



Arrays de caracteres: strings

# Definição

- **String**

- Sequência de caracteres adjacentes na memória.
- Em outras palavras, strings são arrays do tipo **char**.

- **Ex:**

- `char str[6];`

# Definição

- **String**

- Devemos ficar atentos para o fato de que as strings têm no elemento seguinte a última letra da palavra/frase armazenada, um caractere `'\0'`.
- O caractere `'\0'` indica o fim da sequência de caracteres.

- Ex: `char str[6] = "Oi";`

O	i	\0	:	?	x
---	---	----	---	---	---

# Definição

- **Importante**

- Ao definir o tamanho de uma string, devemos considerar o caractere `'\0'`.
- Isso significa que a string **str** comporta uma palavra de no máximo 5 caracteres.

- Ex: `char str[6] = "Teste";`

T	e	s	t	e	\0
---	---	---	---	---	----

# Definição

- Por se tratar de um array, cada caractere podem ser acessados individualmente por indexação
  - Ex: `char str[6] = "Teste";`

T	e	s	t	e	\0
---	---	---	---	---	----

- `str[0] = 'L';`

L	e	s	t	e	\0
---	---	---	---	---	----

# Definição

- **IMPORTANTE:**

- Na inicialização de palavras, usa-se **“aspas duplas”**.

- Ex: `char str[6] = “Teste”;`

T	e	s	t	e	\0
---	---	---	---	---	----

- Na atribuição de caracteres, usa-se **‘aspas simples’**.

- `str[0] = ‘L’;`

L	e	s	t	e	\0
---	---	---	---	---	----

# Manipulando strings

- Strings são arrays. Portanto, **não** se pode atribuir uma string para outra!

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(){
04      char str1[20] = "Hello World";
05      char str2[20];
06
07      str1 = str2; //ERRADO!
08
09      system("pause");
10      return 0;
11  }
```

- O correto é copiar a string
- elemento por elemento.

# Copiando uma string

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(){
04      int i;
05      char str1[20] = "Hello World";
06      char str2[20];
07      for (i = 0; str1[i]!='\0'; i++)
08          str2[i] = str1[i];
09      str2[i] = '\0';
10      system("pause");
11      return 0;
12  }
```



# Manipulando strings

- Felizmente, a biblioteca padrão C possui funções especialmente desenvolvidas para esse tipo de tarefa
  - `#include <string.h>`

# Manipulando strings - Leitura

- `gets(str)`: lê uma string do teclado e coloca em `str`.

```
char str[10];
```

```
gets(str);
```

# Manipulando strings – Limpeza do buffer

- Às vezes, podem ocorrer erros durante a leitura de caracteres ou strings. Para resolver esses pequenos erros, podemos limpar o buffer do teclado

```
char str[10];
```

```
setbuf(stdin, NULL);//limpa o buffer
```

```
gets(str);
```

# Manipulando strings - Escrita

- Basicamente, para se escrever uma string na tela utilizamos a função **printf()**.

```
char str[20] = "Hello World";  
printf("%s",str);
```

# Manipulando strings - Tamanho

- `strlen(str)`: retorna o tamanho da string `str`. Ex:  
    `char str[15] = "teste";`  
    `printf("%d",strlen(str));`
- A função retornará 5, que é o número de caracteres na palavra “teste” e não 15, que é o tamanho do array.

# Manipulando strings - Copiar

- `strcpy(dest, fonte)`: copia a string contida na variável **fonte** para **dest**.
- Ex:  

```
char str1[100], str2[100];  
printf ("Entre com uma string: ");  
gets(str1);  
strcpy(str2, str1);  
printf("%s",str2);
```

# Manipulando strings - Concatenar

- `strcat(dest, fonte)`: concatena duas strings. A string contida em **fonte** permanecerá inalterada e será anexada ao fim da string de **dest**.

```
char str1[15] = "bom ";  
char str2[15] = "dia";  
strcat(str1, str2);  
printf("%s", str1);
```

# Manipulando strings - Comparar

- `strcmp(str1, str2)`: compara duas strings. Se as strings forem iguais, a função retorna ZERO.

```
if (strcmp(str1, str2) == 0)
    printf("Strings iguais");
else
    printf("Strings diferentes");
```



# Manipulando strings

- Basicamente, para se ler uma string do teclado utilizamos a função **gets()**.
- No entanto, existe outra função que, utilizada de forma adequada, também permite a leitura de strings do teclado. Essa função é a **fgets()**, cujo protótipo é:

```
char *fgets(char *str, int tamanho, FILE *fp);
```

# Manipulando strings

- A função **fgets** recebe 3 argumentos
  - a string a ser lida, **str**;
  - o limite máximo de caracteres a serem lidos, **tamanho**;
  - A variável FILE **\*fp**, que está associado ao arquivo de onde a string será lida.
- E retorna
  - NULL em caso de erro ou fim do arquivo;

# Manipulando strings

- Para ler do teclado, basta substituir `FILE *fp` por `stdin`, o qual representa o dispositivo de entrada padrão (geralmente o teclado).

```
fgets (str, tamanho, stdin);
```

# Manipulando strings

- A função lê a string até que um caractere de nova linha seja lido ou *tamanho-1* caracteres tenham sido lidos.
- Se o caractere de nova linha ('\n') for lido, ele fará parte da string, o que não acontecia com **gets**.
- A string resultante sempre terminará com '\0' (por isto somente *tamanho-1* caracteres, no máximo, serão lidos).

# Manipulando strings

- A função **fgets** especifica o tamanho máximo da string de entrada. Evita estouro no buffer;

Ex: fgets (str, tamanho, stdin);

# Manipulando strings

- Além do printf() pode-se usar o fputs para se escrever na tela

```
int fputs (char *str, FILE *fp);
```

- Retorno da função
  - Se o texto for escrito com sucesso um valor inteiro diferente de zero é retornado.

# Manipulando strings

- Para escrever no monitor, basta substituir `FILE *fp` por **`stdout`**  
    `char str[20] = "Teste de escrita";`  
    `fputs (str,stdout);`