



Algoritmos e Programação II

Prof. Joilson dos Reis Brito
Prof.^a Noeli A. Pimentel Vaz

Agenda da Aula

- Correção do exercício de vetor de caracteres
 - Slide 06
- Cadeia de Caracteres – Strings
- Manipulação de String em C – Vetor de char
- Biblioteca string.h

Correção Exercício Slide 06

```
1  #include <stdio.h>
2  #include <ctype.h>
3  #define TAMANHO 12
4  int main()
5  {
6      char Caracteres[TAMANHO];
7      int I;
8      printf("Informe 12 caracteres:");
9      for(I=0; I<TAMANHO; I++)
10     {
11         fflush(stdin);
12         scanf("%c", &Caracteres[I]);
13     }
14     for(I=0; I<TAMANHO; I++)
15         if(isalpha(Caracteres[I]) != 0)
16             Caracteres[I] = toupper(Caracteres[I]);
17     else
18         Caracteres[I] = '*';
19     for(I=0; I<TAMANHO; I++)
20         printf("%c", Caracteres[I]);
21     return 0;
22 }
```

CADEIA DE CARACTERES - **STRINGS**

Cadeia de Carateres - String

Os termos **Cadeia de Caracteres** e **String** representam a mesma estrutura.

O termo Cadeia de Caracteres está em português e o termo String está em Inglês.

Nos próximos slides utilizaremos o termo String

String

Uma string é uma estrutura que armazena uma ou mais palavras.



A collection of various string literals in different fonts and styles, illustrating the concept of a string. The strings include: "AFFF", "Kkkkkkkk", "PUTS!!!!", "Eu?", "ORRA MEU!!!", "UAAAA", "Mano véio", "NÉ?", "Ressaca", "ô Loko", "Vê se pode!", "psiu", "PIREI!", "Aff", "Obs: RS", and "AHHAHA".

Representação de uma string

- Uma **string** é um conjunto de caracteres armazenados em uma única variável.
- Como em C as **strings não são um tipo básico**, elas são armazenadas em um vetor de caracteres.
- A declaração de strings obedece a sintaxe de declaração de um vetor de caracteres.

Exemplo:

```
char Nome[30];
```

String

A linguagem C utiliza um código para representar que um **vetor de caracteres** será considerado uma **string**.

Esse código é: **\0**

Assim uma string é um vetor de caracteres que tem o código **\0** na sua última posição.

String

A linguagem C tem comandos e funções que são próprios para trabalhar com string.

Quando precisamos manipular palavras e frases na linguagem C é muito mais fácil trabalharmos com **string** do que com **vetor de caracteres**.

Nos próximos slides vamos aprender como trabalhar com strings na linguagem C.

String

Como já vimos anteriormente, um caracter é representado entre aspas simples.

```
char Letra, Palavra[5] = {'M','u','n','d','o'};  
Letra= 'A';
```

As string, por sua vez, são representadas entre aspas duplas.

```
char Mensagem[10] = "Ola Mundo";
```

A linguagem C insere um `\0` automaticamente no final do vetor, para marcar que este vetor armazena uma string

O	l	a		M	u	n	d	o	\0
A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]

String

Para imprimir uma string utilizados o código %s

```
#include <stdio.h>

int main(){
    char Mensagem[10] = "Ola Mundo";

    printf("Mensagem: %s.\n",Mensagem);

    return 0;
}
```

Vetores de Caracteres e Strings

Qual a diferença entre
vetor de caracteres e
strings?



Vetores de Caracteres e Strings

```
char VetorCaracter[] = {'a', 'e', 'i', 'o', 'u'};
```

Vetor de caracteres, não é string, pois nesse caso o caracter delimitador não é adicionado ao final do vetor. Trata-se de um vetor com cinco caracteres, que serão utilizados individualmente e não como um todo.

```
char CadeiaCaracter[] = "aeiou";
```

O vetor CadeiaCaracter é iniciado com a string "aeiou", logo podemos considerá-la como uma string, pois o compilador coloca automaticamente o código delimitador `\0`. O vetor CadeiaCaracter fica com seis caracteres de comprimento.

Leitura e Impressão de Strings

Entrada

Para ler uma string utilizamos o código de formatação **%s**

O comando de entrada **scanf** realiza a leitura de strings, porém considera o primeiro espaço digitado como fim da string.

Leitura e Impressão de Strings

Exemplo de entrada

```
1  #include<stdio.h>
2  |
3  int main()
4  {
5      char NomeCliente[40];
6      printf("Nome do cliente:");
7      scanf("%s",&NomeCliente);
8      printf("\n\nNome digitado:%s\n",NomeCliente);
9      return 0;
10 }
```

Nome do cliente:Juarez Barbosa Castro

Nome digitado:Juarez

Leitura e Impressão de Strings

Entrada

Para resolver este problema do **scanf** é necessário utilizar um código de formatação específico para ler strings com espaços em branco.

“% [^ \ N] S”

Exemplo:

Leitura e Impressão de Strings

Saída

- ❑ o comando de saída `printf` realiza a impressão de uma string utilizando o código de formatação `%s`.

```
printf("Nome digitado:%s\n",NomeCliente);
```

Leitura e Impressão de Strings

Exemplo de Entrada

```
1  #include <stdio.h>
2  #define T 40
3  int main()
4  {
5      char NomeCliente[T];
6      printf("Nome do Cliente: ");
7      scanf("%[^\n]s", NomeCliente);
8      printf("\nNome digitado: %s\n", NomeCliente);
9      return 0;
10 }
```

```
Nome do cliente:Noeli Pimentel
```

```
Nome digitado:Noeli Pimentel
```

```
-----
Process exited with return value 0
Press any key to continue . . .
```

Exemplo 1

Programa que lê nomes até que seja pressionado somente <ENTER>.

```
#include<stdio.h>
#include<stdlib.h>
# define T 100
int main()
{
    char Nome[T];
    do
    {
        printf("Digite um nome ou <ENTER> para finalizar: ");
        scanf(" %[^\\n]s",Nome);
        if (Nome[0] != '\\0')
        {
            printf("\\nNome digitado: %s\\n",Nome);
            system("Pause");
            system("cls");
        }
        else
        {
            printf("Leitura de nomes finalizada!\\n");
            system("Pause");
        }
    }while(Nome[0] != '\\0');
    return 0;
}
```

Como percorrer uma string para processar seus caracteres

- ❑ Lembre-se que uma string é um vetor de caracteres que tem um código **\0** no seu último elemento.
- ❑ Assim para acessarmos todos os caracteres de uma string precisamos percorrer o vetor enquanto o elemento do vetor for diferente de **\0**

Exemplo 2

Programa que lê uma cadeia de caracteres e conta quantos caracteres foram digitados.

```
#include<stdio.h>
#include <locale.h>
#define T 100
int main()
{
    setlocale(LC_ALL, "Portuguese");
    char Texto[T];
    int ContaCaracteres=0;
    printf("Digite um texto(Máximo 100 caracteres): ");
    scanf(" %[^\\n]s",Texto);
    while (Texto[ContaCaracteres] != '\\0')
    {
        ContaCaracteres++;
    }
    printf("\\n %s possui %d caracteres.",Texto,ContaCaracteres);
    return 0;
}
```

Exemplo 2

Programa que lê uma cadeia de caracteres e conta quantos caracteres foram digitados.

```
#include<stdio.h>
#include <locale.h>
#define T 100
int main()
{
    setlocale(LC_ALL, "Portuguese");
    char Texto[T];
    int ContaCaracteres=0;
    printf("Digite um texto(Máximo 100 caracteres): ");
    scanf(" %[^\\n]s",Texto);
    while (Texto[ContaCaracteres] != '\\0')
    {
        ContaCaracteres++;
    }
    printf("\\n %s possui %d caracteres.",Texto,ContaCaracteres);
    return 0;
}
```

Altere o código fonte para que os espaços sejam desconsiderados.

Exemplo 3

Contagem de caracteres de uma string

```
#include<stdio.h>
#include <locale.h>
#define T 40
int main()
{
    setlocale(LC_ALL, "Portuguese");
    char NomeCliente[T],CaracterProcurado;
    int ContaCaracteres=0,l=0;
    printf("Nome do cliente: ");
    scanf(" %[^\\n]s",NomeCliente);
    printf("Caracter para procurar: ");
    scanf("%c",&CaracterProcurado);
    while (NomeCliente[l] != '\\0')
    {
        if(NomeCliente[l]==CaracterProcurado)
        {
            ContaCaracteres++    ;
        }
        l++;
    }
    printf("\\n O caracter %c aparece %d vezes no nome do cliente.",CaracterProcurado,ContaCaracteres);
    return 0;
}
```

Exemplo 3

Contagem de caracteres de uma string

```
#include<stdio.h>
#include <locale.h>
#define T 40
int main()
{
    setlocale(LC_ALL, "Portuguese");
    char NomeCliente[T],CaracterProcurado;
    int ContaCaracteres=0,l=0;
    printf("Nome do cliente: ");
    scanf(" %[^\\n]s",NomeCliente);
    printf("Caracter para procurar: ");
    scanf("%c",&CaracterProcurado);
    while (NomeCliente[l] != '\\0')
    {
        if(NomeCliente[l]==CaracterProcurado)
        {
            ContaCaracteres++    ;
        }
        l++;
    }
    printf("\\n O caracter %c aparece %d vezes no nome do cliente.",CaracterProcurado,ContaCaracteres);
    return 0;
}
```

Altere o programa para o usuário digitar 2 caracteres e o programa calcular e mostra quantas vezes os dois caracteres aparecem no nome do cliente

Bibliotecas

Existem bibliotecas específicas para manipulação de strings em C:

Strings: string.h

Funções - Manipulação de Strings

- A Linguagem C possui funções especiais para análise e manipulação de **strings**.
- Tais funções estão definidas na biblioteca ***string.h***.
- A biblioteca ***string.h*** possibilita a manipulação de strings completas (sem considerar caractere a caractere).

Manipulação de Strings - Funções

- **strlen()**
 - Retorna o tamanho da string
- **strcat()**
 - Concatena duas strings
- **strcmp()**
 - Compara duas strings
- **strcpy()**
 - Copia uma string em outra

Exemplo 4

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

```
int main ()
{
    char Nome[20];
    int QuantidadeCaracter;
    printf("Digite seu nome : ");
    scanf(" %[^\\n]s",Nome);
    QuantidadeCaracter = strlen(Nome);
    printf ("%s possui %d caracteres.",Nome, QuantidadeCaracter);
    return 0;
}
```

Exemplo 5

```
#include <stdio.h>
#include <string.h>
int main ( )
{
    char Nome[30], Sobrenome[20];
    printf("Digite seu primeiro nome: ");
    scanf(" %[^\\n]s",Nome);
    printf("Digite seu Sobrenome: ");
    scanf(" %[^\\n]s",Sobrenome);
    strcat(Nome, " ");
    strcat(Nome, Sobrenome);
    printf("Seu nome completo e: %s\\n",Nome);
    return 0;
}
```

Exemplo 6

```
#include <stdio.h>  
#include <string.h>  
  
int main ()  
{  
    char Nome1[20], Nome2[20];  
    printf("Digite seu nome:");  
    scanf(" %[^\\n]s",Nome1);  
    strcpy (Nome2, Nome1);  
    printf ("Nome Copiado: %s \\n",Nome2);  
    return 0;  
}
```

Strings - Funções

- **stricmp()**
 - Compara duas strings, com ignore case (não diferencia maiúsculas de minúsculas)
- **strlwr()**
 - Converte todos os caracteres de uma string para minúsculas
- **strupr()**
 - Converte todos os caracteres de uma string para maiúsculas

Strings - Funções

■ strcmp (string1, string2)

- A função **strcmp** compara **string1** e **string2**, verificando se são iguais ou diferentes. A função **strcmp** retorna como resultado :

Retorno	Se
> 0	string1 > string2
0	string1 == string2 (são iguais)
< 0	string1 < string2