

Strings

- [gets](#)
- [strcpy](#)
- [strcat](#)
- [strlen](#)
- [strcmp](#)

Strings são vectores de [chars](#) (caracteres). Nada mais e nada menos. As strings são o uso mais comum para os vectores. Devemos apenas ficar sempre atentos para o facto de que as strings têm como último elemento um '\0'. A declaração geral para uma string é:

```
char nome_da_string [tamanho];
```

Devemos lembrar que o tamanho da string deve incluir o '\0' final. A biblioteca do C possui diversas funções que manipulam strings. Estas funções são úteis pois não se pode, por exemplo, igualar duas strings:

```
string1=string2;          /* NAO faca isto */
```

Ao fazer-mos programas que tratam de strings muitas vezes podemos fazer bom proveito do facto de que uma string termina com '\0'. Veja, por exemplo, o programa abaixo:

Este algoritmo serve para igualar duas strings (isto é, copia os caracteres de uma string para o vector da outra):

```
#include <stdio.h>
main ()
{
  int count;
  char str1[100],str2[100];
  printf("digite uma string: ");
  gets(str1);

  for (count=0;str1[count];count++)
    {
      str2[count]=str1[count];
    }
  str2[count]='\0';

  printf("string original  -- %s",str1);
  printf("\n cópia da string -- %s",str2);
  scanf("%d");
}
```

A condição no loop [for](#) acima é baseada no facto de que a string que está a ser copiada termina em '\0'. Este tipo de raciocínio é a base do C. Quando o elemento encontrado em **str1[count]** é o '\0', o valor retornado para o teste condicional é falso (nulo). Desta forma a expressão que vinha sendo verdadeira (não zero) continuamente, torna-se falsa.

FUNÇÕES BÁSICAS PARA MANIPULAÇÃO DE STRINGS.

gets

A função **gets()** lê uma string do teclado. Sua forma geral é:

```
gets (nome_da_string);
```

O programa abaixo demonstra o funcionamento da função **gets()**:

```
#include <stdio.h>
main ()
{
char string[100];
printf ("Digite o seu nome: ");
gets (string);
printf ("\n\n olá %s",string);
}
```

strcpy

A sua forma geral é:

```
strcpy (string_destino,string_origem);
```

A função **strcpy()** copia a string-origem para a string- destino. O seu funcionamento é semelhante ao da rotina apresentada na [secção anterior](#). As funções apresentadas nestas sessões estão na biblioteca do C **string.h** como tal é necessário chamar a biblioteca no início do programa.

A seguir apresentamos um exemplo de uso da função **strcpy()**:

```
#include <stdio.h>
#include <string.h>
main ()
{
char str1[100],str2[100],str3[100];
printf ("Digite uma string: \n");
gets (str1);
strcpy (str2,str1);
strcpy (str3,"de uma string ao vector");

printf ("\n String original  --  str1 = %s",str1);
printf ("\n Cópia da string 1 -- str2= %s",str2);

printf ("\n Atribuição directa str3= %s",str3);
scanf ("%d");
}
```

Neste exemplo inicializamos 3 vectores com 100 posições cada um. No primeiro vector (**str1**) vamos colocar a string digitada pelo utilizador, no segundo vector (**str2**) colocamos o valor do vector um (**str1**), por fim no vector 3 (**str3**) atribuímos um valor directamente, no caso: *"de uma string ao vector"*.

strcat

A função **strcat()** tem a seguinte forma geral:

strcat (string_destino,string_origem);

A string de origem permanecerá inalterada e será anexada ao fim da string de destino.

Um exemplo:

<pre>#include <stdio.h> #include <string.h> main () { char str1[100],str2[100]; printf ("Digite uma string: "); gets (str1); strcpy (str2,"digitou a string"); strcpy (str2,str1); printf ("\n\n%s",str2); scanf ("%d"); }</pre>	<pre>#include <stdio.h> #include <string.h> main () { char str1[100],str2[100]; printf ("Digite uma string: "); gets (str1); strcpy (str2,"digitou a string"); strcat (str2,str1); printf ("\n\n%s",str2); scanf ("%d"); }</pre>
---	---

Repare na diferença entre os dois programas, que apenas existe nas linhas a negrito, e mostra a diferença entre os dois programas.

No primeiro o output será aquilo que utilizador digitar porque ao utilizar o **strcpy** apagamos o conteúdo do vector um.

No segundo o output será: *digitou a string* mais aquilo que o utilizador digitar isto porque o **strcat** não apaga o conteúdo do vector de destino mas acrescenta a informação do vector de origem.

strlen

A sua forma geral é:

strlen (string);

A função **strlen()** retorna o comprimento da string fornecida. O terminador nulo não é contado. Isto quer dizer que, de facto, o comprimento do **vector** da string deve ser um a mais que o inteiro retornado por **strlen()**. Um exemplo do seu uso:

```
#include <stdio.h>
#include <string.h>
main ()
{
int size;
char str[100];
printf ("Digite com uma string: ");
gets (str);
size=strlen (str);
printf ("\n\nA string que digitou tem tamanho %d",size);
}
```

strcmp

A sua forma geral é:

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

A função **strcmp()** compara a string 1 com a string 2. Se as duas forem idênticas a função retorna zero. Se elas forem diferentes a função retorna não-zero. Um exemplo da sua utilização:

```
#include <stdio.h>
#include <string.h>
main ()
{
char str1[100],str2[100];
printf ("Digite uma string: ");
gets (str1);
printf ("\n\nDigite outra string: ");
gets (str2);
if (strcmp(str1,str2))
    printf ("\n\nAs duas strings são diferentes.");
else printf ("\n\nAs duas strings são iguais.");
}
```

Resumindo:

→ O **gets** já nosso conhecido lê a string digitada pelo utilizador tal como faz o scanf.

→ O **strcpy** e o **strcat** - atribui o valor de uma string a outra, com a diferença que o strcpy apaga o conteúdo da string de destino e o strcat não.
Sendo a forma geral destes dois comandos a mesma:
(string_destino,string_origem);

→ O **strlen** diz-nos o tamanho de uma string.

→ O **strcmp** compara duas strings.

Exercício 1

Faça um programa que leia duas strings digitadas pelo utilizador, e armazene cada string num vector. Depois, concatene as strings lidas numa única string. Por fim apresente esta como resultado ao final do programa indicando o seu tamanho. Nota as strings devem ter um espaço entre elas.

```
#include <stdio.h>
#include <string.h>
main ()
{
char str1[100],str2[100];
int tamanho;
printf ("Digite uma string: ");
gets (str1 );
printf ("\n\nDigite outra string: ");
gets (str2);

strcat(str1," "); /*apenas serve para colocar um espaço entre strings*/

strcat(str1,str2);
printf("%s ",str1);
tamanho =strlen (str1);
printf(" \na string digitada tem %d caracteres",tamanho);
scanf("%d");
}
```

Exercício 3

Refaça o programa da página anterior. Use o comando while para fechar o loop

```
#include <stdio.h>
#define MAX 31
int main()
{
char str[MAX], copia[MAX];
int i = 0, compstr;
printf("\n\n Digite uma palavra (max 30 caracteres): ");
gets(str);
/* Determina o comprimento da string atraves
de um for sem conteudo : no final dele, compstr
contem a posicao do '\0' da string
é bem mais facil fazer isto com o strlen*/
for(compstr=0; str[compstr]; compstr++);

while (str[i] != '\0')
{
copia[i] = str[compstr-i-1]; /* Linha em destaque */
i++;
}
copia[i] = '\0';
printf("\n\nString Invertida: %s\n", copia);
scanf("%d");
}
```

Exercício 2

[Página c430.html](#)

Faça um programa que inverta uma string: leia a string com gets e armazene-a invertida numa outra string. Use o comando for para varrer a string até o seu final.

```
#include <stdio.h>
#define MAX 31
int main()
{
char str[MAX], copia[MAX];
int i, compstr;
printf("\n\n Digite uma string (max 30 caracteres): ");
gets(str);
/* Determina o comprimento da string: */
compstr=strlen(str);
/* Inverte a string */
for(i=0; str[i]; i++)
{
copia[i]=str[compstr-i-1]; /* Linha em destaque */
}
copia[i] = '\0'; /* finaliza a string */
printf("\n\nString Invertida: %s\n", copia);
printf("%d",compstr);
scanf("%d");
}
```

Neste exercício existem alguns detalhes interessantes. Em primeiro lugar, é necessário determinar o tamanho da string. Tarefa que nos fica facilitada com o uso do strlen.

Lembre-se sempre que se temos uma string de 10 posições, o seu índice pode variar entre 0 a 9, e o '\0' vai estar na posição 9. Logo, devemos inverter a posição 8 com a 0, a 7 com a 1, e assim por diante. É por isso que a expressão fica:

```
copia[i] = str[compstr-i-1]
```

Não podemos também deixar de colocar o '\0' no final da nova string.

Traçagem da linha em destaque

```
for(i=0; str[i]; i++)
{
copia[i]=str[compstr-i-1]; /* Linha em destaque */
}
copia[i] = '\0';
```

	Z	I	T	A	/0
Posição	0	1	2	3	4
compstr	4-0-1=3	4-1-1=2	4-2-1=1	4-3-1=0	/0
Copia	A	T	I	Z	/0