

INSTITUTO FEDERAL

Rio Grande do Sul

Campus Porto Alegre

Linguagem de Programação I

Prof. Fabio Okuyama

Curso Superior em Tecnologia em Sistemas para Internet

Strings

Trabalhando com textos

O que é uma String?

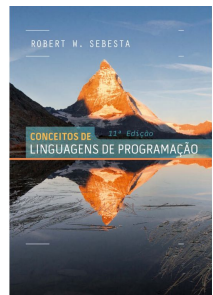


O termo string **serve para identificar uma sequência de caracteres para representar textos.**

Para representar uma string em C, devemos **criar uma variável do tipo char com o tamanho da String** que desejamos ler.

Texto, Sequência Caracteres, String

Qual o TAMANHO máximo para o Nome?
Quantas LETRAS o Nome PODE ter?
100 é o suficiente???



Strings na Linguagem C

Na Linguagem C ANSI não existe o tipo String

É utilizado **vetores** de caracteres terminados em “\0”: caracter nulo.

(veremos vetores nas próximas aulas).

Por esta razão é **necessário reservar uma posição a mais** para armazenar a string.

Para armazenar a String “**casa**” é necessário um vetor de **5 caracteres**

Vetor com tamanho 5

```
char str[5] = "casa";
```

c	a	s	a	\0
----------	----------	----------	----------	-----------

Ao inicializar um vetor com uma constante string, não é necessário inserir o ‘\0’:

Variáveis char como String

Sintaxe:

```
// tamanho explícito  
char nome[TAMANHO];
```

```
// tamanho implícito  
char nome[] = "Texto inicial";
```

Onde:

<char> especifica o tipo de dados da variável como char

<nome> nome/identificador atribuído a posição de memória

<valor inicial> valor inicial opcional da variável

= operador realiza atribuição de valores em variáveis;

[] informação de conjunto de caracteres

TAMANHO Tamanho máximo do conjunto de caracteres

Strings - Textos na Linguagem C

Atribuindo um valor diretamente
usando **tamanho implícito**

Exemplo:

```
char nome[] = "Maria Silva";//11+1 caracteres  
printf ("\nO nome informado foi: %s", nome);
```

Saída:

```
O nome informado foi: Maria Silva
```

Strings - Textos na Linguagem C

Atribuindo um valor diretamente
usando tamanho **explícito de limite**

Exemplo:

```
char nome[100] = "Maria Silva";  
printf ("\nO nome informado foi: %s", nome);
```

Saída:

```
O nome informado foi: Maria Silva
```

Strings - Textos na Linguagem C

Atribuindo um valor com scanf

Exemplo:

```
char nome[100];  
printf ("\nDigite seu Nome: ");  
scanf ("%s", nome); // Nome digitado: Maria Silva  
printf ("\nO nome informado foi: %s", nome);
```

Saída:

O nome informado foi: Maria

scanf com %s só captura a primeira palavra

Strings - Textos na Linguagem C

Atribuindo um valor com scanf com espaços no meio

Exemplo:

```
char nome[100];  
printf ("\nDigite seu Nome: ");  
scanf ("%[^\\n]s", nome); // Nome digitado: Maria Silva  
printf ("\nO nome informado foi: %s", nome);
```

Saída:

```
O nome informado foi: Maria Silva
```

A instrução **gets**

A instrução **gets** é usada em C para ler entradas String, ou seja: palavras, frases ou conjuntos de números com os quais não se realizará **NENHUMA OPERAÇÃO MATEMÁTICA**.

Sintaxe:

```
gets (variável) ;
```

Onde:

<variável> nome da variável do **tipo texto**

Exemplo:

```
gets (nome) ;
```

Strings - Textos na Linguagem C

Atribuindo um valor com gets
usando tamanho **explícito de limite**

Exemplo:

```
char nome[100];  
printf ("\nDigite seu Nome: ");  
gets (nome); // Nome digitado: Maria Silva  
printf ("\nO nome informado foi: %s", nome);
```

Saída:

```
O nome informado foi: Maria Silva
```

A instrução **getchar**

A instrução **getchar** é usada em C para ler um caractere, ou seja, uma única Letra.

Sintaxe:

```
variável = getchar();
```

Onde:

<variável> nome da variável do **tipo texto**

Exemplo:

```
char inicial;  
printf ("\nDigite a letra inicial do seu Nome: ");  
inicial = getchar(); // Letra digitada: M  
printf ("\nA inicial do seu Nome é: %c", inicial);
```

Saída:

A inicial do seu Nome é: **M**

A instrução **puts**

A instrução **puts** significa "**put string**" (colocar string), é utilizada para "colocar" uma string na saída de dados, ou seja: palavras, frases e caracteres

Sintaxe:

insere uma quebra de linha após cada string, mesmo que não haja um caractere um **'\n'** no final

```
puts (variável) ;
```

Onde:

<variável> nome da variável do **tipo texto**

Exemplo:

não utilizar para números

```
puts (nome) ;
```

A instrução **putchar**

A instrução **putchar** significa "**put caractere**" (colocar um caractere), é utilizada para "colocar" um caractere na saída de dados, ou seja: uma letra.

Sintaxe:

```
putchar (variável) ;
```

Onde:

<variável> nome da variável do **tipo caractere**

Exemplo:

```
putchar (nome) ;
```

Exemplificando...

```
int main() {
    char usuario[50], nomeCompleto[100];
    char noticia;
    int idade;
    puts("Digite seu Usuário: ");
    gets(usuario);
    puts("Digite seu Nome Completo: ");
    gets(nomeCompleto);
    puts("Deseja receber notícias? Digite 's' para SIM ou 'n' para NÃO:");
    noticia = getchar();
    puts("Digite sua Idade: ");
    scanf("%d", &idade);

    puts("Seu Usuário é: ");
    puts(usuario);
    puts("Seu Nome Completo é: ");
    puts(nomeCompleto);
    puts("Sua resposta para notícias foi: ");
    putchar(noticia);
    printf("\nSua Idade é : \n %d", idade);
}
```

```
Digite seu Usuário:
mariasilva
Digite seu Nome Completo:
Maria Silva
Deseja receber notícias?? Digite 's' para SIM ou 'n' para NÃO:
s
Digite sua Idade:
33
Seu Usuário é:
mariasilva
Seu Nome Completo é:
Maria Silva
Sua resposta para notícias foi:
s
Sua Idade é :
33
-----
Process exited after 16.98 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Esquema de Funcionamento do Computador

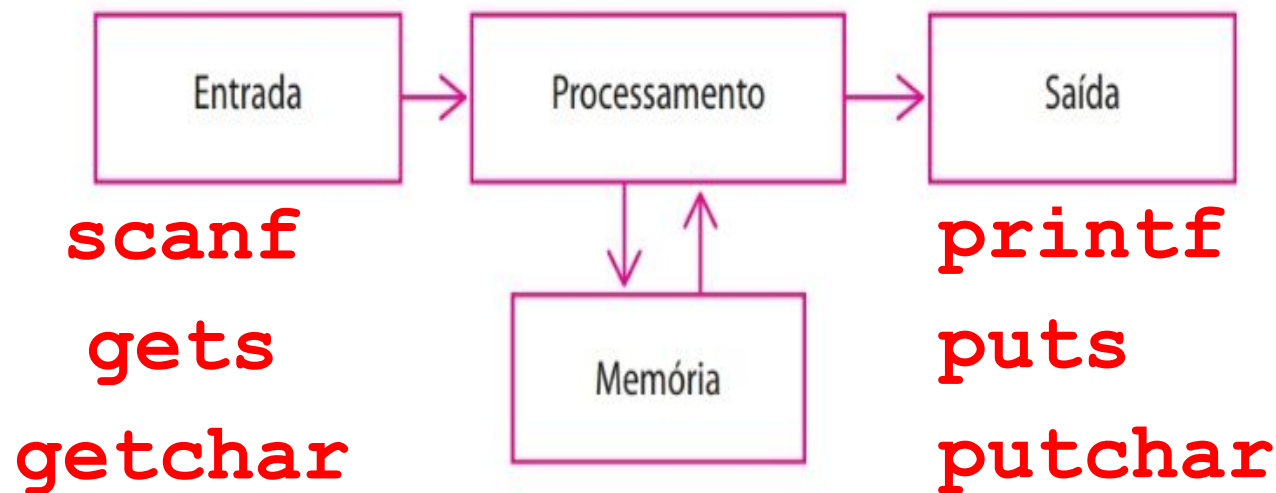


Figura 1.3 Esquema básico de funcionamento do computador.

Tamanho da String `sizeof` e `strlen`

`sizeof` – devolve o tamanho em bytes **da variável**

`strlen` – devolve o tamanho em bytes **da string**

Sintaxes:

```
sizeof(variavel);
```

```
strlen(variavel);
```

Onde:

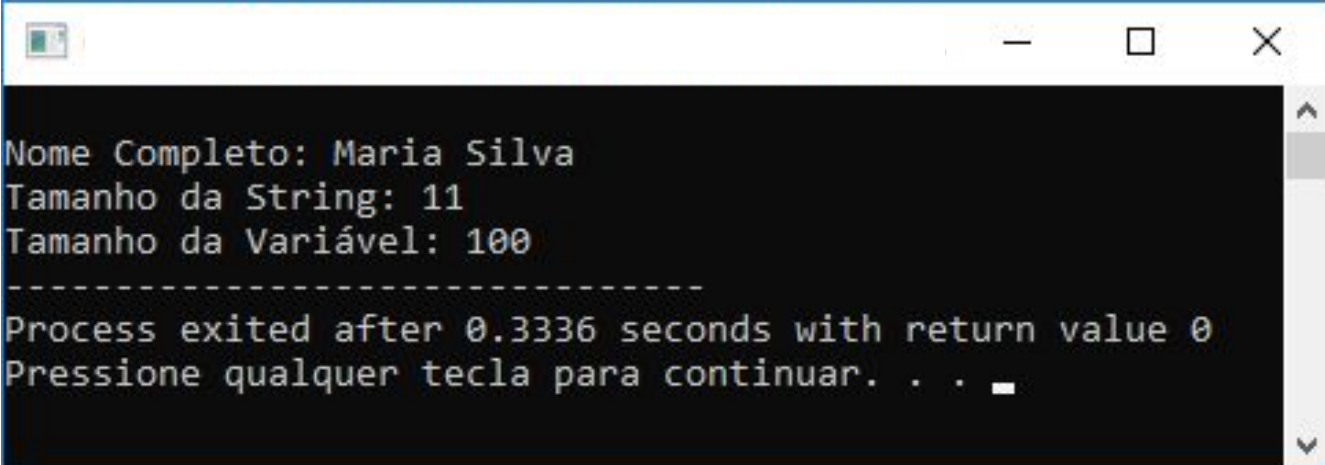
`<variável>` nome da variável do **tipo texto**

Tamanho da String `sizeof` e `strlen`

Exemplo 1: Fazer programa que informe a capacidade de uma variável e o espaço ocupado

```
int main() {  
    char nomeCompleto[100] = "Maria Silva";  
    int tamanhoDaString = strlen(nomeCompleto);  
    int tamanhoDaVariavel = sizeof(nomeCompleto);  
  
    printf ("\nNome Completo: %s", nomeCompleto);  
    printf ("\nTamanho da String: %d", tamanhoDaString);  
    printf ("\nTamanho da Variável: %d", tamanhoDaVariavel);  
}
```

declaração com tamanho explícito



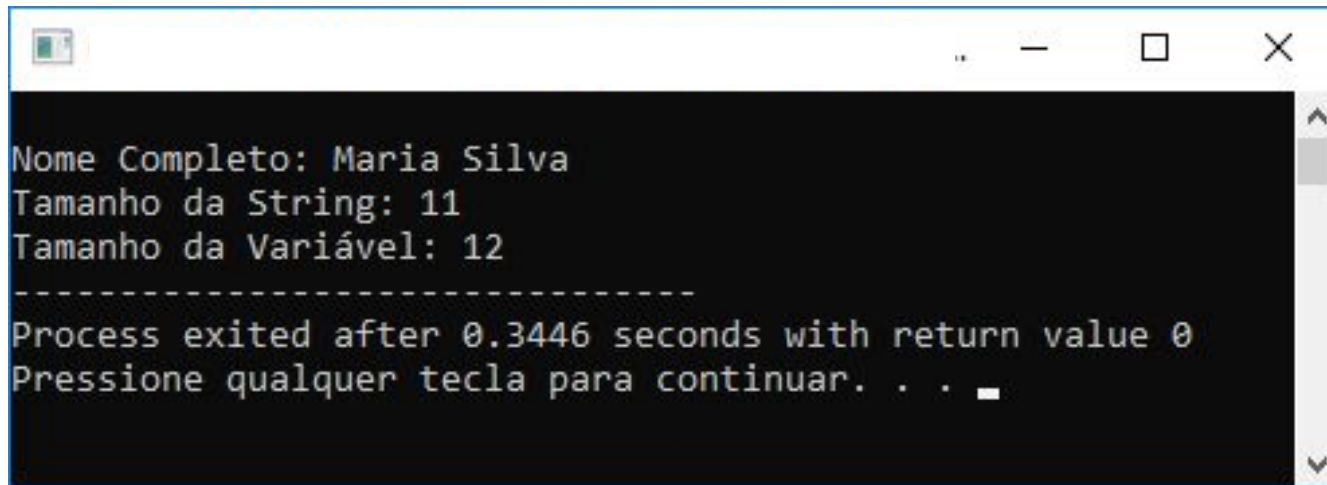
```
Nome Completo: Maria Silva  
Tamanho da String: 11  
Tamanho da Variável: 100  
-----  
Process exited after 0.3336 seconds with return value 0  
Pressione qualquer tecla para continuar. . .
```

Tamanho da String `sizeof` e `strlen`

declaração com tamanho implícito

Exemplo 2:

```
int main() {  
    char nomeCompleto[] = "Maria Silva";  
    int tamanhoDaString = strlen(nomeCompleto);  
    int tamanhoDaVariavel = sizeof(nomeCompleto);  
    printf ("\nNome Completo: %s", nomeCompleto);  
    printf ("\nTamanho da String: %d", tamanhoDaString);  
    printf ("\nTamanho da Variável: %d", tamanhoDaVariavel);  
}
```



```
Nome Completo: Maria Silva  
Tamanho da String: 11  
Tamanho da Variável: 12  
-----  
Process exited after 0.3446 seconds with return value 0  
Pressione qualquer tecla para continuar. . .
```

Manipulando Strings

Para manipulação de Strings a Linguagem C conta com a biblioteca **string.h** com um **conjunto de funções para exame e manipulação de textos**:

strcpy – copia o conteúdo de uma string para outra e coloca um terminador de string

strcmp – utilizado para **comparar ordem alfabética** da primeira com a segunda String. Retorna um inteiro **-1** se a string1 for menor, **1** se a string1 for maior e **0** se forem iguais.

strcat – adiciona o conteúdo da segunda ao final da primeira String

Sintaxes:

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

```
strcmp (string1, string2);
```

```
strcat (string1, string2);
```

ATENÇÃO

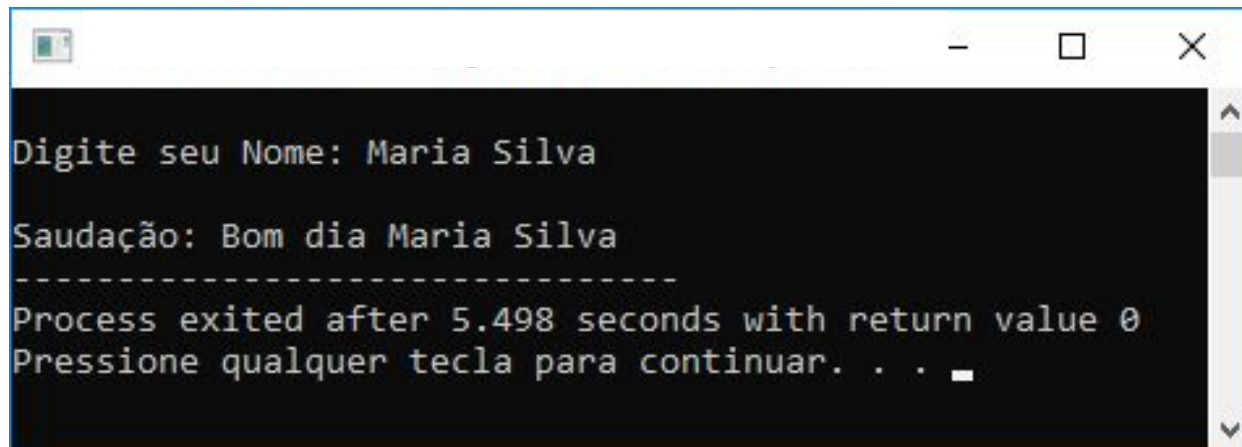
```
#include <string.h>
```

a biblioteca deve ser incluída no cabeçalho do programa sempre que utilizarmos os comandos

Manipulando Strings - Copiar Texto

Exemplo: realizar a cópia do conteúdo de uma string em outra string

```
int main(){
    char nome[100];
    char copiaNome[100];
    printf ("\nDigite seu Nome: ");
    gets(nome);
    strcpy (copiaNome, nome);
    printf ("\nSaudação: \nBom dia %s", copiaNome);
}
```



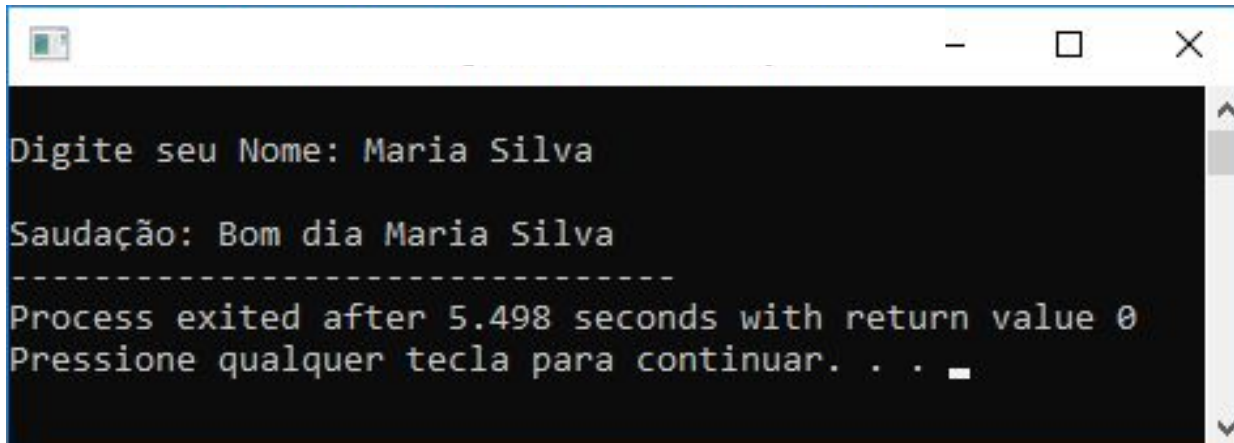
```
Digite seu Nome: Maria Silva

Saudação: Bom dia Maria Silva
-----
Process exited after 5.498 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Manipulando Strings - Concatenar Texto

Exemplo: Faça um programa que leia do usuário seu nome e apresente uma saudação personalizada. **Ex:** "Bom dia Fabio!"

```
int main(){
    char saudacao[200] = "\nBom dia ";
    char nome[100];
    printf ("\nDigite seu Nome: ");
    gets(nome);
    strcat (saudacao, nome);
    printf ("\nSaudação: %s", saudacao);
}
```



```
Digite seu Nome: Maria Silva

Saudação: Bom dia Maria Silva
-----
Process exited after 5.498 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Manipulando Strings - Comparar Strings

strcmp – utilizado para comparar a primeira com a segunda String

Sintaxe:

`strcmp(str1, str2)`

retorna 0 se forem iguais

retorna valor positivo se

$str1 > str2$

retorna valor negativo se

$str1 < str2$

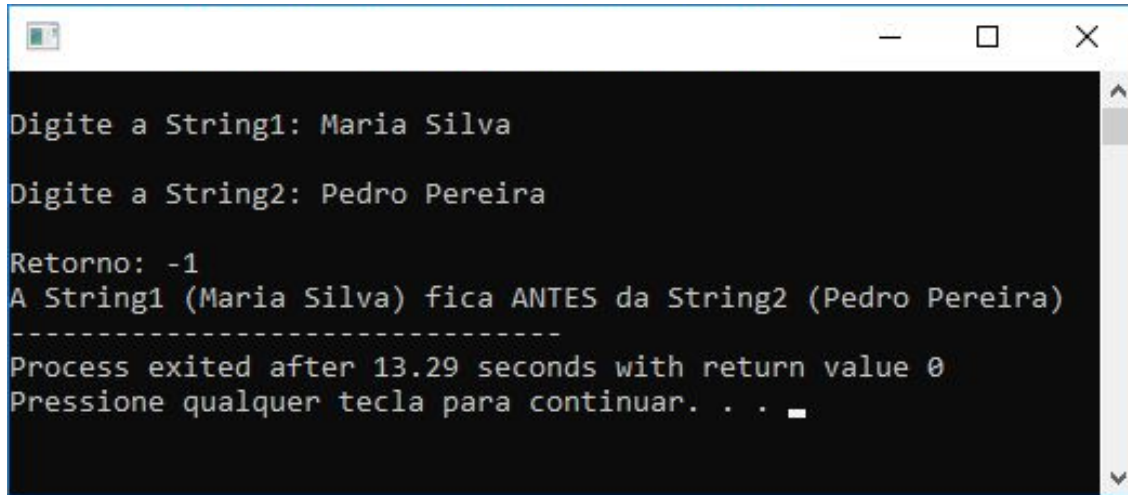
```
1  #include <stdio.h>
2
3  int main(void) {
4      char s[80];
5      printf("Digite a senha:");
6      gets(s);
7      if (strcmp(s, "segredo"))
8          printf("senha inválida\n");
9      else
10         printf("senha ok!\n") ;
11 }
```

**a função strcmp diferencia maiúsculas de minúsculas
então "Oi" é diferente de "oi"**

Manipulando Strings - Ordem Alfabética

Exemplo:

```
int retorno;  
char string1[100], string2[100];  
  
printf ("\nDigite a String1: ");  
gets(string1);  
printf ("\nDigite a String2: ");  
gets(string2);
```



```
Digite a String1: Maria Silva  
Digite a String2: Pedro Pereira  
Retorno: -1  
A String1 (Maria Silva) fica ANTES da String2 (Pedro Pereira)  
-----  
Process exited after 13.29 seconds with return value 0  
Pressione qualquer tecla para continuar. . .
```

```
retorno = strcmp (string1, string2);  
  
printf ("\nRetorno: %d", retorno);  
  
if(retorno < 0){ // menor  
    printf ("\nA String1 (%s) fica ANTES da String2 (%s)", string1, string2);  
} else {  
    if(retorno > 0){ // maior  
        printf ("\nA String1 (%s) fica DEPOIS da String2 (%s)", string1, string2);  
    } else { // igual  
        printf ("\nA String1 (%s) e a String2 (%s) são IGUAIS", string1, string2);  
    }  
}
```


Importante!

#include <string.h>

Deve ser incluído no cabeçalho do programa sempre que utilizarmos os comandos:

```
strcpy  
strcmp  
strcat  
strlen
```



Exercício

Escreva um programa que leia duas strings, compare as duas e indique o tamanho de cada string, o espaço em memória ocupado por cada string. Concatenar as duas strings. Antes de concatenar verifique se a string de destino tem espaço suficiente para a concatenação, caso o espaço não seja suficiente, escreva uma mensagem de erro.

Exercício

Escreva um programa que: a - leia duas strings; b- compare as duas: informe se as duas são iguais ou diferentes entre si; c - indique a capacidade de cada variável string; d - informe o espaço usado pelo conteúdo em cada string. e - Concatenar as duas strings. Antes de concatenar verifique se a string de destino tem espaço suficiente para a concatenação, caso o espaço não seja suficiente, escreva uma mensagem de erro.

DICA NO PRÓXIMO SLIDE
Use apenas as dicas que precisar ;)

Dica 1

Escreva um programa que: a - leia duas strings; b- compare as duas: informe se as duas são iguais ou diferentes entre si; c - indique a capacidade de cada variável string; d - informe o espaço usado pelo conteúdo em cada string. e - Concatenar as duas strings. Antes de concatenar verifique se a string de destino tem espaço suficiente para a concatenação, caso o espaço não seja suficiente, escreva uma mensagem de erro.

Para saber se uma a junção das duas strings cabem juntas:

$\text{Capacidade da String de Destino} > \text{Tamanho String1} + \text{Tamanho da String2}$

Dica 2

Escreva um programa que: a - leia duas strings; b- compare as duas: informe se as duas são iguais ou diferentes entre si; c - indique a capacidade de cada variável string; d - informe o espaço usado pelo conteúdo em cada string. e - Concatenar as duas strings. Antes de concatenar verifique se a string de destino tem espaço suficiente para a concatenação, caso o espaço não seja suficiente, escreva uma mensagem de erro.

Para saber se uma a junção das duas strings cabem juntas:

$\text{Capacidade da String de Destino} > \text{Tamanho String1} + \text{Tamanho da String2}$

função `sizeof` indica a capacidade de uma variável
função `strlen` indica o espaço ocupado em uma string

Dica 3

Escreva um programa que: a - leia duas strings; b- compare as duas: informe se as duas são iguais ou diferentes entre si; c - indique a capacidade de cada variável string; d - informe o espaço usado pelo conteúdo em cada string. e - Concatenar as duas strings. Antes de concatenar verifique se a string de destino tem espaço suficiente para a concatenação, caso o espaço não seja suficiente, escreva uma mensagem de erro.

Para saber se uma a junção das duas strings cabem juntas:

$\text{Capacidade da String de Destino} > \text{Tamanho String1} + \text{Tamanho da String2}$

função `sizeof` indica a capacidade de uma variável

função `strlen` indica o espaço ocupado em uma string

Se o destino é `str1` e a outra string é `str2`, temos:

$\text{sizeof}(\text{str1}) > \text{strlen}(\text{str1}) + \text{strlen}(\text{str2})$

Dica 4

Escreva um programa que: a - leia duas strings; b- compare as duas: informe se as duas são iguais ou diferentes entre si; c - indique a capacidade de cada variável string; d - informe o espaço usado pelo conteúdo em cada string. e - Concatenar as duas strings. Antes de concatenar verifique se a string de destino tem espaço suficiente para a concatenação, caso o espaço não seja suficiente, escreva uma mensagem de erro.

Se o destino é str1 e a outra string é str2, temos:
 $\text{sizeof(str1)} > \text{strlen(str1)} + \text{strlen(str2)}$

Se (condição) concatenar
senão mensagem de erro