# Linguagem C – Matrizes

## Objetivos

Neste artigo estudaremos os conceitos relacionados às matrizes unidimensionais (vetores) e multidimensionais.

## Definição de Matrizes em Linguagem C

As matrizes em geral são caracterizadas por se tratarem de uma única variável de um determinado tamanho que guarda varias informações do mesmo tipo. Essas informações são gravadas na memória seqüencialmente e são referenciadas através de índices. As matrizes podem ser tanto unidimensionais (vetores) como multidimensionais.

## Matrizes unidimensionais

São matrizes de uma única dimensão. Essas matrizes também são chamadas de vetores. A declaração de vetores em C deve obedecer a seguinte sintaxe:

**Tipo nome_vetor [tamanho];**

O tipo deve ser especificado conforme a tabela 2 do primeiro artigo. E o tamanho representa a quantidade de elementos que esta matriz irá conter. É importante dizer que na linguagem c as matrizes começam pelo índice 0 que guarda o primeiro elemento da matriz. Para entender melhor, considere que seja necessário declarar um vetor do tipo inteiro que contenha 10 elementos. Isto é feito da seguinte forma:

**int vetor_exemplo [9];**

Isso por que a matriz “vetor_exemplo” vai de 0 a 9, ou seja, contém 10 elementos. Também é possível inicializar o vetor no momento de sua declaração. Para isso veja a sintaxe abaixo:

**Tipo nome_vetor [tamanho]={lista_de_valores};**

Sendo que todos os elementos da lista de valores devem ser separados por virgula e serem todas do mesmo tipo de dados especificado. A seguir temos a declaração do “vetor_exemplo” com os valores atribuídos.

**int** vetor_exemplo [9]={0,1,2,3,4,5,6,7,8,9};

Exemplo de um programa completo utilizando vetores e outros conceitos estudados até o momento.

**1º Programa**

```
 #include<stdio.h>

void main ()

{

int vet1[5]={1,2,3,4,5}; /*declaração e inicialização do vetor vet1*/

int vet2[5]={6,1,2,2,5}; /*declaração e inicialização do vetor vet2*/

int vetSoma[5]; /*declaração do vetor vetSoma que vai guardar o resultado da soma dos dois vetores(vet1 e vet2).*/

int x;

printf ("Este programa soma os dois vetores abaixo:\n");

 printf ("vet1={1,2,3,4,5}\n");

printf ("vet2={6,1,2,2,5}\n");

printf ("\n");

printf ("vetor resultante da soma:\n");

for (x=0; x<5; x++)

{

 vetSoma [x]=vet1[x]+vet2[x]; /*soma os valores*/

 printf ("vetSoma [%d]:%d\n”, x, vetSoma [x]); /*exibe na tela*/

} /*fim do for*/

} /*fim do programa*/
```

O programa acima soma dois vetores (vet1 e vet2) e exibe sua resposta na tela. O objetivo deste exemplo é mostrar como declarar e inicializar um vetor. Veja a seguir a saída do programa:

Este programa soma os dois vetores abaixo:

vet1={1,2,3,4,5}

vet2={6,1,2,2,5}

vetor resultante da soma:

vetSoma [0]: 7

vetSoma [1]: 3

vetSoma [2]: 5

vetSoma [3]: 6

vetSoma [4]: 10

Os vetores são muito usados para criar uma string de caracteres, pois em C não existe nenhum tipo de dados para definir uma string. A declaração de um vetor contendo uma string sempre deve ser maior que o número de caracteres, pois o compilador acrescenta automaticamente no final da string um espaço nulo que indica o seu término. Este segundo exemplo é muito simples, mostra apenas como podem ser feitas a declaração e inicialização de um vetor de string.

**2º Programa**

```
#include<stdio.h>

void main ()

{

char c1 = 'a';

char vet1[30]="Aprendendo a mexer com string\n";

 /*Imprimindo os dados na tela*/

printf ("O tipo de dado char guarda apenas um caractere\n");

printf ("A variavel c1 do tipo char contem o caractere: %c\n", c1);

printf ("\n");

printf ("Para trabalhar com uma string deve ser declarado um vetor do tipo char");

printf ("\nO vetor do tipo char contem a string: %s", vet1);

} /*fim do programa*/
```

**Saída do programa:**

O tipo de dado char guarda apenas um caractere

A variavel c1 do tipo char contem o caractere: a

Para trabalhar com uma string deve ser declarado um vetor do tipo char

O vetor do tipo char contem a string: Aprendendo a mexer com string

**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. Veja o programa abaixo:

**3º Programa**

```
#include<stdio.h>
void main ()
{
int mat [2][2];
float det;
int x, y;
printf ("Este programa calcula a determinante de uma matriz quadrada de ordem 2");
printf ("\n\n Entre com os valores a 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*/
} /*fim do for*/
det=mat.[0] [0]*mat.[1][1]-mat.[1][0]*mat[0][1]; /*formula para calcular a determinante */
printf ("Determinante da matriz = %f\n", det);
} /*fim do programa*/
```

Esse exemplo calcula a determinante de uma matriz de ordem 2 e através dele podemos ver como deve ser feita a declaração de uma matriz bidimensional.

**int** mat [2] [2];

Que a forma geral é:

tipo nome_matriz [numero_linhas] [numero_colunas];

Observe na formula que calcula a determinante que os valores são acessados através de seus índices (linhas e colunas). A seguir são apresentadas as saídas do programa, lembrando que os valores das matrizes são definidos pelo usuário através da função scanf que será explicada mais adiante.

Este programa calcula a determinante de uma matriz quadrada de ordem 2

Entre com os valores a da matriz:

Mat [1] [1] =4

Mat [1] [2]=-3

Mat [2] [1] =6

Mat [2] [2] =-1

Determinante da matriz = 14.0000

## Matrizes multidimensionais

A linguagem c permite também utilizar matriz de três ou mais dimensões, porém não é freqüentemente usada, pois requer uma quantidade grande de memória e os acessos aos seus elementos são mais lentos. Por esses motivos este tipo de matrizes não serão abordados com detalhes

## Declaração de matrizes multidimensionais:

Tipo nome [tamanho1] [tamanho2] [tamanho3]... [tamanho n]

## 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:

**4º Programa**

```
#include<stdio.h>
void main ()
{
char vet1[37]="Estou aprendendo a programar em C!!!";
char vet2[]="Estou aprendendo a programar em C!!!"; /*vetor não-dimensionado*/
printf ("O vetor abaixo foi declarado com o seu tamanho especificado\n");
printf ("%s\n", vet1);
printf ("\n");
printf ("E este outro foi declarado com o seu tamanho nao especificado\n");
printf ("%s\n", vet2);
}
```

Observe na saída abaixo 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).

O vetor abaixo foi declarado com o seu tamanho especificado

Estou aprendendo a programar em C!!!

E este outro foi declarado com o seu tamanho nao especificado

Estou aprendendo a programar em C!!!

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.

**5º Programa**

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

Imprimindo a matriz mat1 cujo tamanho foi especificado:

mat1[0] [0] =4

mat1[0] [1] =5

mat1[1] [0] =-2

mat1[1] [1] =1

Imprimindo a matriz mat2 cujo tamanho nao foi especificado:

mat2[0] [0] =4

mat2[0] [1] =5

mat2[1] [0] =-2

mat2[1] [1] =1

## Conclusão

Conclui-se que as matrizes podem ser desde unidimensionais até multidimensionais e que seus tamanhos não necessitam serem obrigatoriamente especificados