

Volta Redonda - Rio de Janeiro - Brasil

# LINGUAGEM

# PASCAL

Autor: Desconhecido

Data: Aproximadamente entre 1994 a 1996

Revisado por: Alexandre Valim Rocha

Data: Março de 2000

Email: [valimania@uol.com.br](mailto:valimania@uol.com.br)

<http://www.terravista.pt/copacabana/1700>



# 1 - PREFÁCIO

Este curso destina-se a todos aqueles que desejam aprender a linguagem Pascal, através do seu mais famoso compilador para a linha IBM/PC, o Turbo Pascal. O Turbo Pascal é muito mais que um compilador, pois ele é uma associação entre um compilador, um editor de textos e um linkeditor. Desta forma, o Turbo Pascal facilita o ato de programar. Além de tudo isto, o Turbo permite muitas facilidades e atividades que, com certeza, não estavam planejadas por Niklaus Wirth, o criador da linguagem Pascal. Levando-se em conta todas essas considerações, podemos até mesmo dizer que o Turbo Pascal seria uma nova linguagem, mais poderosa que a Pascal.

Gostaria de salientar que a melhor forma de aprender uma linguagem, programando, assim como a melhor forma de aprender a dirigir, entrar num automóvel e sair com ele por aí, ou seja, o simples fato de ler este pequeno e simples curso de linguagem Pascal, não basta para aprender a programar em Pascal.

Por fim, estou a disposição de todos, que se aventurem a acompanhar este curso, para tirar dúvidas, assim como para receber críticas.

## I - Introdução

### I.1 - A linguagem Pascal

Considero que a programação deve ser entendida como uma arte ou técnica de se construir algoritmos, sendo que estes são métodos ou "receitas" para se resolver problemas. Existem diversas linguagens para se programar, umas mais adequadas a certos tipos de algoritmos, outras a outros tipos. No entanto, uma linguagem de programação não deve ser um fim em si mesma, mas um meio, uma ferramenta para se traduzir os algoritmos em programas a serem executados por computadores. Desta forma, é importante que os cursos de programação não tenham como objetivo primordial, a perfeição do conhecimento de uma linguagem específica. A linguagem deve tão somente, refletir de maneira clara e facilmente compreensível os aspectos principais dos algoritmos.

Por tudo isso, devemos ter a preocupação de ensinarmos aos estudantes a formulação sistemática e metódica de algoritmos, através de técnicas que são características da programação.

Como já disse, existem diversas linguagens de programação, podemos aprender e utilizar quantas desejarmos. Dizer qual a melhor é muito relativo. Há os que defendem o Basic, o Cobol, a C, o Pascal e tantas outras. Bom, mas a pergunta crucial que fato aqui é: Qual a primeira linguagem a ser aprendida? Neste ponto, defendo a linguagem Pascal.

De acordo com observações feitas por diversos professores, inclusive por mim, a maior parte das pessoas ficam ligadas para sempre à primeira linguagem que aprenderam, e quando aprendem uma nova linguagem, têm uma certa tendência em desenvolver os algoritmos segundo o vocabulário e regras sintáticas da primeira linguagem, só que escritas na nova.

Por este motivo, acho que a escolha da primeira linguagem a ser ensinada deve ser feita de forma judiciosa.

A primeira linguagem deve, desta forma, ser tal que forneça ao aprendiz a possibilidade de desenvolver algoritmos lógicos, sistemáticos, facilmente compreensíveis segundo os métodos modernos de programação e deve até possibilitá-lo a "dar asas à sua imaginação".

## **I.2 - Por que Turbo Pascal?**

Um computador não pode entender nem tão pouco executar instruções em linguagens de alto nível. Ele só entende linguagem de máquina. Desta forma, os programas em linguagens de alto nível devem ser traduzidos antes de serem executados pelo computador. Quem faz essa tradução são os programas tradutores.

Existem basicamente 2 tipos de programa tradutor: o interpretador; e o compilador. Os dois aceitam como entrada um programa em linguagem de alto nível (fonte) e produzem como saída um programa em linguagem de máquina (objeto). A diferença entre eles está na forma de executar a tarefa de tradução. O interpretador traduz para a linguagem de máquina e roda uma linha por vez, até que todo programa seja executado. Já o compilador traduz para a linguagem de máquina todo o programa fonte e só então ele é executado.

Existem linguagens de programação interpretadas e compiladas. O cobol é compilado, o basic pode ser tanto compilado como interpretado e assim por diante. A linguagem Pascal, tradicionalmente compilada.

Por outro lado, o processo de compilação, de certa forma moroso, pois deve seguir as seguintes etapas:

- Devemos utilizar um editor de textos para escrever e armazenar em disco o nosso programa fonte.
- Utilizar um compilador para traduzir o programa fonte para um programa em linguagem de máquina.
- Finalmente, devemos juntar ao programa compilado as diversas rotinas necessárias que, normalmente, ficam armazenadas numa biblioteca.

Após todo esse processo, suponha que você chegue à conclusão de que o programa tenha que sofrer modificações, pois bem, você terá que repetir os três passos descritos, e assim sucessivamente até que o programa fique ao seu gosto.

O compilador Turbo Pascal facilita todo esse processo, pois ele possui numa forma integrada, um editor de textos compatível com o Wordstar, um compilador e um linkeditor. O processo de compilação pode ser feito tanto em disco como em memória, o que faz com que ele seja muito rápido. Além disso, o Turbo Pascal atende aos padrões da linguagem Pascal definidos por Niklaus Wirth, "o pai da linguagem".

Na realidade, o Turbo Pascal vai muito além, pois ele possui inúmeras Procedures e funções a mais do que as existentes no padrão da linguagem Pascal.

## **I.3 - Equipamento necessário**

Todos os exemplos e programas contidos neste curso, foram escritos num compatível 486DX 50 Mhz, com dois drivers de discos de dupla face de alta densidade, um HD de 340 Mb, um monitor monocromático e 640 Kbytes de memória RAM.

No entanto, a configuração mínima poderia ser um IBM/PC-XT com um HD de 40 Mb.

## II - Um programa em Pascal

### II.1 - O primeiro programa

Bom, acho que aqueles que nunca tiveram a oportunidade de fazer um programa em Pascal, devem estar muito curiosos para saber como deve ser o seu aspecto. Por isso, antes de prosseguir com os meandros da linguagem Pascal, eu mostrarei um pequeno programa devidamente comentado.

Programa Exemplo: Pequeno exemplo de um programa em Pascal. Tem a finalidade única e exclusiva de mostrar os diversos componentes de um programa em Pascal.

{ Tudo que estiver entre chaves são comentários e não são levados em conta pelo compilador. }

```
Program Primeiro_exemplo;      { Este é o cabeçalho do programa }
USES Crt;      { Aqui estou utilizando uma UNIT, chamada CRT, existem várias,
e inclusive você pode criar as suas. Nestas units temos Procedures e Functions
previamente compiladas. }
```

```
Label
```

```
    fim;      { A partir deste instante posso utilizar o label fim. }
```

```
Const
```

```
    Meu_Nome = 'Alex';
```

```
{ Nesta área podemos definir todas as constantes que quisermos utilizar no
programa. }
```

```
Type
```

```
    n =      (Brasileira, portuguesa, inglesa, francesa, alemã, americana);
```

```
{ O Turbo Pascal possui diversos tipos de variáveis predefinidas, mas também
permite definir novos tipos na sub-área type. }
```

```
Var  idade      : integer;
     altura     : real;
     nome       : string[30];
     sexo       : char;
     nacionalidade : n;
```

```
{ Todas as variáveis que forem utilizadas no corpo do programa deverão ser
declaradas na sub-área Var. }
```

```
Procedure Linha;      { A Procedure equivale ao conceito de sub-rotina. Sua
estrutura pode se tornar tão complexa como de um programa. Esta Procedure,
traçar uma linha na posição atual do cursor. }
```

```
Var i:integer;
```

```
Begin
```

```

        For i:=1 to 80 do Write('-');
End;

Function Soma(x,y:integer):integer;

{ O Turbo Pascal possui diversas funções pré-definidas, mas o programador
também pode definir as suas próprias. }

Begin
    Soma:=x+y;
End;

{ Podemos definir quantas Procedures e Functions quisermos. }

{ Aqui começa o programa propriamente dito. }

Begin
    ClrScr;      { Apaga a tela. }
    Linha;      { Executa a Procedure linha. }
    Writeln('Meu nome e -> ',Meu_Nome);
    Linha;
    Write('Qual o seu nome -> ');
    Readln(Nome);
    Linha;
    Write('Qual a sua idade -> ');
    Readln(idade);
    Linha;
    Writeln('Nossas idades somam -> ',Soma(20,idade));
    Linha;
    goto fim;
    { Estas linhas serão puladas. }
    nacionalidade:=Brasileira;
    Write('Minha nacionalidade é brasileira');
fim:
    Write('Prazer em conhece-lo(a)');
End.

```

## II.2 - Estrutura de um programa em Pascal

Todo programa em Pascal, subdividido em 3 áreas:

- Cabeçalho do programa.
- Área de declarações.
- Corpo do programa.

Na definição padrão da linguagem Pascal, o Cabeçalho do programa, obrigatório, no entanto, no Turbo Pascal ele é opcional. A área de declarações é subdividida em seis sub-áreas, a saber:

- Label.
- Const.
- Type.
- Var.
- Procedures.
- Functions.

Darei agora, uma breve explicação de cada sub-área, pois mais para frente estudaremos cada uma delas com profundidade. Na sub-área Label, devemos declarar todos os labels que forem utilizados no corpo do programa. Os labels são utilizados em conjunto com a instrução goto. Todas as constantes que formos utilizar no nosso programa, podem se assim desejarmos, ser definidas na sub-área Const.

O Turbo Pascal tem basicamente 6 tipos de variáveis pré-definidas a saber: Integer, Real, Byte, Boolean, Char e String. No entanto, podemos definir novos tipos de variáveis na sub-área Type.

Todas as variáveis utilizadas no programa devem ser declaradas na sub-área Var, pois a alocação de espaço de memória para as variáveis, feita durante a compilação. Na sub-área Procedures, podemos definir quantas sub-rotinas quisermos. Elas são chamadas durante o programa pelos seus respectivos nomes.

Finalmente, na sub-área Functions podemos definir novas funções que depois poderemos utilizar no programa embora o Turbo Pascal possua inúmeras funções pré-definidas. Estas sub-áreas só são obrigatórias caso nós estejamos precisando. Exemplo: se não vamos utilizar variáveis no nosso programa (coisa rara) então não precisamos utilizar a sub-área Var. De acordo com a definição padrão da Linguagem Pascal, estas sub-áreas devem aparecer na seqüência que foi dada anteriormente, ou seja, Label - Const - Type - Var - Procedures - Functions. Mas no Turbo Pascal isto é livre.

Por fim, como dito no programa exemplo, existe a possibilidade de se usar a declaração USES, que nos permite utilizar UNITS que nada mais são do que bibliotecas de funções e Procedures previamente declaradas.

### **III - Noções Básicas preliminares**

#### **III.1 - Elementos básicos do Turbo Pascal**

##### **III.1.1 - Caracteres utilizados**

Os caracteres que podem ser utilizados no Turbo Pascal são divididos em:

Letras : 'A' até 'Z', 'a' até 'z'  
 Números : 0,1,2,3,4,5,6,7,8 e 9  
 Especiais : + - \* / = ^ < > ( ) [ ] { } . , : ; ' # \$

Observações:

1) O Turbo Pascal não faz distinção entre letras maiúsculas e minúsculas, de tal forma que no desenvolvimento deste curso eu utilizarei os dois tipos da forma

que achar mais conveniente.

2) Embora na maioria das linguagens o sinal de atribuição de valores a variáveis seja o =, em Pascal, o símbolo de atribuição é := .

Exemplos:

```
A = 100      Em Basic.  
A := 100     Em Pascal.
```

3) Dois pontos em seguida (..) indica um delimitador de faixa. Exemplo:

1..30 -> Todos inteiros entre 1 e 30 inclusive.

### III.1.2 - Palavras reservadas

As palavras reservadas do Turbo Pascal são palavras que fazem parte da sua estrutura e têm significados pré-determinados. Elas não podem ser redefinidas e não podem ser utilizadas como identificadores de variáveis, Procedures, Functions etc. Algumas das palavras reservadas são:

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	Packed	Procedure
Program	Record	Repeat	Set
Shl(*)	Shr(*)	String(*)	Then
To	Type	Until	Var
While	With	xor(*)	

(\*) -> Não definidos no Pascal Standard.

### III.1.3 - Identificadores pré-definidos

O Turbo Pascal possui inúmeros identificadores pré-definidos, que não fazem parte da definição padrão da linguagem Pascal. Esses identificadores consistem em Procedures e Functions, que podem ser utilizados normalmente na construção de programas.

Exemplos:

```
ClrScr   : Limpa a tela de vídeo.  
DelLine  : Deleta a linha em que está o cursor e assim por diante.
```

Constantemente, novas Procedures e Functions estão sendo criadas pela Borland (criadora do Turbo Pascal), aumentando desta forma o número de identificadores. São UNITS que tornam o Turbo Pascal mais poderoso do que ele já é.

Regras para formação de identificadores:

O usuário também pode definir seus próprios identificadores, na verdade nós somos obrigados a isso. Nomes de variáveis, de labels, de Procedures, Functions, constantes etc. São identificadores que devem ser formados pelo programador. Mas para isso existem determinadas regras que devem ser seguidas:

- 1) O primeiro caractere do identificador deve ser obrigatoriamente uma letra ou um underscore (\_).
- 2) Os demais caracteres podem ser letras, dígitos ou underscores.
- 3) Um identificador pode ter no máximo 127 caracteres.
- 4) Como já dissemos anteriormente, não pode ser palavra reservada.

Exemplos de identificadores válidos:

```
Meu_Nome
MEU_NOME      Igual ao anterior.
__Linha
EXemplo23
```

Exemplos de identificadores não válidos:

```
2teste        Começa com número.
Exemplo 23    Tem um espaço.
```

### III.1.4 - Comentários

Comentários são textos que introduzimos no meio do programa fonte com a intenção de torná-lo mais claro. É uma boa prática em programação inserir comentários no meio dos nossos programas. No Turbo Pascal, tudo que estiver entre os símbolos (\* e \*) ou { e } ser considerado como comentário.

### III.1.5 - Números

No Turbo Pascal, podemos trabalhar com números inteiros e reais, sendo que os números inteiros podem ser representados na forma hexadecimal, para tanto, basta precedê-los do símbolo \$. Os números reais também podem ser representados na forma exponencial.

Isso tudo varia de versão para versão do turbo Pascal, citarei aqui as faixas de valores válidas para a versão 7.0:

Tipo	Faixa	Formato
Shortint	-128..127	Com sinal 8-bit
Integer	-32768..32767	Com sinal 16-bit
Longint	-2147483648..2147483647	Com sinal 32-bit
Byte	0..255	Sem sinal 8-bit
Word	0..65535	Sem sinal 16-bit

Tipo	Faixa	Digitos	Bytes
real	2.9e-39..1.7e38	11-12	6
single	1.5e-45..3.4e38	7-8	4
double	5.0e-324..1.7e308	15-16	8
extended	3.4e-4932..1.1e4932	19-20	10
comp	-9.2e18..9.2e18	19-20	8

### III.1.6 - Strings

Strings são conjunto de caracteres entre aspas simples.

Exemplos:

'isto é uma string'.

'123456'.

Etc.

### III.1.7 - Caracteres de controle

Existem alguns caracteres que têm significados especiais. São os caracteres de controle.

Exemplos:

Control G -> Bell ou beep.

Control L -> Form Feed.

Etc.

Em Turbo Pascal, também podemos utilizar estes caracteres. Para tanto, eles devem ser escritos pelo seus valores ASCII correspondentes, precedidos do símbolo #, ou então a letra correspondente precedida do símbolo ^.

Exemplo:

Control G -> #7 ou ^G.

## III.2 - Definição de variáveis

Como já dissemos, todas as variáveis que forem utilizadas no corpo do programa, devem ser declaradas numa sub-área específica chamada Var.

Para estudarmos essa sub-área devemos primeiro ver os tipos de variáveis pré-definidos em Turbo Pascal.

### III.2.1 - Tipos de dados pré-definidos

Os tipos de dados pré-definidos em Turbo Pascal são divididos em duas categorias:

Escalares Simples:

- Char.
- Boolean.
- Todos os tipos de inteiros citados acima.
- Todos os tipos de reais citados acima.

Escalares estruturados:

- String.
- Array.
- Record.
- File.
- Set.
- Text.

Inicialmente, iremos estudar os escalares simples e o tipo String pela sua utilização prática inicial. Os demais tipos estruturados serão vistos mais para a frente.

Char: O tipo char corresponde a todos os caracteres que podem ser gerados pelo teclado tais como dígitos, letras e símbolos tais como &, #, \* e etc.

Os caracteres devem vir entre aspas simples.

Boolean: O tipo boolean só pode assumir os valores FALSE e TRUE.

String: Este tipo é chamado de estruturado ou composto pois, constituído a partir de um tipo simples que é o char. O tipo string, composto por um conjunto de caracteres entre aspas simples.

Shortint, Integer, Longint, Byte, Word: Ver tabela acima.

Real, Single, Double, Extended, Comp: Ver tabela acima.

### III.2.2 - A declaração Var

Esta é a sub-área onde devemos declarar todas as variáveis que iremos utilizar em nosso programa.

Exemplo:

```
Program Exemplo;      (* Cabeçalho do programa. *)
```

```
Var
```

```
    idade, numero_de_filhos : byte;
    altura                   : real;
    sexo                     : char;
    nome                     : string[30];
    sim_ou_ao                : boolean;
    quantidade               : integer;
```

```
(* Aqui começa o programa. *)
```

```
Begin
```

```
    idade:=34;
    numero_de_filhos:=2;
```

```
sexo:='M';
nome:='José';
sim_ou_nao:=TRUE;
quantidade:=3245;
```

End.

Observações importantes:

- 1) A palavra reservada Var aparece uma única vez num programa.
- 2) A sintaxe geral para declaração de variáveis:

```
variável_1, variável_2, ..., variável_n : tipo;
```

3) Os espaços e comentários separam os elementos da linguagem. Você pode colocar quantos espaços quiser. Observe:

Var idade:integer;	O compilador não reconhece a palavra Var.
Var idade:integer;	Agora sim, ou se preferir.
Var idade : integer;	Assim dá na mesma.

4) As instruções são separadas entre si por ponto e vírgula ';'. Se você quiser, pode colocar mais de uma instrução numa única linha.

Lembre-se que o limite de caracteres numa linha é de 127.

5) O tipo string deve ser precedido da quantidade máxima de caracteres que a variável pode assumir. Lembre-se que a alocação de espaço de memória para as variáveis é feita durante a compilação, portanto o compilador precisa saber desse dado. Por outro lado, o fato de termos, por exemplo, atribuído o valor máximo de 30 não significa que tenhamos que utilizar os 30 caracteres e sim no máximo 30.

6) Como última observação, acho muito mais claro e elegante declarar variáveis e ao mesmo tempo informar com linhas comentários os devidos motivos.

Exemplo:

```
Var
  idade,      (* Idade de determinada pessoa. *)
  i,j        (* Utilizadas em loops. *)
  : integer;

  nome1,     (* nome genérico de pessoas. *)
  nome2     (* nome genérico de pessoas. *)
  : string[50];
```

### III.2.3 - A declaração type

Além dos tipos de dados pré-definidos no Turbo Pascal, podemos também definir novos tipos através da declaração Type. A sua sintaxe geral é:

```
Type identificador = (valor1, valor2, valor3, valor4 ,... , valorN);
```

O identificador deve seguir as regras dadas anteriormente e entre os parênteses estão os valores que podem ser assumidos.

Exemplos:

Type

```
cor      = (azul, vermelho, branco, verde, amarelo);
dia_util = (segunda, terça, quarta, quinta, sexta);
linha    = string[80];
idade    = 1..99;
```

(\* A partir deste instante, além dos tipos de dados pré-definidos, podemos também utilizar os novos tipos definidos cor, dia\_util, linha e idade. \*)

Var

```
i      : integer;
d      : idade;
nome   : linha;
dia    : dia_util;
cores  : cor;
```

(\* Etc. \*)

Observação: Quando damos os valores que os dados podem assumir através da declaração type, o Turbo Pascal assume, automaticamente, que o valor da direita vale mais que o da esquerda e assim por diante. Por exemplo: no caso da definição de cor, amarelo vale mais que verde, que por sua vez vale mais que branco e assim por diante.

### III.3 - Constantes

#### III.3.1 - A declaração Const

Nesta sub-área, podemos definir tantas constantes quantas quisermos.  
Sintaxe:

Const

```
meu_nome = 'Alex';
cor_preferida = 'verde';
numero_maximo = 24345;
```

(\* E assim por diante. \*)

Toda vez que nos referirmos às constantes acima, o Turbo Pascal substituí-las-á pelos seus respectivos valores.

#### III.3.2 - Constantes pré-definidas

Existem algumas constantes pré-definidas e que podemos utiliza-las sem ter que declará-las. São elas:

```
PI = 3.1415926536E + 00
FALSE
TRUE
NIL           Pointer nulo, veremos mais adiante.
MAXINT = 32767
```

### III.3.3 - Constantes tipadas

A declaração de variáveis na sub-área Var, apenas reserva espaço de memória para elas, mas não as inicializa, ou seja, até que se atribua valores a elas, seus valores serão desconhecidos. Sob certas circunstâncias, seria interessante que pudéssemos ao mesmo tempo em que declaramos a variável, dar seu valor inicial. Isto é possível com o conceito de constante tipada cuja sintaxe é:

```
Const variável : tipo = valor;
```

Exemplos:

```
Const Contador : integer = 100;
  c : char = 'A';
```

Estamos definindo duas variáveis, uma chamada contador que é inteira e vale inicialmente 100, e outra chamada c que é do tipo char e cujo valor inicial é 'A'.

## III.4 Operadores

### III.4.1 - Operadores aritméticos

```
+  adição.
-  subtração.
*  multiplicação.
/  divisão entre números reais.
DIV divisão entre números inteiros.
MOD resto da divisão.
```

Programa exemplo: Mostra como utilizar operadores aritméticos.

```
Program Operadores_aritmeticos;
Uses CRT;

Var  x,y,z : integer;
     r1,r2 : real;
Begin
```

```

    ClrScr;    (* Limpa a tela. *)
    x:=10;
    y:=20;
    z:=x+y;
    Writeln(z);    (* Escreve o valor de z na tela de vídeo. *)
    x:= 20 DIV 3;
    y:= 20 MOD 3;
    Writeln(x);    (* Escreve 6 na tela. *)
    Writeln(y);    (* Escreve 2 na tela. *)
    r1:=3.24;
    r2:=r1/2.3;
    Writeln(r2);
End.

```

### III.4.2 - Operadores lógicos

```

AND      E lógico.
OR       OU lógico.
XOR     OU EXCLUSIVO lógico.

```

Estes operadores só aceitam como operandos, valores lógicos, ou seja: TRUE e FALSE.

A operação AND resulta em TRUE se e somente se todos os operandos forem TRUE, se um deles ou mais de um for FALSE então o resultado ser FALSE.

A operação OR resulta TRUE quando pelo menos um dos operandos for TRUE.

A operação XOR resulta TRUE quando os operandos forem diferentes entre si, isto é, quando um for TRUE o outro dever ser FALSE.

Exemplo:

Programa Utilizando os operadores lógicos.

```

Program operadores_logicos;
Uses CRT;

Var x,y : boolean;

Begin
    x:=TRUE;
    y:=FALSE;
    Writeln( x OR y );    (* Escreve TRUE. *)
    Writeln( x AND y );  (* Escreve FALSE. *)
    Writeln( x XOR y );  (* Escreve TRUE. *)
End.

```

### III.4.3 - Operadores relacionais

O Turbo Pascal possui ao todo 7 operadores relacionais que são muito utilizados nas tomadas de decisões, são eles:

= Igual.  
<> Diferente.  
> Maior que.  
< Menor que.  
>= Maior ou igual que.  
<= Menor ou igual que.  
IN Testa se um elemento está incluso em um conjunto.

Exemplos:

1) Se A=30 e B=50 então.

( A = B ) FALSE.  
( A < B ) TRUE.

2) Se A=TRUE e B=FALSE.

( A <> B ) TRUE.  
( A = B ) FALSE.

3) Se A=50, B=35, C='A' e D='B'.

( ( A < B ) OR ( C < D ) ) TRUE.

A avaliação ser verdadeira se uma ou outra expressão for verdadeira, no caso, como C < D então a resposta é TRUE.

### III.4.4 - Operadores entre bits

Os operadores entre bits só podem ser aplicados em dados dos tipos byte ou integer e o resultado é do tipo integer. Eles agem bit a bit e podem ser aplicados na notação hexadecimal ou decimal. São eles:

SHL - SHift Left

Desloca n bits à esquerda. Durante o deslocamento, os bits à esquerda são perdidos e dígitos zeros preenchem a posição direita.

Exemplos:

1) Se X = 00010101 então.

X Shl 2 = 01010100.  
X Shl 5 = 10100000.

2) 55 Shl 3 = 184.

55 = 00110111 deslocando 3 à esquerda ficaria:  
10111000 que é igual a 184.

3) \$F0 Shl 2 = \$C0.

\$F0 = 11110000 deslocando 2 à esquerda ficaria:  
11000000 que é igual a \$C0.

SHR - SHift Right.

Desloca n bits à direita. Durante o deslocamento, os bits à esquerda são preenchidos com zeros e os da direita são perdidos.

Exemplos:

1) Se X = 10101100 então:

X Shr 3 = 00010101.

X Shr 6 = 00000010.

2) 55 Shr 3 = 6.

55 = 00110111 deslocando 3 à direita ficaria:  
00000110 que é igual a 6.

3) \$F0 Shr 2 = \$3C.

\$F0 = 11110000 deslocando 2 à direita ficaria:  
00111100 que é igual a \$3C

OBS: Já sei, você não entende a numeração de base 2, bem vou tentar em poucas palavras explicar a base 2. Nós operamos na base 10, porque trabalhamos com 10 algarismos, 0..9, certo? Bem na base 2 operamos somente com 2 algarismos, o 0 e o 1. Dessa forma, temos que representar todos os números da base 10 utilizando somente o 0 e 1. Parece complicado? Nem tanto, veja abaixo a correspondência:

Base 10	Base 2
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010

11                    1011

E assim por diante.

Para converter um número da base 10 para a base 2, basta dividir o número, o qual queremos converter, por dois sucessivamente até que o resto seja 0, depois pegamos os restos de baixo para cima.

Exemplo:

(23)  $\rightarrow$  (     )  
10                    2

23 / 2 = 11 e sobra 1  
11 / 2 = 5 e sobra 1  
5 / 2 = 2 e sobra 1  
2 / 2 = 1 e sobra 0  
1 / 2 = 0 e sobra 1

Portanto (23)  $\rightarrow$  (10111)  
10                    2

Para converter da base 2 para a base 10, devemos fazer ao contrário:

(10111)  $\rightarrow$  (     )  
2                    10

4	3	2	1	0		4		3		2		1		0													
(	1	0	1	1	1	)		1	x	2	+	0	x	2	+	1	x	2	+	1	x	2	+	1	x	2	=
								16			+	0			+	4			+	2			+	1			= 23

NOT

O operador NOT nega os bits, isto é, os bits iguais a 1 se tornam 0 e os bits zero se tornam 1. Devemos lembrar, no entanto, que os inteiros possuem 2 bytes, portanto, ao se trabalhar com números decimais inteiros ser afetado o byte de mais alta ordem e também o sinal.

Exemplo:

NOT (255) = -256.

Para suprimir este problema, você deve trabalhar com bytes:

Program Exemplo;

Uses CRT;

Var i,j : Byte;

Begin

```
    ClrScr;
    i:=255;
    j:=NOT(i);
    Writeln(j);    (* Será escrito 0. *)
End.
```

## AND

Este operador realiza a operação E lógico bit a bit. Relembrando, a operação E resulta em 1 se e somente se os dois operandos forem iguais a 1, caso contrário, o resultado será igual a 0.

Exemplos:

- 1) \$0F AND \$F0 = \$0 pois.  
\$0F = 00001111.  
\$F0 = 11110000.  
00001111 AND 11110000 = 00000000.
- 2) 255 AND 55 = 55 pois.  
255 = 11111111.  
55 = 00110111.  
11111111 AND 00110111 = 00110111.
- 3) 34 AND 76 = 0 pois.  
34 = 00100010.  
76 = 01001100.  
00100010 AND 01001100 = 00000000.

## OR

Este operador realiza a operação OU lógico bit a bit. Relembrando, a operação OU resulta em 1 se um ou os dois operandos forem iguais a 1.

Exemplos:

- 1) \$0F OR \$F0 = \$FF pois.  
\$0F = 00001111.  
\$F0 = 11110000.  
00001111 OR 11110000 = 11111111.
- 2) 255 OR 55 = 255 pois.  
255 = 11111111.  
55 = 00110111.  
11111111 OR 00110111 = 11111111.
- 3) 34 OR 76 = 110 pois.  
34 = 00100010.  
76 = 01001100.  
00100010 OR 01001100 = 01101110.

XOR

Este operador realiza a operação OU EXCLUSIVO lógico bit a bit. Relembrando, a operação OU EXCLUSIVO resulta em 1 se os operandos forem diferentes entre si.

Exemplos:

- 1) \$0F XOR \$F0 = \$FF pois.  
\$0F = 00001111.  
\$F0 = 11110000.  
00001111 XOR 11110000 = 11111111.
- 2) 255 XOR 55 = 200 pois.  
255 = 11111111.  
55 = 00110111.  
11111111 XOR 00110111 = 11001000.
- 3) 34 XOR 76 = 110 pois.  
34 = 00100010.  
76 = 01001100.  
00100010 XOR 01001100 = 01101110.

### III.4.5 - Concatenação

Esta operação, representada pelo sinal de adição, ou seja, +. Os operandos devem ser do tipo string ou char.

Exemplo:

```
'Isto é uma ' + 'String' = 'Isto é uma String'.
```

## IV - Entrada e saída de dados

### IV.1 - Write e Writeln

Estas são as principais Procedures destinadas a exibir todos os tipos de dados no vídeo. A diferença entre Write e Writeln reside no fato de que a Procedure Write escreve o parâmetro, e mantém o cursor do lado daquilo que foi escrito, enquanto que Writeln passa o cursor para a próxima linha. Estas Procedures possuem 3 formas de sintaxes, a saber:

Primeira forma:

```
Write(parâmetro_1, Parâmetro_2, ...);
```

Exemplo:

```
Program Exemplo;  
Uses CRT;
```

```

Var  i : integer;
     r : real;
     c : char;
     s : string[20];

Begin
  ClrScr;      (* Apaga a tela e coloca o cursor em 1,1. *)
  Writeln('Exemplos de aplicação de Writeln e Write. ');
  Writeln;    (* Apenas pula uma linha. *)
  i:=100;
  r:=3.14;
  c:='A';
  s:='Interessante.';
  Writeln('Valor de i é igual a ',i);
  Write('Valor de r = ');
  Writeln(r);
  Writeln(c,' ',s);
End.

```

Este programa resultaria na seguinte tela:

---

Exemplos de aplicação de Writeln e Write.

Valor de i é igual a 100  
 Valor de r = 3.1400000000E+00  
 A interessante.

---

Segunda forma:

```
Write(parâmetro : n);
```

Onde n é um número inteiro que determina quantas colunas o cursor deve ser deslocado à direita, antes do parâmetro ser escrito. Além disso, o parâmetro é escrito da direita para a esquerda.

Exemplo:

```

Program Exemplo;
Uses CRT;
Begin
  Writeln('A');
  Writeln('A':5);
End.

```

Resultaria a seguinte tela:

---

A

.....A

---

Os pontos representam espaços em branco.

Terceira forma:

```
Write(parâmetro : n : d);
```

Neste caso, n tem a mesma função que o caso anterior sendo que d representa o número de casas decimais. Obviamente, parâmetro terá que ser do tipo Real.

Exemplo:

```
Program Exemplo;
Uses CRT;
Var r : real;

Begin
  ClrScr;
  r:=3.14156;
  Writeln(r);
  Writeln(r:10:2);
End.
```

Resultaria a seguinte tela:

---

```
3.1415600000E+00
   3.14
```

---

## IV.2 - Read e Readln

Estas Procedures são utilizadas para fazer leitura de dados via teclado. A Procedure Read lê um dado do teclado até que se pressione a tecla ENTER, sendo que cada tecla digitada é ecoada para o vídeo. Após pressionarmos ENTER, o cursor permanecer no mesmo lugar. Já, a Procedure Readln faz a mesma coisa só que o cursor passa para a próxima linha. A sintaxe geral para estas Procedures é:

```
Read (Var_1,Var_2,Var_3,...);
```

Ao se digitar os valores das variáveis pedidas, deve-se separá-los por espaços.

Exemplo 1:

```
Program Teste;
Uses CRT;

Var a,b,c:integer;
```

```
Begin
  ClrScr;
  Readln(a,b,c);
  Writeln (a,' ',b,' ',c);
End.
```

Exemplo 2:

```
Program teste;
Uses CRT;
```

```
Var  i : integer;
     r : real;
     c : char;
     s : string[10];
```

```
Begin
  ClrScr;
  Write('Digite um numero inteiro -> ');
  Readln(i);
  Write('Digite um numero real -> ');
  Readln(r);
  Write('Digite um caractere -> ');
  Readln(c);
  Write('Digite uma String -> ');
  Readln(s);
  Writeln;Writeln;      (* Pula duas linhas. *)
  Writeln(i);
  Writeln(r);
  Writeln(c);
  Writeln(s);
End.
```

Exemplo 3:

Programa Área de triângulos : Calcula área de triângulos.

```
Program Area_de_triangulos;
Uses CRT;
```

```
Var base,      (* Base do triângulo. *)
    altura: Real;  (* Altura do triângulo. *)
```

```
Begin
  ClrScr;
  Writeln('Calculo da área de triângulos':55);
  Writeln;
  Write('Valor da base -> ');
  Readln(base);
```

```

    Writeln;
    Write('Valor da altura -> ');
    Readln(altura);
    Writeln;
    Writeln;
    Writeln('Área do triângulo = ',base*altura/2 : 10 : 2);
End.

```

ReadKey: Lê uma tecla do teclado, sem que seja necessário pressionar a tecla ENTER.

```

Program Exemplo;
Uses CRT;
Var tecla:char;
Begin
    Write('Digite uma tecla ->');
    Tecla:=Readkey;
    Writeln;
    Writeln('Você digitou ',tecla);
End.

```

### IV.3 - Impressora

Podemos enviar dados para a impressora através das Procedures Write e Writeln. Para tanto, devemos colocar, antes dos parâmetros a serem enviados à impressora, o nome lógico LST.

Exemplo:

```

Writeln('Isto vai para o vídeo');
Writeln(LST,' Isto vai para a impressora',' e isto também');

```

## IV.4 - Funções e Procedures para controle de vídeo

### IV.4.1 - ClrScr

Esta Procedure tem a finalidade de limpar a tela de vídeo e colocar o cursor na primeira coluna da primeira linha. A tela de vídeo é dividida em 80 colunas e 25 linhas. O canto superior esquerdo tem coordenadas (1,1) e o inferior direito (80,25).

### IV.4.2 - Gotoxy(x,y)

Move o cursor para a coluna x e linha y.

Exemplo:

```

Program Exemplo;
Uses CRT;

```

```

Var x,y : Byte;

Begin
  ClrScr;
  Gotoxy(10,2);
  Write('Coluna 10 da linha 2');
  x:=40;
  y:=10;
  Gotoxy(x,y);
  Write('Coluna 40 da linha 10');
End.

```

### IV.4.3 - ClrEol

Esta Procedure limpa desde a posição atual do cursor até o final da linha.

### IV.4.4 - CrtExit

Envia para a tela de vídeo a String de finalização definida na instalação.

### IV.4.5 - CrtInit

Envia para a tela de vídeo a String de inicialização definida na instalação.

### IV.4.6 - DelLine

Procedure que elimina a linha em que está o cursor. As linhas posteriores sobem, ocupando a que foi eliminada.

Exemplo:

```

Program exemplo;
Uses CRT;
Begin
  ClrScr;
  Writeln('Linha 1');
  Writeln('Linha 2');
  Writeln('Linha 3');
  Writeln('Linha 4');
  Gotoxy(1,2);    (* Posicionei o cursor no início da linha 2. *)
  DelLine;
End.

```

O programa anterior resultaria a seguinte tela:

---

```

Linha 1
Linha 3
Linha 4

```

---

Repare que a string 'Linha 2' foi eliminada.

#### **IV.4.7 - HighVideo**

Coloca o vídeo no modo normal. Esta Procedure é equivalente a NormVideo.

#### **IV.4.8 - InsLine**

Esta Procedure faz exatamente o contrário de DelLine, ou seja, insere uma linha na posição atual do cursor.

Exemplo:

```
Program Exemplo;
```

```
Begin
```

```
  ClrScr;
```

```
  Writeln('Linha 1');
```

```
  Writeln('Linha 2');
```

```
  Writeln('Linha 3');
```

```
  Writeln('Linha 4');
```

```
  Gotoxy(1,3);    (* Cursor na 1º coluna da 3º linha. *)
```

```
  InsLine;
```

```
  Write('Teste');
```

```
  Gotoxy(1,20);
```

```
End.
```

Este programa resultaria a seguinte tela:

---

```
Linha 1
```

```
Linha 2
```

```
Teste
```

```
Linha 3
```

```
Linha 4
```

---

#### **IV.4.9 - LowVideo**

Coloca o vídeo em baixa intensidade até que se execute a Procedure NormVideo ou HighVideo.

#### **IV.4.10 - NormVideo**

O mesmo que HighVideo.

#### **IV.4.11 - TextBackground**

Esta Procedure seleciona a cor do fundo sobre o qual o texto ser escrito. Sua sintaxe geral é:

```
TextBackground(cor);
```

Tabela de cores:

0	Black	Preto
1	Blue	Azul
2	Green	Verde
3	Cyan	Ciano
4	Red	Vermelho
5	Magenta	Magenta
6	LightGray	Cinza-claro

Nós podemos entrar com o número ou o nome da cor em inglês. Exemplo:

```
Program Exemplo;  
Uses CRT;  
  
Begin  
  ClrScr;  
  Writeln('Teste');  
  TextBackground(7);  
  Writeln('Teste');  
  TextBackground(Brown);  
  Writeln('Teste');  
End.
```

#### IV.4.12 - TextColor

Esta Procedure permite selecionar a cor com que o texto ser imprimido.

Tabela de cores:

0	Black	Preto
1	Blue	Azul
2	Green	Verde
3	Cyan	Ciano
4	Red	Vermelho
5	Magenta	Magenta
6	Brown	Marrom
7	LightGray	Cinza-claro
8	DarkGray	Cinza-escuro
9	LightBlue	Azul-claro
10	LightGreen	Verde-claro
11	LightCyan	Ciano-claro

12	LightRed	Vermelho-claro
13	LightMagenta	Magenta-claro
14	Yellow	Amarelo
15	White	Branco
16	Blink	Piscante

Exemplo:

```

Program Exemplo;
Uses CRT;

Begin
  ClrScr;
  TextBackground(7);
  TextColor(Black);
  Writeln('Teste');
  TextColor(Black+Blink);
  Write('Teste');
End.

```

#### IV.4.13 - Window

Sintaxe: `Window(x1,y1,x2,y2);`

Esta Procedure tem o poder de definir uma janela de texto cujo canto esquerdo superior é x1,y1 e canto inferior direito é x2,y2. Após esta instrução, as instruções ClrScr, Write Writeln agem somente dentro da janela recém definida. A instrução Gotoxy passa a utilizar como referencial o ponto x1,y1 que passa a ser considerado 1,1.

Exemplo:

```

Program Exemplo;
Uses CRT;
Begin
  Window(10,10,70,20);
  ClrScr; (* Limpa somente a janela. *);
  Writeln('Teste'); (* Escreve 'Teste' em 10,10. *)
End.

```

#### IV.4.14 - WhereX

Função que retorna o número da coluna onde está o cursor.

#### IV.4.15 - WhereY

Função que retorna o número da linha onde está o cursor.

### IV.5 - Controle do teclado

### IV.5.1 - Kbd

Quando quisermos ler dados do teclado e que não sejam ecoados para o monitor de vídeo até que sejam processados e aceitos, nós podemos utilizar a seguinte sintaxe:

```
Read(Kbd, variável);
```

No caso de números inteiros ou reais, o número só ser aceito quando pressionarmos a tecla <Enter>, no caso de variáveis do tipo char, o caractere ser aceito sem que seja necessário pressionar a tecla <Enter>, idem para o tipo string.

Exemplo:

```
Program Exemplo;
Uses CRT;
Var i:integer;

Begin
  ClrScr;
  Write('Entre com um inteiro -> ');
  Readln(Kbd, i);
  Writeln(i);
End.
```

### IV.5.2 - BufLen

BufLen é uma variável interna pré-definida em Turbo Pascal cujo valor inicial é 126. Ela contém o número máximo de caracteres aceitos por Read.

Exemplo:

```
Program Exemplo;
Uses CRT;

Var i : Integer;

Begin
  ClrScr;
  Writeln(BufLen);    (* Escreve 126. *)
  BufLen:=2;
  Write('Digite um inteiro -> ');
  Readln(i);        (* Se você tentar digitar inteiros com mais de dois
dígitos, Readln não permitirá. *)
End.
```

### IV.5.3 - KeyPressed

O identificador KeyPressed é uma função especial do Turbo Pascal que retorna um valor booleano - TRUE se uma tecla foi pressionada, ou FALSE caso contrário. Ela é muito utilizada para detectar teclas pressionadas no teclado.

Exemplo:

```
Program Exemplo;  
Uses CRT;  
Begin  
  ClrScr;  
  Write('Pressione uma tecla -> ');  
  Repeat Until KeyPressed;    (* Repita até que uma tecla seja  
pressionada. O comando Repeat Until será estudado mais adiante. *)  
End.
```

Aguarde uma possível continuação.  
Se interessar entre em contato,  
para criticas e sugestões!

[valimania@uol.com.br](mailto:valimania@uol.com.br)

