Strings em C

A linguagem C não possui um tipo específico de dados Strings. Para fazer uma string, o C utiliza um vetor de caracteres, onde cada posição do vetor representa uma letra. É importante lembrar que a linguagem C identifica o fim de uma cadeia por meio do caracter nulo (\0). Sendo assim, para termos uma string, sempre temos que ter uma posição a mais de tamanho no vetor para este caracter no final. Por exemplo, para armazenarmos a palavra CADEIA, temos que declarar um vetor do tipo *char* com sete posições, e elas ocuparão posições seqüenciais na memória.

char palavra [7];

Índice	0	1	2	3	4	5	6
Valor	C	A	D	E	I	A	\0
memória	863	864	865	866	867	868	869

Printf(“%c”,palavra[3]);
Na tela apareceria a letra **E**

Manipulação de Strings

Para trabalharmos com esses vetores especiais que chamamos de Strings precisaremos incluir a biblioteca **string.h**

strcpy (str1, str2)

Esta função é utilizada para copiar o conteúdo de uma string em outra. A primeira string terá o mesmo valor da segunda string. Podemos também colocar uma string qualquer entre aspas ao invés de uma variável no lugar de str2.

Importante lembrar que o tamanho de str2 deve ter no máximo o mesmo tamanho de str1. Str2 pode ser menor, nunca maior que str1. Mesmo no caso de colocarmos uma string manualmente (fazer uma cópia sem utilizarmos str2 como demonstrado no exemplo abaixo) não podemos ultrapassar o tamanho de str1 menos 1.

Exemplo

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

int main()
{
   char str1[40],str2[40];
   strcpy(str2,"TESTE INICIAL ");
   strcpy(str1,str2);
   printf("%s",str1);
   printf("\n%s",str2);
return 0;
}
```

Resultado na tela: TESTE INICIAL
TESTE INICIAL

strcat (str1,str2)

Esta função é utilizada para concatenar (unir / juntar) duas strings. A segunda string será adicionada no final da primeira string indicada.

Lembre-se que a soma dos valores de caracteres da str1 + str2 não podem exceder o tamanho da str1. Podemos também substituir str2 por um conjunto de caracteres manualmente, como no exemplo do strcpy.

Exemplo

```
#include<stdio.h>
#include<string.h>
int main()
{
  char str1[40],str2[10];
  strcpy(str1,"TESTE INICIAL ");
  strcpy(str2,"teste 2");
  strcat(str1,str2);
  printf("%s",str1);
return 0;
}
```

Resultado na tela: TESTE INICIAL teste 2

strchr (str1,ch)

Esta função é utilizada para procurar a posição da primeira ocorrência do caracter ch em uma string. Ou seja, a função retorna qual posição dentro de uma string a letra especificada em ch se encontra (a primeira ocorrência caso haja repetições). A função retorna um ponteiro para a posição de memória. Para termos o valor exato, precisamos subtrair o valor da string multiplicado por -1. Veja o exemplo:

```
#include<stdio.h>
#include<string.h>
int main()
{
  char str1[50],ch;
  strcpy(str1,"Procura uma letra inicial de caracteres");
  ch='l';
  printf("%d",-(str1-strchr(str1,ch)));
return 0;
}
```

Resultado na tela: 12

strcmp (str1,str2)

Esta função é utilizada para comparar se o conteúdo de str2 é igual ao conteúdo de str1. Nesse caso, a função retorna o valor 0 (zero) se as duas cadeias forem iguais, um valor menor que zero se str1 for alfabeticamente menor que str2 ou um valor maior que

zero se str1 for alfabeticamente maior que str2. Esta função diferencia maiúsculas de minúsculas.

Exemplo:

```
#include<stdio.h>
#include<string.h>
int main()
{
  char str1[50],str2[50],str3[50];
  strcpy(str1,"Procura uma letra inicial de caracteres");
  strcpy(str2,"Procura uma letra inicial de caracteres");
  strcpy(str3,"Cadeia diferente");
  printf("%d\n",strcmp(str1,str2));
  printf("%d\n",strcmp(str1,str3));
  printf("%d\n",strcmp(str2,str3));
  printf("%d\n",strcmp(str3,str1));
  printf("%d\n",strcmp(str2,str1));
return 0;
}
```

Resultado na tela:
```
0
13
13
-13
0
```

stricmp (str1,str2)

Esta função é utilizada para comparar se o conteúdo de str2 é igual ao conteúdo de str1. Nesse caso, a função retorna o valor 0 (zero) se as duas cadeias forem iguais, um valor menor que zero se str1 for alfabeticamente menor que str2 ou um valor maior que zero se str1 for alfabeticamente maior que str2. Esta função **não** diferencia maiúsculas de minúsculas.

Exemplo:

```
#include<stdio.h>
#include<string.h>
int main()
{
  char str1[50],str2[50],str3[50];
  strcpy(str1,"Procura uma letra inicial de caracteres");
  strcpy(str2,"PROCURA UMA LETRA INICIAL DE CARACTRES");
  strcpy(str3,"Cadeia diferente");
  printf("%d\n",stricmp(str1,str2));
  printf("%d\n",stricmp(str1,str3));
  printf("%d\n",stricmp(str2,str3));
  printf("%d\n",stricmp(str3,str1));
  printf("%d\n",stricmp(str2,str1));
return 0;
}
```

Resultado na tela:
```
0
13
```

```
13
-13
0
```

strlen (str1)

Esta função retorna o tamanho (quantidade de letras) de uma string, desprezando o caractere nulo final (\0). Ela retorna o valor exato de caracteres.

Exemplo:

```
#include<stdio.h>
#include<string.h>
int main()
{
  char str1[50];
  strcpy(str1,"Procura uma letra inicial de caracteres");
  printf("%d",strlen(str1));
}
```

Resultado na tela: 39

strstr (str1,str2)

Esta função retorna um ponteiro que aponta para uma string dentro de uma string. Em outras palavras, ela retorna a posição de memória em uma variável de memória que mostra onde começa uma possível repetição de str2 dentro de str1. Se não houver repetição, ela retorna o caracter nulo (\0)

Exemplo:

```
#include<stdio.h>
#include<string.h>
int main()
{
   char str1[40],str2[40];
   char *ptr;
   strcpy(str1,"Eu adoro programar em C");
   strcpy(str2,"programar");
   ptr = strstr(str1, str2);
   printf("A repeticao e: %s\n", ptr);
}
```

Observação: A matéria de ponteiros não será abordada neste ponto da matéria.

Inicialização de Strings

As variáveis que armazenam seqüências de caracteres (strings) em C devem ser inicializadas de forma diferente das variáveis comuns, como int, float, etc. (onde utilizamos apenas o sinal de = para sua inicialização).

Os tipos particulares de inicializações de Strings são:

Inicialização por meio de atribuição

Esta é a forma tradicional de inicialização, que foi demonstrada nos exemplos acima, onde utilizamos a função **strcpy** para atribuir o valor inicial a uma string. Por exemplo:

```
char vet1[10], vet2[10];

strcpy (vet1,"String1"); ou
strcpy (vet2,vet1);
```

Inicialização no momento da declaração

Com esse método de inicialização, podemos declarar uma string sem preocuparmos com seu tamanho, atribuindo uma quantidade de caracteres que serão utilizadas no vetor. Lembrando que, após esse tipo de atribuição, para o resto do programa a variável terá sempre o tamanho fixo do número de letras atribuído.

Por exemplo:

```
char nome [ ] = { 'S', 'e', 'r', 'g', 'i', 'o', ' ', 'P', 'o', 'r', 't', 'a', 'r', 'i','\0' };
ou
char nome[ ] = "Sergio Portari";
```

Nos dois casos, após a inicialização, a variável nome possui o tamanho 14.

Inicialização por meio do teclado

Essa inicialização é feita pelo clássico scanf utilizado até hoje em nossos exercícios, onde o usuário utiliza o teclado para informar quais serão os caracteres que estarão armazenados.

```
char nome[40]
...
printf("Digite o nome: ");
scanf("%s",&nome);
```

Podemos ainda utilizar a função **gets** da biblioteca string ao invés do scanf.

```
char nome[40]
...
```

```
printf("Digite o nome: ");
gets(nome);
```

Observação: Para imprimirmos o conteúdo de uma string podemos utilizar a função **puts** da biblioteca string ao invés do printf. Exemplo

```
puts(nome);
```

mostra o conteúdo da string nome na tela, equivalente a printf("%s",nome);

Exercícios Strings

1) Faça um programa em C que receba uma frase qualquer fornecida pelo usuário, calcule e mostre quantas palavras a frase possui.

2) Faça um programa em C que receba uma frase e troque as vogais existentes nesta frase por um asterisco (*). Por exemplo: Frase "EU ESTOU NA ESCOLA" resultado na tela "** *ST** N* *SC*L*"

3) Faça um programa em C que se comporte como um vírus. Este programa irá duplicar as palavras digitadas em uma frase.

4) Faça um programa em C que receba uma frase do usuário e mostre a frase, palavra por palavra, uma em cada linha diferente.

5) Faça um programa em C que receba uma frase, inverta a frase letra a letra, da última para a primeira, e mostre esta frase ao final.