String em C

Disciplina: PROGRAMAÇÃO II

Prof. Jean Eduardo Glazar
Curso de Sistemas de Informação
Campus Colatina-ES

INSTITUTO FEDERAL
Espírito Santo

Definição

- **Strings** são **vetores de char** que possuem o char **'\0'** definindo o término da string. Exemplo:

Declaração de Strings

- Como as strings são **vetores de char** podemos declarar strings da seguinte forma:

char <nome_string> [<tamanho>] ;

Onde:

<tamanho> ➔ quantidade máxima de caracteres.

Declaração de Strings

- Podemos também alocar dinamicamente as strings assim como fazemos com qualquer vetor:

```
char *<nome_string>;

<nome_string> = (char *) malloc ( tamanho * sizeof(char) );
```

Uso básico

- Podemos declarar uma string e pré carregá-la com valores:

```
char str[10]  = "João";
```

- Não é permitido declarar uma string e posteriormente carregá-la com valores usando o operador de atribuição:

```
char str[10];
str = "João";
```

ERRO

Uso básico

- Podemos alterar valores de posições da string. Exemplo:

```
int main() {
    char str[10] = "Joao";
    printf("\n\nString: %s", str);
    printf("\nSegunda letra: %c", str[1] );
    str[1] = 'U';
    printf("\nAgora a segunda letra eh: %c", str[1] );
    printf("\n\nString resultante: %s", str);
    return(0);
}
```

Resultado: Será impresso na tela “JUao”.

Funções para manipulação

- **fgets ou gets** → ler uma string do teclado.
- **strcpy** → copia uma string para outra
- **strcat** → concatena duas strings
- **strlen** → retorna o tamanho de uma string
- **strcmp** → compara duas strings
- **strchr** → procura um caracter em uma string
- **strstr** → procura uma string em outra string
- **strupr** → converte para maiúsculas
- **strlwr** → converte para minúsculas

gets

- Lê uma string do teclado:

gets (<nome_da_string>);

Evitem! Pois essa função não limita a quantidade de caracteres lido e pode ocorrer estouro (ou invasão) de memória.

fgets

- Essa função serve para ler uma string do teclado ou de um arquivo, limitada ao tamanho de caracteres:

fgets (<nome_da_string> , <tamanho> , <local>);

<local> → stdin (teclado) ou
nome do arquivo

Os caracteres não lidos ficam na fila de entrada para as próximas leituras.

scanf

- Pode ser usado também o **scanf**, porém ele não lê os espaços em branco:

```
scanf ("%s" , <nome_da_string> );
```

- Para ler os espaços em branco e limitar a quantidade de caracteres:

```
scanf ("%10[^\n]s" , <nome_da_string> );
```

Os caracteres não lidos ficam na fila de entrada para as próximas leituras.

Leituras - exemplo

```
#include <stdio.h>
#include <string.h>
int main () {
        char str1[10];
        gets(str1);
        printf("\n\tGETS: %s\n", str1);

        fgets(str1, 10, stdin);
        printf("\n\tFGETS: %s\n", str1);

        scanf("%9[^\n]s", str1);
        printf("\n\tSCANF: %s\n", str1);
}
```

strcpy

- Copia a **<string_origem>** para a **<string_destino>** e pode ser definida da seguinte forma:

strcpy (<string_destino> , <string_origem>);

- **OBS.:** Deve-se incluir a biblioteca **<string.h>**

strcpy - exemplo

```
#include <stdio.h>
#include <string.h>
int main () {
      char str1[100], str2[100], str3[100];
      printf ("Entre com uma string: ");
      scanf("%99[^\n]s", str1);
      strcpy (str2, str1);   // Copia str1 em str2
      strcpy (str3,"Voce digitou a string ");
      printf ("\n\n%s %s", str3, str2);
}
```

strcat

- Copia a **<string_origem>** para o final da **<string_destino>.**
- Essa operação é chamada de **concatenação** e pode ser definida da seguinte forma:

strcat (<string_destino> , <string_origem>);

- **OBS.:** Deve-se incluir a biblioteca **<string.h>**

strcat - exemplo

```
#include <stdio.h>
#include <string.h>
int main () {
      char str1[100], str2[100];
      printf ("Entre com uma string: ");
      scanf("%99[^\n]s", str1);
      strcpy (str2,"Voce digitou a string ");
      strcat (str2, str1);    // Adiciona str1 em str2
      printf ("\n\n%s", str2);
}
```

strlen

- Retorna o tamanho de uma string, ou seja, a quantidade de caracteres.

int strlen (<string>);

- **OBS. 1:** Deve-se incluir a biblioteca **<string.h>**
- **OBS. 2:** Conta a quantidade de caracteres até encontrar o **'\0'** (final da string).

strlen - exemplo

```
#include <stdio.h>
#include <string.h>
int main () {
      char str[100];
      int tamanho;
      printf ("Entre com uma string: ");
      scanf("%99[^\n]s", str);
      tamanho = strlen (str);
      printf ("A string tem tamanho %d", tamanho );
}
```

strcmp

- Compara a **<string1>** com a **<string2>**.
- Se forem idênticas, a função retorna **0**.
- Se **<string1>** for **maior** que **<string2>**, então retorna um valor **positivo**.
- Se **<string1>** for **menor** que **<string2>**, então retorna um valor **negativo**.

```
int strcmp (<string1> , <string2> );
```

strcmp - exemplo

```
#include <stdio.h>
#include <string.h>
int main () {
        char str1[100], str2[100];
        printf ("Entre com uma string: ");
        scanf("%99[^\n]s", str1);
        printf ("\n\nEntre com outra string: ");
        scanf("%99[^\n]s", str2);
        if ( strcmp(str1, str2) == 0 ) {
                printf ("\n\nAs duas strings são IGUAIS.");
        } else {
                printf ("\n\nAs duas strings são DIFERENTES.");
        }
}
```

strchr

- Retorna um ponteiro para a primeira ocorrência de um caractere **<ch>** em uma **<string>**.

char * strchr (<string> , <ch>);

- **OBS.:** Se o caractere não for encontrado, retorna **NULL**

strchr - exemplo

```
#include <stdio.h>
#include <string.h>
int main () {
      char str[50] = "Exemplo de string para teste";
      char * pch;   // Ponteiro para char
      printf ("Procurando o caractere 'e' na string %s\n", str);
      pch = strchr(str,'e');
      while ( pch != NULL)
      {
            printf ("encontrado na posição %d\n", pch-str+1);
            pch = strchr(pch+1,'e');  // Procura novamente a partir da
                                      //    última posição encontrada
      }
}
```

strstr

- Retorna um ponteiro para a primeira ocorrência da **<string2>** na **<string1>**.

char * strstr (<string1> , <string2>);

- **OBS.:** Se não for encontrado, retorna **NULL**

strstr - exemplo

```
#include <stdio.h>
#include <string.h>
int main () {
      char str[50] = "Exemplo de string para teste";
      if ( strchr(str, 'o') ) {
            printf("A letra 'o' está em %s\n", str);
      }

      if ( strstr(str,"teste") ) {
            printf("A palavra 'teste' foi encontrada em %s\n", str);
      }
}
```

strupr e strlwr - exemplo

```
#include <stdio.h>
#include <string.h>
int main () {
        char str1[10];
        scanf("%9[^\n]s", str1);
        strupr(str1);
        printf("\n\tMaiúsculas: %s\n", str1);

        strlwr(str1);
        printf("\n\tMinúsculas: %s\n", str1);
}
```

Exercícios

1) Ler uma string e imprimir se a mesma é ou não palíndrome. Uma palavra é palíndrome se lendo de frente para trás ou de trás para frente temos exatamente a mesma string.

Exemplos: “mutum” é palíndrome.

“socorram me subi no onibus em marrocos” é palíndrome. Nesse caso deve-se ignorar os espaços em branco.

2) Ler uma data no formato: “dd/mm/aaaa” e imprimir a data no formato americano “aaaa-mm-dd”.

3) Implemente a função contCaractere, que dado uma string e um caractere, determina quantas vezes o caractere aparece na string.

INSTITUTO
FEDERAL
Espírito Santo