MANIPULAÇÃO DE CADEIA DE CARACTERES (STRING)

A linguagem C não possui um tipo específico de dados `strings`. Para fazer uma `string`, a linguagem 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 caractere nulo (`'\0'`). Sendo assim, para ter uma `string`, sempre tem que ter uma posição a mais de tamanho no vetor para este caractere no final.

Exemplo: Para armazenar a palavra `CADEIA`, temos que declarar um vetor do tipo `char` com sete posições, e elas ocuparão posições sequenciais na memória.

```
char palavra [7];
```

palavra	C	A	D	E	I	A	\0
	0	1	2	3	4	5	6

```
printf ("%c",palavra[2]);
```

- **Manipulação de strings**

Para trabalhar com esses vetores especiais que chamamos de `strings é necessário` incluir a biblioteca `string.h`

Função `strcpy()`

A função `strcpy()` é utilizada para copiar o conteúdo de uma `string` em outra. A primeira `string` terá o mesmo valor da segunda `string`. Pode-se também colocar uma `string` qualquer entre aspas ao invés de uma variável no lugar de `str2`.

Sintaxe: `strcpy(str1, 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`.

Exemplo:

```
#include<stdio.h>
#include<string.h>
int main() {
  char str1[40], str2[40];
  strcpy (str2,"ALGORITMOS I");
  strcpy(str1,str2);
  printf("%s",str1);
  printf("\n%s",str2);
}
```

Obs: Mesmo no caso de colocar uma `string` manualmente (fazer uma cópia sem utilizar `str2` como demonstrado no exemplo anterior) não pode-se ultrapassar o tamanho de `str1` menos 1.

Exemplo:

```
#include<stdio.h>
#include<string.h>
int main() {
  char str1[40];
  strcpy (str1,"ALGORITMOS I");
  printf("%s",str1);
}
```

Função `strlen()`

A função `strlen()` retorna o tamanho (quantidade de caracteres) de uma `string`, desprezando o caractere nulo final (`'\0'`). Ela retorna o número exato de caracteres.

Sintaxe: `<var> = strlen(str);`

Exemplo:

```
#include<stdio.h>
#include<string.h>
int main() {
  char str[50], ch;
  strcpy(str,"Tamanho de caracteres da frase.");
  printf("%d\n\n",strlen(str));
}
```

Função `strcat()`

A função `strcat()` é utilizada para concatenar (unir / juntar) duas `strings`. A segunda `string` será adicionada no final da primeira `string`. Lembre-se que a soma dos valores de caracteres da `str1 + str2` não podem exceder o tamanho da `str1`.

Sintaxe: `strcat(str1, str2);`

Exemplo:

```
#include<stdio.h>
#include<string.h>
int main() {
  char str1[40], str2[40];
  strcpy (str1,"ALGORITMOS I");
  strcpy (str2," - BSI");
  strcat (str1,str2);
  printf("%s",str1);
}
```

Obs: Pode-se também substituir `str2` por um conjunto de caracteres manualmente, como no exemplo do `strcpy`.

Função `strchr()`

A função `strchr()` é utilizada para procurar a posição da primeira ocorrência do caractere uma `string`.

Sintaxe: `<var> = strchr(str, ch);`

Assim, a função `strchr` retorna qual posição dentro de uma `string` o caractere especificado em `ch` se encontra (a primeira ocorrência caso haja repetições). A função retorna um ponteiro para a posição de memória. Para obter o valor exato, é necessário subtrair o valor da `string` multiplicado por -1.

Exemplo:

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

Função `strcmp()`

Sintaxe: `<var> = strcmp(str1, str2);`

A função `strcmp()` é utilizada para comparar se o conteúdo de `str2` é igual ao conteúdo de `str1`. Nesse caso, a função `strcmp()` 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`. A função `strcmp()` diferencia caracteres maiúsculos de minúsculos.

Exemplo:

```
#include<stdio.h>
#include<string.h>
int main() {
  char str1[50],str2[50],str3[50];
  strcpy (str1,"Sequencia de caracteres.");
  strcpy (str2,"Sequencia de caracteres.");
  strcpy (str3,"Cadeia de caracteres diferente.");
  printf ("str1: %s\n",str1);
  printf ("str2: %s\n",str2);
  printf ("str3: %s\n\n",str3);
  printf ("str1 e str2: %d\n",strcmp(str1,str2));
  printf ("str1 e str3: %d\n",strcmp(str1,str3));
  printf ("str2 e str3: %d\n",strcmp(str2,str3));
  printf ("str3 e str1: %d\n",strcmp(str3,str1));
  printf ("str2 e str1: %d\n",strcmp(str2,str1));
}
```

Função `stricmp()`

Sintaxe: `<var> = stricmp(str1, str2);`

A função `stricmp()` é utilizada para comparar se o conteúdo de `str2` é igual ao conteúdo de `str1`. Nesse caso, a função `stricmp()` 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`. A função `stricmp()` **não** diferencia caracteres maiúsculos de minúsculos.

Exemplo:

```
#include<stdio.h>
#include<string.h>
int main() {
  char str1[50],str2[50],str3[50];
  strcpy (str1,"Sequencia de caracteres.");
  strcpy (str2,"SEQUENCIA de CARACTERES.");
  strcpy (str3,"Cadeia de caracteres diferente.");
  printf ("str1: %s\n",str1);
  printf ("str2: %s\n",str2);
  printf ("str3: %s\n\n",str3);
  printf ("str1 e str2: %d\n",stricmp(str1,str2));
  printf ("str1 e str3: %d\n",stricmp(str1,str3));
  printf ("str2 e str3: %d\n",stricmp(str2,str3));
  printf ("str3 e str1: %d\n",stricmp(str3,str1));
  printf ("str2 e str1: %d\n",stricmp(str2,str1));
}
```

Iniciação de cadeias de caracteres (strings)

As variáveis que armazenam sequências de caracteres (`strings`) na linguagem C devem ser iniciadas de forma diferente das variáveis comuns, como `int`, `float`, etc. (onde utiliza-se apenas o sinal de `=` para sua iniciação).

Os tipos particulares de inicializações de `strings` são:

- **Iniciação por meio de atribuição**

Esta é a forma tradicional de iniciação, onde utiliza-se a função `strcpy()` para atribuir o valor inicial a uma `string`.

Exemplo:

```
char vet1[10], vet2[10];
strcpy (vet1,"String1");
strcpy (vet2,vet1);
```

- **Iniciação no momento da declaração**

Com esse método de iniciação, pode-se declarar uma `string` sem se preocupar 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 caracteres atribuído.

Exemplo:

```
char nome[ ]={'A','l','g',' ','I','\0' };
```

ou

```
char nome[ ] = "Alg I";
```

- **Iniciação por meio do teclado**

Iniciação feita através do `scanf()` ou, através da função `gets()` da biblioteca `string.h`.

Na leitura usando a função scanf() deve-se utilizar %s e, a leitura é realizada até encontrar um espaço, depois encerra a leitura e coloca o caractere terminador `'\0'`. A variável que vai armazenar a `string` não necessita ser precedida por `&`.

A função `gets()` armazena tudo que foi digitado, inclusive os espaços, até que a tecla `ENTER` seja pressionada.

Exemplo:

```
#include<stdio.h>
#include<string.h>
int main() {
  char nome[30];
  printf ("\nDigite seu nome: ");
  gets (nome);
  printf ("\nSeu nome: %s\n",nome);
}
```

Observação: Para imprimir o conteúdo de uma `string` podemos utilizar a função `puts` da biblioteca `string.h` ao invés do `printf`.

Exemplo:

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

Função `atoi()`

Sintaxe: `<var> = atoi (str);`

A função `atoi()` recebe uma `string` que representa um número inteiro em notação decimal e converte essa `string` no número inteiro correspondente. É responsabilidade do usuário garantir que o número representado pela `string` pertence ao intervalo fechado `[INT_MIN..INT_MAX]`.

Exemplo:

```
int main() {
  char str[10];
  int num;
  printf ("Digite uma string(numero): ");
  gets(str);
  num = atoi(str)+10;
  printf ("Numero fornecido acrescido de 10 unidades: %d",num);
}
```

Função `itoa()`

Sintaxe: `<var> = itoa (num,str,base);`

A função `itoa()` recebe um número inteiro `num` e o converte em `string` na `base` solicitada.

Exemplo:

```
int main() {
  char str[10];
  int num;
  printf ("Digite um numero: ");
  scanf("%d",&num);
  itoa (num,str,2);  // base binária
  printf ("Valor binário: %s",str);
  itoa (num,str,10);  // base decimal
  printf ("Valor decimal: %s",str);
  itoa (num,str,16);  // base hexadecimal
  printf ("Valor hexadecimal: %s",str);
}
```

Exercícios

1) Faça um programa que leia cinco cadeias de caracteres e as exibe no monitor.

```
#include <stdio.h>
#include <stdlib.h>
int main () {
  char texto[5][100];
  int i;
  for (i = 0; i < 5; i++) {
    printf ("\nDigite uma cadeia de caracteres: ");
    gets (texto[i]);
  }
  printf ("\n\nAs cadeias de caracteres digitadas foram:\n");
  for (i = 0; i < 5; i++)
    printf ("%s\n",texto[i]);
  system ("pause");
}
```

2) Escreva uma função que classifique um conjunto alfanumérico em ordem crescente.

```
#include <stdio.h>
#include <string.h>
void troca (char n1[], char n2 []) {
  char aux[10];
  strcpy (aux,n1);
  strcpy (n1,n2);
  strcpy (n2,aux);
}
void BubbleSort (int n, char v[][10]) {
  int i, j, aux;
```

```
  int trocado = 1;
  for (i=0; i<n-1 && trocado; i++) {
    trocado = 0;
    for (j=0; j<n-i-1; j++) {
      if (stricmp(v[j],v[j+1])>0) {
        trocado = 1;
        troca (v[j], v[j+1]);
      }
    }
  }
}
int main () {
  char nomes[5][10]={"bbbbb","CCCCC","aaaaa","eeeeee","dddddd"};
  int i;
  for (i = 0; i < 5; i++)
    printf ("%s  ",nomes[i]);
  BubbleSort (5,nomes);
  printf ("\n");
  for (i = 0; i < 5; i++)
    printf ("%s  ",nomes[i]);
}
```

3) Faça um programa que conte o número de vogais de um texto fornecido pelo usuário.

```
#include <stdio.h>
#include <stdlib.h>
int contaVogais(char s[]) {
  int i, j, numVogais = 0;
  char vogais[] = "aeiouAEIOU";
  for (i = 0; s[i] != '\0'; ++i) {
     for (j = 0; vogais[j] != '\0'; ++j)
     if (vogais[j] == s[i]) {
       numVogais += 1;
       break;
     }
  }
  return numVogais;
}
int main () {
  char texto[20];
  gets (texto);
  printf    ("\n\nNumero    de    vogais    no    texto    \"%s\":
%d.\n",texto,contaVogais(texto));
  system ("pause");
}
```

4) Escreva uma função que receba uma *string* e imprima uma tabela com o número de ocorrências de cada caractere na *string*. Escreva um programa para testar a função.

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
void mostraCaracteres (char texto[]) {
  int i, j, achou, cont;
  for (i = 0; i < strlen(texto); i++) {
```

```
      achou = 0;
      for (j = 0; j < i && !achou; j++)
        if (toupper(texto[i]) == toupper(texto[j]))
          achou = 1;
      if (!achou) {
        cont = 0;
        for (j = i; j < strlen(texto); j++)
          if (toupper(texto[i]) == toupper(texto[j]))
            cont++;
        printf ("%c - %d vez(es)\n",texto[i],cont);
      }
    }
  }
  int main () {
    char texto[20];
    printf ("Digite um texto: ");
    gets (texto);
    mostraCaracteres (texto);
    printf ("\n\n");
  }
```