Programação II
AULA 04 – Matrizes

8 Matrizes

Os vetores são estruturas capazes de armazenar diversos valores do mesmo tipo. Assim como os vetores, existem as matrizes, que também são estruturas capazes de armazenar diversos valores do mesmo tipo. A diferença entre eles é que os vetores são estruturas unidimensionais, já as matrizes são estruturas multidimensionais.

Declaração de Matriz em C:

Para definirmos uma variável do tipo matriz, utilizamos a seguinte sintaxe:

Sintaxe:

tipo lista-de-identificadores [índice1, índice2, ..., índiceN];

onde:
tipo é o tipo dos componentes da variável
lista-de-identificadores é o nome da matriz que se deseja declarar;
índice1 é o tamanho (número de elementos) do índice 1;
índice2 é o tamanho (número de elementos) do índice 2;
...

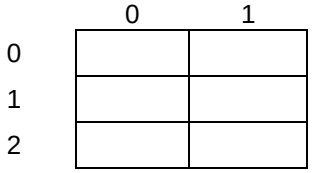
As matrizes em geral são caracterizadas por se tratarem de uma única variável de um determinado tamanho que armazena várias informações do mesmo tipo. Essas informações são gravadas na memória seqüencialmente e são referenciadas através de índices.

Matrizes Bidimensionais

São matrizes linha-coluna, onde o primeiro índice indica a linha e o segundo a coluna. Esse tipo de matriz é considerado o caso mais simples de matrizes multidimensionais.

Exemplo:

```
int m[3][2];
```



Veja o programa abaixo:

```
#include<stdio.h>
#include<conio.h>
int main()
{
     int mat[2][2];
     float det;
     int x,y;

     printf("Este programa serve para cadastrar os dados na matriz e exibir o dobro de
     cada elemento");
     printf("\n\nEntre com os valores da matriz:\n");
     for(x=0;x<2;x++)
     {
        for(y=0;y<2;y++)
        {
           printf("mat[%d][%d]=",x+1,y+1);
           scanf("%d",&mat[x][y]);
        } /*fim do for y*/
     } /*fim do for x*/

     printf("\n\nExibindo o dobro de cada elemento da matriz:\n");

     for(x=0;x<2;x++)
     {
        for(y=0;y<2;y++)
        {
           printf("\nmat[%d][%d]=%d",x,y, mat[x][y] * 2);
        } /*fim do for*/
     } /*fim do for*/

     getch();
     return 0;
} /*fim do programa*/
```

OBS: Note que foi preciso utilizar dois comandos para, um dentro do outro. Isto porque o comando para mais externo controla as linhas da matriz m e o comando para mais interno controla as colunas.

Matrizes Multidimensionais

A seguir vamos mostrar um exemplo de uma matriz de três dimensões, lembrando que o tipo matriz é multidimensional, ou seja, podemos declarar uma matriz com quantas dimensões quisermos, apesar da matriz bidimensional ser mais comumente utilizada.

Exemplo:

```
ma[3][3][2]: inteiro;
```

Matrizes Não-dimensionadas

As matrizes não-dimensionais são aquelas cujo tamanho não é especificado. Nesse caso o compilador cria uma matriz grande para conter todos os seus elementos. As declarações de matrizes unidimensionais com essa característica podem ser vista no programa abaixo:

```
#include<stdio.h>
#include<conio.h>
int main()
{
     char vetor1[37]="Estou aprendendo a programar em C!!!";
     char vetor2[]="Estou aprendendo a programar em C!!!"; /*vetor não-dimensionado*/
     printf("O vetor abaixo foi declarado com o seu tamanho especificado\n");
     printf("%s\n",vetor1);
     printf("\n");
     printf("E este outro foi declarado com o seu tamanho nao especificado\n");
     printf("%s\n",vetor2);
     getch();
     return 0;
}
```

Observe na saída, que os vetores (vet1 e vet2) declarados de forma diferentes obtiveram os mesmo efeitos. A diferença é que no vetor vet2 não foi especificado a quantidades de caracteres (vetor não-dimensionado).

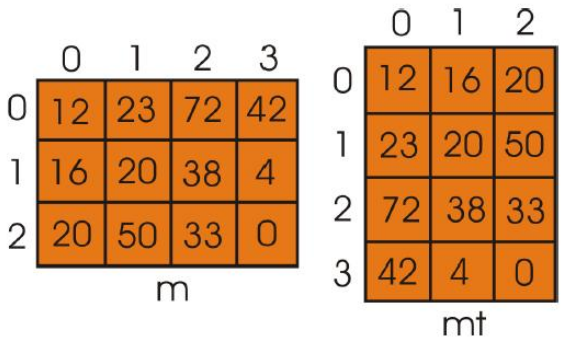
As matrizes multidimensionais não-dimensionadas devem ter apenas seu primeiro índice não especificado, os outros devem ser indicados para que o compilador possa indexar de forma correta as matrizes. Desta forma pode-se criar tabelas com diversos tamanhos, sem a necessidade de mudar as dimensões da matriz.

```
#include<stdio.h>
#include<conio.h>
int main()
{
     int mat1[2][2]={4,5,-2,1};
     int mat2[][2]={4,5,-2,1}; /*Matriz não-dimensionada*/
     int x,y;
     printf("\n Imprimindo a matriz mat1 cujo o tamanho foi especificado:\n");
     for(x=0;x<2;x++)
     {
        for(y=0;y<2;y++)
        {
           printf("\n mat1[%d][%d]=%d",x,y,mat1[x][y]);
        } /*fim do for*/
     } /*fim do for*/
     printf("\n\n Imprimindo a matriz mat2 cujo o tamanho nao foi especificado:\n");
     for(x=0;x<2;x++)
     {
        for(y=0;y<2;y++)
        {
           printf("\n mat2[%d][%d]=%d",x,y,mat2[x][y]);
        } /*fim do for*/
     } /*fim do for*/
     getch();
     return 0;
} /*fim do programa*/
```

Do mesmo modo verifica-se através da saída abaixo que tanto a matriz mat1 como a matriz mat2, obtiveram os mesmos resultados embora tenham sido declaradas de maneira diferente.

EXERCÍCIO PARA LABORATÓRIO

EL_10) Faça um algoritmo que leia uma matriz bidimensional 3x4, em seguida, calcule e mostre sua transposta. Exemplo: a figura abaixo mostra a matriz m e sua transposta mt.



EL_11) Faça um algoritmo e um programa capaz de ler uma matriz A e uma matriz B com três linha e cinco colunas, em seguida faça a soma das matrizes e armazene na matriz C.

EL_12) Faça um algoritmo e um programa capaz de ler uma matriz A de ordem 5x5, em seguida calcule a multiplicação dos elementos da diagonal principal.