UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO
CIC107 – PROGRAMAÇÃO DE COMPUTADORES I

10ª Aula prática

Estruturas homogêneas : matrizes

Uma matriz (*array bidimensional),* assim como um vetor, também é uma variável que serve para guardar vários valores do mesmo tipo na memória. A diferença é que agora são necessários dois índices: um para indicar a linha e outro para indicar a coluna onde o elemento está situado.

Também de forma semelhante aos vetores, para definir uma matriz precisamos indicar o número máximo de elementos que ela poderá ter; para isso estabelecemos o número máximo de linhas e de colunas que esperamos ser suficientes. Também é necessário indicar o tipo de seus elementos.

Isso é feito da seguinte forma:

<tipo> identificador [<número de linhas>][<número de colunas>];

- Primeiro o tipo de dado: **int, float, double**, ...
- Segundo o nome da variável
- E por fim, o número de linhas e de colunas, entre colchetes separados.

No exemplo a seguir, observamos que para percorrer uma matriz são necessários dois laços de repetição: um faz variar o índice de linha (i) e o outro faz variar o índice de coluna (j).

```
#include <iostream>
using namespace std;
main()
{
        int mat[3][3];    // matriz de 3 linhas e 3 colunas
        int i,j;
        //Inserção de dados na matriz
        for (i = 0; i < 3; i++) {
            cout << ”Digite os numeros da linha " << i << endl;
            for (j = 0; j < 3; j++)
                cin >> mat[i][j];
        }
        //Visualização dos dados
        cout << "Conteudo da matriz" << endl;
        for (i = 0; i < 3; i++) {
            for (j = 0; j < 3; j++)
```

```
                cout << mat[i][j] << '\t';
            cout << endl;
        }
        system("pause");
}
```

Para facilitar a entrada de dados, pode ser conveniente definí-los junto com a matriz, da seguinte forma:

```
#include <iostream>
using namespace std;
main()
{
        int mat[3][3] = { {1,2,3}, {4,5,6}, {7,8,9} };
        int i,j;
        //Visualização dos dados
        cout << "Conteudo da matriz" << endl;
        for (i = 0; i < 3; i++) {
            for (j = 0; j < 3; j++)
                cout << mat[i][j] << '\t';
            cout << endl;
        }
        system("pause");
}
```

Naturalmente, essa forma de atribuição de valores serve também para vetores.

Veja a seguir um exemplo de como criar um tipo matriz que pode ser usado em vários lugares no programa.

```
#include <iostream>
using namespace std;

typedef int matriz[3][5];

int contaZeros(matriz k, int m, int n) {
    // conta os zeros em uma matriz mxn
    int i,j;
    int cont=0;
    for (i=0; i<m; i++)
       for (j=0; j<n; j++)
          if (k[i][j]==0)  cont++;
    return cont;
}

main() {
   matriz m = {{0,20,6}, {30,0,1}, {-1,-1,-1}};
```

```
   int i,j,zeros;
   zeros = contaZeros(m,2,2);
   cout << "Quantidade de zeros na matriz = " << zeros << endl;
   system("pause");
}
```

Embora todos os exemplos e exercícios aqui apresentados tratem de matrizes bidimensionais, na verdade podemos criar matrizes de três ou mais dimensões, utilizando um número de índices correspondente.

Exercícios

1. Somar e imprimir os elementos de cada linha de uma matriz constituída por 4 linhas e 3 colunas de números reais.
2. Somar e imprimir os elementos de cada coluna de uma matriz constituída por 4 linhas e 3 colunas de números reais.
3. Dada uma matriz quadrada de números inteiros, contar quantos números negativos existem em sua diagonal principal.
4. Dada uma matriz formada por m linhas e n colunas de números reais, criar um vetor de n elementos onde cada elemento seja a soma dos valores encontrados na coluna correspondente da matriz.
5. Escreva uma função para realizar a soma de duas matrizes formadas por m linhas e n colunas.
   O cabeçalho dessa função será:
   void somaMatrizes(matriz a, matriz b, matriz c, int m, int n)
   Escreva uma função "main" que defina duas matrizes, chame a função acima para somá-las e finalmente imprima as três matrizes.