



Vetores e Matrizes

Neste tópico abordaremos a manipulação de vetores e matrizes em C.

Prof. Ciro Cirne Trindade
Prof. Thiago Ferauche

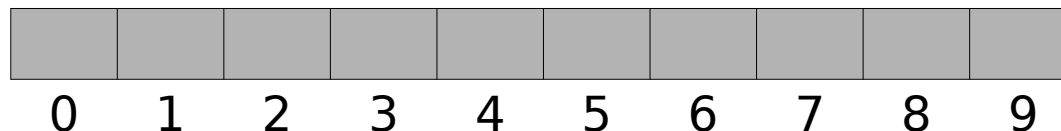


Introdução

- Um vetor é uma coleção de variáveis do mesmo tipo que são referenciadas por um nome comum
- Um elemento específico em um vetor é acessado através de um índice
- Em C, todos os vetores consistem em posições contíguas de memória
 - O endereço + baixo corresponde ao 1º elemento
 - O endereço + alto, ao último elemento

Declaração de um vetor

- A forma geral de um vetor é:
`tipo nome-da-variável[tamanho];`
 - `tipo`: declara o tipo base do vetor
 - `tamanho`: define quantos elementos o vetor conterá
- Todos os vetores têm 0 como o índice do 1º elemento
- Exemplo: `int v[10];`
 - declara um vetor de inteiros que tem 10 elementos, `v[0]` a `v[9]`





Acessando os elementos de um vetor

- Não é possível fazer referência a todos os elementos de um vetor de uma vez
 - Com exceção de vetores de caracteres (strings)
- Os elementos devem acessados individualmente através de um índice
- Forma geral:
 - `nome-vetor[índice]`



Percorrendo um vetor

- Para armazenar e ler dados de um vetor, geralmente usamos o comando **for**
- Em C, não é feita a verificação do tamanho do vetor
 - É responsabilidade do programador incluir a verificação dos limites do vetor quando isso for necessário
- Um vetor pode armazenar qualquer tipo de dado
- O índice de um vetor é sempre um inteiro



Exemplo: uso do `for` para acessar os elementos do vetor

```
#include <stdio.h>
int main() {
    int numeros[10], i;
    printf("Informe 10 números inteiros positivos: ");
    for (i = 0; i < 10; i++) {
        scanf("%d", &numeros[i]);
    }
    printf("\nNúmeros pares informados : ");
    for (i = 0; i < 10; i++) {
        if (numeros[i] % 2 == 0)
            printf("%d ", numeros[i]);
    }
    printf("\n");
    return 0;
}
```



Definindo o tamanho do vetor através de uma variável

- A partir da versão C99 é possível definir do vetor através de uma variável do tipo int.
- Por exemplo:

```
int n;  
scanf("%d", &n);  
int numeros[n];
```



Inicializando vetores (1/2)

- Você pode inicializar vetores na mesma instrução de sua declaração

```
#define LIM 7  
LIM é uma constante
```

```
int notas[LIM] = { 100, 50, 20, 10, 5, 2, 1 };
```

- A lista de valores é colocada entre chaves e os valores são separados por vírgulas
- Os valores são atribuídos na sequência
- Se nenhum número for fornecido para dimensionar o vetor, o compilador contará o número de itens da lista de inicialização e o fixará como dimensão do vetor

```
int notas[] = { 100, 50, 20, 10, 5, 2, 1 };8
```


Inicializando vetores (2/2)

- É possível inicializar parcialmente os elementos do vetor, os demais são inicializados com 0

```
int notas[LIM] = { 100, 50, 20 };
```

- Neste caso, os elementos das posições 3, 4, 5 e 6 do vetor seriam inicializadas com 0
- Também é possível inicializar um elemento particular:

```
int notas[LIM] = { [3] = 10 };
```

notas[3] = 10,
demais iguais a 0

- Para declarar um vetor somente para leitura:

```
const int notas[LIM]={100,50,20,10,5,2,1};
```



Matrizes

- A linguagem C permite matrizes de qualquer tipo, incluindo matrizes com mais de duas dimensões
- Com 2 pares de colchetes obtemos uma matriz de 2 dimensões e p/ cada par de colchetes adicionais obtemos uma matriz com uma dimensão a mais:

```
tipo nome-da-variável[tamanho 1][tamanho 2]...  
                        [tamanho n];
```



Matrizes bidimensionais (1/2)

- Para declarar uma matriz bidimensional devemos usar 2 pares de colchetes
 - No **1º** par de colchetes definimos o número de **linhas** da matriz
 - No **2º** par de colchetes definimos o número de **colunas** da matriz
- Exemplo:
 - `int matriz[4][6];`

Declara uma matriz de inteiros com 4 linha e 6 colunas



Matrizes bidimensionais (2/2)

- Podemos dizer que `matriz` é um vetor de 4 elementos, cada elemento, por sua vez, é um vetor de 6 elementos
- Usando esta lógica, `matriz[0]` que é o elemento de `matriz` é um vetor de 6 valores `int`
- Se `matriz[0]` é um vetor, seu primeiro elemento é `matriz[0][0]`, seu segundo elemento é `matriz[0][1]`, e assim por diante



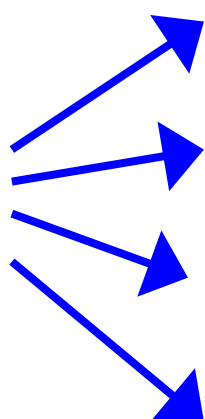
Acessando os elementos da matriz (1/3)

- Para acessar um elemento específico da matriz devemos indicar a linha e coluna desse elemento
 - Semelhante a referência a uma célula de uma planilha eletrônica
- Usamos índices de linha e coluna para referenciar um elemento específico da matriz

Acessando os elementos da matriz (2/3)

- Exemplo de uma matriz 4x6

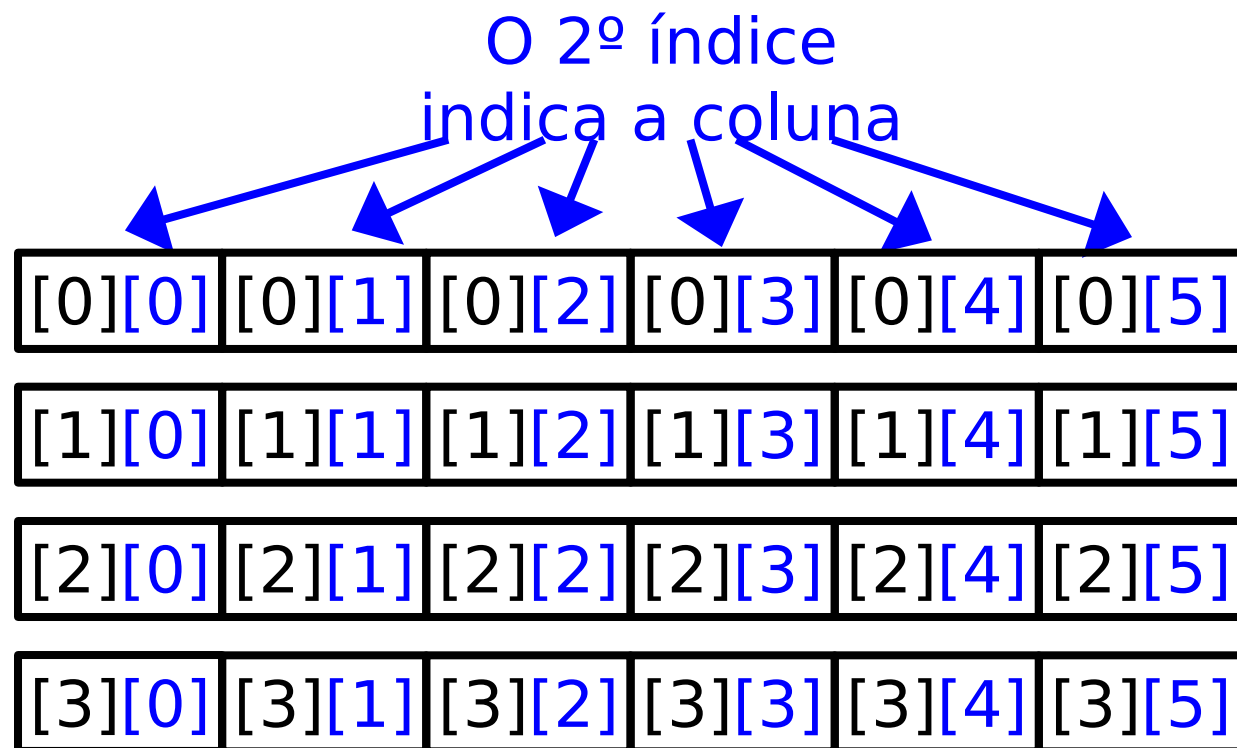
O 1º
índice
indica a
linha



[0][0]	[0][1]	[0][2]	[0][3]	[0][4]	[0][5]
[1][0]	[1][1]	[1][2]	[1][3]	[1][4]	[1][5]
[2][0]	[2][1]	[2][2]	[2][3]	[2][4]	[2][5]
[3][0]	[3][1]	[3][2]	[3][3]	[3][4]	[3][5]

Acessando os elementos da matriz (3/3)

- Exemplo de uma matriz 4x6





Percorrendo uma matriz

- Para percorrer uma matriz normalmente utiliza-se dois laços **for** encaixados
- As variáveis de controle dos laços são utilizadas como índices da matriz



Exemplo da leitura e impressão de uma matriz

```
#include <stdio.h>
int main() {
    int matriz[4][6], i, j;
    printf("Informe os elementos da matriz 4x6:\n");
    for (i = 0; i < 4; i++) {
        for (j = 0; j < 6; j++) {
            scanf("%d", &matriz[i][j]);
        }
    }
    printf("Conteudo da matriz:\n");
    for (i = 0; i < 4; i++) {
        for (j = 0; j < 6; j++) {
            printf("%d\t", matriz[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Inicializando matrizes

- As matrizes são inicializadas da mesma maneira que os vetores

```
const char inimigo[LIN][COL] = {  
    { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },  
    { 0, 1, 1, 1, 1, 0, 0, 1, 0, 1 },  
    { 0, 0, 0, 0, 0, 0, 0, 1, 0, 1 },  
    { 1, 0, 0, 0, 0, 0, 0, 1, 0, 0 },  
    { 1, 0, 1, 1, 1, 0, 0, 0, 0, 0 } };
```

```
#define LIN 5  
#define COL 10
```

- Uma matriz pode ser vista como um vetor onde seus elementos são vetores

Inicializando matrizes

- A partir da versão C99 permite indicar explicitamente que elemento se deseja inicializar, é possível inicializar uma linha ou um elemento específico da matriz:

```
const char inimigo[][COL] =  
    {  
        [1] = { 0, 1, 1, 1, 1, 0, 0, 1, 0, 1 },  
        [2][7] = 1, [2][9] = 1,  
        { 1, [7] = 1 },  
        { 1, [2] = 1, 1, 1 }  
    };
```

Quando a matriz é inicializada, o número de linhas pode ser omitido



Referências Bibliográficas

- DEITEL, Paul; DEITEL, Harvey. C Como Programar. 6. ed., Pearson, 2011.
- MIZHARI, Victorine Viviane. Treinamento em Linguagem C. 2. ed., Pearson, 2008.
- PRATA, Stephen. C Primer Plus. 6. ed. Addison Wesley, 2014.
- SCHILDT, Herbert. C Completo e Total. 3. ed., Makron Books, 1996.