

# Introducao a Programacao: Vetores e *strings*

periclesmiranda@gmail.com

# Plano de aula

- 1 Vetores unidimensionais
- 2 Vetores multidimensionais
- 3 Cadeia de caracteres

- 1 Vetores unidimensionais
- 2 Vetores multidimensionais
- 3 Cadeia de caracteres

## Vetor unidimensional: Definição

- Tipo de dado usado para definir um vetor com um **número fixo** de elementos do **mesmo tipo**;
- Cada elemento pode ser acessado diretamente;
- Cada elemento é indexado por um inteiro.

## Sintaxe: Declaração

```
tipo v[tamanho];
```

- **tipo**: tipo dos componentes. Pode ser qualquer tipo (`int`, `char`, ...);
- **tamanho**: Número de elementos.

## Sintaxe: uso

```
v[i]
```

- Retorna o elemento de índice `i` do vetor `v`;
- Devemos ter  $0 \leq i < \text{tamanho}$ .

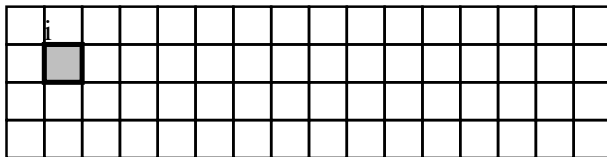
**Exemplo:**

```
int i;  
int v[10];  
i = 3;  
v[i] = 5;
```


Memória

## Exemplo:

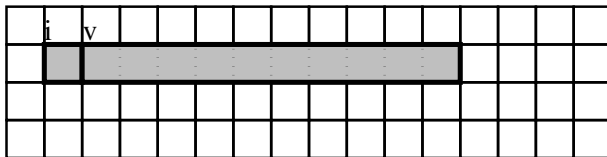
```
int i;  
int v[10];  
i = 3;  
v[i] = 5;
```



Memória

## Exemplo:

```
int i;  
int v[10];  
i = 3;  
v[i] = 5;
```

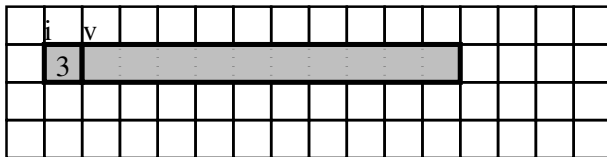


Memória



## Exemplo:

```
int i;  
int v[10];  
i = 3;  
v[i] = 5;
```



Memória

## Exemplo:

```
int i;  
int v[10];  
i = 3;  
v[i] = 5;
```

	i	v															
	3	-	-	5	-	-	-	-	-	-	-						
		v[0]	v[1]	v[2]	v[3]	v[4]	v[5]	v[6]	v[7]	v[8]	v[9]						

Memória

## Observações

- As operações num vetor são feitas elemento por elemento;
- Tamanho do vetor definido no momento da programação. Pensar em definir um tamanho suficiente!

## Exercícios: Inicialização de vetores

- Escrever um programa que define um vetor de 20 elementos reais e que inicializa esse vetor com 0 em todos os elementos;
- Escrever um programa que define um vetor de 20 elementos inteiros, que inicializa cada elemento com  $i$ ,  $i$  sendo o índice do elemento no vetor. Em seguida, imprimir na tela o vetor;
- Escrever um programa que inicializa um vetor de 100 elementos inteiros da forma seguinte: O  $i^{\text{ésimo}}$  elemento recebe  $i$  se  $i$  for ímpar.

## Índice

Sempre garantir que o índice  $i < \text{tamanho}$ . Se o índice for desconhecido, testar sua validade:

```
if ((0 <= i) && (i < tamanho)) then  
    v[i] = ...
```

### Exercícios: Máximo de $n$ valores

Escrever um programa que:

- Inicialize um vetor de tamanho 10 com inteiros positivos aleatórios menores ou iguais a 1000;
- Imprima na tela o vetor e o maior elemento.

### Observação

Gerar um número aleatório de 0 até 1000:

```
rand() * 1000.0 / RAND_MAX;
```

### Exercícios: Média de $n$ valores

Escrever um programa que peça ao usuário:

- um valor inteiro  $1 \leq n \leq 100$ ;
- $n$  valores reais;

e que retorne:

- Um erro se  $n$  for inválido;
- A média dos valores;
- A quantidade de valores acima ou iguais à média.

- 1 Vetores unidimensionais
- 2 Vetores multidimensionais
- 3 Cadeia de caracteres



## Vetor multidimensional: Definição

- É um vetor que contém vetores:

```
char v1[10][25];
```

define um vetor de 10 vetores de 25 caracteres;

- Podemos definir um vetor de vetores de vetores:

```
int v2[10][25][3];
```

E assim por diante.

## Sintaxe: Acesso

Acessar um elemento do vetor multidimensional:

```
c = v1[5][9];  
i = v2[1][9][3];
```

- `c` recebe o nono elemento do quinto vetor;
- `i` recebe o terceiro elemento do nono vetor do primeiro vetor.

### Exercícios: Inicialização

Escrever um programa que define e inicializa com 0 todos os elementos de um vetor de 50 vetores de tamanho 30 contendo números reais.

### Dica

Usar estruturas `for` aninhadas.

## Observação

- Um vetor de vetores pode ser visto como uma matriz!

**float** `mat[10][25];`

é equivalente a uma matriz de 10 linhas e 25 colunas de reais.

- Acessar o elemento da linha  $i$  e da coluna  $j$  da matriz:  
`mat[i][j].`

### Exercícios: Impressão de matriz identidade

Escrever um programa que define e inicializa uma matriz de reais de tamanho 6x6, de forma que seja a matriz identidade. Em seguida, imprimir a matriz na tela.

### Lembrando

$$I_6 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

## Exercícios: Soma de matrizes

Escrever um programa que define e inicializa randomicamente duas matrizes de tamanho 6x6 de inteiros positivos estritamente menores que 1000. Fazer a computação da soma numa terceira matriz.

Imprimir na tela as três matrizes, de forma que, em cada matriz, os valores fiquem bem alinhados.

## Observação

Gerar um número aleatório de 0 até 1000:

```
rand() * 1000.0 / RAND_MAX;
```

## Exercícios: Produto de matrizes

Escrever um programa que peça ao usuário um número  $2 \leq n \leq 6$ , e que:

- Gere aleatoriamente 2 matrizes  $A$  e  $B$  de tamanho  $n$  por  $n$  contendo valores inteiros de -5 até 5;
- Calcule o produto  $P = A \times B$ ;
- Imprima  $A$ ,  $B$  e  $P$ .

## Lembrando

Com  $a_{i,j}$  (respectivamente  $b_{i,j}$ ) o elemento da linha  $i$  e da coluna  $j$  de  $A$  (resp.  $B$ ), o elemento da linha  $i$  e da coluna  $j$  de  $P$  vale:

$$p_{i,j} = \sum_{k=1}^n a_{i,k} b_{k,j}$$

- 1 Vetores unidimensionais
- 2 Vetores multidimensionais
- 3 Cadeia de caracteres



## Tipo `string`

- Uma cadeia de caracteres (`string`) é um vetor de `char`!

```
char s[50];
```

Declara uma cadeia de caracteres de 50 caracteres no máximo;

- Acesse o carácter de índice `i` da cadeia de caracteres com  
`s[i]`

### Sintaxe: Declaração

```
char s[15] = "Hello world!";
```

Declara uma variável `s` como cadeia de 15 caracteres no máximo, e atribui a cadeia de caracteres `Hello world!` a variável `s`;

### Sintaxe: Entrada no teclado

```
gets(s);
```

Atribui à variável `s` a cadeia de caracteres digitada;

### Sintaxe: Impressão na tela

```
printf("%s", s);
```

Imprime a cadeia de caracteres `s` na tela.

### Sintaxe: Tamanho

```
strlen(s);
```

Retorna o tamanho da cadeia de caracteres `s`;

### Sintaxe: Comparação de cadeias de caracteres

```
strcmp(s1, s2)
```

Retorna um inteiro `n` tal que:

- $n < 0$  se `s1` vem antes de `s2`;
- $n = 0$  se `s1` e `s2` são iguais;
- $n > 0$  se `s1` vem depois de `s2`;

### Sintaxe: Cópia

```
strcpy(s1, s2);
```

Copia em `s1` a cadeia de caracteres `s2`.

## Cuidado!

Nem a inicialização, nem os comandos `gets` e `strcpy`, verificam se a variável tem um tamanho suficiente para receber a cadeia de caracteres desejada.

## Observação

Apenas na declaração podemos usar o operador `=`

```
char s[20] = "Hello world!";
```

é correto. Agora, no meio do programa, isso é incorreto:

```
s = "Bom dia mundo!";
```

Temos que usar

```
strcpy(s, "Bom dia mundo!");
```

### Exercício: Ordem contrária

Escrever um programa que peça ao usuário seu nome, e que imprima na ordem contrária os caracteres do nome entrado.

### Exemplo

O usuário entra com:

Gurvan

O programa retorna `navruG`