

Strings

emanoelim@utfpr.edu.br

Strings

- Assim como vetores e matrizes, strings são tipos de dados homogêneos.
- 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:
 - 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

```
o, 140732694233256
l, 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:

```
1  #include <stdio.h>
2
3  int main(void) {
4      char str[] = "fundamentos de programação";
5      printf("%s", str);
6
7      return 0;
8  }
```


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'):

```
1  #include <stdio.h>
2
3  int main(void) {
4      char str[50];
5      printf("Digite seu nome: ");
6      scanf("%[^\n]s", str); // não é preciso o &
7      printf("Bom dia, %s!", str);
8      return 0;
9  }
```

Percorrendo uma string

- 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 caracteres.

Percorrendo uma string

- Exemplo:

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

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

Percorrendo uma string

- 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 }
```

Percorrendo uma string

- Exemplo: percorra uma string e conte a quantidade de letras “a” que aparecem:

```
1  #include <stdio.h>
2
3  int main(void) {
4      char str[50] = "fundamentos de programação";
5      int i = 0;
6      int cont_a = 0;
7      while(str[i] != '\0') {
8          if(str[i] == 'a')
9              cont_a++;
10         i++;
11     }
12     printf("A string contém %d letras a.", cont_a);
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.

Funções com string

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

Funções com string

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

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main(void) {
5      char str[20] = "Maria";
6      printf("Tamanho da string: %d", strlen(str));
7      return 0;
8  }
9
```

Stdout

Tamanho da string: 5

Funções com string

- **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;

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main(void) {
5      printf("%d\n", strcmp("aaa", "bbb"));
6      printf("%d\n", strcmp("bbb", "aaa"));
7      printf("%d\n", strcmp("ccc", "ccc"));
8      return 0;
9  }
```

Stdout

-1

1

0

Funções com string

- **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:

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

Stdout

0

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

Funções com string

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

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

Stdout

Copia

Funções com string

- **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`

Funções com string

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

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

Stdout

Concatenando Strings

Funções com string

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