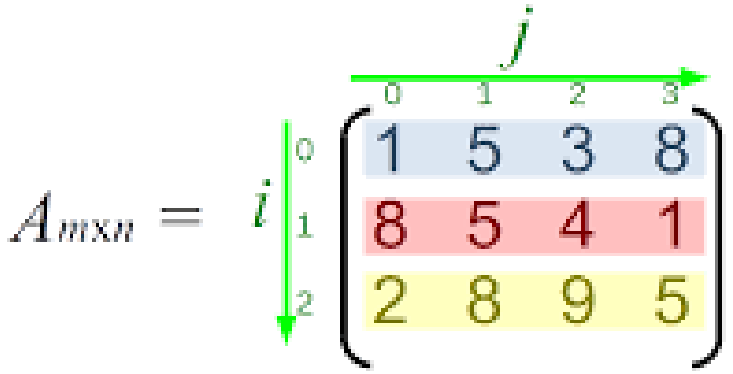
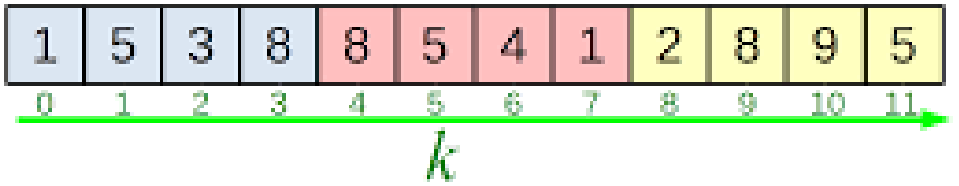
UNIDADE III: VETORES E MATRIZES



int *vet*[12];



Vetores

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.

Vetores

- 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 vetor, com cinco elementos inteiros, escrevemos:

```
int vetor[5];
```

Vetores

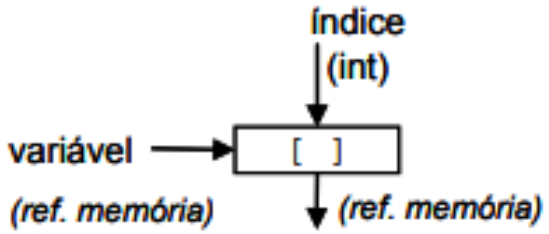
- Declaramos vetores de maneira muito semelhante à declaração de variáveis normais
- A única diferença é que depois do nome da variável deve ser informada a quantidade de elementos do vetor.
- Para declarar um vetor chamado vetor, com cinco elementos inteiros, escrevemos:

```
int vetor[5];

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

Vetores

- 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.



Regras para 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.

Regras para acesso

Atribuir o valor de todos os elementos de uma só vez

- 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;
}
```

Regras para acesso

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.

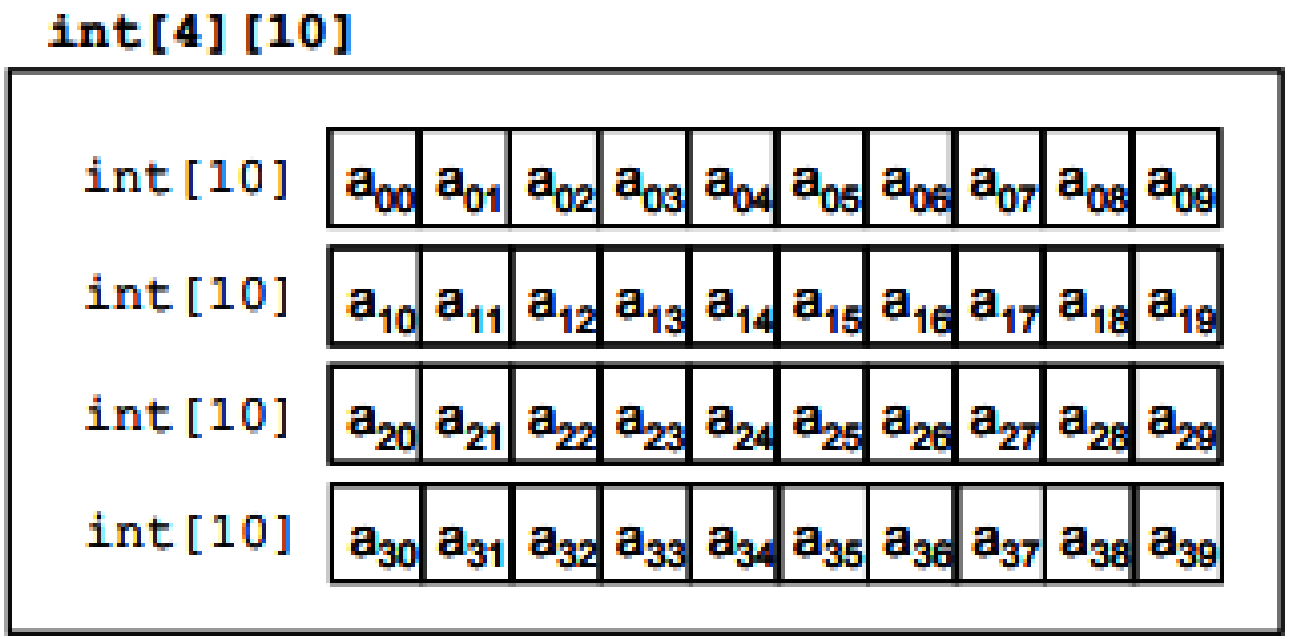
```
int vetorA[10], vetorB[10];
int indice;
// copiar o conteúdo do vetor B para o vetor A
for (indice = 0; indice < 10; indice++) {
    vetorA[indice] = vetorB[indice];
}
```

Matrizes e vetores multidimensionais

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

Matrizes e vetores multidimensionais



Matrizes e vetores multidimensionais

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];
```

Matrizes e vetores multidimensionais

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 End