

INSTITUTO FEDERAL

Norte de Minas Gerais

Campus Januária

Estruturas de Dados I

- Strings -



Strings em C

- Em C não existe um tipo básico “**string**”.
- Uma **string** é, na verdade, uma estrutura do tipo *Array*. Ou seja, um caso específico de **vetor de elementos do tipo *char***.
- **Vetor de caracteres...**

```
int main() {  
    char umaString[100];  
    char variasStrings[10][100];  
}
```



Propriedades Especiais

- Uma string sempre é finalizada com um caractere nulo (0)

Exemplo:

```
char texto[10];  
scanf(" %s", texto);
```

Isso é necessário para determinar até onde, de fato, a string é válida!

I	F	N	M	G	0	\$	%	@	#
0	1	2	3	4	5	6	7	8	9




Propriedades Especiais

Para usar a função *scanf()* em estruturas do tipo *string* (vetor de caracteres) **não é necessário** preceder o nome da variável com o operador **&**

```
#include <stdio.h>

int main() {
    char nome[100];
    scanf(" %s", nome);
    printf(" %s", nome);
    return 0;
}
```





Atividade Prática

- Faça um programa que declare uma string **nome**, com limite de 100 caracteres.
p.e: `char nome[100];`
- Faça a leitura da string, usando a função **scanf()**, informando o seu **nome completo**.
- Imprima na tela o conteúdo da string **nome**, usando a função **printf()**.



Provável Solução

■ Provável Solução Implementada:

```
#include <stdio.h>

int main() {
    char nome[100];
    scanf(" %s", nome);
    printf(" %s", nome);
    return 0;
}
```



Propriedades Especiais

A função *scanf()* considera, por padrão, que **espaços em branco** finalizam a leitura da string.

Solução: Dizer explicitamente à função quando ela deve interromper a leitura da string.

```
#include <stdio.h>

int main() {
    char nome[100];
    scanf(" %[^\n]s", nome);
    printf(" %s", nome);
    return 0;
}
```



I/O para Strings

- Leitura de *strings* com o `scanf`
- Impressão de *strings* com o `printf`

```
#include <stdio.h>

int main() {
    char nome[100];
    scanf(" %[^\n]s", nome);
    printf("Nome: %s", nome);
    return 0;
}
```




Funções para Strings

```
char* strcpy(destino, origem);
```

- Biblioteca <string.h>
 - Copia o conteúdo de uma string da origem para o destino.

```
char* strcat(str1, str2);
```

- Biblioteca <string.h>
 - Concatena duas na primeira;
 - Não verifica o tamanho.



Funções para Strings

```
int strcmp(str1, str2);
```

■ Biblioteca <string.h>

- Verifica se duas *strings* são idênticas (**retorno 0**)
- Se **str1 > str2** então o retorno é **positivo**.
- Se **str2 > str1** então o retorno é **negativo**.

```
int strlen(string);
```

■ Biblioteca <string.h>

- Retorna o comprimento da *string* fornecida.
 - *O caractere final (nulo) não é considerado.*



Vamos à Prática!

1. Desenvolva um programa em C que conte e imprime o número de ocorrências de uma letra, fornecida pelo usuário, em uma *string* também informada pelo usuário.

2. Construir um programa que pede para o usuário:

- (i) Uma string *S*.
- (ii) Um caractere *c1*.
- (iii) Um caractere *c2*.

O programa deve substituir todas as ocorrências do caractere **c1** na string **S** pelo caractere **c2**. Imprima o resultado.

3. Faça um programa que leia 3 palavras. O programa deve imprimir as três palavras lidas em **ordem alfabética**.

4. Faça um programa que leia o nome completo de uma pessoa. O programa deve imprimir o nome com todas as iniciais no formato maiúsculo, e demais letras no formato minúsculo.

5. Faça um programa que leia, em formato de string, um valor binário. O programa deve imprimir o número lido no formato decimal correspondente.

Exemplo: 10110 == 22