

# Arrays multidimensionais

Um *array* pode ter qualquer número de dimensões.

A declaração de vectores unidimensionais não é mais do que um caso particular da declaração de *arrays* com qualquer número de dimensões.

## Declaração

```
tipo_elementos nome[nº_elementos1] ... [nº_elementosk] ... [nº_elementosn];
```

- **tipo\_elementos:** tipo de dados de cada um dos elementos do *array*
- **nome:** indica o nome pelo qual o *array* vai ser conhecido
- **nº\_elementos<sub>k</sub>:** valor constante que indica quantos elementos tem o *array* na dimensão “*k*”

Por exemplo, a declaração seguinte, cria um **array bidimensional** ou **matriz** (em linguagem matemática):

```
int m[5][9];
```

Esta declaração diz que a variável **m** é um *array* bidimensional de inteiros, com **5 linhas** e **9 colunas**.

Os índices das linhas e das colunas começam em zero, como mostra a figura abaixo.

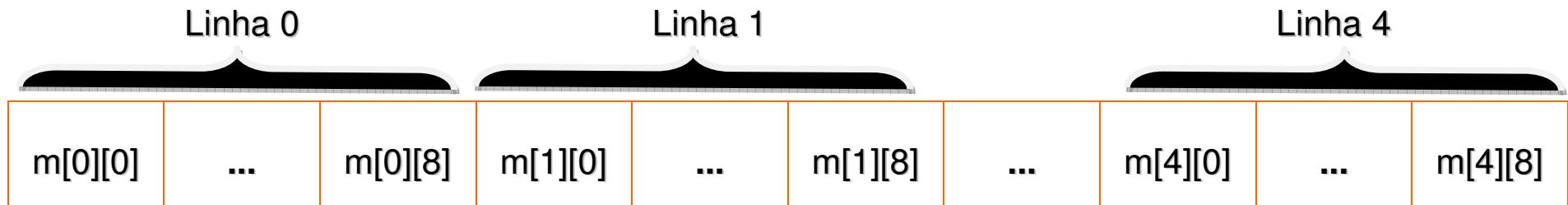
<b>m</b>	<b>0</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>
<b>0</b>									
<b>1</b>									
<b>2</b>									
<b>3</b>									
<b>4</b>									

O acesso ao elemento da **linha i** e da **coluna j**, faz-se através de **m[ i ][ j ]**.

A expressão **m[ i ]** designa a **linha i** da matriz **m**; a expressão **m[ i ][ j ]** selecciona o elemento da **posição j** dessa linha.

Apesar de ser habitual visualizarem-se os arrays bidimensionais como tabelas, não é dessa forma que os mesmos são armazenados na memória. De facto estes são armazenados por linhas, do seguinte modo: primeiro a linha 0, depois a linha 1, a seguir a linha 2, etc...

Por exemplo, o *array* **m** declarado atrás, seria armazenado em memória da seguinte forma:



**Nota:** Em C, um **array** declarado com **duas dimensões**, não é na realidade uma **matriz**, mas sim um **vector de vectores**. O mesmo se aplica a *arrays* com dimensão superior a 2.

Da mesma maneira que os ciclos **for** “andam de mão dada” com os *arrays* unidimensionais, os ciclos **for** encadeados são ideais para o processamento de *arrays* bidimensionais.

Considere-se, por exemplo, o problema de inicialização de uma matriz para ser usada como *matriz identidade*. (Em matemática, uma *matriz identidade* é constituída por 1's na diagonal principal e por 0's nos restantes elementos).

Vamos ter que aceder a todos os elementos do array de uma forma sistemática. Para tal vamos utilizar dois ciclos **for** encadeados: um para percorrer os índices das linhas e outro para percorrer os índices das colunas.

```
#define N 10

(...)

float ident[N][N];
int linha, coluna;

for(linha=0;linha<N;linha++)
    for(coluna=0;coluna<N;coluna++)
        if(linha==coluna)
            ident[linha][coluna]=1.0;
        else
            ident[linha][coluna]=0.0;

(...)
```

## Inicialização explícita de arrays multidimensionais

```
int m[5][9] = {{1,1,1,1,1,0,1,1,1},
               {0,1,0,1,0,1,0,1,0},
               {0,1,0,1,1,0,0,1,0},
               {1,1,0,1,0,0,0,1,0},
               {1,1,0,1,0,0,1,1,1}};
```

Cada conjunto de valores entre as chavetas mais internas, inicializa os elementos de uma linha da matriz. (A inicialização de *arrays* de dimensão superior faz-se de modo semelhante.)

A linguagem C faculta várias formas de abreviar a inicialização de arrays multidimensionais:

- Se o conjunto de valores de inicialização não for suficiente para preencher todo array, os elementos restantes serão inicializados a zero. O exemplo que se segue mostra uma situação em que apenas são preenchidas as 3 primeiras linhas do array, ficando as restantes com valores zero:

```
int m[5][9] = {{1,1,1,1,1,0,1,1,1},
               {0,1,0,1,0,1,0,1,0},
               {0,1,0,1,1,0,0,1,0}};
```

- Se um dos conjuntos de valores de inicialização (entre as chavetas mais internas) não for suficiente para preencher uma linha, os elementos restantes serão inicializados a zero:

```
int m[5][9] = {{1,1,1,1,1,0,1,1,1},  
               {0,1,0,1,0,1,0},  
               {0,1,0,1,1,0,0},  
               {1,1,0,1,0,0,0},  
               {1,1,0,1,0,0,1,1,1}};
```

- Podem omitir-se as chavetas mais internas:

```
int m[5][9] = {1,1,1,1,1,0,1,1,1,  
               0,1,0,1,0,1,0,1,0,  
               0,1,0,1,1,0,0,1,0,  
               1,1,0,1,0,0,0,1,0,  
               1,1,0,1,0,0,1,1,1};
```

A partir do momento em que o compilador encontra valores suficientes para preencher uma linha, começa a preencher a seguinte.

## Exemplo 1

---

```
/*          Quantos elementos nulos existem          */
/*          numa matriz identidade de dimensão 6x6?  */

#include<stdio.h>

void main (void)
{
    int tab[6][6]={ {1},
                    {0, 1},
                    {0, 0, 1},
                    {0, 0, 0, 1},
                    {0, 0, 0, 0, 1},
                    {0, 0, 0, 0, 0, 1}};

    int linha, coluna, num;

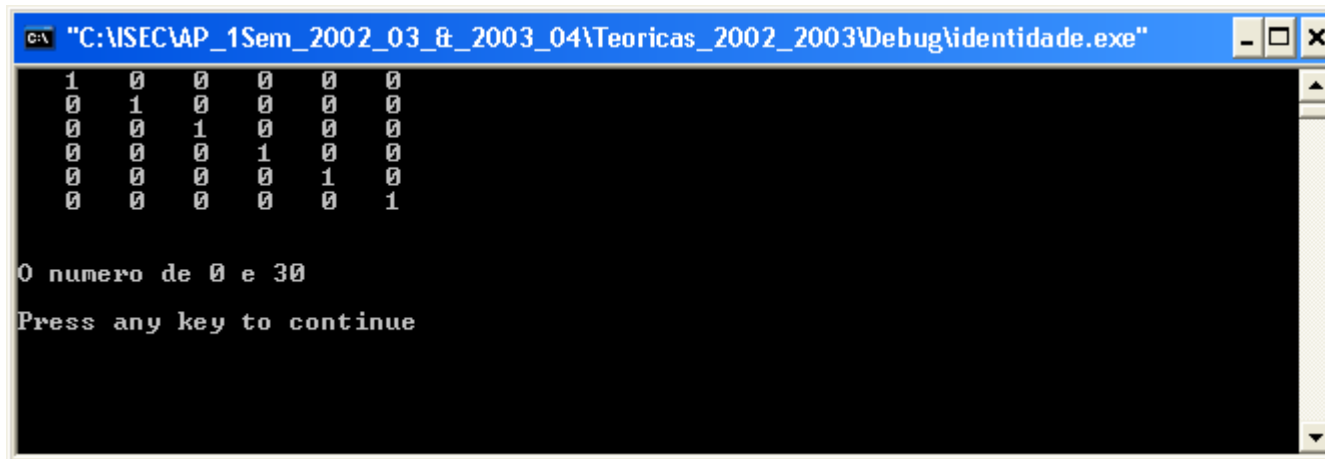
    /* Mostra as 6 linhas e 6 colunas correspondentes à tabela */
    for (linha=0; linha<6; linha++)
    {
        for (coluna=0; coluna<6; coluna++)
            printf("%4d",tab[linha][coluna]);
        printf("\n");
    }
    printf("\n");
}
```

```

/* Conta os zeros presentes nas 6 linhas e 6 colunas da
tabela */
    for (linha=0, num=0; linha<6; linha++)
        for (coluna=0; coluna<6; coluna++)
            if (tab[linha][coluna] == 0)
                num++;
    printf("\nO numero de 0 é %d\n\n", num);
}

```

Exemplo de execução:



```

C:\MSECVAP_1Sem_2002_03_&_2003_04\Teoricas_2002_2003\Debug\identidade.exe
1 0 0 0 0 0
0 1 0 0 0 0
0 0 1 0 0 0
0 0 0 1 0 0
0 0 0 0 1 0
0 0 0 0 0 1

O numero de 0 e 30
Press any key to continue

```



## Passagem de arrays multidimensionais para funções

Quando um parâmetro é um array multidimensional, **só o tamanho da 1ª dimensão é que pode ser omitido.**

Por exemplo, considere-se a seguinte função que calcula a soma dos elementos de um array bidimensional:

```
#define TAM 10

int SomaArray(int a[][TAM],int n)
{
    int i,j,soma=0;

    for(i=0;i<n;i++)
        for(j=0;j<TAM;j++)
            soma+=a[i][j];

    return soma;
}
```

Como se conclui, **não é possível passar *arrays* multidimensionais para funções, com um nº arbitrário de colunas.** (Esta dificuldade poderá ser ultrapassada, usando *arrays* de ponteiros.)

## Exemplo 2

---

```
/*          Quantos elementos nulos existem          */
/*          numa matriz identidade de dimensão 6x6?   */

#include<stdio.h>

#define DIM 6

void MostraMatriz (int t[][DIM], int dim_l, int dim_c)
{
/* Mostra as dim_l linhas e dim_c colunas correspondentes à
tabela */
    int linha, coluna;

    for (linha=0; linha<dim_l; linha++)
    {
        for (coluna=0; coluna<dim_c; coluna++)
            printf("%4d",t[linha][coluna]);
        printf("\n");
    }
    printf("\n");
}
```

```

void main (void)
{
    int tab[DIM][DIM]={ {1},
                        {0, 1},
                        {0, 0, 1},
                        {0, 0, 0, 1},
                        {0, 0, 0, 0, 1},
                        {0, 0, 0, 0, 0, 1}};

    int linha, coluna, num;

    /* Mostra as 6 linhas e 6 colunas correspondentes à tabela */
    MostraMatriz(tab, DIM, DIM);

    /* Conta os zeros presentes nas 6 linhas e 6 colunas
    da tabela */
    for (linha=0; linha<DIM; linha++)
        for (coluna=0; coluna<DIM; coluna++)
            if (tab[linha][coluna] == 0)
                num++;
    printf("\nO numero de 0 é %d\n\n", num);
}

```