



Linguagem de Programação I

- Vetores e Matrizes -

Prof. Ulysses Santos Sousa
ulyssessousa@ifma.edu.br

Aula 04

Roteiro

- Vetores
- Vetor de *string*
- Funções para manipulação de *strings*
- Matrizes

Vetores

- Vetor é uma estrutura de dados capaz de armazenar vários dados do mesmo tipo.
- Por isso é denominada estrutura de dados homogênea.
- O tipo de dado que um vetor armazena é definido no momento da sua declaração.
- Os valores de um vetor são acessados através de índices.

Vetores

- Estrutura

Índice	0	1	2	3	4
Elemento	30	29	43	12	18

O índice é a forma de fazer referência a cada um dos elementos do vetor, ou seja, é uma maneira de indicar a posição do elemento que estamos querendo utilizar naquele momento.

Na linguagem C, os índices iniciam de zero.

Vetores

- Declaração

- Sintaxe:

```
tipo_de_dado identificador[tamanho_vetor];
```

- Onde:

- tipo_de_dado: tipo de dado (int, char, float etc);
 - identificador: nome do vetor;
 - tamanho_vetor: número total de posições do vetor.

- Exemplos:

```
int numeros[5];  
char nome[40];
```

Vetores

- Exemplo 1:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main()
5 {
6     int numeros[5];
7     int i;
8     numeros[0] = 30;
9     numeros[1] = 29;
10    numeros[2] = 43;
11    numeros[3] = 12;
12    numeros[4] = 18;
13
14    for(i = 0; i < 5; i++)
15        printf("Valor da posicao %d: %d\n", i, numeros[i]);
16
17    return 0;
18 }
```

Vetores

- Exemplo 2:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main()
5 {
6     int numeros[5];
7     int i;
8     printf("Digite os numeros: ");
9     for(i = 0; i < 5; i++)
10         scanf("%d", &numeros[i]);
11     printf("Vetor:\n");
12     for(i = 0; i < 5; i++)
13         printf("Valor da posicao %d: %d\n", i, numeros[i]);
14
15     return 0;
16 }
```

Vetores

- Inicialização (exemplo)

```
int vetor[5] = {10, 20, 30, 40, 50};  
float n[] = {1.2, 2.3, 3.4, 4.5};
```


Vetor de String

- **Vetor de String**

- Na linguagem C, uma *string* é, na realidade, um vetor que armazena vários caracteres, terminando com o caractere `'\0'`.
- Logo, ao declararmos um vetor devemos lembrar de deixar um espaço reservado para o caractere `'\0'`.

Vetor de String

- Vetor de String

- Exemplo

- `char nome[9]`

0	1	2	3	4	5	6	7	8
'E'	'd'	'u'	'c'	'a'	'c'	'a'	'o'	'\0'

- **char** – O vetor declarado é do tipo **char**, ou seja, este vetor armazenará valores do tipo **char**.
 - **nome[9]** – O vetor armazenará até 8 caracteres, já que precisamos garantir uma posição para o caractere `'\0'`.

Vetor de String

- **Função *gets()***
 - Utilizada para realizar a leitura de uma string.
 - Essa função faz a leitura e coloca o terminador nulo (`'\0'`) no final da *string*, assim que a tecla **<enter>** for pressionada.

Vetor de String

- Função *gets()*

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main()
5 {
6     char mensagem[50];
7     printf("Digite a mensagem: ");
8     gets(mensagem);
9     printf("A mensagem digitada foi: %s\n", mensagem);
10    system("pause");
11    return 0;
12 }
```

Funções que manipulam *Strings*

- **Função *strcpy()***

- Tem a função de copiar o valor de uma *string* para outra variável.
- Sintaxe:
 - `strcpy(copia, original);`

Funções que manipulam *Strings*

- Função *strcpy()*
 - Exemplo:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main()
5 {
6     char nome[20], copiaNome[20];
7     printf("Digite a mensagem: ");
8     gets(nome);
9     strcpy(copiaNome, nome);
10    printf("Conteudo da variavel copiaNome: %s \n", copiaNome);
11    system("pause");
12    return 0;
13 }
```

Funções que manipulam *Strings*

- **Função *strcmp()***

- Tem a função de realizar a comparação entre duas strings.
- Após a comparação a função retorna o valor 0 (zero) se os valores comparados forem idênticos.
- Sintaxe:
 - `strcmp(string1, string2);`

Funções que manipulam *Strings*

- Função *strcmp()*

- Exemplo:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main()
5 {
6     char nome1[20], nome2[20];
7     printf("Digite a primeira mensagem: ");
8     gets(nome1);
9     printf("Digite a segunda mensagem: ");
10    gets(nome2);
11    if (strcmp(nome1, nome2)==0)
12    {
13        printf("Conteudo das variaveis sao identicos = %s, %s\n",
14              nome1,nome2);
15    }else{
16        printf("Conteudo das variaveis sao diferentes %s, %s\n",
17              nome1,nome2);
18    }
19    system("pause");
20    return 0;
21 }
```


Funções que manipulam *Strings*

- **Função *strcat()***

- Concatena uma *string* à outra, ou seja, acrescenta uma *string* ao final de outra.
- Recebe dois endereços de variáveis do tipo char e concatena a segunda na primeira.
- A segunda *string* não é alterada.
- Sintaxe:
 - Strcat(string1, string2)

Funções que manipulam *Strings*

- Função *strcat()*

```
#include <stdlib.h>
#include <string.h>

int main()
{
    char mensagem[50], nome[20];
    gets(mensagem);
    gets(nome);
    strcat(mensagem, nome);
    printf("%s\n", mensagem);
    getchar();
    return 0;
}
```

Matrizes

- Semelhante aos vetores, as matrizes são estruturas de dados capazes de armazenarem vários dados do mesmo tipo.
- Porém, enquanto os vetores são estruturas unidimensionais, as matrizes podem ter duas ou mais dimensões.

Matrizes

- **Matrizes Bidimensionais**

- Possuem duas ou mais linhas e duas ou mais colunas.
- Costumam ser representadas por meio de tabelas.
- Precisam de dois índices para que seus valores sejam acessados.

Matrizes

- Estrutura

		Índices de coluna		
		0	1	2
Índices de linha	0	15.00	20.00	25.00
	1	14.50	21.60	40.22
	2	30.11	12.02	20.12
	3	29.12	20.18	83.01

Matrizes

- Declaração:

- Sintaxe:

```
tipo_de_dado identificador[maxLin] [maxCol] ;
```

- Onde:

- tipo_de_dado: tipo de dado (int, char, float etc);
 - identificador: nome da matriz;
 - maxLin: total de linhas;
 - maxCol: total de colunas.

- Exemplo:

```
float dados[4][3] ;
```

Matrizes

- Exemplo:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main()
5 {
6     float notas[2][3];
7     int i, j;
8     for (i = 0; i < 2; i++)
9     {
10         printf("Digite as notas do aluno %d:\n", i + 1);
11         for(j = 0; j < 3; j++)
12         {
13             printf("nota %d: ", j + 1);
14             scanf("%f", &notas[i][j]);
15         }
16     }
```

Matrizes

- Exemplo (cont.):

```
17     for (i = 0; i < 2; i++)
18     {
19         printf("Aluno %d\n", i + 1);
20         printf("Notas:\n");
21         for (j = 0; j < 3; j++)
22             printf("%f\n", notas[i][j]);
23     }
24     system("pause");
25     return 0;
26 }
```


Matrizes

- Inicialização (exemplo):

```
int matriz[2][3] = {{10, 20, 30}, {40, 50, 60}};
```

Matrizes de caracteres

- **Exemplo:**

```
char nome[ ] = "Programa"; // {'P','r','o','g','r','a','m','a','\0'};
```

```
char dias_semana[7][14] = {"Domingo", "Segunda-feira", "Terça-  
feira", "Quarta-feira", "Quinta-feira", "Sexta-feira", "Sabado"};
```

Matrizes de caracteres

linhas colunas




```
char dias_semana[7][14] = {
```

```
    "Domingo",  
    "Segunda-feira",  
    "Terça-feira",  
    "Quarta-feira",  
    "Quinta-feira",  
    "Sexta-feira",  
    "Sabado"
```

```
};
```

Matrizes de caracteres

linhas colunas



char dias_semana[7][14] = {

D	o	m	i	n	g	o							
S	e	g	u	n	d	a	-	f	e	i	r	a	
T	e	r	c	a	-	f	e	i	r	a			
Q	u	a	r	t	a	-	f	e	i	r	a		
Q	u	i	n	t	a	-	f	e	i	r	a		
S	e	x	t	a	-	f	e	i	r	a			
S	a	b	a	d	o								

};

Resumo

- **Vetor**

- É uma estrutura indexada que armazena dados de um mesmo tipo (homogêneos).

- **Vetor de *string***

- É um vetor de caracteres que deve ser lido pela função `gets()`.

- **Matriz**

- É uma estrutura na qual vários dados podem ser armazenados, desde que eles sejam do mesmo tipo. Porém possui duas ou mais dimensões.

Referências

- MIZRAHI, V. V. Treinamento em Linguagem C. 2ª Edição. São Paulo: Person Prentice Hall, 2008.
- SANTANNA, Solimara Ravani de. Lógica de programação e automação. Curitiba: Livro Técnico, 2012. 144 p. ISBN 8563687340.