



**INSTITUTO FEDERAL**

Norte de Minas Gerais

Campus Januária

# Estruturas de Dados I

## - *Strings* -



# Strings em C

- **STRING** é, basicamente, uma sequência de caracteres que permite representar informações textuais.

*Exemplos...*

- *"Este texto é uma string"*
- *"Maior campeão do Brasil"*
- *"1º Campeão Mundial"*



# Strings em C

- **STRING** é, basicamente, uma sequência de caracteres que permite representar informações textuais.

*Exemplos...*

- *"Este texto é uma string"*
- *"Maior campeão do Brasil"*
- *"1º Campeão Mundial"*
- *"Quando surge o alviverde imponente  
No gramado em que a luta o aguarda  
Sabe bem o que vem pela frente  
Que a dureza do prélio não tarda!"*



# Strings em C

- **STRING** é, basicamente, uma sequência de caracteres que permite representar informações textuais.

*Exemplos...*

- *"Este texto é uma string"*
- *"Maior campeão do Brasil"*
- *"1º Campeão Mundial"*

**Como definir o tamanho máximo de uma String?**





# Strings em C

- **STRING** é, basicamente, uma sequência de caracteres que permite representar informações textuais.

*Exemplos...*

- *"Este texto é uma string"*
- *"Maior campeão do Brasil"*
- *"1º Campeão Mundial"*

- **Em C não existe um tipo primitivo “string”**

Uma **string** é uma estrutura do tipo **Array**.

*Outras linguagens implementam bibliotecas de alto nível para criação e tratamento de strings. [Leia Aqui](#) e [Leia Aqui](#)*



# Strings em C

*Ou seja, toda string é um caso de aplicação de Array de elementos do tipo char*

## ■ String == Vetor de Caracteres

```
int main() {  
    char nome[100];  
    char estados_BR[27][100];  
}
```

*uma única string*

*várias strings*



# Não Confunda

## char != string

- "IFNMG" => string
- 'I' 'F' 'N' 'M' 'G' => caracteres
- 9 => inteiro
- '9' => caractere
- 'A' => caractere
- "A" => ???



# Não Confunda

## char != string

- "IFNMG" => string
- 'I' 'F' 'N' 'M' 'G' => caracteres
- 9 => inteiro
- '9' => caractere
- 'A' => caractere
- "A" => **string** *(especificamente na linguagem C)*





# Propriedades Especiais

- Uma string sempre é finalizada com um caractere nulo (valor inteiro igual a 0 ou /0)

*Exemplo:*

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

Isso é necessário para determinar até onde, de fato, a string é válida, prevenindo a “impressão de lixo”.

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

A função `scanf("%s")` já se encarrega de “marcar” o final da string após a leitura



# Propriedades Especiais

- Uma string sempre é finalizada com um caractere nulo (valor inteiro igual a 0 ou /0)

	0	1	2	3	4	5	6	7	8	9
arr [0]	G	e	e	k	\0					
arr [1]	G	e	e	k	s	\0				
arr [2]	G	e	e	k	s	f	o	r	\0	




# Propriedades Especiais

Na leitura de strings (*array de char*) com a função `scanf()` **não deve-se utilizar o operador &** antes do nome da variável.

```
#include <stdio.h>

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



*Entenderemos o porquê disso  
nas próximas aulas...*



# Atividade Prática

- Faça um programa que declare uma string **nome**, com limite de 100 caracteres.  
*p.ex:* **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("Nome Lido: %s\n", nome);
}
```





# Propriedades Especiais


## 1ª Situação: Definir quando o scanf deve interpretar o final da string

A função `scanf()` considera por padrão, que **espaços em branco** finalizam a entrada de uma string.

*Solução: Dizer explicitamente à função `scanf()` quando considerar o final da leitura de uma string...*

```
#include <stdio.h>

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





# 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 Lido: %s\\n", nome);
    return 0;
}
```



# Atividade Prática

- *Continuando a atividade anterior...*
- Faça um programa que leia três nomes.
- Faça a leitura das strings usando a função **scanf()**, como aprendemos.
- Imprima na tela o conteúdo das 03 strings na ordem de leitura.



# Provável Solução

## ■ Provável Solução Implementada:

```
#include <stdio.h>

int main() {
    char nome1[100];
    char nome2[100];
    char nome3[100];
    scanf("%[^\\n]s", nome1);
    scanf("%[^\\n]s", nome2);
    scanf("%[^\\n]s", nome3);
    printf("Nome Lido: %s\\n", nome1);
    printf("Nome Lido: %s\\n", nome2);
    printf("Nome Lido: %s\\n", nome3);
}
```



# Escape da Tecla ENTER

## 2ª Situação:

*Tratar o problema de escape da tecla ENTER...*

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

```
int main()
```

```
char n
```

```
char nome2[100];
```

```
char nome3[100];
```

```
scanf("%[^\n]s", nome1);
```

```
scanf("%[^\n]s", nome2);
```

```
scanf("%[^\n]s", nome3);
```

```
printf("Nome Lido: %s\n", nome1);
```

```
printf("Nome Lido: %s\n", nome2);
```

```
printf("Nome Lido: %s\n", nome3);
```

```
}
```

**SOLUÇÃO:** Colocar um espaço em branco antes da flag de formato do scanf().





# Atividade Prática

- ***Continuando a atividade anterior...***
- Reduza o tamanho dos arrays para tamanho 3 (*situação de um problema hipotético*).
- Continue a leitura dos nomes, com tamanhos superiores ao tamanho do array.
- Imprima na tela o conteúdo das 03 strings na ordem de leitura.



# Provável Solução

## ■ Provável Solução Implementada:

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

```
int main() {
```

```
    char nome1[3];
```

```
    char nome2[3];
```

```
    char nome3[3];
```

```
    scanf("%[^\\n]s", nome1);
```

```
    scanf("%[^\\n]s", nome2);
```

```
    scanf("%[^\\n]s", nome3);
```

```
    printf("Nome Lido: %s\\n", nome1);
```

```
    printf("Nome Lido: %s\\n", nome2);
```

```
    printf("Nome Lido: %s\\n", nome3);
```

```
}
```

```
Terminal
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda

adriano antunes
joao fulano
maria ciclana
Nome Lido: adrjoamaria ciclana
Nome Lido: joamaria ciclana
Nome Lido: maria ciclana
*** stack smashing detected ***: terminated
Aborted (core dumped)

-----
(program exited with code: 134)
Press return to continue
```



# Escape da Tecla ENTER

## 3ª Situação:

*Delimitar o tamanho de entrada de dados do scanf()*

```
#include <stdio.h>

int main() {
    char nome1[3];
    char nome2[3];
    char nome3[3];
    scanf(" %2[^\n]s", nome1);
    while(getchar() != '\n');
    scanf(" %2[^\n]s", nome2);
    while(getchar() != '\n');
    scanf(" %2[^~\n]s", nome3);
    while(getchar() != '\n');
    (...)
}
```

**SOLUÇÃO:** Delimitar o tamanho máximo da entrada de dados após o símbolo %

**Lembrar do espaço do Char NULL**

**SOLUÇÃO:** "Consumir" todos os caracteres excedentes no input.



# Atividade Prática

- *Continuando a atividade anterior...*
- Leia o nome de 03 pessoas, com tamanho máximo de 10 caracteres.
- Imprima os nomes lidos em **ORDEM ALFABÉTICA**.



# Funções para Strings

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

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

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

- Biblioteca <string.h>
- Compara a str1 com str2 e...
  - Se **str1 == str2** então o retorno é **igual a zero**.
  - Se **str1 > str2** então o retorno é **maior que zero**.
  - Se **str1 < str2** então o retorno é **menor que zero**.





# Funções para Strings

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

- Concatena str2 junto à str1
- *Obs.: Não verifica a capacidade das strings.*

```
int sprintf(str, format, variaveis...);
```

- Similar ao printf, mas direcionando a saída da impressão para a string str.

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

- Retorna o comprimento válido da *string* fornecida.
  - Desconsidera o char NULL.



# BORA CODAR!!!



1. Faça um programa que simula um campo *input* para digitação de senha, oculto por caracteres '\*'. Após pressionar ENTER revele a senha digitada. Repita a operação 5 vezes na mesma execução.

*Dica: use a função `getch()` da lib `gconio.h`*

2. Construa um programa que peça para o usuário:

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

O programa deve substituir todas as ocorrências de **c1** na string **S** pelo caractere **c2**.

3. Uma senha forte é uma string contendo entre 10 e 16 caracteres, com a presença obrigatória de pelo menos 1 caractere numérico, 1 caractere maiúsculo, 1 minúsculo e 1 caractere especial, com tamanhos e posições aleatórias. Faça um programa que faça a geração e impressão de 10 senhas fortes aleatórias.

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

5. 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.

6. Faça um programa que alimenta uma matriz [10][5] com n°s aleatórios entre 0 e 1000. Faça a impressão dos valores em formato de tabela, e cuja largura de cada coluna contenha exatamente o tamanho de N espaços (N deverá ser informado pelo usuário). Dica: `sprintf()`

7. 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