

```
+-----+
| +-----+ |
| | LINGUAGEM PASCAL | |
| +-----+ |
+-----+
```

```
+-----+
|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 sobre-maneira 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 é programando, assim como a melhor forma de aprender a dirigir é 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 faço 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 é tradicionalmente compilada.

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

- 1-) Devemos utilizar um editor de textos para escrever e armazenar em disco o nosso programa fonte.
- 2-) Utilizar um compilador para traduzir o programa fonte para um programa em linguagem de máquina.
- 3-) 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 com dois acionadores de discos de dupla face e alta densidade, um winchester de 340 megabytes, um monitor monocromático e 640 Kbytes de memória RAM. No entanto, a configuração mínima poderia ser um IBM/PC-XT com um winchester de 40M.

+-----+
|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.PAS -> 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 e o cabeçalho do  
programa }
```

```
USES Crt; { Aqui estou utilizando uma UNIT, chamada CRT,  
existem várias, e inclusive vc 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 = 'Thelmo'; { nesta área podemos definir todas  
as constantes que quisermos  
utilizar no programa }
```

```
Type  
n = (BRASILEIRA, PORTUGUESA, INGLESA, FRANCESA, ALEMA, AMERICANA);
```

```
{ o Turbo Pascal possui diversos tipos de variáveis pre-  
definidas, mas também permite definir novos tipos na  
subárea 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 subárea Var }

Procedure Linha; { a procedure equivale ao conceito de sub-rotina.
                  Sua estrutura pode se tornar tão complexa como
                  de um programa. Esta procedure, traça 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(34,idade));
    Linha;
    goto fim;
    { estas linhas serão puladas }
    nacionalidade:=BRASILEIRA;
    Write('Minha nacionalidade e brasileira');
fim:
    Write('Prazer em conhece-lo');
End.

```

II.2 - Estrutura de um programa em Pascal

 Todo programa em Pascal é subdividido em 3 áreas:

- cabeçalho do programa
- área de declarações
- corpo do programa

Na definição padrão da linguagem Pascal, o Cabeçalho do programa é 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 subárea, pois mais para frente estudaremos cada uma delas com profundidade. Na subárea 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 subárea 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 subárea Type.

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

Finalmente, na subárea Functions podemos definir novas funções que depois poderemos utilizar no programa embora o Turbo Pascal possua inúmeras funções pré-definidas. Estas subáreas só são obrigatórias caso nós estejamos precisando. Exemplo: se não vamos utilizar variáveis no nosso programa (coisa rara) então não precisamos utilizar a subárea Var. De acordo com a definição padrão da Linguagem Pascal, estas subáreas devem aparecer na sequência 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 International (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 deverá 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 } será 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	Signed 8-bit
Integer	-32768..32767	Signed 16-bit
Longint	-2147483648..2147483647	Signed 32-bit
Byte	0..255	Unsigned 8-bit
Word	0..65535	Unsigned 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 subárea específica chamada Var.

Para estudarmos essa subárea 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 &, #,* 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 é constituído a partir de um tipo simples que é o char. O tipo string é 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 subárea onde devemos declarar todas as variáveis que iremos utilizar em nosso programa. Exemplo:

```
Program Exemplo;          (* cabeçalho do programa *)
```

```
Var
```

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

```
(* aqui começa o programa *)
```

```
Begin
```

```
    idade:=34;
    número_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 é:

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

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

```
Varidade:integer;      o compilador não reconhece a palavra Var
```

```
Var idade:integer;    agora sim, ou se preferir
```

```
Var                    idade
                        : integer;                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 procedido 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];

```

```

+-----+
| Aguarde a aula No. 2 |
+-----+

```

```

[]s .""| _lCa |"".
```