

Estruturas de Dados Homogêneas: Vetores e Matrizes

Prof. Gabriel Barbosa da Fonseca

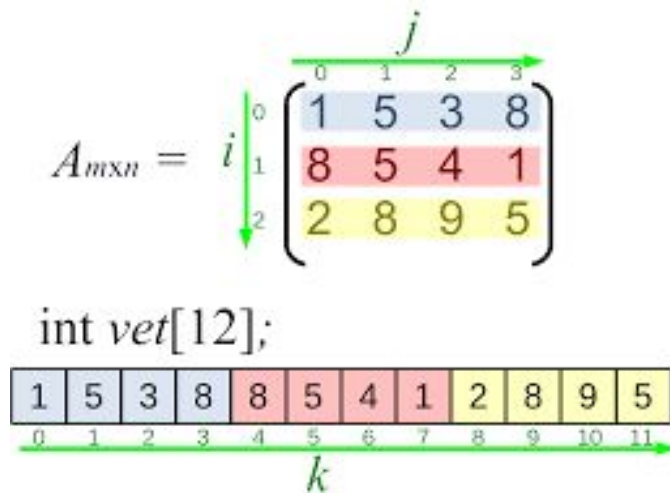
Email: gbfonseca@sga.pucminas.br

A dark blue diagonal gradient bar that starts from the bottom left and extends towards the top right, covering the lower half of the slide.

Introdução

Vetores, também chamados **arrays** (do inglês) ou **arranjo**, são uma maneira de **armazenar vários dados num mesmo nome de variável** através do uso de índices numéricos.

Em C, vetores devem sempre conter dados do mesmo tipo de variável.



Declaração

- Declaramos vetores de maneira muito semelhante à declaração de variáveis normais
- A única diferença é que depois do nome da variável deve ser informada a quantidade de elementos do vetor.
- Para declarar um vetor chamado “vetor1”, com cinco elementos inteiros, escrevemos:

```
int vetor1[5];
```

Declaração

- Podemos também declarar um vetor e iniciá-lo atribuindo valores

```
int vetor1[5];
```

```
int vetor1[5] = {1,2,3,4,5};
```

Acesso

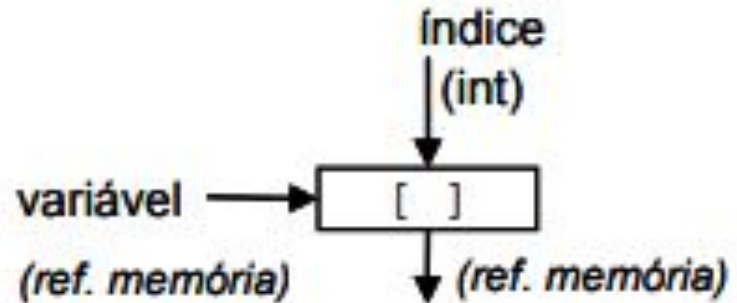
- Para fazer referência a um valor de um elemento contido em um vetor, usamos a notação **vetor[índice]**, que serve tanto para obter quanto para definir o valor de um elemento específico, dada sua posição.
- Note que os elementos são numerados a começar do **ZERO**, e, portanto, se o número de elementos é **N**, o índice ou posição do último elemento será **N - 1**.

```
int vetor1[5];  
vetor1[0] = 1;  
vetor1[1] = 12;  
vetor1[2] = 13;  
vetor1[3] = 15;  
vetor1[4] = 2;
```

Acesso

- Para fazer referência a um valor de um elemento contido em um vetor, usamos a notação **vetor[índice]**, que serve tanto para obter quanto para definir o valor de um elemento específico, dada sua posição.
- Note que os elementos são numerados a começar do **ZERO**, e, portanto, se o número de elementos é **N**, o índice ou posição do último elemento será **N - 1**.


```
int vetor1[5];  
vetor1[0] = 1;  
vetor1[1] = 12;  
vetor1[2] = 13;  
vetor1[3] = 15;  
vetor1[4] = 2;
```



Acesso

- **Vetor1** (o nome da variável) funciona como um ponteiro que aponta para o primeiro elemento do vetor.

```
short int vetor1[3];  
vetor1[0] = 15;  
vetor1[1] = 14;  
vetor1[2] = 13;
```



Memória RAM		
Endereço	Variável	Valor
7ffe5367e000	vetor1[0]	15
7ffe5367e001		
7ffe5367e002	vetor1[1]	14
7ffe5367e003		
7ffe5367e004	vetor1[2]	13
7ffe5367e005		

Acesso – Índices inválidos

- Os elementos são numerados sempre de 0 até tamanho-1.
- Caso o programa tente acessar erroneamente um elemento de índice negativo ou de índice além do tamanho do vetor, **as consequências poderão ser imprevisíveis.**

Atribuição de valores

- **Não é possível** atribuir valores a todos os elementos em **uma só linha**.
- Cada elemento precisa ser acessado **individualmente**.
- Tampouco é possível usar um único **scanf** para ler todo o conteúdo do vetor

```
int vetor[10];  
int indice;  
// inicializar todos os elementos com  
// o valor 0  
for (indice = 0; indice < 10; indice++)  
{  
    vetor[indice] = 0;  
}
```

Copiar Vetores

- Não é possível copiar o conteúdo de um vetor para um outro, mesmo que os dois sejam de mesmo tamanho e os elementos sejam de mesmo tipo.
- Devemos copiar índice a índice.

```
int vetorA[10], vetorB[10];  
int indice;  
// copiar o conteúdo do vetor B para o  
vetor A  
for (indice = 0; indice < 10; indice++) {  
    vetorA[indice] = vetorB[indice];  
}
```

Vetores multidimensionais – MATRIZES

- A noção de **matriz** é a generalização imediata da noção de vetor. Uma matriz é **uma tabela de vários valores do mesmo tipo**, armazenados sequencialmente e fazendo uso de um mesmo nome de variável para acessar esses valores
- Cada elemento da tabela pode ser acessado individualmente através de **dois índices** com valores inteiros. Estes índices poderiam ser interpretados como a **linha e a coluna da matriz**

Vetores multidimensionais – MATRIZES

Exemplo:

`int[4][10]`

`int[10]`

a_{00}	a_{01}	a_{02}	a_{03}	a_{04}	a_{05}	a_{06}	a_{07}	a_{08}	a_{09}
----------	----------	----------	----------	----------	----------	----------	----------	----------	----------

`int[10]`

a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	a_{17}	a_{18}	a_{19}
----------	----------	----------	----------	----------	----------	----------	----------	----------	----------

`int[10]`

a_{20}	a_{21}	a_{22}	a_{23}	a_{24}	a_{25}	a_{26}	a_{27}	a_{28}	a_{29}
----------	----------	----------	----------	----------	----------	----------	----------	----------	----------

`int[10]`

a_{30}	a_{31}	a_{32}	a_{33}	a_{34}	a_{35}	a_{36}	a_{37}	a_{38}	a_{39}
----------	----------	----------	----------	----------	----------	----------	----------	----------	----------

Vetores multidimensionais – MATRIZES

A declaração de matrizes obedece a mesma sintaxe que a declaração de vetores, exceto pela adição de uma nova dimensão escrita entre colchetes [].

tipo **variável**[**linhas**][**colunas**];

Ex:

int **matriz1**[**5**][**10**];

Vetores multidimensionais – MATRIZES

Para percorrer os elementos de uma matriz, são necessárias **duas estruturas de repetição for**, uma dentro da outra. O **for externo** percorre as **linhas** da matriz, o **for interno** percorre as **colunas** de uma determinada linha que está fixada pelo for externo. Por exemplo, para imprimir todos os elementos de uma matriz 4x10, linha por linha:

```
int linha, coluna;
int matriz[4][10];
...
for (linha = 0; linha < 4; linha++) {
    for (coluna = 0; coluna < 10; coluna++) {
        printf("%d ", matriz[linha][coluna]);
    }
    printf("\n");
}
```

The image features a red background with a series of concentric circles that create a tunnel-like effect, drawing the eye towards the center. In the middle of this pattern, the words "The End" are written in a white, elegant script font. The text is slightly offset to the left and has a subtle drop shadow, making it stand out against the dark red circular backdrop.

The End