Capitulo I – Introdução ao Turbo Pascal

1. Instalação e inicialização do Turbo Pascal 7.0

Junto com sua documentação você estará recebendo dois disquetes: um intitulado Turbo Pascal 7.0 que é uma versão compacta deste software. O outro é um disquete vazio para você armazenar seus programas desenvolvidos durante o curso.

Para instalar o Turbo Pascal, proceda da seguinte forma:

- Coloque o disquete no drive A:
- No menu Iniciar – Executar do Windows 95/98 digite: a:\Install.exe e pressione enter ou clique em OK;

Capitulo II – Itens Básicos

2.1Tipos de dados

2.1.1 Numérico

Os dados numéricos podem ser inteiros ou reais.

- Os números inteiros podem ser positivos ou negativos e não possuem parte fracionária.

A faixa de valores inteiros possíveis está relacionada com a definição do seu tipo

Tipo	Faixa	Tamanho em bytes
Shortint	De –128 a 127	1
Integer	De –32768 a 32767	2
Longint	De 2147483648 a 2147483648	4
Byte	De 0 a 255	1
Word	De 0 a 65535	2

Exemplo: 4 -123 57

- Os números reais podem positivos ou negativos e possuem parte fracionária

A faixa de valores inteiros possíveis está relacionada com a definição do seu tipo

Tipo	Faixa	Tamanho em bytes
Real	De $2,9 \times 10^{-39}$ a $1,7 \times 10^{38}$ (10 a 12 dígitos)	6
Single	De $1,5 \times 10^{-45}$ a $3,4 \times 10^{38}$	4
Double	De $5,0 \times 10^{-324}$ a $1,7 \times 10^{308}$	8
Extended	De $3,4 \times 10^{-4932}$ a $1,1 \times 10^{4932}$	10
Comp	De $-9,2 \times 10^{18}$ a $9,2 \times 10^{18}$	8

Observe-se que:
Os números reais, a separação da parte inteira da parte fracionária é representada pr um(.) ponto e não pela (,) virgula.
Não pode haver espaços em branco entre os algarismos usados
Se existir o ponto decimal, pelo menos um digito deve preceder e um digito deve suceder o ponto
Podemos representar os valores reais pela notação escalar ou exponencial: por exemplo:
5.32 -34.89
7.8E3 – onde a letra E significa 10 elevado a, assim teremos 7.8 x 1000 = 78000
7.8 E-1 – significa 7.8 x 0,1= 0,78

2.1.2 Lógico

Os dados lógicos podem ser TRUE ou FALSE, isto é verdadeiro ou falso. Este tipo de dado ocupa apenas 1 byte e não pode ser lido ou escrito sendo usado para controle de fluxo do programa.

Tipo	Faixa	Tamanho em bytes
Boolean	1	1

2.1.3 Literal ou caracter

São formados por um único caracter ou por um conjunto de caracteres. Os caracteres podem ser letras maiúsculas, minúsculas símbolos especiais ou números que serão tratados como letras.

Este tipo de dados é referenciado por

Tipo	Faixa	Tamanho em bytes
string	De 1 a 255	1 byte por caracter + 1 byte inicial com o tamanho da cadeia
Char	1	1

Os dados destes tipos, dentro dos programas, são representados da seguinte forma:

- Entre apóstrofos: Ex: ‘Alexandre’

‘1234’

‘arojas@ime.uerj.br’

- Pelo seu valor ASCII – neste caso o símbolo # precede o código ASCII

#166 significa ª.

- Pelo seu valor em hexadecimal – neste caso o símbolo $ precede o código

$FFFF

2.2 Formação dos Identificadores

Os identificadores são os nomes das variáveis, dos programas, das constantes, das rotinas e das unidades.

As regras para formação dos identificadores são:

- Podem ter qualquer tamanho, porém somente os 63 primeiros caracteres serão considerados;
- Podem ter letras e números ou underscore (sublinhado _)
- Não podem ter espaços em branco ou caracteres especiais ( )*&%$#@!+-=/?><.
- Iniciam sempre por uma letra
- Não pode ser uma palavra reservada
- Podem ser usadas letras maiúsculas ou minúsculas

Palavras reservadas				
AND	ASM	ARRAY	BEGIN	CASE
CONST	DIV	DO	DOWNTO	ELSE
END	FILE	FOR	FUNCTION	GOTO
IF	IN	LABEL	MOD	NIL
NOT	OF	OR	PACKED	PROCEDURE

Palavras reservadas				
PROGRAM	RECORD	REPEAT	SET	THEN
TO	TYPE	UNTIL	VAR	WHILE
WITH				

Exemplos:
Nome
Idade
Telefone
Nome_do_aluno
A1234

2.3 Estrutura de um programa Pascal

O Pascal é uma linguagem procedural, ou seja, obedece a um procedimento sendo as instruções executas em ordem seqüencial.
Todo programa escrito em Pascal é subdividido em 3 áreas: área de cabeçalho, área de declarações e corpo do programa

2.3.1 Área de cabeçalho

O cabeçalho de um programa é representado pela palavra PROGRAM <identificador>;

2.3.2 Área de declarações

Em um programa não é obrigatório existir todas as seções.

Seção	Função	Exemplo
USES	Informa as unidades a serem utilizadas. Caso uma rotina não esteja no corpo do programa, a instruções necessárias serão procuradas nas unidades relacionadas	USES CRT;(usa o monitor) USES PRN;(usa a impressora)
CONST	Atribui a identificadores valores constantes que não poderão ser alterados dentro do programa	CONST NMAX = 100;
LABEL	Define o nome dos rótulos	LABEL FIM, INICIO;
VAR	Define as variáveis usadas e seu tipo	VAR NOME: STRING[30]; IDADE:INTEGER;
TYPE	Define um tipo de variável criada pelo usuário	TYPE ALUNO=STRING[60];
PROCEDURE	Define as instruções de um sub programa	PROCEDURE LER;
FUNCTION	Define as instruções de uma função	FUNCTION SOMA:INTEGER;

2.3.3 Corpo do Programa

O programa propriamente dito está contido no corpo do programa. Esta área tem início com a instrução BEGIN e término com a instrução END seguida de (.).

Chamamos de BLOCO ao conjunto de instruções entre o begin e o end.

2.3.4 Exemplo de programa Pascal

```
Program um;
Uses crt;
Var
A,b,c: integer;
Begin
        Clrscr;
        Write('Digite A ');readln(a);
        Write('Digite B ');readln(b)
        C:=a+b;
        Writeln('A soma é ',c);
End.
```

2.4 Comentários

Os comentários são introduzidos no corpo do programa para aumentar sua legibilidade.

Usa-se { para iniciar e } para terminar.

Ex. {Programador: Alexandre }

2.5 Declaração de variáveis

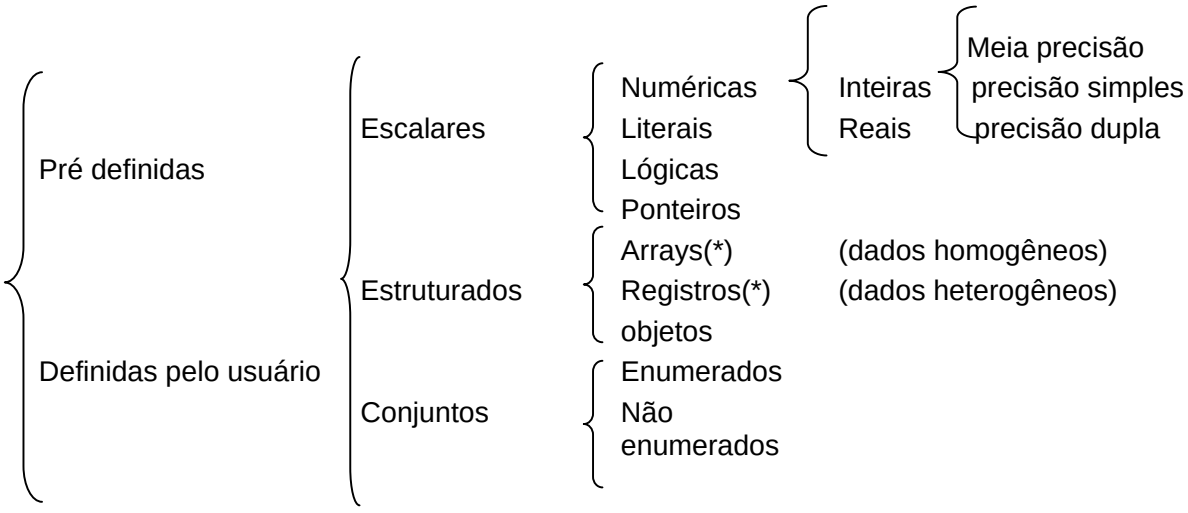
As declarações de variáveis são obrigatórias para validar os identificadores. Todas as variáveis devem ser incluídas em uma única declaração do tipo:

VAR

Lista de variáveis : tipo;

Lista de variáveis: tipo;

Os tipos de variáveis podem ser:



(*) estes tipos serão apresentados durante o curso

- Tipos Pré definidos

Os tipos pré definidos escalares já foram examinados no item 2.1

Exemplo de trecho de programa com declaração de variáveis:

```
PROGRAM DOIS;{nome do programa}
```

```
USES CRT; {carrega as rotinas para a manipulação da tela}
VAR
        NOME:STRING[30]; {a variável nome armazena até 30 caracteres}
        IDADE: INTEGER;
        SEXO: CHAR;
        SALARIO: REAL;
        ACHEI: BOOLEAN;
BEGIN


END.
```

- Tipo definido pelo usuário

O comando TYPE permite a definição de um tipo de dados utilizando os tipos pré definidos na linguagem ou outro tipo previamente definido pelo usuário e permite uma maior legibilidade do programa.

A declaração é do tipo:

```
TYPE
        VARIÁVEL = tipo;
```

A criação de um tipo não atribui a nenhuma variável que deve ser realizada pela declaração VAR.

Exemplo de trecho de programa com declaração de tipo e de variáveis:

```
PROGRAM TRES;
 USES CRT;
TYPE
        TIPO_DE_NOME = STRING[30];
VAR
        NOME: TIPO_DE_NOME;
        IDADE: INTEGER;
        SEXO: CHAR;
        SALARIO: REAL;
BEGIN
....
END.
```

Podemos definir um conjunto heterogêneo do tipo RECORD da seguinte forma:

```
PROGRAM TRES1;
USES CRT;
TYPE
        TIPO_ALUNO = RECORD
        NOME: STRING[30];
        IDADE:INTEGER;
        SEXO:CHAR;
        END;{encerra a definição do RECORD}
VAR
```

```
        ALUNO:TIPO_ALUNO
BEGIN
        ALUNO.NOME:='ALEX';
        ALUNO.IDADE :=20;
…..
END.
```

- Conjuntos

No Turbo Pascal um conjunto é um grupo de números ou caracteres inter-relacionados . Os conjuntos são basicamente utilizados para verificar se um caracter ou um número pertence ao conjunto. O operador IN verifica se um determinado valor pertence ao conjunto

O tipo SET define o conjunto.

Exemplo usando conjunto:

```
PROGRAM QUATRO;
 USES CRT;
VAR
        LETRAS: SET OF 'A'..'Z';
        NUMERO: SET OF 0..9;
        TODOS: SET OF CHAR;
BEGIN
....
END.
```

Os conjuntos podem ser enumerados, como por exemplo:

```
PROGRAM QUATRO;
 USES CRT;
VAR
        INGREDIENTES:(OVOS,LEITE,MANTEIGA,FARINHA);
        NUMERO: SET OF 0..9;
        TODOS: SET OF CHAR;
BEGIN
....
END.
```

- Constantes

As definições de constantes são realizadas antes das declarações das variáveis. Todas as declarações devem ser incluídas da seguinte forma:

```
CONST
        Variável = valor;
```

Algumas constantes estão pré definidas. Por exemplo:

PI – 3.1416

Maxint - 32767

Exemplos de definição de constante:

```
PROGRAM CINCO;
USES CRT;
CONST
        NOME = ‘ANA FERREIRA’;
        NUM = 100.
        LETRA =’A’;
```

Capitulo III - Comando de Atribuição

3.1 Um comando de atribuição é sempre do tipo:

variável := expressão;

As expressões podem ser:

Expressões matemáticas – São escritas linearmente usando a notação matemática. Os operadores são:

Operador	Função	Operandos	Resultado
+	Somar	Inteiro ou real	Inteiro ou real
-	Subtrair	Inteiro ou real	Inteiro ou real
*	Multiplicar	Inteiro ou real	Inteiro ou real
/	Dividir	Inteiro ou real	Real
Div	Quociente inteiro	Inteiro	Inteiro
Mod	Resto da divisão	Inteiro	inteiro

Deve-se observar as seguintes regras:

Não pode existir nenhuma operação implícita;

Dois operadores não podem aparecer juntos – usa- se o parêntesis para separar;

Por não existir operador para a exponenciação, temos: $A^B = EXP(B*LN(A))$;

Para cada (deve existir um);

O símbolo := significa *atribua a* e tem sentido diferente do igual da matemática;

Hierarquia dos operadores:

1° sinal unário

2° parêntesis

3° funções

4° */ div mod

5° + -

Em caso de mesma hierarquia resolve-se da esquerda para a direita.

Exemplos de operação matemática:

A=2 B=3

Expressão	Pascal	Resultado
$Z= \frac{A+B}{2}$	Z:=(A+B)/2;	Z=2,5
$Z=A^B$	Z:=EXP(LN(A)*B);	Z=8
	Z:= 11 MOD A	Z=1
	Z:= 11 DIV A	Z= 5

3.2 Principais Funções Matemáticas pré definidas

Função		Pascal	Argumentos	Resultado
| x |	Valor absoluto	ABS(X)	Inteiro ou real	Inteiro ou real
e^X	Exponencial de e	EXP(X)	Inteiro ou real	Real
Sen(X)	Seno de X	SIN(X)	X em radianos	Real
Cos(X)	Co seno de X	COS(X)	X em radianos	Real
Arctg(x)	Arco tangente de X	ARCTAN(X)	X em radianos	Real
LN(X)	Logaritmo neperiano de X	LN(X)	Inteiro ou real	Real
	Arredondar	ROUND(X)	Real	Inteiro
	Parte inteira de X	TRUNC(X)	Real	Inteiro
	Parte fracionária de X	FRAC(X)	Real	Real
	Raiz quadrada	SQRT(X)	Inteiro ou real	Real
X^2		SQR(X)	Inteiro ou real	Real
Semente de no. randômico		RANDOMIZE		
No. randômico entre 0 e 1		RANDOM		Real
No. randômico entre 0 e o parâmetro		RANDOM(x)	Inteiro	real

Exemplos:

Considere que A= 3.0 B=2.0 C=7 D=4

O valor da expressão Z:= A/B *(C MOD D); será igual a:

3.0/2.0 * (7 MOD 4)

3.0/2.0 * 3

1.5 * 3 = 4.5

3.3 Expressões Lógicas

São expressões cujos operadores são lógicos e cujos operandos são relações tendo como resposta um valor booleano (true ou false).

Os operadores relacionais são

Operador	Significado
=	Igual
<>	Diferente
<=	Menor ou igual
>=	Maior ou igual
<	Menor
>	maior

Exemplos

Para X=1 Y=2 Z=5 considere a expressão: X * X + Y > Z assim teremos: 1 * 1 + 2 > 5

3> 5 o resultado é FALSE

Para X=4 Y=3 Z=1, considere a expressão X * X + Y > Z, assim teremos 4 * 4 + 3 > 1

19 > 1 logo o resultado é TRUE

Os operadores lógicos AND, OR e NOT significam e ou não sendo usados para conjunção disjunção e negação

3.4 Expressões Literais

São expressões cujas respostas são valores literais.

- Principais funções literais pré definidas

Função	Descrição	Exemplo
ORD(X)	Retorna um valor inteiro que indica o valor de X no código ASCII	Z:=ORD('A');Z:=65
CHR(X)	Retorna um caracter representado por X – inteiro – no código ASCII	Z:=CHR(65); Z:='A';
SUCC(X)	Retorna o sucessor de X no código ASCII	Z:=SUCC('A'); Z:='B';
PRED(X)	Retorna o predecessor de X no código ASCII	Z:=PRED(166); Z:=165;
UPCASE(X)	Retorna X maiúsculo	Z:=UPCASE('a'); z:='A';

3.5 Principais funções para tratamento de cadeias de caracteres (string)

Função	Descrição
COPY(cadeia, posição, número)	Copia da cadeia, a partir da posição o numero de caracteres
LENGTH(cadeia)	Retorna o numero de caracteres da cadeia
POS(cadeia1,cadeia2)	Retorna em que posição a cadeia1 aparece dentro da cadeia2
DELETE(cadeia, posição, número)	Apaga, a partir da posição o número de caracteres
INSERT(cadeia1, cadeia2, posição)	Insere na cadeia2 a cadeia1 a partir da posição
CONCAT(cadeia1,cadeia2) ou Cadeia1+cadeia2+...	Soma as cadeias
VAL(cadeia, variável, código)	Transforma a cadeia no equivalente numérico inteiro ou real conforme a variável. Se tiver êxito o código vale zero.
STR(valor: tamanho, cadeia)	Transforma um valor em uma cadeia

Exemplo de programa usando as funções de manipulação de cadeias

```
PROGRAM FUNCAO;
USES CRT;
CONST
   CADEIA = 'UNIVERSIDADE DO ESTADO DO RIO DE JANEIRO';
   SEQ = 'VER';
   SIGLA =  'UERJ';
   NUM1='123';
   NUM2='123.4';
   NUM3 = 567.8;
VAR
   A,B:INTEGER;
   C,S,D,E,H:STRING[60];
```

```
    F,COD:INTEGER;
    G:REAL;
BEGIN
CLRSCR;
A:=POS(SEQ,CADEIA); {posição em que seq aparece na cadeia}
WRITELN('A = ',A);{a partir da 4 posição}
B:=LENGTH(CADEIA);{quantidade de caracteres de cadeia}
WRITELN('B = ',B);{40 caracteres}
C:=COPY(CADEIA,17,6);{copia de cadeia, a partir da posição 17,6 caracteres}
WRITELN('C = ',C);{ESTADO}
D:=CADEIA;
DELETE(D,1,16);{retira da frase 16 caracteres a partir da posição 1}
WRITELN('D = ',D);
E:=CADEIA + ' - ' +SIGLA;{soma as duas cadeias}
WRITELN('E = ',E);
VAL(NUM1,F,COD); {transforma 123 em numero inteiro}
WRITELN('F = ',F, ' E INTEIRO');
VAL(NUM2,G,COD); {transforma 123.4 em numero real}
WRITELN('G = ',G:6:2,' E REAL');
STR(NUM3:5:2,H);{transforma 567.8 em string}
WRITELN('H = ',H);
READKEY;
END.
```

Capítulo IV– Comandos de Entrada e Saída

4.1 Os comandos de entrada são:

READ(arq, lista de variáveis e/ou constantes e/ou expressões);

ou

READLN(arq, lista de variáveis e/ou constantes e/ou expressões);

O comando READ lê permanece na mesma linha. O comando READLN lê e muda de linha.

Os valores somente serão atribuídos as variáveis após pressionarmos a tecla <enter>

Exemplos:

READ(NOME);

READLN(IDADE);

READKEY;

O comando READKEY; lê a tecla pressionada atribuindo a uma variável CHAR. Não é necessário pressionar <enter>

Exemplo:

RESP:=READKEY; a tecla pressionada será atribuída a RESP.

4.2 Os comandos de saída são:

WRITE(arq, lista de variáveis e/ou constantes e/ou expressões);

ou

WRITELN(arq, lista de variáveis e/ou constantes e/ou expressões);

Onde:

O comando WRITE exibe e permanece na mesma linha. O comando WRITELN exibe e muda de linha.

Arq: nome do arquivo selecionado para ler ou exibir. O uso de arquivos será examinado no decorrer do curso.

Podemos enviar dados para a impressora, para tanto o nome do arquivo é lst. Devemos incluir o comando USES PRINTER; Obs: Deve existir uma impressora conectada ao computador.

Se a unidade selecionada para exibir for o monitor, não é necessário especificar o arquivo.

Exemplos:

WRITE('A =' , A); exibe a mensagem entre apóstrofos A= e exibe o conteúdo da variável A, mantendo o cursor na mesma linha.

WRITELN(A+B); exibe a soma da variável A + B

WRITE(LST, 'Isto vai para a impressora');

4.3 Comandos e funções para controle do vídeo

4.3.1 Comando para limpar a tela

CLRSCR; significa clear screen

Para usarmos este comando é necessário especificar USES CRT; no início do programa

4.3.2 Comando para apagar a partir do cursor até o final da linha

CLREOL; significa clear end of line

4.3.3 Comando para posicionar o cursor na tela

GOTOXY(X,Y);

O comando posiciona na coluna X e na linha Y, sabendo-se que a tela tem 80 colunas e 25 linhas e que a primeira posição é 1,1.

Exemplo:

```
program exemplo333;
uses crt;
var x,y:byte;
begin
   clrscr;
   gotoxy(10,5);
   writeln('Estou na coluna 10, linha 5');
   x:=40;
   y:=20;
   gotoxy(x,y);
   writeln('Estou na coluna 40 linha 20');
   x:=x-20;{x = 40-20 =20}
   y:=y+1;  {y =20+1=21}
   gotoxy(x,y);
   writeln('Estou na linha seguinte pressione qq tecla para
   continuar');
   readkey;
   gotoxy(29,21);
   clreol;
   writeln('<------- observe o que apagou');
   readkey;
end.
```

4.3.4 Comando para formatar os valores exibidos

WRITE(variável: número total de casas: casas decimais);

Exemplo:

```
PROGRAM exemplo334;
uses crt;
var
   r: real;
   i: integer;
   nome: string[10];
begin
   clrscr;
   r:=3.1416;
   writeln(r);
   writeln(r:10:4);
   i:=10;
   writeln(i);
   writeln(i:5);
```

```
   nome:= 'Alex';
   writeln(nome);
   writeln(nome:10);
   readkey;
end.
```

Resultaria na seguinte tela:

```
3.141599999902E00
   3.1416
10
    10
ALEX
       ALEX
```

4.3.5 Comando para alterar a cor das letras:

TEXTCOLOR(código da cor);

Exemplo: TEXTCOLOR(1); todas as letras passam para azul até que outro comando altere.

Os códigos das cores podem ser agrupados em 0-7 cores pasteis, 8-15 cores fortes, 16-23 cores piscantes.

Numero da cor	cor	Numero da cor	cor	Numero da cor	cor
0	Preto	8	Cinza escuro	16	Preto piscante
1	Azul	9	Azul claro	17	Azul piscante
2	Verde	10	Verde claro	18	Verde piscante
3	Ciano	11	Ciano claro	19	Ciano piscante
4	Vermelho	12	Vermelho claro	20	Vermelho piscante
5	Magenta	13	Magenta claro	21	Magenta piscante
6	Marrom	14	amarelo	22	Amarelo piscante
7	Cinza claro	15	Branco	23	Cinza piscante

A cor da letra padrão é branca (15)

4.5 Comando para alterar a cor do fundo da tela

TEXTBACKGROUND(código da cor);

Os códigos das cores são

Numero da cor	cor	4	Vermelho
0 (padrão)	Preto	5	Magenta
1	Azul	6	Marrom
2	Verde	7	Cinza claro
3	Ciano		

3.3.6 Comando para eliminar a linha onde esta o cursor

DELLINE;

As demais linhas abaixo sobem uma linha

3.3.7 Comando para inserir linha na posição do cursor

INSLINE;

As demais linhas descem uma linha

Exemplo

```
program exemplo334;
uses crt;
begin
     clrscr;
     writeln('linha 1');
     writeln('linha 2');
     writeln('linha 3');
     writeln('linha 4');
     writeln('pressione qq tecla para continuar');
     readkey;
     gotoxy(1,2);{vá para coluna 1 linha 2}
     delline;
     gotoxy(20,2);
     write('<--- observe que linha 2 apagou');
     readkey;     {aguarda pressionar qq tecla}
     gotoxy(20,2);   {VA para coluna 20 linha 2}
     clreol;   {limpa a partir do cursor}
     insline; {insere uma linha na posição do cursor}
     gotoxy(1,2);
     write('inseri a linha 2');
     readkey;
end.
```

Exemplos:

Elabore um programa Pascal que:

1. Leia dois valores determine e exiba a soma eles;

```
program exercicio1;
uses crt;
var
a,b,c: real;
begin
clrscr;
write('Digite A ');readln(A);
write('Digite B ');readln(B);
c:=A+B;
writeln('A soma e ',C:10:2);
readkey;
end.
```

2. Leia uma determinada quantia em Reais e exiba a menor quantidade de cédulas de 100,50,10 e 1 real que serão necessários para obtermos esta importância.

```
program troco;
uses crt;
```

```
var

valor,nota100,nota50,nota10,nota1,resto100,resto50,resto10:in
teger;
begin
     clrscr;
     write('Digite o valor R$');readln(valor);
     nota100:=valor div 100;
     resto100:=valor mod 100;
     nota50:= resto100 div 50;
     resto50 :=resto100 mod 50;
     nota10:=resto50 div 10;
     resto10:=resto50 mod 10;
     nota1:= resto10;
     writeln(nota100, '  notas de R$100');
     writeln(nota50,' notas de R$ 50');
     writeln(nota10,' notas de R$10');
     writeln(nota1,' notas de R$1');
     readkey;
end.
```

3. Leia uma frase e exiba-a centralizada na tela:

```
program centra;
uses crt;
var
frase: string[80];
begin
     clrscr;
     write('Digite a frase ');readln(frase);
     writeln(frase:(40+trunc(length(frase)/2)));
     readkey;
     end.
```

Capitulo V – Estruturas de decisão (ou seleção)

5.1 Decisão simples

IF condição THEN comando;

Ou

IF condição THEN

BEGIN

Comando 1;

Comando 2;

END;

O comando ou o bloco de comandos é executado caso a condição seja verdadeira;

Os operadores relacionais são:

Operador	Significado
=	Igual
<>	Diferente
<=	Menor ou igual
>=	Maior ou igual
<	Menor
>	maior

Também podemos usar os operadores lógicos

Operador	Significado	
AND	E	As duas condições devem ser verdadeiras
OR	ou	Uma das duas condições deve ser verdadeira
NOT	não	A condição não deve ser verdadeira

Exemplo

```
program decisao1;
uses crt;
var
   a,b:real;
begin
     clrscr;
     write('Digite A ');readln(A);
     write('Digite B ');readln(B);
     if A=B then
        write('Os valores sao iguais ',a:10:2);
     readkey;
end.
```

5.2 Decisão composta

IF condição THEN

Comando ou bloco de comandos 1

ELSE

Comando ou bloco de comandos 2;

Se a condição for verdadeira o bloco de comandos 1 é executado. Se a condição for falsa o bloco de comandos 2 é executado.

O comando anterior ao ELSE não tem (;)

Exemplo:

Programa para determinar o maior de dois valores reais lidos

```
program maior_de_dois;
uses crt;
var
   a,b:real;
begin
     {maior de dois valores lidos}
     clrscr;
     write('Digite A ');readln(A);
     write('Digite B ');readln(B);
     if A>B then
        write('A e'' o maior ',A:10:2)
     else
        write('B e'' o maior ',B:10:2);
readkey;
end.
```

Observe que se os valores forem iguais será impresso que B é o maior. Para corrigirmos isto iremos colocar outro IF dentro do primeiro IF.

```
program maior_de_dois;
uses crt;
var
   a,b:real;
begin
     {maior de dois valores lidos}
     clrscr;
     write('Digite A ');readln(A);
     write('Digite B ');readln(B);
     if A>B then
        write('A e'' o maior ',A:0:2)
     else
        if A = B then
           write('A e B sao  iguais a ',A:0:2)
        else
           write('B e'' o maior ',B:0:2);
readkey;
end.
```

Programa para determinar o maior de 3 valores lidos. Os valores são diferentes entre si.

```
program maior_de_tres;
uses crt;
var
   a,b,c:real;
begin
     {maior de tres valores lidos}
     {os valores sao diferentes entre si}
     clrscr;
     write('Digite A ');readln(A);
     write('Digite B ');readln(B);
     write('Digite C ');readln(C);
 if (A>B) and (A>B) then
        write('A e'' o maior ',A:0:2)
     else
        if (B>A) and (B>C) then
           write('B e'' o maior ',B:0:2)
        else
           write('C e'' o maior ',C:0:2);
readkey;
end.
```

4.3 Seleção múltipla

CASE opcao OF

Lista de alvos 1 : BEGIN

END;

Lista de alvos 2 : BEGIN

END;

ELSE comando 3;

END;

Obs: a variável opcao deve ser do tipo INTEGER ou CHAR

A lista de alvos pode ser uma constante numérica inteira, do tipo caracter um conjunto de valores ou um intervalo de valores

```
program escolha;
uses crt;
var
  opcao:char;
begin
```

```
     {exemplo do CASE}
     clrscr;
     write('Digite a opcao ');readln(opcao);
     {escolha entre as opções listadas}
     case opcao of
          '1' : write('Voce digitou 1');
          '2'..'6':write('Voce digitou entre 2 e 6');
          'A','E','I','O','U':write('Voce digitou vogal
maiuscula');
     else
     {se não for nenhuma das opcoes}
         begin
          textcolor(19);
          write('voce digitou outro caracter');
         end; {este end encerra o else }
     end; {este end encerra o case}
readkey;
end.
```

Capitulo VI – Estruturas de repetição

6.1 Comando WHILE

Repete um conjunto de instruções enquanto uma condição for verdadeira.

Devemos estar atentos para que a condição se torne falsa durante a execução para interromper a repetição.

A condição é testada antes de entrar na estrutura de repetição

WHILE condição DO

Comando ou bloco de comandos;

FLAG é um tipo de valor que é lido e que representa o final dos dados. Seu valor não pode ser considerado nos cálculos realizados.

Exemplos:

Programa para calcular a soma de um conjunto de valores reais positivos. O Flag é um valor negativo. O termino ocorrerá quando o flag for lido

```
program repete1;
uses crt;
var
   num,soma:real;
begin
     clrscr;
     write('Digite um numero ');readln(num);
     while num >= 0 do {enquanto nao for o flag repita}
           begin
             soma:=soma+num;
             write('Digite um numero ');readln(num);
           end;
     writeln('A soma e'' ',soma:0:2);
     readkey;
end.
```

Programa para calcular a soma de 5 valores reais.

Neste caso o termino da repetição ocorrerá quando o contador de valores lidos chegar a 5

```
program repete2;
uses crt;
var
   num,soma:real;
   cont:integer;
begin
     clrscr;
     cont:=0;
     while cont < 5 do {enquanto o contador for menor que 5
repita}
           begin
             write('Digite um numero ');readln(num);
             soma:=soma+num;
```

```
            cont:=cont+1;
          end;
     writeln('A soma e'' ',soma:0:2);
     readkey;
end.
```

6.2 Comando REPEAT ..UNTIL

A lógica é semelhante ao comando While somente que repete até a condição seja verdadeira.

A condição é testada no final do conjunto de comandos.

REPEAT

Comando1;

Comando 2;

Comando n;

UNTIL condição;

Exemplos:

Programa para calcular a soma de 5 valores reais

```
program repete3;
uses crt;
var
   num,soma:real;
   cont:integer;
begin
     clrscr;
     cont:=0;
     repeat
            write('Digite um numero ');readln(num);
            soma:=soma+num;
            cont:=cont+1;
          until cont=5;{repete ate que cont = 5}
     writeln('A soma e'' ',soma:0:2);
     readkey;
end.
```

6.2 Comando REPEAT ..UNTIL

A lógica é semelhante ao comando While somente que repete até a condição seja verdadeira.

A condição é testada no final do conjunto de comandos.

REPEAT

Comando1;

Comando 2;

Comando n;

UNTIL condição;

Exemplos:

Programa para calcular a soma de 5 valores reais

```
program repete3;
uses crt;
var
   num,soma:real;
   cont:integer;
begin
     clrscr;
     cont:=0;
     repeat
             write('Digite um numero ');readln(num);
             soma:=soma+num;
             cont:=cont+1;
           until cont=5;{repete ate que cont = 5}
     writeln('A soma e'' ',soma:0:2);
     readkey;
end.
```

Programa para ler nome, idade e sexo de um conjunto de pessoas, calcular e exibir

a) Nome da pessoa mais velha;
b) Média das idades dos homens;
c) Média das idades das mulheres;

O flag é a palavra *pare* no campo do nome. Usando o comando WHILE e o comando REPEAT

```
program exemplo521;
uses crt;
var
   nome,nomev:string[30];
   idade,idadev,contm,contf,somam,somaf:integer;
   sexo:char;
   mediam,mediaf:real;
begin
  clrscr;
  writeln('Entrada dos Dados':45);writeln;writeln;
  {leitura da primeira informacao}
  write('Digite o Nome--->');readln(nome);
  write('Digite a Idade-->');readln(idade);
  write('Digite o sexo--->');readln(sexo);
  nomev:=nome;
  idadev:=idade;
  while nome <> 'pare' do
    begin
```

```
     if idade > idadev then
        begin
          idadev:=idade;
          nomev:=nome;
        end;
     if sexo = 'm' then
        begin
          somam:=somam+idade;
          contm:=contm+1;
         end
        else
         begin
          somaf:=somaf+idade;
          contf:=contf+1;
        end;
     write('Digite o Nome---> ');readln(nome);
     if nome <> 'pare' then
      begin
        write('Digite a Idade--> ');
        readln(idade);
        write('Digite o sexo---> ');
        readln(sexo);
      end;
  end;
    mediam:=somam/contm;
    mediaf:=somaf/contf;
    clrscr;
    gotoxy(1,10);
    writeln('Nome da pessoa mais velha e''',nomev,' com
',idadev:3,' anos');
    writeln('Media da idade dos homens ',mediam:0:2);
    writeln('Media da idade das mulheres ',mediaf:0:2);
    writeln;writeln;
    textcolor(20);
    write('Pressione qq. tecla para encerrar');
    readkey;
end.
```

O mesmo programa usando o REPEAT

```
program exemplo522;
uses crt;
var
   nome,nomev:string[30];
   idade,idadev,contm,contf,somam,somaf:integer;
   sexo:char;
   mediam,mediaf:real;
begin
  clrscr;
```

```
  writeln('Entrada dos Dados':45);writeln;writeln;
  {leitura da primeira informacao}
  write('Digite o Nome--->');readln(nome);
  write('Digite a Idade-->');readln(idade);
  write('Digite o sexo--->');readln(sexo);
  nomev:=nome;
  idadev:=idade;
  repeat
     if idade > idadev then
        begin
          idadev:=idade;
          nomev:=nome;
        end;
     if sexo = 'm' then
        begin
          somam:=somam+idade;
          contm:=contm+1;
         end
     else
         begin
          somaf:=somaf+idade;
          contf:=contf+1;
        end;
     write('Digite o Nome---> ');readln(nome);
     if nome <> 'pare' then
      begin
        write('Digite a Idade--> ');
        readln(idade);
        write('Digite o sexo---> ');
        readln(sexo);
      end;
  until nome= 'pare' ;
    mediam:=somam/contm;
    mediaf:=somaf/contf;
    clrscr;
    gotoxy(1,10);
    writeln('Nome da pessoa mais velha e''',nomev,' com
',idadev:3,' anos');
    writeln('Media da idade dos homens ',mediam:0:2);
    writeln('Media da idade das mulheres ',mediaf:0:2);
    writeln;writeln;
    textcolor(20);
    write('Pressione qq. tecla para encerrar');
    readkey;
end.
```

5.3 Comando FOR

Repete uma quantidade determinada de vezes

- FOR I:= valor inicial TO valor final DO
  Comando ou bloco de comandos;

A variável I chamada de variável de controle é inicializada com o valor inicial, executa todas as instruções do bloco de comandos e retorna para o início do comando. Acrescenta uma unidade a variável de controle e verifica se é superior ao valor final se afirmativo encerra o comando passando para a instrução seguinte ao termino do bloco.
O bloco é executado pelo menos uma vez.
A variável de controle não pode ter seu valor alterado dentro do bloco e é do tipo INTEGER ou CHAR desde que a variação seja ordinal(obedecendo a ordem dos caracteres ASCII)

- FOR I:= valor final DOWNTO valor inicial
  Comando ou bloco de comandos;

Irá decrescer os valores da variável de controle

Exemplos

Impressão dos valores de 1 até 10

```
program ex531;
uses crt;
var
   i:integer;
begin
clrscr;
{exibe os valores de 1 ate 10}
for i:=1 to 10 do
    writeln(i);
readkey;
end.
```

Impressão das letras de a até z

```
program ex532;
uses crt;
var
   i:char;
begin
clrscr;
{exibe os valores de a ate z}
for i:='a' to 'z' do
    writeln(i);
readkey;
end.
```

Fatorial de um número n lido

```
program fatorial;
uses crt;
var
   i,fat,n:integer;
begin
 clrscr;
 write('Digite o valor ');
 readln(n);
 fat:=1;
 for i:=n downto 1 do
     fat:=fat*i;
 writeln('O fatorial de ',n,'e'' ', fat);
 readkey;
end.
```

Impressão das letras de z até a

```
program ex533;
uses crt;
var
   i:char;
begin
clrscr;
{exibe os valores de z ate a}
for i:='z' downto 'a' do
    writeln(i);
readkey;
end.
```

Capitulo VII Estruturas de Dados Homogêneas

As variáveis compostas homogêneas, conhecidas como Arrays, matrizes, variáveis indexadas, variáveis subscritas ou tabelas em memória, são conjunto de elementos de mesma natureza identificados por um índice.

Os arrays podem ser unidimencionais (também são chamados de vetores) quando possuem apenas uma dimensão ou multidimensionais quando posuem mais de uma dimensão.

7.1 Vetores

Vetores são conjunto de dados com uma única dimensão.

Por exemplo

Supondo que o vetor NOTA contivesse os seguintes valores:

NOTA

10.0	7.5	4.3	8.2	6.0
1	2	3	4	5

NOTA[4] conteria o valor 8.2

Declaramos as variáveis como Array da seguinte forma

VAR

NOME: ARRAY[índice inicial .. índice final] OF tipo de variável;

O índice inicial e o índice final são constantes inteiras o caracteres ordinais.

Exemplos

VAR nome : array[1..100] of string[30] {nome é uma variável com 100 elementos e cada elemento pode ter até 30 caracteres}

VAR idade : array [-10..10] of integer; { o índice da variável idade

VAR: cod : array [‘a’..’z’] of char;

No corpo do programa as variáveis são identificadas pelo seu nome e pelo índice que a localiza.

Nome[1]:=’Alexandre’; Idade[-5]:=52; Cod[‘a’]:= ‘x’;

```
Vetor1;
Uses crt;
Var
   nota:array[1..10] of real;
   i:char;
begin
   clrscr;
   for I:= 1 to 10 do
       begin
        write('Digite o ',i,'
termo') ;
        readln(nota[i]);
       end;
   for I:='a' to 'e' do
       writeln(nota[i]);
   readkey;
   end.
```

```
program vetor2;
uses crt;
var
   nota:array['a'..'z'] of real;
   i:char;
begin
   clrscr;
   for I:= 'a' to 'e' do
       begin
        write('Digite o ',i,'
termo');
        readln(nota[i]);
       end;
   for I:='a' to 'e' do
       writeln(nota[i]);
   readkey;
   end.
```

7.2 Matrizes

As matrizes são conjunto de dados de duas ou mais dimensões. Até 3 dimensões existe representação no espaço cartesiano.

A declaração de uma variável tipo matriz é do tipo:

VAR

X: ARRAY[1..10,1..20] OF REAL; {onde x tem 2 dimensões uma com 10 elementos e outra com 20 elementos totalizando 200 elementos reais}

A referência a variável é realizada por intermédio de seus índices.

X[1,1]:= 50.7;

Ou

I:=5; J:=8;

X[i,j]:= 23;

Exemplos

- Programa que leia duas matrizes de 3x3 elementos reais determine sua soma. Imprimir sob a forma de tabela

```
program exemplo62;
uses crt;
     var
     a,b,c:array[1..5,1..5] of real;
     i,j:integer;
begin
  {ler duas matrizes e determinar sua soma}
  clrscr;
  write('Matriz A':44);writeln;writeln;
  for i:=1 to 3 do
   for j:=1 to 3 do
    begin
     write('Digite A[',i:2,',',j:2,'] ');
     readln(A[i,j]);
    end;
   clrscr;
  writeln('Matriz B':44);writeln;writeln;
  for i:=1 to 3 do
   for j:=1 to 3 do
    begin
     write('Digite B[',i:2,',',j:2,'] ');
     readln(B[i,j]);
     C[i,j]:=A[i,j]+B[i,j];
    end;
   {imprimir as duas matrizes}
   clrscr;
   writeln('Matriz Soma':27);
   for i:= 1 to 3 do
    begin
      for j:=1 to 3 do
       write(C[i,j]:0:2,'  ');
      writeln;
     end;
```

```
     readkey;
end.
```

- Programa que leia uma matriz de 3x3 elementos reais e determine a soma ddos termos cada linha e de cada coluna

```
program matriz2;
uses crt;
var
   centra:integer;
   titulo:string[30];
   a:array[1..3,1..3] of real;
   slinha, scol:array[1..3] of real ;
   i,j:integer;
begin
  clrscr;
  titulo:='Leitura da Matriz';
  {centra determina o formato para centralizar o titulo}
  centra:=trunc(40+length(titulo)/2) ;
  writeln(titulo:centra);
  for i:= 1 to 3 do
   for j:= 1 to 3 do
    begin
     write('Digite A[',i:1,',',j:1,'] = ');readln(A[i,j]);
    end;
    {calculo da soma da linha}
  for i:=1 to 3 do
    for j:=1 to 3 do
     slinha[i]:=slinha[i]+a[i,j];
  {calculo da soma da coluna}
  for j:= 1 to 3 do
    for i:=1 to 3 do
     scol[j]:=scol[j]+a[i,j];
{impressao do resultado}
  clrscr;
  writeln('     Impressao do Resultado'); writeln;
  write('Matriz Lida':15);
  gotoxy(25,3);
  writeln('Soma ');
  writeln;
  for i:=1 to 3 do
   begin
    for j:=1 to 3 do
      write(a[i,j]:5:2,'  ');
    writeln('  ',slinha[i]:5:2);
   end;
   writeln;
   for i:= 1 to 3 do
    write(scol[i]:5:2,'  ');
```

```
  readkey;
  end.
```

7.2 Exercícios Resolvidos

- Procura seqüencial em uma tabela. Consideraremos uma tabela com nomes e leremos um determinado nome para verificar se o mesmo se encontra na tabela. O programa deverá prever a procura de vários nomes até que o usuário responda que não que continuar.

```
program pesqseq;
uses crt;
var
   nome:array[1..100] of string[30];
   nomep:string[30];
   achei:boolean;
   i,n:integer;
   continua:char;
begin
clrscr;
{leitura da tabela com os nomes}
write('Digite a quantidade de dados ');readln(n);
for i:=1 to n do
  begin
   write('Digite o ',i,' nome ');readln(nome[i]);
  end;
while continua <> ('N')  do
begin
clrscr;
write('Digite o nome a ser procurado ');readln(nomep);
achei:=false;
i:=0;
while (i< n) and (achei=false) do
  begin
   i:=i+1;
   if nomep = nome[i] then
      achei:= true;
  end;
  if achei = true then
   writeln('O  nome  ',nomep,'  foi  localizado  na  ',i,#166,'
posicao')
  else
   writeln('O nome ',nomep,' nao foi localizado');
  write('Deseja continuar? S/N  ');continua:=readkey;
  continua:=upcase(continua);
  end;
end.
```

- Ordenação de valores

Existem vários processos para ordenar valores. O que será apresentado abaixo é denominado método da seleção

```
program classifica;
uses crt;
var
   nome:array[1..100] of string[30];
   aux:string[30];
   i,j,n:integer;

begin
clrscr;
{leitura da tabela com os nomes}
write('Digite a quantidade de dados ');readln(n);
for i:=1 to n do
  begin
   write('Digite o ',i,' nome ');readln(nome[i]);
  end;
{rotina de ordenacao}
for i:= 1 to n-1 do
 for j:=i+1 to n do
     if nome[i] > nome[j] then
       begin
        aux:=nome[i];
        nome[i]:=nome[j];
        nome[j]:=aux;
       end;
{impressao dos valores}
clrscr;
writeln('Valores ordenados');
for i:= 1 to n do
  writeln(nome[i]);
readkey;
end.
```

- Outro processo de ordenação é o chamado Bubble Sort

```
program bubble;
uses crt;
var
   a:array[1..23] of integer;
   aux:integer;
   k:boolean;
   i,j:integer;
begin
clrscr;
{para facilidade iremos preencher a tabela com valores
aleatorios de 0-100}
randomize;{cria a semente para os numeros aleatorios}
for i:= 1 to 23 do
    a[i]:=trunc(random(100));
writeln('Valores inciciais');
```

```
for i:=1 to 23 do
    writeln(a[i]:7);
k:=false;
while k = false do
 begin
   k:=true;
   for i:= 1 to 22 do
    if a[i] > a[i+1]  then
      begin
        aux:=a[i];
        a[i]:=a[i+1];
        a[i+1]:=aux;
        k:=false;
      end;
 end;
 {readkey;}
 gotoxy(20,1);writeln('Valores ordenados');
 for i:=1 to 23 do
   begin
    gotoxy(20,i+1);
    write(a[i]:7);
    end;
readkey;
end.
```

- A manipulação de cadeias tem vários usos práticos. O programa abaixo usa uma cadeia de 80 caracrteres de linha dupla horizontal para dividir a tela em duas partes.

```
program splitscreen;
uses crt;
var
divider:string[255];
begin
clrscr;
fillchar(divider,sizeof(divider),205);
divider[0]:=chr(80);
gotoxy(1,14);
write(divider);
readkey;
end.
```

A primeira instrução da rotina fillchar(divider,sizeof(divider),205);
Preenche a cadeia toda, da posição 0 a 255, com o valor ASCII 205. Para fazer com que a cadeia preencha a linha na tela é necessário alterar o tamanho da cadeia.
O byte de tamanho da cadeia deve ser definido como 80.
Divider[0]:=80;
Agora a cadeia pode ser exibida na tela de forma agradável.

- Rotina para localização e substituição de cadeias.

Dada a frase: *Diga ao Fernando para pagar os R$10,00 que ele me deve*. Substituir Fernando por Raul

```
Program acha_e_troca;
uses crt;
var
   frase:string[80];
   acha,troca:string[20];
   i:integer;
begin
  clrscr;
  acha:='Fernando';
  troca:='Raul';
  frase:='Diga ao Fernando para pagar os R$10,00 que ele me
deve';
  writeln(frase);
  i:=pos(acha,frase);{localiza o texto de acha em frase}
  delete(frase,i,length(acha));{apaga em frase os caracteres
de acha}
  insert(troca,frase,i);{insere na frase o texto troca a
partir de i}
  writeln(frase);
  readkey;
end.
```

- Entrada numérica livre de erro

Um dos problemas para entrada de valores numéricos é que o usuário poderá digitar dados não numéricos e gerar um erro de execução do programa. A forma de solucionar este aspecto é ler o valor para uma variável string e converte-la para numérica. Este programa emite uma mensagem de erro caso seja digitado um valor não numérico.

```
program entra_numero;
uses crt;
var
   idade,cod:integer;
   idade_alfa:string[10];
begin
  repeat
  clrscr;
  write('Digite sua idade ');readln(idade_alfa);
  val(idade_alfa,idade,cod);{converte idade_alfa p/ idade}
  if cod <> 0 then {se retornar com erro}
     begin
     textcolor(red+blink);
     writeln('Idade errada. Tente de novo');delay(1000);
     textcolor(white);
     end;
   until cod = 0; {ate que nao tenha erro}
  writeln('Sua idade e'' ',idade);
  readkey;
end.
```

- Remover espaços em branco de uma frase

```
Program tira_branco;
uses crt;
var
   frase:string[80];
   i,j:integer;
begin
  clrscr;
  write('Digite uma frase ');
  readln(frase);
  for j:= 1 to length(frase) do
  begin
   i:=pos(' ',frase);
   delete(frase,i,1);
  end;
  writeln(frase);
  readkey;
end.
```

- Programa para separar as palavras de uma frase. Cada frase contem um máximo de 10 palavras de 20 caracteres cada.

```
program separa_palavra;
uses crt;
var
   frase:string[200];
   pal:string[20];
   palavra:array[1..10] of string[20];
   i,k,inicio,tam:integer;
begin
clrscr;
write('Digite a frase ');readln(frase);
k:=1;
inicio:=1;
tam:=1;
for i:=1 to length(frase) do
   if (frase[i] = ' ')  then
     begin
     tam:=i-inicio;
     palavra[k]:=copy(frase,inicio,tam);
     k:=k+1;
     inicio:=i+1;
     end;
for i:=1 to k do
    write(palavra[i],' ');
     readkey;
     end.
```

- Programa para traduzir palavras do português para o inglês (palavra a palavra).

```
program tradutor;
```

```
uses crt;
var
 portugues,ingles:array[1..10] of string[20];
 palavra:string[20];
 i:integer;
 achei:boolean;
 cont:char;
begin
clrscr;
{leitura das tabelas}
writeln('                 Leitura das Tabelas');
writeln('            Portugues           Ingles');
for i:=1 to 10 do
   begin
   gotoxy(2,i+2);write(i,#166,' palavra   ');
   readln(portugues[i]);
   gotoxy(35,i+2);
   readln(ingles[i]);
   end;
{leitura da palava a traduzir}
clrscr;
write('Digite a palavra a traduzir ');readln(palavra);
i:=0;
achei:=false;
while (achei = false) and (i<=10) do
    begin
    i:=i+1;
    if portugues[i] = palavra then
       achei:=true;
    end;
if achei = false then
   writeln('A palavra ',palavra,' nao esta'' cadastrada')
else
   writeln('A traducao de ',palavra,' e'' ',ingles[i]);
readkey;
end.
```

- Programa para calcular a “surpresinha” da megasena

Inicialmente sorteamos 6 números, ordenamos os valores e verificamos se cada valor é diferente do seguinte. Caso afirmativo sorteamos um novo valor e repetimos o processo até que todos os valores sejam diferentes.

```
program surpresinha;
uses crt;
var
   sorte:array[1..6] of integer;
   i,j,t:integer;
   igual:boolean;
begin
clrscr;
```

```
{sorteio dos 6 primeiros valores}
randomize;
for i:= 1 to 6 do
    sorte[i]:=trunc(random(60));
igual:=true;
while igual = true do
  begin
   igual:=false;
   {rotina de ordenacao}
   for i:= 1 to 5 do
     for j:=i+1 to 6 do
         if sorte[i] > sorte[j] then
           begin
            t:=sorte[i];
            sorte[i]:=sorte[j];
            sorte[j]:=t;
          end;
    {verifica se existem 2 iguais}
    for i:= 1 to 5 do
        if sorte[i] = sorte[i+1] then
           begin
           sorte[i+1]:=trunc(random(60));
           igual:=true;
           end;
    end;
    writeln('Valores sorteados');
    for i:= 1 to 6 do
     writeln(sorte[i]);
     readkey;
     end.
```

Capitulo VIII Estruturas de Dados Heterogêneas

8.1 Registros

Quando nos referimos a algum elemento, geralmente o caracterizamos por um conjunto de dados lógicamente relacionados e de diferentes tipos de informação. Por exemplo se nos referimos a dados pessoais estamos querendo o nome, idade, sexo, filiação, endereço etc.

Este conjunto chama-se registro ou RECORD em inglês e tem as seguintes características:

a) contém um número fixo de elementos chamados campos;
b) os campos podem ser de tipos diferentes;
c) cada campo tem um nome próprio chamado de identificador do campo.

Estabelecemos os registros da seguinte forma:

```
Program nome;
Type
        Nome_do_registo = record
                Campo1:tipo;
                Campo2:tipo;
                End;
Var
        Nome_da_variavel:nome_do_registro;
```

Por exemplo:

```
Program exemplo;
Uses crt;
Type
         ficha = record
        nome:string[30];
        idade:integer;
        sexo: char;
        salario: real;
        end;
var
        funcionario:ficha;
```

A variável funcionario tem os campos de Ficha.

Para nos referenciarmos a um determinado componente dentro do programa indicamos o nome da variável (.) nome do campo.

Por exemplo

```
Funcionario.nome := 'Alex';
Funcionario.salario:=10000.00;
```

- Os nomes dos campos podem ser arrays:

```
Type
```

```
        Ficha = record;
        Nome:string[30];
        Nota:array[1..3] of real;
        End;
Var
        Aluno:ficha;
```

No programa nos referimos a primeira nota do aluno da seguinte forma:

Aluno.nota[1]:=10;

Aluno.nota[2]:=7.5;

- As variáveis também podem ser arrays:

```
Type
        Ficha = record;
        Nome:string[30];
        Nota:array[1..3] of real;
        End;
Var
        Aluno:array[1..100] of ficha;
```

No programa nos referimos ao 1 aluno da seguinte forma

Aluno[1].nome:=’Alex’;

Aluno[1].nota[1]:=10.;

Exercício:

Elabore um programa pascal que leia nome, idade, sexo e salário de um conjunto de pessoas ordene e exiba em ordem ascendente por nome. A quantidade máxima de pessoas é 100 e é lida no início do programa.

```
program ordena;
uses crt;
type
    cadastro = record
    nome:string[30];
    idade:integer;
    sexo:char;
    salario:real;
    end;
var
   pessoa:array[1..100] of cadastro;
   aux:cadastro;
   i,j,n:integer;
begin
   clrscr;
   write('Digite a quantidade de pessoas ');
   readln(n);
   clrscr;
   writeln('Entrada de dados':48);
   for i:=1 to 80 do
```

```
        write(#205);
    writeln;
    for i:= 1 to n do
        begin
        clrscr;
        writeln(i:2,#167,'  Pessoa '); writeln;
        write('Digite o nome '); readln(pessoa[i].nome);
        write('Digite a idade ');readln(pessoa[i].idade);
        write('Digite o sexo ');readln(pessoa[i].sexo);
        write('Digite o salario ');readln(pessoa[i].salario);
        end;
     {rotina de ordenacao}
    for i:=i to n-1 do
       for j:=i+1 to n do
         if pessoa[i].nome > pessoa[j].nome then
           begin
            aux:=pessoa[i];
            pessoa[i]:=pessoa[j];
            pessoa[j]:=aux;
          end;
      {rotina de exibicao}
    clrscr;
    writeln('Valores Ordenados':48);
    for i:=1 to 80 do
        write(#205);
    writeln('Nome':15,'Sexo':15,'Idade':15,'Salario':15);
    for i:= 1 to n do
        with pessoa[i] do
          writeln(nome:15,sexo:15,idade:15,salario:15:2);
    readkey;
end.
```

Capitulo IX – Sub Rotinas

A complexidade do desenvolvimento dos softwares exigem algoritmos complexos que necessitam ser divididos em partes para poderem ser resolvidos.

O uso da modularização permitirá uma maior legibilidade dos software, uma melhor manutenção enfim um menor custo e prazo de desenvolvimento.

9.1 Utilização da Sub Rotinas

As sub rotinas podem ser as PROCEDURES e as FUNCTION que são trechos de programa que podem ser invocados em qualquer parte do programa retornando após o comando que a chamou.

A diferença entre os dois tipos está no fato que uma PROCEDURE pode ou não retornar um valor, enquanto uma FUNCTION sempre irá retornar um valor.

O Turbo Pascal dispõe de um conjunto de rotinas pré definidas. Algumas destas rotinas já foram examinadas nos capítulos anteriores.

As sub-rotinas aparecem no inicio do programa com a seguinte sintaxe:

```
Program nome;
Uses crt; {e demais unidades necessárias}
Var {variáveis globais}
…..
PROCEDURE <nome> [(parâmetros)]
  Var {variáveis locais}
  Begin
 ….
 End;
Begin

  <nome> [(parâmetros)]

end;
```

Procedures e function

Programa principal

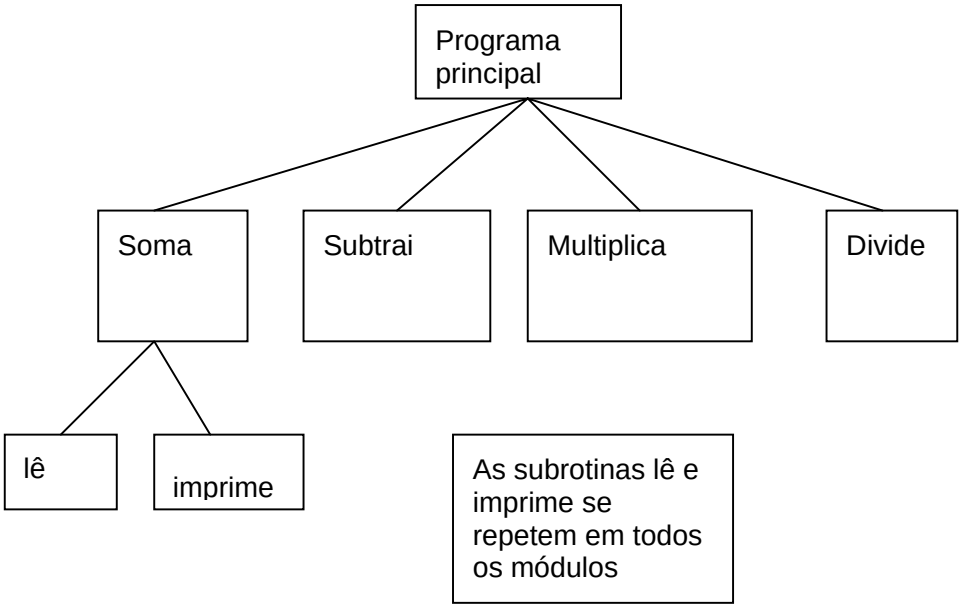
9.2 Variáveis Globais e variáveis Locais

Uma variável global é definida no início do programa (antes das procedures) e é vista por todo o programa e procedures.

Uma variável local é definida dentro de uma procedure e somente é vista pelo sub programa onde foi criado

9.3 Exemplo de programa usando procedures e variáveis globais

Este programa é para realizar as 4 operações matemáticas. O diagrama abaixo mostra como o programa principal está ligado as várias tarefas.




```
program subrotina;
uses crt;
var
{todas as variaveis sao globais}
   resp,a,b:real;
   opcao:char;
procedure le;
begin
 write('Digite o primeiro valor  ');readln(a);
 write('Digite o segundo valor   ');readln(b);
end;
procedure imprime;
begin
 writeln('A resposta e''  ',resp:0:2);
 write('Pressione qq tecla para voltar ao menu');
 readkey;
end;
procedure soma;
begin
 clrscr;
 writeln ('Rotina de Soma':30);
 le;
 resp:=a+b;
```

```
 imprime;
end;
procedure subtrai;
begin
 clrscr;
 writeln ('Rotina de Subtracao':30);
 le;
 resp:=a-b;
 imprime;
end;
procedure multiplica;
begin
 clrscr;
 writeln ('Rotina de Multiplicacao':30);
 le;
 resp:=a*b;
 imprime;
end;
procedure divide;
begin
 clrscr;
 writeln ('Rotina de Divisao':30);
 le;
 resp:=a/b;
 imprime;
end;
begin
{programa principal}
opcao:='0';
while opcao <> '5' do
 begin
 clrscr;
 gotoxy(33,1);write('Menu Principal');
 gotoxy(28,6);write('1.....................Soma');
 gotoxy(28,8);write('2..............Subtracao');
 gotoxy(28,10);write('3...........Multiplicacao');
 gotoxy(28,12);write('4.................Divisao');
 gotoxy(28,14);write('5.........Fim do Programa');
 gotoxy(28,18);write('Escolha uma opcao........');
 opcao:=readkey;
 if opcao <> '5' then
    case opcao of
      '1':soma;
      '2':subtrai;
      '3':multiplica;
      '4':divide;
     else
```

```
      gotoxy(27,24);writeln('Opcao invalida - tecle algo');
      readkey;
     end;{fim do case}
 end;{fim do while}
end.
```

9.4 Parâmetros

Os parâmetros tem como finalidade servir como pont de comunicação bidirecional entre a sub-rotina e o programa principal, ou com uma sub-rotina hierarquicamente de nível mais alto. Os parâmetros podem ser ***formais*** ou ***reais***
Serão parâmetros formais quando forem declarados através das variáveis juntamente com a identificação do nome da sub-rotina, os quais serão tratados exatamente da mesma forma como são tratadas as variáveis globais ou locais.
Considere o programa abaixo:

```
program parametro;
uses crt;

procedure soma(a,b:integer);
var
z:integer;
begin
 z:=a+b;
 writeln(z);{z e' variavel local}
 readkey;
end;

{programa principal}
var
 x,y,w,z:integer;
begin
 clrscr;
 write('Digite x ');readln(x);
 write('Digite y ');readln(y);
 soma(x,y);
 soma(3,4);
 end.
```

As variáveis a e b são parâmetros formais da sub-rotina soma. Z é uma variável local e somente é reconhecida dentro da sub-rotina. As variáveis X e Y são parâmetros reais assim como os valores 3 e 4.

9.5 Passagem de Parâmetros

A passagem de parâmetros pode ser por ser por ***valor*** ou por ***referência***

9.5.1 Passagem por valor

A passagem por valor caracteriza-se pela não alteração do valor do parâmetro real quando o parâmetro formal é manipulado dentro da sub-rotina.

```
program parametro_valor;
uses crt;

procedure soma(a,b,c:integer);
begin
 c:=a+b;
end;
{programa principal}
var
 w:integer;
begin
 clrscr;
 w:=9876;
 soma(3,4,w);
 writeln(w);readkey;
end.
```

Observe que a sub-rotina soma os valores de a com b armazenando em c.
No programa principal, passamos os parâmetros 3 e 4 para a sub rotina. O parâmetro w é um parâmetro real que foi passado para a sub-rotina como valor. Assim não pode ser alterado dentro da sub-rotina permanecendo com o valor existente antes da chamada.
O valor impresso será 9876, atribuído no início do programa.

9.5.2 Passagem por referência

A passagem de parâmetro por referência caracteriza-se pela ocorrência de valor do parâmetro real quando o parâmetro formal é manipulado dentro da sub-rotina.
A alteração efetuada no parâmetro formal corresponde a alteração no parâmetro real.
Identificamos a passagem por referência colocando na relação de parâmetros a expressão VAR.
Por exemplo

```
Procedure troca (var a,b :real);
Var aux:real;
Begin
 Aux:=a;
 A:=b;
 B:=aux;
End;
```

- Calcular o fatorial de um numero lido usando procedure.

```
program fatorial;
uses crt;
procedure fat(var val:longint;n:integer);
var
i:integer;
begin
 val:=1;
 for i:= 1 to n do
```

```
     val:=val*i;
end;
var
 resp:longint;
 k:integer;
begin
 clrscr;
 write('Digite um valor inteiro ');readln(k);
 fat(resp,k);
 writeln('O fatorial de ',k,' e'' ',resp);
 readkey;
end.
```

Neste caso a variável n é passada como valor e a variável val é passada como referência.

9.6 Function

Uma function é um bloco de programa, ao qual são validadas todas as regras vistas anteriormente. A diferença é que uma function sempre retorna um valor no próprio nome da função cujo tipo é definido na sua definição.

Uma function possui a seguinte sintaxe:

Function <nome> [(parâmetros)] :<tipo>;
var
<variáveis>
begin
<comandos>
end;

Por exemplo; Função para determinar o fatorial de um número

```
program fatorial2;
uses crt;
function fat(n:integer): longint;
var
fatorial:longint;
i:integer;
begin
 fatorial:=1;
 for i:= 1 to n do
     fatorial:=fatorial*i;
     fat:=fatorial;
end;
{programa principal}
var
 k:integer;
begin
 clrscr;
 write('Digite um valor inteiro ');readln(k);
 writeln('O fatorial de ',k,' e'' ',fat(k));
{fat(k) retorna com o fatorial de k}
 readkey;
end.
```

Capitulo X Arquivos

A manipulação de um conjunto de dados requer uma forma de armazenamento que permita ser acesso e tratamento de um grande volume de informações com facilidade.

10.1 Definição do Arquivo

Os arquivos em Pascal podem ser texto ou de tipo definido chamado de “tipados” e compostos de registros.

A sintaxe é a seguinte:	Onde:
Type <arquivo> = [text] ou [file of <tipo>] var <variável> : <arquivo>;	<arquivo> nome do tipo do arquivo; <tipo> tipo do arquivo (text, string, integer, record) <variável> nome que será usada dentro do programa para representar o arquivo

10.2 Principais comandos com arquivos

Instrução	Sintaxe	Descrição
Assign	Assign(<variável>,’caminho:\ arquivo);	Associa a variável ao arquivo gravado
Rewrite	Rewrite(<variável>);	Cria o arquivo com o nome da variável.Se o arquivo já existir, apaga o antigo e re-cria um novo.
Reset	Reset(<variável>);	Abre um arquivo existente e posiciona o ponteiro no 1° registro
Append	Append(<variável>)	Abre um arquivo texto existente e posiciona o ponteiro no ultimo registro
Write	Write(<variável>, dados);	Grava no arquivo <variável> os dados
Writeln	Writeln(<variável>,dados);	Grava em um arquivo texto os dados
Read	Read(<variável>,dados);	Le do arquivo <variável> os dados
Readln	Readln(<variável>,dados);	Le de um arquivo texto <variável> os dados
Close	Close(<variável>);	Fecha o arquivo
Seek	Seek(<variável>,posição);	Posiciona o ponteiro na posição indicada. O primeiro registro é 0 (zero)
Filesize	Filesize(<variável>);	Retorna o numero de registros do arquivo
Filepos	Filepos(<variável>);	Retorna o numero do registro onde está posicionado. O primeiro registro
Not eof	Not eof(<variável>);	Verifica o final do arquivo
Not eoln	Not eoln(<variável>);	Verifica o final da linha em um arquivo texto

10.3 Métodos de acesso

Os arquivos são um conjunto de informações gravadas fisicamente de forma seqüencial. Entretanto, podemos acessar as informações de 3 formas distintas:

Acesso seqüencial: as informações são acessadas uma após a outra assim como em uma fita k-7. Há a necessidade da leitura da informação anterior para o prosseguimento da leitura.

Acesso direto ou randômico: as informações são acessadas diretamente sem a necessidade da leitura da informação anterior. Os discos CD permitem que localizemos uma determinada música sem ouvirmos a trilha anterior.

Acesso seqüencial indexado: O arquivo tem um outro arquivo auxiliar que contém o endereço da informação desejada. É como se procurássemos no índice a informação desejada e fossemos diretamente a página que procuramos.

No nosso curso veremos apenas os acessos seqüenciais e de acesso direto.

10.4 Exemplos:

- Arquivo tipo texto.

O programa abaixo lê uma frase e grava a frase lida. Depois inclui novas frases. Finalmente exibe todas as frases gravadas.

```
program arq_texto;
uses crt;
var
   arq:text;
   frase:string[255];
   cont:char;
begin
clrscr;
assign(arq,'a:arq1.txt');{associa arq ao arquivo a:\arq1.txt}
rewrite(arq);{cria o arquivo arq);
{trecho para ler uma frase e gravar no arquivo}
cont:='s'; {continua = sim}
writeln('Leitura dos dados');
while cont <> 'N' do
      begin
      write('Digite uma frase ');
      readln(frase);{le do teclado uma frase}
      writeln(arq,frase);{escreva em arq a frase lida}
      write('Deseja continuar ? S/N ');
      readln(cont);
      cont:=upcase(cont); {transforma em maiuscula}
      end;
close(arq);{fecha o arquivo arq}
{trecho para incluir novas frases no final do arquivo}
append(arq); {abre posicionando no final do arquivo}
writeln('Inclusao de novos dados');
cont:='s'; {continua = sim}
```

```
while cont <> 'N' do
      begin
      write('Digite uma frase ');
      readln(frase);{le do teclado uma frase}
      writeln(arq,frase);{escreva em arq a frase lida}
      write('Deseja continuar ? S/N ');
      readln(cont);
      cont:=upcase(cont); {transforma em maiuscula}
      end;
close(arq);{fecha o arquivo arq}
{trecho para exibir as informacoes do arquivo}
reset(arq);{abre posicionando no inicio do arquivo}
writeln('Informacoes do arquivo');
while not eof(arq) do {enquanto nao for o final do arquivo
repita}
      begin
      readln(arq,frase);{leia em arq uma frase}
      write(frase,' '); {exiba na tela a frase lida}
      end;
readkey;
end.
```

- Arquivos “Tipados”

Para facilidade de entendimento adotaremos um arquivo de empregados de uma empresa onde cada registro contem o código do empregado, seu nome, idade e sexo.

- Cadastra os empregados

```
program arq_tipado;
uses crt;
type
    {define o registro empregado}
    dados_empreg = record
    codigo:string[5];
    nome:string[30];
    idade:integer;
    sexo:char;
    end;
var
   cad: file of dados_empreg;
   empregado:dados_empreg;
   cont:char;
begin
clrscr;
assign(cad,'a:\empreg.dat');{associa cad a empreg.dat}
rewrite(cad);{cria cad}
writeln('Entrada de dados');
cont:='S';
```

```
while cont <> 'N' do
      with empregado do {simplifica a descricao do campo}
      begin
      write('Digite o nome ');readln(nome);
      write('Digite a idade ');readln(idade);
      write('Digite o sexo ');readln(sexo);
      write(cad,empregado);
      write('Deseja continuar ? S/N '); readln(cont);
      cont:=upcase(cont);
      end;
{filesize mostra o numero de registros do arquivo}
writeln(filesize(cad),' registros cadastrados');
close(arq);
readkey;
end.
```

- Inclusão de um novo empregado

```
program arq_tipado;
uses crt;
type
    {define o registro empregado}
    dados_empreg = record
    codigo:string[5];
    nome:string[30];
    idade:integer;
    sexo:char;
    end;
var
   cad: file of dados_empreg;
   empregado:dados_empreg;
   cont:char;
   novo_empreg:dados_empreg;
   achei: boolean;
   n_empreg:integer;
begin
clrscr;
assign(cad,'a:\empreg.dat');{associa cad a empreg.dat}
reset(cad);{abre o arquivo criado anteriormente}
{leitura dos dados do novo empregado}
writeln('Empregado a incluir');
while cont <> 'N' do
begin
 with novo_empreg do
  begin
      write('Digite o codigo do empregado ');readln(codigo);
      write('Digite o nome ');readln(nome);
      write('Digite a idade ');readln(idade);
      write('Digite o sexo ');readln(sexo);
```

```
   end;
 {rotina de critica dos dados}
 {verifica se o empregado ja e cadastrado}
 achei:=false;
 while not eof(cad) and (achei=false) do
      begin
      read(cad,empregado);
      if empregado.codigo = novo_empreg.codigo then
         achei:=true;
      end;
 {se tiver achado nao pode ser cadastrado de novo}
 if achei = true then
   write('Empregado ja cadastrado ')
 else
    write(cad,novo_empreg);
 write('Deseja continuar ? s/n ');
 readln(cont);
 cont:=upcase(cont);
 end;
 close(cad);
 readkey;
end.
```

- Edita os dados de um empregado

```
program arq_tipado;
uses crt;
type
    dados_empreg = record
    codigo:string[5];
    nome:string[30];
    idade:integer;
    sexo:char;
    end;
var
   cad: file of dados_empreg;
   empregado:dados_empreg;
   cont:char;
   k:integer;
   novo_empreg:dados_empreg;
   achei: boolean;
   n_empreg:integer;
begin
clrscr;
assign(cad,'a:\empreg.dat');{associa cad a empreg.dat}
reset(cad);{abre o arquivo criado anteriormente}
{leitura dos dados do novo empregado}
writeln('Empregado a Alterar');
```

```
while cont <> 'N' do
begin
write('Digite o codigo do empregado
');readln(novo_empreg.codigo);
 {rotina de critica dos dados}
 {verifica se o empregado ja e cadastrado}
 achei:=false;
 while not eof(cad) and (achei=false) do
      begin
      read(cad,empregado);
      if empregado.codigo = novo_empreg.codigo then
         achei:=true;
      k:=k+1;
      end;
{a posicao em que foi encontrado e k-1}
 if achei = true then
  begin
  writeln('Dados atuais');
  with empregado do
       writeln(codigo,'  ',nome,'  ',idade:2,'  ',sexo);
   writeln('Dados a alterar');
   with novo_empreg do
      begin
      write('Digite o nome ');readln(nome);
      write('Digite a idade ');readln(idade);
      write('Digite o sexo ');readln(sexo);
   end;
   seek(cad,k-1);{posiciona onde foi encontrado}
   write(cad,novo_empreg);
   end
 else
   write('Empregado nao cadastrado');
  write('Deseja continuar ? s/n ');
  readln(cont);
  cont:=upcase(cont);
 end;
 seek(cad,0);
 writeln;writeln('Cadastro atual');
 while not eof(cad) do
      begin
      read(cad,empregado);
      with empregado do
       writeln(codigo,'  ',nome,'  ',idade:2,'  ',sexo);
      end;
 close(cad);
 readkey;
end.
```

- Retira empregado do cadastro

```
program arq_tipado;
uses crt;
type
    {define o registro empregado}
    dados_empreg = record
    codigo:string[5];
    nome:string[30];
    idade:integer;
    sexo:char;
    end;
var
   cad: file of dados_empreg;
   aux: file of dados_empreg;
   empregado:dados_empreg;
   cont:char;
   k:integer;
   cod_demitido:string[5];
   achei: boolean;
begin
clrscr;
assign(cad,'a:\empreg.dat');{associa cad a empreg.dat}
assign(aux,'a:\auxiliar.dat');
reset(cad);{abre o arquivo criado anteriormente}
rewrite(aux);
{leitura dos dados do novo empregado}
writeln('Empregado a Retirar');
while cont <> 'N' do
begin
write('Digite o codigo do empregado ');readln(cod_demitido);
 {rotina de critica dos dados}
 {verifica se o empregado ja e cadastrado}
 achei:=false;
 while not eof(cad)  do
      begin
      read(cad,empregado);
      if empregado.codigo = cod_demitido then
         begin
         writeln(empregado.nome,' ',empregado.idade,'
',empregado.sexo);
         achei:=true;
         end
       else
         write(aux,empregado);
        end;
 if achei = true then
   writeln('Empregado removido')
```

```
 else
  writeln('Empregado nao cadastrado');
  write('Deseja continuar ? s/n ');
  readln(cont);
  cont:=upcase(cont);
 end;
 rewrite(cad);{apaga cad e mantem aberto}
 seek(aux,0);{posiciona no 1 registro}
 while not eof(aux) do
  begin
   read(aux,empregado);
   write(cad,empregado);
  end;
 erase(aux);
 seek(cad,0);
 writeln;writeln('Cadastro atual');
 while not eof(cad) do
      begin
      read(cad,empregado);
      with empregado do
       writeln(codigo,'  ',nome,'  ',idade:2,'  ',sexo);
      end;
 close(cad);
 readkey;
end.
```

Capitulo XI Conjuntos

O conceito de conjunto é usado em matemática como uma coleção de objetos, números etc.
Em Pascal, um conjunto pode ser composto por zero elementos – conjunto vazio – ou por vários elementos – no máximo 256.
Os elementos integrantes de um conjunto são representados em colchetes e separados por virgulas, como por exemplo:

[1,3,5,7,9] - alguns inteiros
[3..7] - inteiros entre 3 e 7
[‘A’..’Z’] - letras maiúsculas de A até Z
[golf, marea,audi] - marcas de carro
[] - conjunto vazio

Declaração
A forma geral para a definição de conjuntos é:

```
Type
<identificador> = set of <tipo>;
```

Exemplos

```
Type
        Caracteres = set of char;
        Letras_maiusculas = set of ‘A’..’Z’;
        Digitos = set of 0..9;
        Carros = set of (golf,marea,audi);
Var
        Letras:letras_maiusculas;
        Marca: carros;
```

Operações em tipo SET
Atribuição
A atribuição é realizada de forma semelhante aos demais tipos:

```
C:=[‘a’,’e’, ‘i’, ‘o’, ‘u’];
Numero:=[3,4,5];
```

União
O operador de união é representado pelo sinal + fazendo que esta união resulte em um terceiro conjunto constituido por elementos dos outros dois.
Por exemplo:

```
a:=[1,2,3];
B:=[2,3,4,5];
C:=a+b;   resulta em c=[1,2,3,4,5]
```

Interseção
Ë representada pelo operador * resultando por um terceiro conjunto com os elementos comuns aos outros dois conjuntos:
Por exemplo:

A:=[1,2,3];
B:=[2,3,4,5];
C:=a*b; resulta em c= [2,3]

Diferença
É representada pelo operador – resultando em um terceiro conjunto cujos elementos estão em um conjunto e não no outro;
Por exemplo:
A:=[1,2,3,6];
B:=[2,3,4,5];
C:=a-b; resulta em c=[1,6];
C:=b-a; resulta em c=[4,5]

Operadores relacionais
A= B todos os elementos estão em ambos conjuntos;
A<>B alguns ou todos os elementos não estão em ambos conjuntos
A>= B todos os elementos de B estão em A
A<= B todos os elementos de A estão em B
A IN B A é um elemento de B – este operador é o que apresenta maior uso nos programas.
Exemplos

```
program set1;
uses crt;
{le uma tecla e informa se e' maiuscula, minuscula ou numero}
type
simbolo= set of char;
var
maiu,minu,nume:simbolo;
tecla:char;
begin
     clrscr;
     maiu:=['A'..'Z'];
     minu:=['a'..'z'];
     nume:=['0'..'9'];
     repeat
           write('Pressione uma tecla '); tecla:= readkey;
           if tecla IN maiu then
              writeln('Voce pressionou letra maiuscula')
           else
              if tecla IN minu then
                 writeln('Voce pressionou letra minuscula')
              else
                  if tecla IN nume then
                     writeln('Voce pressionou numero')
                  else
                      writeln('Voce pressionou outra tecla');
     until tecla = #27;
end.
program altera;
```

```
uses crt;
var
nomel:string[30];
nome:array[1..30] of char;
i,j,l:integer;
letra: set of char;
t:char;
begin
clrscr;
writeln('Alteracao de dados de Clientes':45);
gotoxy(1,3);
write('Digite o Nome do Cliente ');readln(nomel);
letra:=['A'..'Z']+['a'..'z']+[' '];
        gotoxy(1,4);Write('Nome ');
        gotoxy(10,4);write(nomel);
        gotoxy(10,4);
        t:=readkey;
        if (t in letra) then
           begin
           gotoxy(10,4);clreol;
           nome[1]:=t;
           end
           else
           begin
            writeln('Voce pressionou dif de letra');
            readkey;
            exit;
           end;
        j:=1;
        l:=11;

        while (nome[j] in letra) do
              begin
              gotoxy(l,4);
              write(nome[j]);
              l:=l+1;
              j:=j+1;
              nome[j]:=readkey;
              end;
              writeln;
        writeln('Nome Alterado ');
        i:=1;
        while nome[i] <> #13 do
        begin
        write(nome[i]);
        i:=i+1;
        end;
        readkey;
end.
```

Capitulo XII – Unit

Na linguagem Pascal existem várias Unit pré definidas que já vem sendo utilizadas como por exemplo a CRT, DOS, entre outras.

Entretanto, podemos criar nossas próprias unidades que funcionarão como programas autônomos que serão ligados aos programa principal.

Compilar um programa Pascal para a memória irá gerar um arquivo .PAS. Compilar um programa Pascal para o disco irá gerar um arquivo .EXE.

Compilar uma Unit irá gerar um arquivo TPU.

A sintaxe de uma Unit é

UNIT nome da unidade;

INTERFACE

Declaração de outras unidades utilixzadas;

Declaração das procedures e funções;

IMPLEMENTATION

Desenvolvimento das procedures e funções;

END.

Exemplo: Criar uma unidade que escreva “Estou na procedure”.

1) Digite o programa:

```
unit uni1;
interface
         uses crt;
         procedure achei;
implementation
         procedure achei;
         begin
         clrscr;
         writeln('Estou na procedure');
         end;
end.
```

1.1 Não esqueça de alterar o destino da compilação para o disco !!

1.2 O nome usado para salvar o arquivo será a referência para a ligação com o programa principal

2 Digite agora o programa principal:

```
program unidade2;
uses crt,uni1;
     begin
     achei;
     readkey;
end.
```