2** for subconjunto de **conj1**.
conj1 **in** conj2	S	O mesmo que < = exceto que **conj1** pode ser um dado único de um conjunto básico (formado por valores escalares – caracteres ou inteiros).

RESULTADOS ENTRE OPERAÇÕES

Operador	Operação	Tipo	Resultado
–	subtração	real	real
–	subtração	int.	int.
–	subtração	real, int.	real
*	multiplicação	real	real
*	multiplicação	int.	int.
*	multiplicação	real, int.	real
/	divisão	real, int.	real

Operador	Operação	Tipo	Resultado
/	divisão	int.	real
/	divisão	real	real
div	divisão int.	int.	int.
mod	resto da div.	int.	int.
and	aritmético	int.	int.
and	lógico	bool.	bool.
shl	rotação esq.	int.	int.
shr	rotação dir.	int.	int.
+	adição	real	real
+	adição	int.	int.
+	adição	real, int.	real
or	aritmético	int.	int.
or	lógico	bool.	bool.
xor	aritmético	int.	int.
xor	lógico	bool.	bool.

TABELA VERDADE PARA OPERADORES NOT, AND, OR e XOR

Primeiro valor	Operador	Segundo valor	Resultado
verd.	not	---	falso
falso	not	---	verd.

Primeiro valor	Operador	Segundo valor	Resultado
verd.	and	verd.	verd.
verd.	and	falso	falso
falso	and	verd.	falso
falso	and	falso	falso
verd.	or	verd.	verd.
verd.	or	falso	verd.
falso	or	verd.	verd.
falso	or	falso	falso
verd.	xor	verd.	falso
verd.	xor	falso	verd.
falso	xor	verd.	verd.
falso	xor	falso	falso

ROTINAS DE VÍDEO E TECLADO

Observação

Nenhuma destas rotinas faz parte do Pascal Padrão.

Identificador (parâmetro)	Procedimento /função	Descrição
ClrEol	P	Limpa do cursor até o fim da linha sem mover o cursor.
ClrScr	P	Limpa o vídeo e volta o cursor para a primeira posição da tela (canto superior esquerdo).
CrtExit	P	Envia uma string de reset para o terminal. Esta string é definida na instalação da tela.
CrtInit	P	Envia uma string de inicialização do terminal.
DelLine	P	Apaga linha na posição do cursor.
GotoXY (col, lin)	P	Posiciona o cursor na coluna **col**, linha **lin**.

Identificador (parâmetro)	Procedimento /função	Descrição
		O canto superior esquerdo vale 1,1. Valores inteiros.
HighVideo	P	Próxima saída em alta intensidade (IBM PC) ou com o fundo normal (APPLE). Depende da instalação do sistema.
InsLine	P	Insere linha na posição do cursor.
IOresult	F	Devolve o status de I/O (entrada/saída) para o tratamento de erros. Deve ser usado com a diretiva {$I−}. Retorna valor inteiro com o código do erro.

Identificador (parâmetro)	Procedimento /função	Descrição
KeyPressed	F	TRUE se existir um caractere no buffer do teclado.
LowVideo	P	Próxima saída em baixa intensidade (IBM PC) ou fundo reverso (APPLE). Depende da instalação do sistema.
NormVideo	P	O mesmo que HighVideo.

VARIÁVEIS PRÉ-DECLARADAS

Identificador	Pascal padrão	Descrição
Variáveis de Arquivos – Texto (atribuição pode variar dependendo do sistema)		
Aux	N	Atribuída ao dispositivo de comunicação serial ou dispositivo auxiliar (AUX:).

Identificador	Pascal padrão	Descrição
Con	N	Atribuída aos dispositivos de entrada e saída (I/O) do console do sistema (CON:).
Input	S	Atribuída ao dispositivo de entrada padrão (pode ser CON: ou TRM:).
Kbd	N	Atribuída ao dispositivo de entrada do console do sistema (KBD:).
Lst	N	Atribuída à impressora do sistema (LST:).
Output	S	Atribuída ao dispositivo de saída padrão (pode ser CON: ou TRM:).
Trm	N	Atribuída ao dispositivo de entrada e saída (I/O) do terminal do sistema.
Usr	N	Disponível para atribuição do usuário (USR:).

Variáveis de Endereço de Vetor do Acionador de Entrada e Saída (I/O)

(endereços de 16 bits em sistemas de 8 bits e endereços de 32 bits em sistemas de 16 bits)

Identificador	Pascal padrão	Descrição
AuxInPtr	N	Aponta para o procedimento de entrada Aux.
AuxOutPtr	N	Aponta para o procedimento de saída Aux.
ConInPtr	N	Aponta para o procedimento de entrada Con.
ConOutPtr	N	Aponta para o procedimento de saída Con.
ConStPtr	N	Aponta para a função de status Con.
ConLstPtr	N	Aponta para o procedimento de saída Lst.
UsrInPtr	N	Aponta para o procedimento de entrada Usr.
UsrOutPtr	N	Aponta para o procedimento de saída Usr.

Variáveis Tipo Matriz para Memória e Portas de Dados

Mem[ ]	N	Matriz tipo Byte; acessa toda a memória.
MemW[ ]	N	Matriz tipo Integer; acessa toda a memória. Só para

Identificador	Pascal padrão	Descrição
		sistemas MS DOS e CP/M 86.
Port[ ]	N	Matriz tipo Byte; acessa todas as portas de entrada e saída (I/O).
PortW[ ]	N	Matriz tipo Integer; acessa todas as portas de entrada e saída (I/O). Só para sistemas MS DOS e CP/M 86.

Nota

Em CP/M 80, as matrizes tipo Byte devem ser indexadas por valores inteiros. Em MS DOS e CP/M 86, as matrizes devem ser indexadas por valores **segmento: offset**.

Variáveis a Nível do Sistema

BufLen	N	Número máximo de caracteres aceito pelo procedimento Read.
HeapPtr	N	Endereço do primeiro byte livre depois do código-objeto (HEAP).

Identificador	Pascal padrão	Descrição
RecurPtr	N	Endereço do topo da pilha de recursividade, somente se a diretiva {$A−} for usada.
StackPtr	N	Endereço do topo da memória livre (STACK).

Notas:

Endereços em sistemas de 8 bits são valores inteiros. Endereços em sistemas de 16 bits são valores **segmento: offset.**

Os valores dos endereços devem seguir a seguinte ordem:

HeapPtr < RecurPtr < StackPtr

ROTINAS PARA ENTRADA E SAÍDA DE ARQUIVOS E DISPOSITIVOS EXTERNOS

Identificador (parâmetros)	Procedimento/ função	Pascal padrão	Descrição
Append (var_arqtxt)	P	N	Abre um arquivo-texto já existente, posicionando o ponteiro de arquivo no final do arquivo. O arquivo está aberto somente para saída (output). Só disponível em MS DOS e versão Turbo 3.0.
Assign (var_arq, nome_arq)	P	N	Associa o nome de arquivo em **nome_arq** com **var_arq**. Nunca deve ser usado em arquivos que já estão em uso.
BlockRead (var_arq, var, reg [, trans])	P	N	Transferência de um arquivo sem tipo, do disco para a memória. O nome do arquivo deve estar assinalado em **var_arq**; **var** é uma variável qualquer e **reg** é o número de registros de 128 bytes. **Trans** guarda o número de bytes já transferidos (opcional).
BlockWrite (var_arq, var, reg [, trans])	P	N	Transferência de um arquivo sem tipo, da memória para disco. Mesmas características do BlockRead.
Chain (var_arq)	P	N	Carrega e executa um arquivo .**CHN** com um nome assinalado em **var_arq**, usando as rotinas do Turbo Pascal contidas na memória.
Close (var_arq)	P	N	Grava as informações do buffer e fecha um arquivo externo. Nunca esqueça de fechar um arquivo.
Eof (var_arq)	F	S	VERDADEIRO se o ponteiro do arquivo estiver no fim do arquivo (CTRL-Z).

Identificador (parâmetros)	Procedimento/ função	Pascal padrão	Descrição
EoLn (var_arq)	F	S	VERDADEIRO se o ponteiro do arquivo-texto estiver no fim da linha (caractere CR, "carriage return" e LF "line feed") ou fim de arquivo (CTRL-Z).
Erase (var_arq)	P	N	Remove o arquivo do diretório do disco. Nunca apague um arquivo aberto.
Execute (var_arq)	P	N	Carrega e executa um arquivo tipo **.COM** com nome assinalado em **var_arq**.
FilePos (var_arq)	F	N	Retorna a posição do ponteiro do arquivo assinalado em **var_arq** (valor inteiro). Começa pelo número 0 (primeiro componente). Não é válido para arquivos-texto.
FileSize (var_arq)	F	N	Retorna o número de componentes de um arquivo não texto (valor inteiro +1 pois a numeração começa do 0) – não é válido para arquivos-texto.
Flush (var_arq)	P	N	Força a saída dos dados contidos no buffer para o disco, sem fechar o arquivo. Não pode ser usado em arquivos-texto no CP/M, nem tem efeito em arquivos com tipo em MS DOS. É válido para arquivos-texto em MS DOS. Nunca use o Flush em arquivos fechados.
Get	P	S	Entrada de dados do Pascal Padrão – não usada no Turbo.
LongFilePos (var_arq)	F	N	Retorna a posição do ponteiro de um arquivo maior que 32 K – não é válido para arquivos-texto. Só disponível em MS DOS. Valor real.

Identificador (parâmetros)	Procedimento/ função	Pascal padrão	Descrição
LongFileSize (var_arq)	F	N	Retorna o número de componentes de um arquivo não texto maior que 32K – não é válido para arquivos-texto. Para arquivos sem tipo, cada componente possui 128 bytes. Só disponível em MS DOS.
LongSeek (var_arq, var_real)	P	N	Posiciona o ponteiro de arquivo no componente número **var_real** em arquivos maiores que 32 K – não é válido para arquivos-texto. Só disponível em MS DOS.
Put	P	S	Saída de dados do Pascal Padrão – não usada no Turbo.
Read ([var_arq,] var1, var2, . . .)	P	S	Entrada de dados de um arquivo externo em disco. Se **var_arq** não estiver indicado, a entrada de dados deverá ser feita pelo teclado. Não pode ser usado em arquivos sem tipo.
ReadLn ([var_arqtxt,] var1, var2. . .)	P	S	Entrada de dados de um dispositivo ou arquivo texto. Regras iguais ao Read. As variáveis serão lidas até que se encontre a seqüência de caracteres de fim de linha (CR e LF).
Rename (var_arq, str1)	P	N	Muda o nome do arquivo em disco assinalado por **var_arq** para **str1**. Nunca use em arquivos abertos.
Reset (var_arq [, var_int])	P	S	Abre arquivo já existente em disco com nome assinalado em **var_arq**. Posiciona o ponteiro no componente número 0 (arquivo não texto). Se **var_arq** for um arquivo-texto, ele será aberto para leitura, e o ponteiro será posi-

Identificador (parâmetros)	Procedimento/ função	Pascal padrão	Descrição
			cionado no começo do arquivo. **Var_int** especifica o tamanho do bloco de transferência, somente em MS DOS (opcional).
Rewrite (var_arq [,var_int])	P	S	Cria e abre um novo arquivo no disco com nome assinalado em **var_arq**. Se **var_arq** for tipo texto, o arquivo será aberto somente para saída. Posiciona o ponteiro no componente número 0 (arquivo não texto) ou no começo do arquivo texto. **Var_int** especifica o tamanho do bloco de transferência, somente em MS DOS (opcional).
Seek (var_arq, var)	P	N	Posiciona ponteiro do arquivo no componente de número **var** (inteiro em CP/M ou real em MS DOS) do arquivo assinalado em **var_arq** – não é válido para arquivos-texto.
SeekEof (var_arq)	P	N	Pula tabulação, espaço em branco e marcas de fim de linha (CR/LF) antes do teste de fim de arquivo **(Eof)**. Só em versão Turbo 3.0. Retorna valor booleano.
SeekEoln (var_arqtxt)	P	N	Pula tabulação e espaços em branco antes do teste de fim de linha **(Eoln)** e/ou fim de arquivo. Só em versão Turbo 3.0. Retorna valor booleano.
Truncate (var_arq)	P	N	Trunca um arquivo no registro indicado pelo valor corrente do ponteiro do arquivo e deixa-o somente para saída. Só em MS DOS e versão Turbo 3.0.
Write ([var_arq,] var1, var2...)	P	S	Saída de dados para dispositivo ou arquivo em disco. Regras iguais ao Read.
WriteLn ([var_arq,] var1, var2...)	P	S	Saída de dados para dispositivo ou arquivo de texto com uma sequência de caracteres CR e LF após a última variável. Regras iguais ao Read.

ROTINAS PARA TRATAMENTO DE STRINGS

Observação

Nenhuma destas rotinas faz parte do Pascal Padrão.

Identificador (parâmetros)	**Procedimento/ função**	**Descrição**
Concat (str1, str2[, str3, . . .])	F	Concatena duas ou mais strings.
Copy (str1, var_int1, var_int2)	F	Retorna uma sub-string de **var_int2** caracteres da string **str1**, começando na posição **var_int1**.

Identificador (parâmetros)	Procedimento/ função	Descrição
Delete (str1, var_int1, var_int2)	P	Remove **var_int2** caracteres começando na posição **var_int1** da string **str1**.
Insert (str1, str2, var_int1)	P	Insere **str1** em **str2** na posição **var_int1**.
Length (str1)	F	Retorna o comprimento (em valor inteiro) de **str1**.
Pos (str1, str2)	F	Retorna a posição do primeiro caractere de **str1** dentro de **str2**. Se não existir, retornará 0. Valor inteiro.
Str (var, str1)	P	Converte o inteiro ou real de **var** para a string **str1**. **Var** pode conter os

Identificador (parâmetros)	Procedimento/ função	Descrição
		parâmetros de escrita do WRITE.
UpCase (carac1)	F	Retorna o caractere maiúsculo de **carac1**, no limite de 'a' . . . 'z'.
Val (str1, var1, var_int)	P	Converte **str1** para um valor numérico inteiro ou real **var1** (dependendo do tipo declarado para **var1**). **Var_int** retornará o status da operação: sem erro **var_int = 0**; com erro – **var_int** = (posição do primeiro caractere onde ocorreu erro) e **var1** fica indefinida.

ROTINAS DE MANIPULAÇÃO DE MEMÓRIA

Identificador (parâmetros)	Procedimento/ função	Pascal padrão	Descrição
			Alocação de Memória
Dispose (var_pntr)	P	S	Livra memória alocada somente para **var_pntr**. Não use Dispose no mesmo programa com Mark/Release. Versão Turbo 2.0 em diante.
FreeMem (var_pntr, var_int1)	P	N	Deixa livre **var_int1** bytes do heap começando em **var_pntr**.
GetMem (var_pntr, var_int1)	P	N	Aloca **var_int1** bytes do heap começando em **var_pntr**.
Mark (var_pntr)	P	N	Salva o endereço de topo do heap em **var_pntr**. Não use Mark no mesmo programa com Dispose.
MaxAvail	F	N	Retorna o maior espaço livre contíguo do heap. Valor inteiro. Em micros de 16 bits retornará o número de blocos de 16 bytes. Em micros de 8 bits, retornará o número de bytes. Se o valor der maior que 32767, retornará um número negativo.
MemAvail	F	N	Retorna o espaço total remanescente no heap. Mesmas regras que o MaxAvail.
New (var_pntr)	P	S	Aloca espaço no heap para **var_pntr**.
Pack	P	S	Não é usada no Turbo.
Ptr (var_int1)	F	N	Retorna uma variável tipo ponteiro baseado em **var_int1**, compatível com qualquer tipo de ponteiro. Somente sistemas 8 bits.
Ptr (var_int1, var_int2)	F	N	Retorna uma variável tipo ponteiro baseado em **var_int1**. **Var_int1** é o endereço **segmento** e **var_int2** é o endereço **offset**. Só em sistemas de 16 bits.

Identificador (parâmetros)	Procedimento/ função	Pascal padrão	Descrição
Release (var_pntr)	P	N	Restaura topo do heap para o endereço em **var_pntr**. Não use Release no mesmo programa com Dispose.
			Manipulação de Memória
Addr (var)	F	N	Retorna o endereço da variável ou do início de um subprograma **var**. Var pode ser qualquer tipo de variável. Em 8 bits retorna valor inteiro. Em 16 bits, valor em formato **segmento: offset**.
Cseg	F	N	Retorna valor inteiro com o endereço de início do segmento de códigos. O valor é retornado em blocos de 16 bytes. Para saber o endereço exato, faça 16*Cseg (faixa até 640 Kbytes). Somente sistemas de 16 bits.
Dseg	F	N	Retorna valor inteiro com o endereço de início do segmento de dados. O valor é retornado em blocos de 16 bytes. Para saber o endereço exato faça 16*Dseg (faixa até 640 Kbytes). Somente sistemas de 16 bits.
FillChar (var, var_int1, carac1)	P	N	Preenche **var_int1** bytes de memória, a partir do endereço **var**, com **carac1** (pode ser tipo char ou byte).
Hi (var_int1)	F	N	Retorna o byte mais significativo da variável inteira **var_int1**. Valor inteiro.

Identificador (parâmetros)	Procedimento/ função	Pascal padrão	Descrição
Lo (var int1)	F	N	Retorna o byte menos significativo da variável inteira **var_int1**. Valor inteiro.
Move (var1, var2, var int1)	P	N	Copia o bloco de **var_int1** bytes do endereço **var1** para o endereço **var2**.
Ofs (var)	F	N	Devolve endereço **offset** de **var**, que pode ser um identificador de variável ou de subprograma. Valor inteiro. Somente para sistemas de 16 bits.
Seg (var)	F	N	Retorna o endereço **segmento** de **var**, que pode ser um identificador de variável ou de subprograma. Valor inteiro. Somente sistemas de 16 bits.
SizeOf (var)	F	N	Retorna o espaço exigido por **var** que pode ser uma variável ou identificador de tipo. Valor inteiro.
Sseg	F	N	Retorna o endereço **segmento** do início do stack. Valor inteiro. Somente sistemas de 16 bits.
Swap (var int1)	F	N	Troca byte mais significativo pelo menos significativo de **var_int11**. Valor inteiro.

FUNÇÕES ESCALARES E DE CONVERSÃO

Identificador	Pascal padrão	Descrição
Chr (var_int1)	S	Retorna o caractere ASCII de **var_in1** (tipo char).
Odd (var_int1)	S	Retorna TRUE se **var_int1** for ímpar.
Ord (var_escalar)	S	Retorna o valor numérico inteiro **var_escalar**.
Ord (var_pntr)	N	Retorna o endereço (na forma inteiro) apontado por **var_pntr**.
Pred (var_escalar)	S	Retorna o predecessor de **var_escalar** se existir.
Succ (var_escalar)	S	Retorna o sucessor de **var_escalar** se existir.
Tipo_Escalar (var_escalar)	N	Retorna a constante literal de **Tipo_Escalar** que corresponde a **var_escalar**. **Tipo_Escalar** representa qualquer identificador de tipo escalar.

FUNÇÕES ARITMÉTICAS

Identificador	Pascal padrão	Descrição
Abs (var)	S	Retorna o valor absoluto de **var**. O tipo de retornado será o mesmo de **var**.
ArcTan (var)	S	Retorna o ângulo em radianos baseado na tangente de **var**. O resultado é real.
Cos (var)	N	Retorna o valor real do coseno de **var**. **Var** pode ser real ou inteiro, mas deve ser expresso em radianos.
Exp (var)	S	Retorna o valor real do exponencial de **var**. **Var** pode ser inteiro ou real.
Frac (var)	N	Retorna a parte fracionária de **var**.
Int (var)	N	Retorna a parte inteira de **var**. Se **var** for inteiro, retornará um valor real igual.
Ln (var)	S	Retorna o valor real do logarítmo natural de **var** (va-

Identificador	Pascal padrão	Descrição
		lor inteiro ou real).
Random	N	Retorna um valor pseudoaleatório (real) maior ou igual a 0 e menor do que 1.
Random (var_int1)	N	Retorna um valor pseudoaleatório (inteiro) maior ou igual a 0 e menor do que **var_int1**.
Round (var_real)	N	Retorna o valor inteiro de **var_real**, arredondado, se necessário.
Sin (var)	S	Retorna o valor real do seno de **var. Var** pode ser real ou inteiro mas deve ser expresso em radianos.
Sqr (var)	S	Eleva **var** ao quadrado. O tipo retornado será o mesmo de **var.**
Sqrt (var)	S	Retorna a raiz quadrada de **var.** O tipo retornado será real.
Trunc (var_real)	N	Retorna a parte inteira de **var_real**.

Notas:

1) As funções trigonométricas Sin e Cos necessitam que os ângulos sejam dados em radianos.

2) 1 radiano equivale a .017453 graus.

3) 1 grau equivale a 57.2957795 radianos.

4) A tangente de um ângulo pode ser calculada como Sin (var) / Cos (var).

5) As funções Ln e Exp usam como base o número **e**, que vale 2.718282.

ROTINAS DE CONTROLE DO SISTEMA

Observação

Nenhuma destas rotinas faz parte do Pascal Padrão.

Rotinas para todos os Sistemas
(CP/M 80, CP/M 86 e MS DOS)

Identificador (parâmetros)	Procedimento/ função	Descrição
ParamCount	F	Retorna o número de parâmetros no buffer de linha de comando do siste-

Identificador (parâmetros)	Procedimento/ função	Descrição
		ma operacional. Versão Turbo 3.0.
ParamStr (var_int)	F	Retorna uma string de **var_int** parâmetros do buffer de linha de comando do sistema operacional. Versão Turbo 3.0.
Delay (var_int)	P	Faz uma pausa de **var_int** milissegundos.
Halt	P	Termina a execução de um programa e volta para o sistema operacional.
Randomize	P	Dá uma nova semente à rotina de geração de números aleatórios. Versão Turbo 3.0.

Rotinas Exclusivas do CP/M 80

Identificador (parâmetros)	**Procedimento/ função**	**Descrição**
Bdos (int1, [, var_int2])	P	Chama a rotina Bdos de número **var_int1** usando parâmetros opcionais **var_int2**, se necessário.
Bdos (var_int1 [, var_int2])	F	Chama a função Bdos de número **var_int1** usando parâmetros opcionais **var_int2**, se necessário. O resultado será devolvido no registrador A.
BdosHl (var_int1 [, var_int2...])	F	Chama a função Bdos de número **var_int1** usando parâmetros opcionais

Identificador (parâmetros)	Procedimento/ função	Descrição
		var_int2, se necessário. O resultado será devolvido no par de registradores HL.
Bios (var_int1 [, var_int2...])	P	Chama a rotina Bios de número **var_int1** usando parâmetros opcionais **var_int2**, se necessário.
Bios (var_int1 [, var_int2...])	F	Chama a função Bios de número **var_int1** usando parâmetros opcionais **var_int2**, se necessário. O resultado será devolvido no registrador A.

Identificador (parâmetros)	Procedimento/ função	Descrição
BiosHL (var_int1 [, var_intr2...])	F	Chama a função Bios de número **var_int1** usando parâmetros opcionais **var_int2**, se necessário. O resultado será devolvido no par de registradores HL.

Rotina Exclusiva do CP/M 86

Identificador (parâmetros)	Procedimento/ função	Descrição
Bdos (reg_dos) (veja nota)	P	Executa chamadas de função Bdos baseado em **reg_dos**. O resultado é devolvido em **reg_dos**.

Rotinas Exclusivas do MS DOS

Identificador (parâmetros)	Procedimento/ função	Descrição
ChDir (str1)	P	O diretório corrente é alterado para aquele indicado em **str1**. Versão Turbo 3.0.
GetDir (var_int1, str1)	P	Retorna o diretório do drive **var_int1** (0 = drive em uso, 1 = drive A, 2 = drive B – valor inteiro) na variável string **str1**. Versão Turbo 3.0.
MkDir (str1)	P	Cria um novo subdiretório especificado pela string **str1**, cujo último valor deve ser um nome de arquivo inexistente. Versão Turbo 3.0.

Identificador (parâmetros)	Procedimento/ função	Descrição
MsDos (reg_dos) (veja nota)	P	Executa uma chamada às rotinas do sistema MS DOS baseada em **reg_dos**,
OvrPath (str1)	P	Indicam que os arquivos para **overlay** estão no subdiretório indicado por **str1**. Versão Turbo 3.0.
RmDir (str1)	P	Remove o subdiretório indicado por **str1**, se estiver vazio. Versão Turbo 3.0.

Rotinas para CP/M 86 e MS DOS

Identificador (parâmetros)	Procedimento/ função	Descrição
Intr (reg_dos) (veja nota)	P	Causa uma interrupção (por software) baseada em **reg_dos.**
OvrDrive (var_int1)	P	Indicam que os arquivos para **overlay** estão no drive **var_int1**. Versão Turbo 3.0.

Nota

O parâmetro **reg_dos** é do tipo registro e pode ser declarado usando **type**:

```
type    Registro = record
            ax, bx, cx, dx, bp, si,
            di, ds, es, flags          : Integer;
        end;
```

Reg_dos é um parâmetro de variável, podendo ser usado pelo MS DOS para retornar valores à rotina de chamada:

```
var  reg_dos  : registro;
```

ROTINAS DO SISTEMA PARA IBM PC, XT ou AT

Observação

Todas as variáveis usadas devem ser inteiras. A exceção está indicada.

Identificador	Procedimento/ função	Descrição
Não Exige Placa Gráfica		
NoSound	P	Interrompe a rotina de geração de som.
Sound (t)	P	Inicia a geração de som com o tom inteiro **t**.
TextBackGround (c7)	P	Seleciona a cor de fundo número **c7** (entre 0 e 7 da tabela de cores) para o texto.
TextColor (c15)	P	Seleciona a cor do texto de número inteiro **c15**, com

Identificador	Procedimento/ função	Descrição
		valores entre 0 e 15. Somando-se 16 o texto piscará.
TextMode	P	Seleciona o modo texto usando o modo de tela corrente.
TextMode (var_int1)	P	Seleciona o modo texto usando o modo de tela número **var_int1**.
WhereX	F	Retorna a coluna corrente do cursor.
WhereY	F	Retorna a linha corrente do cursor.
Window (x1, y1, x2, y2)	P	Define uma janela de texto. Canto esquerdo superior em **x1, y1** e canto inferior direito em **x2, y2**.

Identificador	Procedimento/ função	Descrição
Exigem Placa Gráfica		
Draw (x1, y1, x2, y2, c)	P	Desenha uma linha ligando o ponto **x1**, **y1** ao ponto **x2**, **y2** com a cor **c**.
GraphBackGround (c15)	P	Seleciona a cor de fundo número **c** para a tela toda de gráficos (resolução de 320 x 200), com valores entre 0 e 15.
GraphColorMode	P	Seleciona o modo gráfico colorido, com resolução de 320 x 200 pixels.
GraphMode	P	Seleciona o modo gráfico monocromático, com resolução de 320 x 200 pixels. Deixa usar as palettes 0 e 1 para fazer tonali-

Identificador	Procedimento/ função	Descrição
		dades diferentes.
GraphWindow (x1, y1, x2, y2)	P	Define uma janela para gráficos. Canto esquerdo superior em **x1**, **y1** e canto inferior direito em **x2**, **y2**. A posição 0,0 passa a ser o canto superior esquerdo da nova janela.
HiRes	P	Seleciona alta resolução de 640 x 200 pixels. Cor de fundo preta.
HiResColor (c15)	P	Seleciona a cor número **c** para os desenhos. Valores entre 0 e 15.
Palette (var_int1)	P	Seleciona combinações de cores para gráficos de resolução 320 x 200 de

Identificador	Procedimento/ função	Descrição
		acordo com **var_int1** e a tabela de palettes.
Plot (x, y, c)	P	Coloca um pixel na coordenada **x**, **y** com a cor **c**.
Exigem Placa Gráfica e Turbo Versão 3.0		
Arc (x1, y1, ang, r, c3)	P	Começando no ponto **x1**,**y1**, desenha um arco de **ang** graus com raio **r**, usando a cor **c3** (veja Observação 1).
Circle (x, y, r, c3)	P	Desenha um círculo com centro no ponto **x**, **y** e raio **r**, usando cor **c3** (veja Observação 1).
ColorTable (c1, c2, c3, c4)	P	Cria uma tabela de conversão de co-

Identificador	Procedimento/ função	Descrição
		res. Todos procedimentos de desenho usarão esta tabela se a cor for igual a –1. Veja exemplo e a tabela do ColorTable adiante.
FillScreen (c3)	P	Limpa janela usando cor **c3** (ver Observação 1).
FillShape (x, y, cp, cr)	P	Preenche área que contenha o ponto **x, y** e que seja rodeada pela cor **cr**, com a cor **cp**. Valores entre 0 e 3, não aceita cor –1.
FillPattern (x1,y1,x2,y2,c)	P	Preenche um retângulo com canto esquerdo superior em **x1, y1** e canto inferior direito em

Identificador	Procedimento/ função	Descrição
		x2, y2 com o padrão reticulado corrente e cor **c**.
GetDotColor (x, y)	F	Retorna o código da cor do pixel **x, y**.
GetPic (b, x1, y1, x2, y2)	P	Armazena uma imagem gráfica definida num retângulo de canto esquerdo superior em **x1, y1** e canto inferior direito em **x2, y2** num buffer definido por **b**.
Pattern (p)	P	Define um padrão reticulado (pattern) utilizado por **FillPatern**, usando os bits contidos em **p** (veja Observação 2).
PutPic (b, x, y)	P	Exibe a imagem gráfica contida no buf-

Identificador	Procedimento/ função	Descrição
		fer **b** na posição **x, y** (antigo canto inferior esquerdo de **GetPic**).

Observações

1) Os valores das cores variam entre 0 e 3 da tabela de palettes. Se **c3** = −1, a cor será escolhida de acordo com ColorTable.

2) O parâmetro **p** é uma matriz de 8 bytes (array [0 . .7] of byte). O estado individual dos bits de cada byte determinam o padrão reticulado (pattern).

Gráficos com Tartaruga.
Exigem Placa Gráfica e Turbo Versão 3.0.

Back (n)	P	Move a tartaruga para trás **n** passos.
ClearScreen	P	Limpa a janela gráfica da tartaruga e desloca-a para a posição 0,0.
Forwd (n)	P	Move a tartaruga para

Identificador	Procedimento/ função	Descrição
		frente **n** passos.
Heading	F	Retorna a direção da tartaruga em graus (0 é para cima, 90 é para a direita, 180 é para baixo etc...).
HideTurtle	P	Esconde a tartaruga.
Home	P	Move a tartaruga para a posição central 0,0 de sua janela.
NoWrap	P	Evita que a tartaruga apareça do outro lado da janela quando ela ultrapassa os limites desta.
PenDown	P	A tartaruga desenhará uma linha ao se mover.
PenUp	P	A tartaruga não de-

Identificador	Procedimento/ função	Descrição
		senhará uma linha ao se mover.
SetHeading(ang)	P	Ajusta a direção da tartaruga em **ang** graus (0 é para cima ou Norte, 90 é para a direita ou Leste, 180 é para baixo ou Sul etc...).
SetPenColor (c)	P	A cor da linha a ser desenhada pela tartaruga ao se mover será definida pela cor **c** (valores inteiros entre −1 e 3).
SetPosition (x, y)	P	Move a tartaruga para a posição **x**, **y**, sem desenhar uma linha.
ShowTurtle	P	Exibe a tartaruga.
TurnLeft (ang)	P	Rotaciona a direção

Identificador	Procedimento/ função	Descrição
		da tartaruga em **ang** graus no sentido anti-horário.
TurnRight (ang)	P	Rotaciona a direção da tartaruga em **ang** graus no sentido horário.
TurtleDelay (var_int)	P	A tartaruga perderá **var_int** milissegundos entre um passo e outro.
TurtleWindow (x, y, l, h)	P	Define a janela gráfica da tartaruga. O centro em **x, y** com largura **l** e altura **h**. A tartaruga não pára ao ultrapassar o limite da janela, porém não é mostrada, nem desenha nada. Ver observação 2.

Identificador	Procedimento/ função	Descrição
TurtleThere	F	Retornará TRUE se a tartaruga estiver na janela gráfica corrente.
Wrap	P	Faz a tartaruga aparecer do outro lado da janela quando ela ultrapassar os limites desta.
Xcor	F	Retorna a coordenada horizontal corrente da tartaruga. Valor inteiro.
Ycor	F	Retorna a coordenada vertical corrente da tartaruga. Valor Inteiro.

Notas

1) A tartaruga é representada pelo desenho de

um pequeno triângulo. Este é um conceito da linguagem LOGO.

2) A coordenada 0,0 fica no centro da janela da tartaruga. Obtêm-se valores positivos à direita (X) e para cima (Y) e valores negativos à esquerda (X) e para baixo (Y) desta coordenada central.

MODOS DE TELA

Tabela a ser usada pelo comando TextMode (var_int), onde var_int é um número, expressão inteira ou uma constante predefinida:

var_int	**const.**	**resulta:**
0	bw40	Branco e preto, 40 colunas
1	c40	Colorido, 40 colunas
2	bw80	Branco e preto, 80 colunas
3	c80	Colorido, 80 colunas

Sempre teremos 25 linhas.

TABELA DE CORES

As cores podem ser indicadas por números ou pelas constantes predefinidas a seguir:

num.	const.	cor	num.	const.	cor
0	Black	Preto	8	DarkGray	Cinza Escuro
1	Blue	Azul	9	LightBlue	Azul Claro
2	Green	Verde	10	LightGreen	Verde Claro
3	Cyan	Azul Ciano	11	LightCyan	Azul Ciano Claro
4	Red	Vermelho	12	LightRed	Vermelho Claro
5	Magenta	Magenta	13	LightMagenta	Magenta Claro
6	Brown	Marrom	14	Yellow	Amarelo
7	LightGray	Cinza Claro	15	White	Branco

Usando-se 16, teremos o efeito "**de piscar**" (a constante, neste caso, é Blink).

TABELA DE CORES PARA O COLORTABLE

Este procedimento cria uma tabela de conversão das cores usadas pelo computador quando tiver de sobrepor duas cores. ColorTable (0, 1, 2, 3) é o default (não haverá mudança de cor se for a mesma).

Exemplo:

ColorTable (2, 3, 1, 0) transformaria:

cor 0 em cor 2
cor 1 em cor 3
cor 2 em cor 1
cor 3 em cor 0

TABELA DE COORDENADAS DA TARTARUGA

Podemos orientar a tartaruga por ângulos dados ou constantes predefinidas:

constante	**Valor (graus)**	**tradução:**
North	0	Norte
East	90	Leste
South	180	Sul
West	270	Oeste

TABELA DE PALETTES

O número da palette define quatro cores diferentes, sendo que uma é sempre a cor do fundo:

	Cor –>	**0**	**1**	**2**	**3**
palette 0	cor	fundo	verde	vermelho	marrom
palette 1	cor	fundo	ciano	magenta	cinza claro
palette 2	cor	fundo	verde claro	vermelho claro	amarelo
palette 3	cor	fundo	ciano claro	magenta claro	branco

APÊNDICES

APÊNDICE 1
MENSAGENS DE ERRO

Erros na Compilação

01	' ; '	faltando.
02	' : '	faltando.
03	' , '	faltando.
04	') '	faltando.
05	' ('	faltando.
06	' = '	faltando.
07	' : = '	faltando
08	' ['	faltando.
09	'] '	faltando.
10	' . '	faltando.
11	' . . '	faltando.
12	BEGIN	faltando.
13	DO	faltando.
14	END	faltando.
15	OF	faltando.
16	PROCEDURE ou FUNCTION faltando.	
17	THEN faltando.	
18	TO ou DOWNTO faltando.	
20	Expressão booleana faltando.	
21	Variável de arquivo faltando.	

22 Constante inteira faltando.
23 Expressão inteira faltando.
24 Variável inteira faltando.
25 Constante inteira ou real faltando.
26 Expressão inteira ou real faltando.
27 Variável inteira ou real faltando.
28 Variável tipo pointer (ponteiro) faltando.
29 Variável tipo record (registro) faltando.
30 Tipo simples faltando.
Qualquer valor escalar, com exceção dos reais, é considerado "tipo simples".
31 Expressão simples faltando.
32 Constante string faltando.
33 Expressão string faltando.
34 Variável string faltando.
35 Arquivo texto faltando.
36 Identificador de tipo faltando.
37 Arquivo sem tipo faltando.
40 Label (etiqueta) indefinida.
Algum comando faz referência a um label (etiqueta) não declarado.
41 Identificador desconhecido ou erro de sintaxe.
Existe um erro de sintaxe ou falta uma definição ou declaração para

um label (etiqueta), constante, variável, identificador de campo ou tipo.

42 Tipo pointer (ponteiro) indefinido na definição de tipo seguinte.
Um pointer definido previamente não corresponde a nenhuma variável declarada.

43 Identificador ou label (etiqueta) já usado neste bloco.

44 Tipo misturado.
Verifique os tipos de variáveis, parâmetros, expressões, operandos e índices das matrizes.

45 Constante fora da faixa.

46 Tipo do seletor do CASE e constante são incompatíveis.

47 Tipos dos operandos são incompatíveis com o operador.

48 Tipo inválido de resultado.
Tipos válidos: tipos escalares, strings e ponteiros.

49 Comprimento da string é inválido (não deve ultrapassar os limites 1 e 255).

50 Comprimento da constante string é incompatível com o tipo.

51 Tipo básico da subfaixa é inválido.
52 Limite inferior é maior que limite superior.
53 Palavra reservada (não pode ser usada como identificador).
54 Atribuição ilegal.
55 Constante string excede linha.
56 Erro na constante inteira.
57 Erro na constante real.
58 Caractere ilegal no identificador.
60 Constantes não são permitidas neste lugar.
61 Arquivos e pointers (ponteiros) não são permitidas neste lugar.
62 Variáveis estruturadas não são permitidas neste lugar.
63 Arquivos de texto não são permitidos neste lugar.
64 Arquivos de texto e arquivos sem tipo não são permitidos neste lugar
65 Arquivos sem tipo não são permitidos neste lugar.
66 Entrada e saída (I/O) não são permitidas neste lugar.
67 Arquivos devem ser parâmetros de VAR.
68 Componentes do arquivo não podem ser outros arquivos.
69 Ordenação dos campos é inválida.

70 Tipo base do conjunto está fora da faixa.

71 GOTO inválido.
Só podemos referenciar um label dentro de um loop FOR se estivermos dentro deste loop.

72 Label (etiqueta) não está dentro do bloco em uso.

73 Procedimento FORWARD indefinido.
Foi usado o comando Forward junto a um subprograma, mas esqueceu-se de definir o subprograma posteriormente.

74 Erro no INLINE.

75 Uso ilegal do ABSOLUTE:
Não pode ser usado em registros, nem aparecer mais de um identificador depois dos dois pontos.

76 Comando FORWARD não pode ser usado com o OVERLAY.

77 OVERLAYS não são permitidos no modo direto (somente em programas compilados em disco).

90 Arquivo não encontrado.

91 Fim inesperado do programa-fonte (deve faltar algum end).

92 Incapaz de criar arquivo overlay.

93 Diretiva de compilação inválida.
97 Muitos WITH entrelaçados (tente usar a diretiva W).
98 Overflow da memória (muitos dados para pouco espaço).
99 Overflow do compilador (não tem espaço na memória para compilação).

Erros na Execução (Run-Time Errors)

01 Overflow de ponto flutuante.
02 Tentativa de divisão por zero.
03 Erro no argumento de Sqrt (argumento negativo).
04 Erro no argumento de Ln (argumento < = 0).
10 Erro no comprimento da String.
11 Índice da string é inválido (fora da faixa entre 1 e 255).
90 Índice fora dos limites (verifique índice da matriz).
91 Escalar ou subfaixa fora do limite.
92 Fora do limite dos inteiros (o valor passou do limite entre – 32768 e 32767).
F0 Arquivo OVERLAY não foi encontrado.
FF Colisão do HEAP com o STACK.

Erros de Entrada e Saída (I/O ERRORS)

Só ocorrerão se a diretiva {$I+} do compilador estiver ativa.

01	Arquivo inexistente.
02	Arquivo não está aberto para entrada de dados.
03	Arquivo não está aberto para saída de dados.
04	Arquivo não está aberto (para BlockRead ou BlockWrite).
10	Erro no formato numérico.
20	Operação não é permitida em dispositivo lógico.
21	Não é permitido no modo direto (o programa deve ser executado a partir do sistema operacional e não do Turbo).
22	Não é permitida atribuição a arquivos standard.
90	Comprimento do registro está misturado (diferente do definido no programa).
91	Procura (seek) depois do fim do arquivo.
99	Fim de arquivo inesperado.
F0	Erro de escrita no disco (Disk Write Error): o disco deve estar cheio.
F1	Diretório cheio.

F2 Overflow no comprimento do arquivo (número do registro é maior que 65535).

F3 Muitos arquivos abertos (somente em MS DOS).

FF Arquivo desapareceu.

APÊNDICE 2
DIRETIVAS DE COMPILAÇÃO

Diretivas são ordens especiais que modificam o comportamento do compilador. Devem ser escritas dentro de chaves e sempre começando com um cifrão. Não pode haver espaço entre o cifrão e a diretiva.

Diretivas para todos os Sistemas

Diretiva default	Descrição
{$B+}	Deixa CON: com buffer de entrada e saída (I/O); {$B–} deixa dispositivo TRM: sem buffer de entrada e saída (I/O).
{$C+}	Admite CTRL-C e CTRL-S durante a execução de READ ou WRITE; {$C–} ignora-os. Veja diretiva {$U–}.
{$I+}	Sistema identifica erros de entrada e saída (I/O) e pára a execução do programa; {$I–} passa o código do

Diretiva default	Descrição
	erro de entrada e saída para o programa através do **IOresult.**
{$I**n_arq** }	O código-fonte de **n_arq** será incluído no seu programa durante a compilação.
{$R–}	{$R+} faz a checagem das faixas de valores dos índices de uma matriz e de valores escalares (é aconselhável usá-la na fase de desenvolvimento do programa); {$R–} não faz esta checagem (se existir um erro, você poderá perder o programa).
{$V+}	Compara se o parâmetro **var** do tipo string tem tamanho e tipo iguais; {$V–} não faz esta comparação.
{$U–}	{$U+} pode-se abortar o programa a qualquer momento com CTRL-C (mas diminui o tempo de processamento); {$U–} o efeito do CTRL-C é dado pela diretiva {$C+}.

Diretivas Exclusivas para CP/M 80

{$A+} Recursividade não é permitida; {$A−} recursividade é permitida (gasta mais memória e é mais lento).

{$Wn} Especifica o número máximo de **with** que pode ser entrelaçado (**n** varia entre 1 e 9 e o default é 2).

{$X+} Otimiza as matrizes em velocidade, mas gasta mais memória; {$X−} otimiza as matrizes em espaço para código-objeto, porém fica mais lento.

Diretivas para MS DOS e PC DOS

{$D+} Dispositivos de entrada e saída (I/O) sem buffer; {$D−} todas as operações de entrada e saída (I/O) usam buffer.

{$Fn} **n** define o número de arquivos que podem ser abertos ao mesmo tempo (default é 16, mas veja se o sistema operacional não está configurado de outra maneira).

{$Gn} A entrada de dados padrão será via CON: ou TRM: (dependendo da diretiva {$B+} também) se **n** for 0 (default). Caso o valor **n** seja maior que 0, será adotado o dispositivo corrente de entrada padrão.

{$Pn} A saída de dados padrão será via CON: ou TRM: (dependendo da diretiva {$B+} também) se **n** for 0 (default). Caso o valor **n** seja maior que 0, será adotado o dispositivo corrente de saída padrão.

Somente para Sistemas 16-Bits

{$K+} Verifica se há disponibilidade de espaço no Stack para as variáveis locais de subprogramas; {$K−} não verifica.

TABELA ASCII

Dec.	Hex.	Caract.		Dec.	Hex.	Caract.	
0	00	^@	NUL	23	17	^W	ETB
1	01	^A	SOH	24	18	^X	CAN
2	02	^B	STX	25	19	^Y	EM
3	03	^C	ETX	26	1A	^Z	SUB
4	04	^D	EOT	27	1B	^[	ESC
5	05	^E	ENQ	28	1C	^\	FS
6	06	^F	ACK	29	1D	^]	GS
7	07	^G	BEL	30	1E	^^	RS
8	08	^H	BS	31	1F	^_	US
9	09	^I	HT	32	20	SP	
10	0A	^J	LF	33	21	!	
11	0B	^K	VT	34	22	"	
12	0C	^L	FF	35	23	#	
13	0D	^M	CR	36	24	$	
14	0E	^N	SO	37	25	%	
15	0F	^O	SI	38	26	&	
16	10	^P	DLE	39	27	'	
17	11	^Q	DC1	40	28	(	
18	12	^R	DC2	41	29	)	
19	13	^S	DC3	42	2A	*	
20	14	^T	DC4	43	2B	+	
21	15	^U	NAK	44	2C	,	
22	16	^V	SYN	45	2D	–	

Dec.	Hex.	Caract.	Dec.	Hex.	Caract.
46	2E	.	70	46	F
47	2F	/	71	47	G
48	30	0	72	48	H
49	31	1	73	49	I
50	32	2	74	4A	J
51	33	3	75	4B	K
52	34	4	76	4C	L
53	35	5	77	4D	M
54	36	6	78	4E	N
55	37	7	79	4F	O
56	38	8	80	50	P
57	39	9	81	51	Q
58	3A	:	82	52	R
59	3B	;	83	53	S
60	3C	<	84	54	T
61	3D	=	85	55	U
62	3E	>	86	56	V
63	3F	?	87	57	W
64	40	@	88	58	X
65	41	A	89	59	Y
66	42	B	90	5A	Z
67	43	C	91	5B	[
68	44	D	92	5C	\
69	45	E	93	5D	]

Dec.	Hex.	Caract.	Dec.	Hex.	Caract.
94	5E	^	111	6F	o
95	5F	_	112	70	p
96	60	`	113	71	q
97	61	a	114	72	r
98	62	b	115	73	s
99	63	c	116	74	t
100	64	d	117	75	u
101	65	e	118	76	v
102	66	f	119	77	w
103	67	g	120	78	x
104	68	h	121	79	y
105	69	i	122	7A	z
106	6A	j	123	7B	{
107	6B	k	124	7C	¦
108	6C	l	125	7D	}
109	6D	m	126	7E	~
110	6E	n	127	7F	DEL

PALAVRAS RESERVADAS

absolute
and
array
begin
case
const
div
do
downto
else
end
external
file
for
forward
function
goto
if
in
inline
label
mod
nil
not
of
or
overlay
packed
procedure
program
record
repeat
set
shl
shr
string
then
to
type
until
var
while
with
xor

IDENTIFICADORES PADRÕES

Addr
ArcTan
Assign
Aux
AuxInPtr
AuxOutPtr
Black
Blink
Blue
BlockRead
BlockWrite
Boolean
Brown
BufLen
BW40
BW80
Byte
C40
C80
Chain
Char
Chr
Close
ClrEol
ClrScr
Con
ConInPtr
ConOutPtr
Concat
ConstPtr
Copy
Cos
CrtExit
CrtInit
Cyan
DarkGray
DelLine
Delay
Delete
Dispose
Draw
East
EOF
EOLN
Erase
Execute

Exit
Exp
False
FilePos
FileSize
FillChar
Flush
Frac
FreeMem
GetMem
GotoXY
GraphBackGround
GraphColorMode
GraphMode
GraphWindow
Green
Halt
HeapPtr
Hi
HiRes
HiResColor
IOresult
Input
InsLine
Insert
Int
Integer
Kbd
KeyPressed
Length
LightBlue
LightCyan
LightGray
LightGreen
LightMagenta
LightRed
Ln
Lo
LowVideo
Lst
LstOutPtr
Mark
MaxInt
Mem
MemAvail
Move
New
NormVideo

North
NoSound
Odd
Ord
Output
Palette
Pi
Plot
Port
Pos
Pred
Ptr
Random
Randomize
Read
ReadLn
Real
Release
Rename
Reset
Rewrite
Round
Seek
Sin
SizeOf
SeekEof
SeekEoln
South
Sqr
Sqrt
Str
Succ
Swap
Text
TextBackGround
TextColor
TextMode
Trm
True
Trunc
Upcase
Usr
UsrInPtr
UsrOutPtr
Val
West
WhereX
WhereY

White
Window
Write
WriteLn
Yellow

OUTROS LIVROS NA ÁREA

Guias do Operador — Comandos Básicos

Burd/Moreira — MSX — Guia do Operador

Compucenter — MS-DOS — Guia do Operador

Garcia — dBase III — Guia do Operador

Gifford — Apple II — Guia do Operador

Ingrahan — CP/M — Guia do Operador

Intercorp — Lotus 1-2-3 — Versão 2.0 — Guia do Operador

Zuccolo — WordStar — IBM PC e seus compatíveis Guia do Operador

Zuccolo — WordStar — Versão 8 bits — CP/M — Guia do Operador

Wilson — IBM PC — Guia do Operador

Wilson — VisiCalc — Guia do Operador

0-07-450363-4