UNISINOS - UNIVERSIDADE DO VALE DO RIO DOS SINOS
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS (C6/6) – Curso: Informática

PROGRAMAÇÃO I – AULA 12

Disciplina: Linguagem de Programação PASCAL
Professor responsável: *Fernando Santos Osório*
Semestre: 2001/2
Horário: 63

E-mail: *osorio@exatas.unisinos.br*
Web:
http://inf.unisinos.br/~osorio/prog1.html
Xerox : *Pasta 54* (Xerox do C6/6)

1. Vetores Uni e Multidimensionais - ARRAY:

Variáveis estruturadas, compostas por um conjunto de variáveis iguais:
- Estruturas homogêneas = Série de elementos iguais => *Vetores / Array / Matriz*

Os arrays unidimensionais (vetores) ou arrays multidimensionais (matrizes) são compostos por um número fixo de elementos do mesmo tipo, onde cada elemento é acessado individualmente através do seu índice. O índice é um valor inteiro e positivo que permite indicar qual dos 'N' elementos do vetor que estamos fazendo referência em um certo momento. Os índices são indicados entre colchetes, por exemplo, Vetor[6] usualmente indica o acesso ao sexto elemento de uma variável chamada de 'Vetor' (para um vetor unidimensional), e Matriz[3,4] usualmente indica o acesso ao elemento da terceira linha e quarta coluna de uma variável chamada 'Matriz' (para uma matriz de duas dimensões – linhas ***x*** colunas).

Sendo assim, nós poderemos ter vetores com uma, duas, três, quatro, até 'N' dimensões; só depende de nossa imaginação (e da capacidade de memória do micro para armazenar este tipo de estruturas de dados). Uma matriz com três dimensões tem seus elementos acessados através de uma referência à uma variável como a deste exemplo:

```
Matriz3D[x,y,z] := Valor;  {onde x, y e z são os três índices, um para cada dimensão da matriz }
```

Os vetores multidimensionais são como vetores de vetores, ou seja, cada elemento do vetor é um novo vetor. Para ler todos elementos de um vetor de 4 x 5 x 6 elementos poderíamos usar um conjunto de FOR aninhados, onde os contadores seriam usados como índices do vetor. Exemplo:

```
FOR  X := 1 TO 4
DO   FOR  Y := 1  TO  5
       DO   FOR  Z :=  1  TO  6
              DO  READLN ( Matriz[ x, y, z] );
```

* SINTAXE:

A declaração de um array multidimensional é feita assim...

```
<nome_do_vetor>  :  ARRAY  [ <início> ..  <fim> , <início> .. <fim> ]  OF  <tipo_da_variável> ;
<nome_do_vetor>  :  ARRAY  [ <início> ..  <fim> , <início> .. <fim> , <inicio> .. <fim> ]
                      OF  <tipo_da_variável> ;
<nome_do_vetor>  :  ARRAY  [ <início> ..  <fim> , <início> .. <fim> , <inicio> .. <fim>  ,  <etc> ]
                      OF  <tipo_da_variável> ;
```

Usando os vetores...

<nome_do_vetor> [<índice_X> , <índice_Y>] := <valor>;

<nome_da_variável> := <nome_do_vetor> [<índice_X> , <índice_Y> , <índice_Z>] ;

Write (<nome_do_vetor> [<IX> , <IY>, <IZ>, <IW>]) ;

Readln (<nome_do_vetor> [<I>, <J>]) ;

Exemplos:

```
PROGRAM Exemplo;
VAR
    Nota :  ARRAY  [ 1.. 10 , 1..3]  OF  REAL;
    Aluno, Grau : INTEGER;
BEGIN
   WriteLn (‘>> Cadastro de um Turma de até 10 alunos, com 3 notas cada um <<’);
   WriteLn;
    FOR  Aluno := 1  TO  10
    DO  FOR Grau := 1 TO 3
         DO  BEGIN
                Writeln (‘Entre com a nota ‘, Grau,’ do aluno ‘,Aluno,’ : ‘ );
                ReadLn ( Nota[Aluno,Grau] );
             END;
    WriteLn (‘A primeira nota do primeiro aluno é :  ‘, Nota [1, 1] );
    WriteLn (‘A última nota do último aluno é : ‘, Nota[10, 3] );
END.
```

Observações:

- Os arrays multidimensionais usualmente ocupam muito espaço em memória. Por exemplo, imagine um array do seguinte tipo: Array[1..10, 1..10, 1..10] of Integer. Este array irá ocupar um total de memória de 2000 bytes (10*10*10*2, onde cada integer ocupa 2 bytes).

- Lembre-se que o Turbo Pascal é um compilador de 16 bits, onde as estruturas de dados são limitadas a um máximo de 64Kbytes de memória. Supondo um array criado para armazenar dados de uma agenda de compromissos (armazena um compromisso/string para cada dia/mês do ano). Se for declarado como sendo um Array [1..31, 1..12] of string, esta estrutura de dados irá ocupar 31*12*256 = 95232 bytes e consequentemente irá provocar um ***estouro de memória***!

- Quando desejamos passar como parâmetro um array inteiro, somos obrigados a criar um tipo de dados específico para definir este array e usar o tipo de dados definido na declaração dos parâmetros da função ou procedure. O Turbo Pascal não aceita que sejam declarados parâmetros do tipo array, sem que estes tenham sido previamente declarados através de um type. Exemplo:

```
Program Jogo_da_Velha;

Type
   Tipo_Tabuleiro = Array [1..3, 1..3] of Char;

Procedure Exibe_Tabuleiro (Tabuleiro: Tipo_Tabuleiro);
Var
    Linha,Coluna:integer;
Begin
   For Linha:=1 to 3
   Do Begin
         For Coluna:= 1 to 3
         Do Write (Tabuleiro[Linha,Coluna],' ');
         WriteLn;
       End;
End;

Var
  T: Tipo_Tabuleiro;
Begin
  ...  Exibe_Tabuleiro(T); ...
End.
```

* **Não** podemos declarar a procedure do seguinte modo:
Procedure Exibe_Tabuleiro (Tabuleiro: Array [1..3, 1..3] of Char); ← O pascal não aceita!

2. Usando a Impressora no Turbo PASCAL

Para escrever na impressora no Turbo Pascal, basta que seja declarada uma nova UNIT junto com as outras Units do comando USES do Pascal:

```
Uses
    Crt, Dos, Printer;
    { Utiliza a Unit CRT e também a Unit DOS e a Unit PRINTER }
```

Depois disso basta que nos comandos Write ou WriteLn o primeiro parâmetro indicado seja a palavra LST. Por exemplo:

```
WriteLn (Lst, ' Hello World ');   { Escreve "Hello World" na impressora }
```

Observações importantes:

- Note que para a impressora não funcionam os comandos como o GOTOXY. Não podemos posicionar o cursor na hora da impressão em qualquer posição do papel. No máximo podemos deslocá-lo para o lado imprimindo espaços em branco...

- E lembre-se de inicializar a impressora na rede Novell (batch DOS = "matric-<sala>").

EXERCÍCIOS – AULA 12

1. Faça um programa que crie uma matriz de inteiros de 5 linhas por 10 colunas. Leia os valores desta matriz linha após linha e exiba a matriz na tela coluna por coluna.

2. Faça um programa que crie uma matriz 3x3x3 onde cada elemento da matriz seja igual a soma dos seus índices (exemplo: M[1,2,1] = 1+2+1 = 4). Crie uma função que obtenha a soma de todos elementos da matriz, e uma outra sub-rotina que obtenha soma dos elementos cujos valores são pares e a soma dos elementos cujos valores são ímpares. Exibir na tela os valores da soma total, soma dos pares e soma dos ímpares.

3. Faça um programa que leia três vetores independentes compostos por 5 números que são fornecidos pelo usuário. Crie uma matriz que reuna estes três vetores em uma única estrutura. Faça uma *procedure* que exiba na tela o conteúdo da matriz. Faça uma *function* que receba a matriz como parâmetro e retorne o maior valor contido nesta matriz, e antes de terminar a execução do programa exiba este valor na tela.

4. Faça um programa para armazenar em uma matriz os compromissos de uma agenda pessoal. Cada dia do mês deve conter 24 horas, onde para cada uma destas 24 horas podemos associar um tarefa específica (compromisso agendado). O programa deve ter um menu onde o usuário indica o dia do mês que deseja alterar e a hora, entrando em seguida com o compromisso, ou então, o usuário pode também consultar a agenda, fornecendo o dia e a hora para obter o compromisso armazenado.

5. Modifique o programa anterior de maneira a guardar os compromissos de todo o ano, organizados por mês, dia e hora (só 8 horas por dia).

6. Crie uma rotina que converta a matriz multi-dimensional de compromissos do programa anterior em uma matriz uni-dimensional.

7. Faça um programa para jogar o jogo da velha. O programa deve permitir que dois jogadores façam uma partida do jogo da velha, usando o computador para ver o tabuleiro. Cada jogador vai alternadamente informando a posição onde deseja colocar a sua peça (‘O’ ou ‘X’). O programa deve impedir jogadas inválidas e determinar automaticamente quando o jogo terminou e quem foi o vencedor (jogador1 ou jogador2). A cada nova jogada, o programa deve atualizar a situação do tabuleiro na tela.

Bom Trabalho!