Strings

A linguagem C, ao contrário de outras linguagens de programação, não possui um tipo de dados correspondente às strings; no lugar, usam-se vetores .

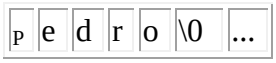
Em C, strings são vetores de caracteres terminados pelo caractere nulo (`'\0'`).

Por exemplo:

```
char nome[] = {'P', 'e', 'd', 'r', 'o', '\0'};
```

No entanto, escrever strings dessa maneira é muito trabalhoso; por isso, foi criada uma notação abreviada que equivale à notação acima e elimina a necessidade de colocar o caractere terminador:

```
char nome[] = "Pedro";
```




```
No caso acima, as duas células não usadas
```

Para se acessar um determinado caracter de uma string, basta "indexarmos", ou seja, usarmos um índice para acessarmos o caracter desejado dentro da string Podemos acessar a **segunda** letra de *nome* da seguinte forma:

```
nome[1] = 'e';
```

****** Nesta unidade não serão tratados caracteres com acentos.**

Lendo e Escrevendo Strings

A biblioteca padrão de E/S, **stdio.h**, fornece duas funções para E/S com strings.

A função **gets()** lê uma string de caracteres do teclado e a coloca no endereço apontado pelo argumento passado.
Os caracteres digitados são transferidos para a memória após um **Enter**.
O **Enter** não se torna parte da string; em seu lugar é colocado o caracter **nulo (‘\0’)**.

gets(str)

Neste protótipo **str** é um vetor de caracteres que recebe os caracteres digitados pelo usuário.

A função **puts()** escreve o seu argumento (uma string) na tela, seguida por um caracter de nova linha.

puts(str);

A função **puts()** reconhece os mesmos caracteres de controle que a função **printf()** (por exemplo, **'\n'**).
puts() pode escrever apenas strings de caracteres, não podendo escrever números ou fazer conversões de formato.

Os comandos a seguir reexibem uma frase lida no monitor.

```
char str[81];
gets(str);
puts(str);
```

A função **scanf()** pode ser utilizada para ler uma string usando o especificador de formato **%s**. O **%s** faz com que **scanf()** leia caracteres até que seja encontrado **um caracter de espaço em branco**.

Outra alternativa

```
int main(){
...
scanf("%[^\n]", str);
...
}
```

A instrução [^\n] diz ao comando scanf() para ler tudo até encontrar retorno de carro (ENTER)

IMPORTANTE *** Os caracteres lidos são colocados em um vetor de caracteres apontado pelo argumento correspondente, e o resultado tem terminação nula **('\0')**.
Para **scanf()**, um caracter de espaço em branco é:
um espaço, ou um retorno de carro (**Enter**), ou uma tabulação.

Logo, uma string como **Alo Mundo!** não pode ser lido com **scanf()** (apenas a substring **Alo** será carregado no vetor).

A função **printf()** escreve uma string através do **%s**. Ao contrário da função **puts()**, não existe mudança automática de linha na exibição de uma string no vídeo.

Funções da biblioteca padrão

A biblioteca padrão fornece várias funções úteis para manipular strings. A seguir mostraremos algumas delas. Para usá-las, você deve incluir o cabeçalho `string.h` no início dos seus arquivos.

1 - strlen

`strlen` retorna o tamanho, em caracteres, de uma string dada. Na verdade o strlen() procura o terminador de string e calcula a distância dele ao início da string. Por exemplo:

```
char nome[] = "Maria da Silva";

int s = strlen (nome);

// s conterá o valor 14
```

2 - strcpy

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

Sua sintaxe é :

```
   strcpy (destino, origem)

char nome[]  = "Clarice Lispector";
char nome2[] = "Oswald de Andrade";
strcpy (nome, nome2);
// agora nome conterá "Oswald de Andrade"
```

Pode-se usar a função **strncpy**, que recebe um terceiro argumento que corresponde ao número máximo de caracteres a serem copiados:

```
char msg[] = "Bom dia!";
char nome[] = "Maria da Silva";
strncpy (msg, nome, strlen(msg));
// agora msg conterá "Maria da"
```

4 - strcmp

Se você tentar criar duas strings com o mesmo conteúdo e compará-las como faria como números, verá que elas "não são iguais". Isso ocorre porque, na verdade, o que está sendo comparado são *os endereços de memória* onde estão guardadas as strings. Para comparar *o conteúdo* de duas strings, você deve usar a função **strcmp** (ou suas variantes):

strcmp (s1 , s2);

O valor de retorno é:

- menor que zero se *s1* for menor que *s2*;
- igual a zero se *s1* e *s2* são iguais;
- maior que zero se *s1* for maior que *s2*.

Costuma parecer estranho dizer que uma string é *menor* ou *maior* que outra; na verdade essa comparação é entre a primeira letra que difere nas duas strings. Assim, se tivermos `s1 = "abc"` e `s2 = "abd"`, diremos que *s2* é **maior** que *s1*, pois na primeira posição em que as duas strings diferem, a letra em *s2* é "maior".

É importante notar que a comparação feita por `strcmp` distingue maiúsculas de minúsculas. Isto é, as strings "ABC" e "abc" **não** são iguais para essa função.

Algumas funções úteis

B - Quadrado de um valor

```
pow(x,2)
{
int x=3;
printf("%f",pow(x,2));
```

C- Converte caracter para maiusculo

toupper(nome[x])

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <ctype.h>
 int main()
{
   char palavra[5];
   int i;

   printf("Digite uma palavra: ");
   gets(palavra);

   for(i=0; i<5; i++)
   palavra[i]=toupper(palavra[i]);

   printf(" palavra convertida em maiusculo: %s",palavra);

      getch();}
```

```
/* Convertendo uma string em minúsculas
* usando a função strlwr() */

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

int main()
  {
    char string[20];

    printf("\n");
    printf("Convertendo uma string para minúsculas\n");
    printf("-------------------------------------\n");
    printf("\n");
    printf("Entre com a string :");
    gets(string);
    printf("\n");
    printf("string digitada : %s\n",string);
    printf("\n");
    printf("Convertendo para minúsculas : %s\n",strlwr(string));

 }
```

/* Convertendo uma string em maiúsculas -OK

 * usando a função strupr() */

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

int main()
  {
    char string[20];

    printf("\n");
    printf("Convertendo uma string para maiúsculas\n");
    printf("--------------------------------------\n");
    printf("\n");
    printf("Entre com a string :");
    gets(string);
    printf("\n");
    printf("string digitada : %s\n",string);
    printf("\n");
    printf("Convertendo para maiúsculas : %s\n",strupr(string));
    return(0);

 }
```