Strings

emanoelim@utfpr.edu.br







Strings

- Assim como vetores e matrizes, strings s\u00e3o tipos de dados homog\u00e9neos.
- Uma string pode ser definida como um vetor de caracteres.







Strings

- A diferença entre um vetor e uma string é que o último caractere da string sempre será o caractere especial **\0**.
- Ele indica que a string chegou ao fim.







Declaração

 A declaração de uma string pode ser feita de maneira semelhante a declaração de um vetor. Por exemplo, para declarar uma string de até 50 caracteres:

char string[50];







Inicialização

- Existem diversas formas de inicializar uma string:
 - 1. O texto **inteiro** é passado entre **aspas duplas**. O compilador irá adicionar o \0 automaticamente.

```
char s[50] = "strings";
char s[] = "strings";
```

S	t	r	i	n	g	S	/0
0	1	2	3	4	5	6	7





Inicialização

- Existem diversas formas de inicializar uma string:
 - 2. Cada caractere é informado separadamente, entre **aspas simples**. É semelhante a inicialização de um vetor. É recomendado adicionar o \0 manualmente (nem todo compilador irá adicionar automaticamente):

```
char s[50] = {'s', 't', 'r', 'i', 'n', 'g', 's', '\0'};
char s[] = {'s', 't', 'r', 'i', 'n', 'g', 's', '\0'};
```

S	t	r	i	n	g	s	\0
0	1	2	3	4	5	6	7







Memória

 Assim como em vetores, uma string é armazenada em posições contíguas de memória (lembrando que cada caractere ocupa 1 byte):

```
1  #include <stdio.h>
2
3 * int main(void) {
4    char str[] = "ola";
5    printf("%c, %ld\n", str[0], &str[0]);
6    printf("%c, %ld\n", str[1], &str[1]);
7    printf("%c, %ld\n", str[2], &str[2]);
8    return 0;
9 }
```

Stdout

- 0, 140732694233256
- 1, 140732694233257
- a, 140732694233258







Saída de strings

- Diferente dos vetores, não é necessário imprimir a string item a item.
- A linguagem C tem o especificador "%s" que permite imprimir e ler strings de forma direta:

```
#include <stdio.h>

int main(void) {
    char str[] = "fundamentos de programação";
    printf("%s", str);

return 0;
}
```







Leitura de strings

- O identificador "%s" também é usado na leitura de uma string.
- Entretanto é preciso indicar para o compilador que a leitura será feita até ser digitado um ENTER ('\n'):

```
#include <stdio.h>

int main(void) {
    char str[50];
    printf("Digite seu nome: ");
    scanf("%[^\n]s", str); // nāo é preciso o &
    printf("Bom dia, %s!", str);
    return 0;
}
```







 Quando uma string é declarada com tamanho igual a 20, por exemplo: char str[20];

não estamos dizendo que o texto armazenado terá exatamente 20 caracteres. Estamos dizendo que ele terá **no máximo** 20 caracteres. Ou seja ele pode ter menos carateres.







Exemplo:

```
#include <stdio.h>

int main(void) {
    char nome[20] = "Tereza";
    int i;
    for(i = 0; i < 20; i++)
        printf("%c\n", nome[i]);
    return 0;
}</pre>
```

A string ocupa 7 posições (com o \0), nas demais posições pode ser impresso lixo de memória.







 Como o último caractere de uma string é sempre o \0, podemos usar o \0 para encontrar o final da string e imprimir apenas as posições que realmente foram usadas:

```
1  #include <stdio.h>
2
3  int main(void) {
4    char nome[20] = "Tereza";
5    int i = 0;
6  while(nome[i] != '\0') {
7     printf("%c\n", nome[i]);
8    i++;
9    }
10    return 0;
11 }
```







 Exemplo: percorra uma string e conte a quantidade de letras "a" que aparecem:

```
#include <stdio.h>
    int main(void) {
        char str[50] = "fundamentos de programação";
        int i = 0;
        int cont a = 0;
        while(str[i] != '\0') {
            if(str[i] == 'a')
                cont_a++;
10
            i++;
11
        printf("A string contém %d letras a.", cont_a);
12
13
        return 0:
14
```

Stdout

A string contém 3 letras a.

Obs.: o 'ã' não é contado, ele tem um código diferente de 'a' na tabela ASCII.







- A linguagem C possui uma biblioteca chamada **string.h**, que contém diversas funções prontas para trabalhar com strings.
- Algumas das funçõos mais usadas são descritas na sequência.







• **strlen(str)**: retorna a quantidade de caracteres de uma string:

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

int main(void) {
    char str[20] = "Maria";
    printf("Tamanho da string: %d", strlen(str));
    return 0;
}
```

Stdout

Tamanho da string: 5







strcmp(str1, str2): recebe duas strings e compara seu conteúdo.
 Ela retorna:

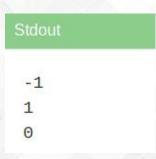
0: se as strings forem iguais;

< 0: se str1 for menor que str2;

> 0: se str1 for maior que str2;

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

int main(void) {
    printf("%d\n", strcmp("aaa", "bbb"));
    printf("%d\n", strcmp("bbb", "aaa"));
    printf("%d\n", strcmp("ccc", "ccc"));
    return 0;
}
```









• **Obs.:** sempre que precisar comparar duas strings, use a função strcmp() ou faça a comparação caractere por caractere. A seguinte comparação NÃO funciona para comparar conteúdo:

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

int main(void) {
    char str1[] = "aaa";
    char str2[] = "aaa";
    printf("%d", str1 == str2);
    return 0;
}
```



0

Esta comparação irá comparar os endereços de memória das strings e não o conteúdo.



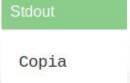




• **strcpy(destino, origem)**: copia o conteúdo da string de origem na string de destino. É uma função sem retorno:

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

int main(void) {
    char str1[20] = "Copia";
    char str2[20];
    strcpy(str2, str1);
    printf("%s", str2);
    return 0;
}
```









• **Obs.:** sempre que precisar copiar o conteúdo de uma string para outra, use a função strcpy() ou copie item a item. O seguinte código não funciona:

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main(void) {
5     char str1[20] = "Copia";
6     char str2[20];
7     str2 = str1;
8     printf("%s", str2);
9     return 0;
10 }
```

Não compila devido ao erro: /teste_str.c|7|error: assignment to expression with array type







• **strcat(str1, str2):** concatena duas strings. Irá colocar a str2 após o final da str1. Não tem retorno:

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

int main(void) {
    char str1[] = "Concatenando ";
    char str2[] = "Strings";
    strcat(str1, str2);
    printf("%s", str1);
    return 0;
}
```

Stdout

Concatenando Strings







• **Obs.:** as funções não precisam receber o tamanho da string como parâmetro, elas usam o \0 para identificar o fim da string.





