
Tutorial de Turbo Pascal

Para se completar este tutorial é necessário ter um mínimo de conhecimento em programação e de inteligência.

- 1- Tipos de Variáveis
- 2- Units
- 3- Comandos Básicos
- 4- Estruturas Condicionais
 - 4.1 If.. Then.. Else
 - 4.2 CASE
- 5- Estruturas de Repetição
 - 5.1 For
 - 5.2 While
 - 5.3 Repeat
- 6- Manipulação de Strings
- 7- Functions e Procedures
- 8- Arquivos e Registros

BEGIN

- Todo programa começa com a declaração **PROGRAM** seguido de um nome do programa.
- Após deve-se declarar as unidades.
- Declarar as variáveis.
- O algoritmo em si começa com **BEGIN** e termina com **END**. (não esqueça do ponto no end);
- Todas as instruções terminam com ; .

Ex.:

```
Program example;  
Uses crt;  
Var x : integer;  
    y : real;
```

```
Begin
  {Aqui vai o algoritmo}
End.
```

1 - Tipos de VARIÁVEIS

- Boolean : ocupa 1 byte, só pode ter os valores True ou False (Verdadeiro ou Falso);
- Real : ocupa 6 bytes, seus valores vão de 1E-38 até 1E+38;
- Integer : ocupa 2 bytes, seus valores vão de -32768 até 32767, do tipo inteiro;
- Word : ocupa 2 bytes, inteiro de 0 a 65535;
- Longint : ocupa 4 bytes, valores inteiros de -2147483648 a 2147483647;
- Shortint : inteiros de -128 a 127;
- Byte : ocupa 1 byte, tipo inteiro de 0 a 255;
- Char : ocupa 1 byte, tipo alfanumérico, seu conteúdo é qualquer valor da tabela ASCII;
- String : ocupa de 2 a 256 bytes, cadeia de caracteres;

As próximas variáveis são só para o Turbo Pascal 5 ou superior

- Single : tipo real com 7 dígitos
- Double : tipo real com 15 dígitos
- Extended : tipo real com 19 dígitos
- Comp : inteiros de -10E18 até 10E18

Obs: A atribuição de um valor a uma variável é com :=

Soma:=(1+2+3+4);

2- UNITS

As units são rotinas separadas do programa principal. Para usar uma unit deve se declarar Uses.

- CRT: rotinas de vídeo e som;
- DOS: Controles do SO;
- GRAPH: Rotinas gráficas;
- PRINTER: Define LST como arquivo de texto direcionado para impressora;

É possível se criar uma unit própria, para ser usada em vários programas.

Ex:

```
UNIT Exemplo;
INTERFACE
  PROCEDURE Logo;
```

IMPLEMENTATION

```
USES Crt;
```

```
VAR C : Integer;
```

```
PROCEDURE Logo;
```

```
BEGIN
```

```
For C:=1 to 15 do
```

```
TextColor(c);
```

```
WriteLn('|-----|');
```

```
WriteLn(' Exemplo de um logo  ');
```

```
WriteLn('|-----|');
```

```
END;
```

```
BEGIN
```

```
END.
```

Quando num programa você declarar USES "Exemplo" e chamar a rotina "Logo" o logo aparecerá piscando na tela.

3- COMANDOS BÁSICOS DE I/O

Write ou WriteLn : escreve algo num dispositivo de saída, se o dispositivo não for especificado o default será a tela do micro.

```
WriteLn('Isto é uma string e sairá na tela');
```

Read ou ReadLn : Permite a entrada de dados via teclado

```
Write('Digite um valor para X: ')
  Read(x);
```

ClrScr (Clear Screen): Permite limpar a tela, posicionando o cursor no canto superior esquerdo. Equivale ao CLS do DOS.

GotoXY : Posiciona o cursor em qualquer parte da tela.

```
GotoXY(Coluna, Linha);
```

Delay : Permite uma pausa no programa em milissegundos. (Este comando funciona conforme o clock do computador, sendo diferente em cada tipo de computador);

```
Delay(1000);
```

4- ESTRUTURAS CONDICIONAIS

As estruturas condicionais impõem uma condição para que uma tarefa seja realizada.

4.1 - Condição IF.. Then.. Else (Se.. Então.. Senão);

Se a condição for satisfeita não executa um bloco de tarefas senão executa outra tarefa ou cai fora da estrutura.

```
If  
  Then ;
```

OU

```
If  
  Then  
  Else ;
```

4.2 - Instrução CASE;

O comando CASE é um seletor de opções, executando a opção que for igual a expressão.

Ex.:

```
CASE Of  
    1:bloco;  
    2:bloco;  
    3:bloco;  
  Else  
    bloco;  
  
END;
```

5- Estruturas de REPETIÇÃO

Uma estrutura de repetição repete um bloco até que a condição seja satisfeita.

5.1 FOR

```
FOR X:=1 to 10 do  
  Begin  
    { O bloco será repetido até que x tenha o valor 10}  
  end;
```

Ex.:

```
ClrScr;  
For L:=1 to 24 do  
begin  
GotoXY(1,L);  
WrteLn('Esta é a linha ', l);  
End;
```

5.2 WHILE

While Do

Enquanto a condição não for satisfeita faça

Ex.:

```
while x<100 do  
begin  
write('Digite um valor para A: ');  
read(a);  
x:=a+b;  
end;
```

5.3 REPEAT

Repete enquanto a condição não for satisfeita.

Ex.:

```
REPEAT  
write('Aperte a tecla A');  
read(tecla);  
UNTIL tecla='a';
```

6- Manipalação de STRINGS

Lenght : Retorna a quantidade de caracteres contidas em uma string

Pos : Retorna a posição de uma sub-string dentro de uma string

Copy : Retorna uma substring de uma string de acordo com a posição e a quantidade de caracteres predefinida.

```
Ex.:  
v:='asdllogprtfacillIdeas'  
Write(copy(v,11,5));  
{na tela aparecerá a palavra 'facil'}
```

Val : converte uma string passada como parametro para o valor numérico.

7- FUNCTIONS & PROCEDURES

PROCEDURE

Uma Procedure Realiza uma série de tarefas quando chamada.

Ex.:

```
Program P_Exemplo;  
Uses CRT;  
Var op:byte;  
  
Procedure Menu;  
begin  
  ClrScr;  
  WriteLN('Digite sua Opção:');  
  WriteLN;  
  WriteLN('1- Mostrar Registros');  
  WriteLN('2- Fechar e Sair');  
  OP:=readkey;  
End;
```

```
BEGIN  
  Clrscr;  
  Write('Aguarde carregando o programa...');  
  menu; {Aqui a procedure é chamada}  
  Case Op of  
    1:  
    2:  
  END.
```

FUNCTION

A FUNCTION é uma rotina que nos retorna um determinado valor. Da mesma forma que uma procedure, uma FUNCTION deve ser declarada antes de ser

utilizada.

Ex.:

```
PROGRAM Ex_Function;
USES CRT;
VAR X:INTEGER;

FUNCTION RESULTADO (Y:INTEGER): INTEGER;
BEGIN
  Y:=Y*Y;
  RESULTADO:=Y;
END;

BEGIN
  CLRSCR;
  WRITE ('DIGITE UM VALOR: ');
  READLN(X);
  X:=RESULTADO(X);
  WRITELN ('O RESULTADO É : ',X);
  DELAY (10000);
END.
```

8 - ARQUIVOS e REGISTROS

ARQUIVO

Para formar um arquivo, devemos formar uma variável do tipo arquivo, além de criar comandos de abertura, leitura e fechamento.

- File: Define uma variável como sendo arquivo.

Ex.:

```
Type Registro=record
  Nome:string[50];
  Endereço:string[100];
  Fone: string[8];
end;

VAR Arquivo:File of Registro;
    Reg:registro;
```

- Assign: Este procedimento associa um nome externo de arquivo a uma variável do tipo arquivo.

Ex.: Assign (Arquivo,' d:\ficha.dat');

- Reset: Este procedimento permite abrir um arquivo já existente. No caso do arquivo não existir ocorrerá erro.

Ex.: Reset(Arquivo);

- ReWrite: Permite Criar e Abrir um arquivo novo. Caso o arquivo já exista, terá seu conteúdo eliminado e será gerado um novo arquivo.

Ex.: ReWrite(Arquivo);

- Close: Este procedimento permite que se feche um arquivo. É necessário fechar um arquivo para não ocorrer erro com a FAT do Sistema Operacional.

Ex.: Close(arquivo);

- Write: Permite gravar em um arquivo.

Ex.: Write (arquivo, reg); {Grava todo o reg no arquivo}

- EOF: Retorna o valor TRUE quando for encontrado o fim do arquivo.

```
Ex.: While not eof(arquivo) do
  begin
  x:=x+1;
  Seek (arquivo,x);
  end;
```

- Seek: Este procedimento permite que movamos(do verbo 'mover') o ponteiro do arquivo para uma posição preestabelecida.

Ex.: Seek(Arquivo,0);

- FilePos: Retorna a posição atual do ponteiro do arquivo.

Ex.: x:=FilePos(Arquivo);

- FileSize: Retorna o numero de registros de um arquivo.

END. { Por Enquanto é só, aguarde novas atualizações, Última atualização: 13/10/97. }
