

# Programação

## Matrizes

**Prof. Silvana Teodoro**

# Matrizes



# Linguagem C

## Matrizes

- Como os vetores, as matrizes são estruturas de dados homogêneas.
- A Principal diferença em relação aos vetores (unidimensionais): possui uma ou mais dimensões adicionais, mas na maioria dos casos: utiliza-se matrizes bidimensionais.
- São utilizadas quando os dados homogêneos necessitam de uma estruturação com mais de uma dimensão.

# Linguagem C

## Matrizes

- A matriz multidimensional funciona como a matriz de uma dimensão (vetor), mas tem **mais de um índice**. As dimensões são declaradas em sequência entre colchetes.

- **Sintaxe:**

tipo nome[qtde\_elementos\_linha] [qtde\_elementos\_colunas];

- **Exemplo:**

```
int dados[2] [5]; // matriz com 2 linhas e 5 colunas do tipo int
```

# Linguagem C

## Matrizes

### Sintaxe

```
1. int mat[2][5];
```

### Exemplo

```
1. #include <stdio.h>
2. main( )
3. {
4.     int mat[2][5];
5.     int i,j;
6.     for(i=0;i<2;i++)
7.     {
8.         for(j=0;j<5;j++)
9.         {
10.             mat[i][j] = j*2;
11.             printf ("%d\n", mat[i][j]);
12.         }
13.     }
14. }
```


	0	1	2	3	4
0	0	2	4	6	8
1	0	2	4	6	8

# Linguagem C

## Matrizes

- Pode-se fornecer valores de cada elemento de uma matriz na declaração, da mesma forma que nos vetores.

- Exemplo:

  
float num[2][3] = {{3.6,2.7,7.4},{5.0,4.1,2.1}};

- A instrução abaixo atribui um valor ao elemento linha zero e coluna um da matriz **mat**:

int i=0, j=1;

mat[0][1] = 15;      ou      mat[i][j] = 15;

# Linguagem C

## Matrizes

- Se houver menos valores do que o número de elementos da matriz, os elementos restantes são inicializados automaticamente com o valor zero.

```
int num[5][3] = {{32, 64, 27}};
```

```
int n[4][4] = {{0}}; //todos elementos são nulos.
```

- A seguinte declaração causa um erro de sintaxe, uma vez que há mais elementos do que espaços de memória alocados.

```
int n[2][5] = {{32, 64, 27, 18, 95, 14}, {12,15,43,17,67,31}};
```

# Linguagem C

## Matrizes

- Um programa que causará problemas ao computador durante sua execução porque o índice da matriz **erro**, dentro do laço de for, excederá o tamanho da matriz (que é 10).

```
# include <stdio.h>
main( )
{
    int erro[10], i;
    for( i = 0; i<100; i++)
    {
        erro[i]=1;
        printf ( " %d\n ", erro[i] );
    }
}
```



# Linguagem C

## Matrizes - STRING

- Em C não existe um tipo de dado **string**, no seu lugar é utilizado uma matriz de caracteres.
- Cadeias de caracteres em C, são representadas por vetores do tipo *char* terminadas, **obrigatoriamente**, pelo caractere nulo: `'\0'` (`\zero`). Portanto, deve-se reservar uma posição para este caractere de fim de cadeia.
- **Exemplos:**  

```
char cidade[4] = {'R', 'i', 'o', '\0'};
```

```
char disc[40] = {'A', 'l', 'g', 'o', 'r', 'i', 't', 'm', 'o', '\0'};
```
- **Equivale:**  

```
char cidade[4] = "Rio";
```

```
char disc[40] = "Algoritmo";
```

# Linguagem C

## Matrizes - STRING

- Para ilustrar a declaração e a inicialização de *strings*, consideremos as seguintes declarações:

```
char s1[] = "" ; // 2 aspas sem espaços entre elas
char s2[] = "Rio de Janeiro";
char s3[81];
char s4[81] = "Rio";
```

- **s1** armazena uma string vazia. Tem um único elemento: `'\0'`;
- **s2** representa um vetor com 15 elementos (caracteres); (rio de janeiro contando os espaços 14 mais um para o `'\0'`);
- **s3** representa uma cadeia de caracteres com até 80 caracteres e não é inicializada;(lembrando que a última posição é sempre do `'\0'`);
- **s4** também é dimensionada para conter até 80 caracteres e é inicializada com a cadeia "Rio".

# Linguagem C

## Matrizes - STRING

- FUNÇÃO GETS()

`gets(nome_matriz);`

É utilizada para leitura de uma **string** através do dispositivo padrão, até que a tecla <ENTER> seja pressionada. A função **gets()** não testa limites na matriz em que é chamada. O programa apresenta um programa simples que utiliza a função **gets**.

```
# include <stdio.h>
main()
{
    char str[80];
    gets(str);
    printf("%s",str);
}
```

- Incluir **#include <string.h>**

# Linguagem C

## Matrizes - STRING

- FUNÇÃO PUTS()

`puts(nome_do_vetor_de_caracteres);`

Imprime uma string na tela seguida de nova linha. O programa apresenta um programa simples que utiliza a função **puts**.

```
# include <stdio.h>
# include <string.h>
main()
{
    puts("mensagem");
}
```

- Incluir **#include <string.h>**

# Linguagem C

## Matrizes - STRING

- FUNÇÃO STRCPY()

**strcpy(destino, origem);**

A função **strcpy()** copia o conteúdo de uma **string** para uma variável do tipo **string** (um vetor de **char**). No programa abaixo, a string “alo” será copiada para a variável (matriz de caracteres) **str**.

```
# include <stdio.h>
# include <string.h>
main()
{
    char str[20];
    strcpy(str, "alo");
    puts(str);
}
```

- Incluir **#include <string.h>**

# Linguagem C

## Matrizes - STRING

- FUNÇÃO STRNCPY()

**strncpy(destino, origem, x);**

Copia os x primeiros caracteres da string origem para a destino.

- Incluir **#include <string.h>**

# Linguagem C

## Matrizes - STRING

- FUNÇÃO STRCAT()

**strcat(string1, string2);**

Concatena duas strings.

```
#include <string.h>
#include <stdio.h>
main()
{
    char um[20], dois[10];
    strcpy (um, "bom");
    strcpy (dois, " dia");
    strcat (um, dois);      // une a string dois à string um
    printf ("%s\n", um);
}
```

- Incluir **#include <string.h>**

# Linguagem C

## Matrizes - STRING

- FUNÇÃO STRNCAT()

**strncat(string1, string2, x);**

Adiciona ao final do vetor destino os x primeiros caracteres do vetor origem.

- Incluir **#include <string.h>**



# Linguagem C

## Matrizes - STRING

- FUNÇÃO STRCMP()

**strcmp(s1, s2);**

Essa função compara duas strings, retorna 0 se s1 é igual a s2, < 0 se s1 é menor que s2 e > 0 se s1 é maior que s2. O programa mostra um exemplo da função **strcmp()**.

```
main( )
{
    char s[80];
    printf("Digite a senha:");
    gets (s);
    if ( strcmp(s,"laranja") ){
        printf("senha inválida\n");
    }
    else {
        printf("senha ok!\n") ;}
}
```

- Incluir **#include <string.h>**

# Linguagem C

## Matrizes - STRING

- **FUNÇÃO STRLEN()**

**strlen(s1);**

Determina o tamanho de uma string.

```
#include <stdio.h>
#include <string.h> //necessário para strlen
main (void)
{
    char str[5] = "Curso";
    int tamanho;

    tamanho = strlen(str);

    printf("O tamanho da string %s vale %d\n", str, tamanho);
}
```

- Incluir **#include <string.h>**

# Linguagem C

## Matrizes - STRING

- **FUNÇÃO STRREV()**

**strrev(s1);**

Inverte a string s1 sobre ela mesma.

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

```
main(){
    char exemplo[6] = "Curso";
    printf("Frase invertida: %s", strrev(exemplo));
}
```

- Incluir **#include <string.h>**

# Linguagem C

## Matrizes - STRING

- FUNÇÃO STRUPR()

**strupr(s1);**

Converte os caracteres da string para caixa alta.

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

main(){
    char exemplo[6] = "Curso";
    printf("Frase maiúscula: %s", strupr(exemplo));
}
```

- Incluir **#include <string.h>**

# Linguagem C

## Matrizes - STRING

- FUNÇÃO STRLWR()

**strlwr(s1);**

Converte os caracteres da string para caixa baixa.

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

main(){
    char exemplo[6] = "Curso";
    printf("Frase minúscula: %s", strlwr(exemplo));
}
```

- Incluir **#include <string.h>**

# Linguagem C

## Matrizes - STRING

- FUNÇÃO STRSET()

**strset(string,caractere) ;**

Substitui todos os caracteres de uma string pelo caractere passado como parâmetro.

```
#include <stdio.h>
#include <string.h>
main (void)
{
    char exemplo[5] = "Curso";
    printf("Frase minúscula: %s ", strset(exemplo, 'C') );

}
```

- Incluir **#include <string.h>**

# Linguagem C

## Matrizes - STRING

tolower	Converter um caractere em minúsculo
toupper	Converte um caractere minúsculo em maiúsculo.
isalnum	Verifica se o caractere é alfanumérico
isalpha	Verificar se o caractere é uma letra do alfabeto
isctrl	Verificar se o caractere é um caractere de controle
isdigit	Verificar se o caractere é um dígito decimal
isgraph	Verifica se o caractere tem representação gráfica
islower	Verifica se o caractere é minúsculo
isprint	Verifica se o caractere é imprimível.
ispunct	Verifica se o caractere é um ponto
isspace	Verificar se o caractere é um espaço em branco
isupper	Verifica se o caractere é uma letra maiúscula
isxdigit	Verifica se o caractere é um dígito hexadecimal

# Linguagem C

## Matrizes - STRING

```
#include<stdio.h>
#include<string.h>
main(){
    char ch;
    printf("Digite uma letra: ");
    scanf("%c", &ch);
    if ( isalnum(ch))
        printf("\nVoce digitou um caractere alfanumérico");
    if ( isalpha(ch))
        printf("\nVoce digitou um letra do alfabeto");
    if ( iscntrl(ch))
        printf("\nVoce digitou um caractere de controle");
    if ( islower(ch))
        printf("\nVoce digitou um caractere minuscuro");
    if ( isprint(ch))
        printf("\nVoce digitou caractere imprimivel");
    if ( ispunct(ch))
        printf("\nVoce digitou um ponto");
    if ( isspace(ch))
        printf("\nVoce digitou um espaço");
    if ( isupper(ch))
        printf("\nVoce digitou um caractere maiusculo");
    if ( isxdigit(ch))
        printf("\nVoce digitou um caractere hexadecimal");
}
```



# Exercícios



# Exercícios

1. Quais são os elementos do vetor referenciados pelas expressões abaixo?

mat	3.2	4.1	2.7
	5.9	0.6	9.0

a) `mat[2][0]`      b) `mat[1][1]`      c) `mat[3][1]`

2. Qual é a diferença entre os números “3” das duas instruções abaixo ?

```
int mat[6][3];
```

```
mat[6][3] = 5;
```

3. A instrução seguinte é correta para inicializar uma matriz ?

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

# Exercícios

4. Quais serão os valores dos elementos da matriz x no final da execução do trecho de programa seguinte:

```
for (i=0; i<=3; i++)  
{  
    for (j = 0; j<=3; j++)  
    {  
        x[i][j] = i*(j+1);  
    }  
}  
for (i = 0; i<=3; i++)  
{  
    x[i][2] = x[2][i];  
}
```