Matrizes

Já vimos como declarar matrizes unidimensionais (vetores). Vamos tratar agora de matrizes bidimensionais. A forma geral da declaração de uma matriz bidimensional é muito parecida com a declaração de um vetor:

tipo_da_variável nome_da_variável [altura][largura];

É muito importante ressaltar que, nesta estrutura, o índice da esquerda indexa as linhas e o da direita indexa as colunas. Quando vamos preencher ou ler uma matriz no C o índice mais à direita varia mais rapidamente que o índice à esquerda. Mais uma vez é bom lembrar que, na linguagem C, os índices variam de zero ao valor declarado, menos um; mas o C não vai verificar isto para o usuário. Manter os índices na faixa permitida é tarefa do programador. Abaixo temos um exemplo do uso de uma matriz:

Exemplo 01:

```
#include <stdio.h>
int main ()
{
int mtrx [20][10];
int i,j,count;
count=1;
for (i=0;i<20;i++){
        for (j=0;j<10;j++)
        {
                mtrx[i][j]=count;
                 count++;
         }
}
return(0);
}
```

No exemplo, a matriz **mtrx** é preenchida, sequencialmente por linhas, com os números de 1 a 200. Você precisa entender o funcionamento do programa acima antes de prosseguir.

Matrizes de strings

Matrizes de strings são matrizes bidimensionais. Imagine uma string. Ela é um vetor. Se fizermos um vetor de strings estaremos fazendo uma lista de vetores. Esta estrutura é uma matriz bidimensional de **chars**. Podemos ver a forma geral de uma matriz de strings como sendo:

char nome_da_variável [num_de_strings][compr_das_strings];

Aí surge a pergunta: como acessar uma string individual? Fácil. É só usar apenas o primeiro índice. Então, para acessar uma determinada string:

nome_da_variável [índice]

Aqui está um exemplo de um programa que lê 5 strings e as exibe na tela:

Exemplo 02:

```
#include <stdio.h>
int main ()
{
        char strings [5][100];
        int count;
        for (count=0;count<5;count++){
                printf ("\n\nDigite uma string: ");
                gets (strings[count]);
        }
        printf ("\n\n\nAs strings que voce digitou foram:\n\n");

        for (count=0;count<5;count++){
                 printf ("%s\n",strings[count]);
        }
        return(0);
}
```

Matrizes multidimensionais

O uso de matrizes multidimensionais na linguagem C é simples. Sua forma geral é:

tipo_da_variável nome_da_variável [tam1][tam2] ... [tamN];

Uma matriz N-dimensional funciona basicamente como outros tipos de matrizes. Basta lembrar que o índice que varia mais rapidamente que o índice mais à direita.

Inicialização

Podemos inicializar matrizes, assim como podemos inicializar variáveis. A forma geral de uma matriz como inicialização:

tipo_da_variável nome_da_variável [tam1][tam2] ... [tamN] = {lista_de_valores};

A lista de valores é composta por valores (do mesmo tipo da variável) separados por vírgula. Os valores devem ser dados na ordem em que serão colocados na matriz. Abaixo vemos alguns exemplos de inicializações de matrizes:

```
float vect [6] = { 1.3, 4.5, 2.7, 4.1, 0.0, 100.1 };
int matrx [3][4] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 };
char str [10] = { 'J', 'o', 'a', 'o', '\0' };
char str [10] = "Joao";
char str_vect [3][10] = { "Joao", "Maria", "Jose" };
```

O primeiro demonstra inicialização de vetores. O segundo exemplo demonstra a inicialização de matrizes multidimensionais, onde **matrx** está sendo inicializada com 1, 2, 3 e 4 em sua primeira linha, 5, 6, 7 e 8 na segunda linha e 9, 10, 11 e 12 na última

linha. No terceiro exemplo vemos como inicializar uma string e, no quarto exemplo, um modo mais compacto de inicializar uma string. O quinto exemplo combina as duas técnicas para inicializar um vetor de strings. Repare que devemos incluir o ; no final da inicialização.

Inicialização sem especificação de tamanho

Podemos, em alguns casos, inicializar matrizes das quais não sabemos o tamanho *a priori*. O compilador C vai, neste caso verificar o tamanho do que você declarou e considerar como sendo o tamanho da matriz. Isto ocorre na hora da compilação e não poderá mais ser mudado durante o programa, sendo muito útil, por exemplo, quando vamos inicializar uma string e não queremos contar quantos caracteres serão necessários. Alguns exemplos:

```
char mess [] = "Linguagem C: flexibilidade e poder.";
int matrx [][2] = { 1,2,2,4,3,6,4,8,5,10 };
```

No primeiro exemplo, a string mess terá tamanho 36. Repare que o artifício para realizar a inicialização sem especificação de tamanho [ ] não especificar o tamanho! No segundo exemplo o valor não especificado será 5.

AUTO AVALIAÇÃO

Veja como você está.

O que imprime o programa a seguir? Tente entendê-lo e responder. A seguir, execute-o e comprove o resultado.

```
# include <stdio.h>
int main()
{
    int t, i, M[3][4];
    for (t=0; t<3; ++t) {
        for (i=0; i<4; ++i){
            M[t][i] = (t*4)+i+1;
        }
    }
    for (t=0; t<3; ++t)
    {
        for (i=0; i<4; ++i){
            printf ("%3d ", M[t][i]);
            printf ("\n");
        }
     }
         return(0);
}
```