UNIVERSIDADE SÃO JUDAS TADEU

ESTRUTURA DE DADOS - PROF. H. Senger

1 CADEIAS DE CARACTERES

As cadeias de caracteres são estruturas de dados muito comuns em diversas linguagens. Praticamente todas as linguagens de programação possuem cadeias de caracteres. Muitos programadores a chamam de string, ou ainda, de variáveis alfanuméricas.
Na linguagem C, uma string nada mais é do que um vetor de caracteres.
Assim como qualquer vetor, as cadeias de caracteres também precisam Ter um tamanho máximo pré-definido, como no exemplo abaixo :

char palavra[10];

É obrigatório declarar o tamanho da cadeia de caracteres, pois o compilador precisa saber quantos bytes devem ser reservados, para armazenar a string.

1.1 Representação interna dos dados

Sabemos que cada caracter armazenado necessita de um byte para armazenar o seu código ASCII.
Exemplo:

char palavra[10]= "ANA";

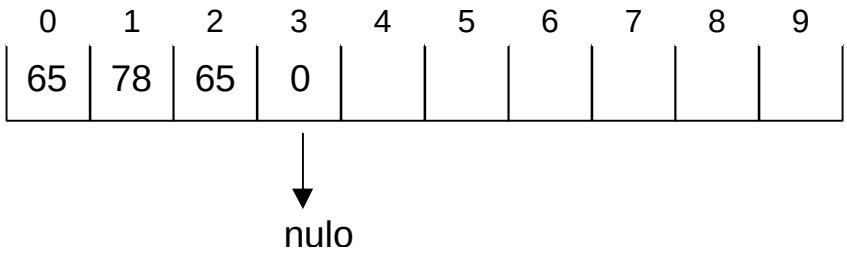
0	1	2	3	4	5	6	7	8	9
A	N	A	\0						

Note que existe um caracter **nulo**, simbolizado pelo "\0" (contra-barra zero), indicando o fim da cadeia. Isto significa que o conteúdo que está nas posições de 4 até 9 é lixo, ou seja não fazem parte da string, e portanto, não devem ser considerados.
Na verdade, lembre-se de que os caracteres são armazenados na memória do computador, através de seu código ASCII :

Código 32 = ' ' (espaço)
Código 65 = 'A'
Código 66 = 'B'
. . .
Código 90 = 'Z'
. . .
Código 97 = 'a'

Código 98 = ‘b’

Assim, se dermos uma olhada dentro da cadeia da variável **palavra**, utilizada no exemplo anterior, veremos que o nome “**ANA”** será armazenado da seguinte forma:



1.2 Funções para a manipulação de cadeias de caracteres

A linguagem C, tal como a maioria das outras linguagens, possui uma série de funções que permitem realizar algumas operações com cadeias e caracteres. A seguir, serão apresentadas apenas algumas dessas funções da linguagem C:

1. strlen (cadeia)

A função **strlen** verifica (conta) quantos caracteres a variável **cadeia** possui. O resultado será devolvido sob a forma de um número inteiro. Veja o exemplo abaixo :

```
#include <stdio.h>
#include <string.h>
void main ( )
{
    char nome [20]= “JOSÉ CARLOS”;
    int n;
    n = strlen(nome);
    printf (“ O nome ‘JOSE CARLOS’ possui &d caracteres \n”,n);
    printf(“Digite seu nome : “);
    scanf(“%20s”,nome);
    n = strlen(nome);
    printf (“O seu nome possui %d caracteres.”,n);
}
```

2. strcpy (dest, orig)

A função **strcpy** copia o conteúdo da cadeia **orig** para a cadeia **dest**. Veja alguns exemplos abaixo :

```
#include <stdio.h>
#include <string.h>
void main ( )
{
    char nome1 [20], nome2[20], seunome[20];

    strcpy (nome1,"JOSÉ CARLOS");
    strcpy (nome2,"DA SLIVA");
    printf ("Nome completo = %s %s \n",nome1,nome2);
    printf("Digite o seu primeiro nome :");
    scanf("%20s",seunome);
    strcpy(nome1,seunome);
    printf ("Nome completo = %s %s \n",nome1,nome2);
}
```

3. strcmp (cadeia1, cadeia2)

A função **strcmp** compara **cadeia1** e **cadeia2**, verificando se são iguais ou diferentes. A função **strcmp** retorna os valores abaixo, como resultado :
Retorno :

1 : indica que **cadeia1** > **cadeia2** (**cadeia1** "vem antes de" **cadeia2**)
0 : indica que **cadeia1** = **cadeia2** (**cadeia1** e **cadeia2** são iguais)
-1 : indica que **cadeia1** < **cadeia2** (**cadeia1** "vem depois de" **cadeia2**)

Obs. Os símbolos de **maior** e **menor** (> e <) possuem um significado especial, relacionado com a ordem alfabética. Veja o exemplo abaixo :

"A" < "B"
"BCD" < "FG"
"MATO" > "MATA"

“MATA” < “MATAR”

Veja o exemplo abaixo

```
. . .
char nome1[20], nome2[20];

printf(“ Digite dois nomes quaiquer : “);
scanf( “%20s %20s”,nome1,nome2);
x = strcmp(nome1,nome2);
if ( x == 0 ) printf (“ Os dois nomes são iguais”);
else
    printf (“São diferentes “);
```

4. strcat (cadeia1, cadeia2)

A função **strcat** concatena cadeia2 no final de **cadeia1**. Note que **cadeia1** deve ser declarada com um tamanho suficiente para a concatenação.
Veja o exemplo abaixo :

```
. . .
char nome1[20], nome2[20];

strcpy(nome1,”JOSÉ”);
strcpy(nome2,”DA SILVA”);

strcat(nome1,” “); /* concatena um espaço ao final de nome1 */
strcat(nome1, nome2);
printf(“%s”,nome1);
```

1.3 Exercícios

1) Leia através do teclado uma frase, contendo no máximo 50 caracteres, e em seguida conte quantas vogais (a,e,i,o,u) existem nessa frase.
2) Faça um programa completo em C, para ler uma frase pelo teclado, e em seguida verificar e exibir uma mensagem na tela, indicando quantas

palavras existem nessa frase. Sugestão : descubra quantos espaços em branco existem na frase.

3) Leia através do teclado uma frase, contendo no máximo 50 caracteres. Em seguida, o programa deverá percorrer essa frase, convertendo de minúscula para maiúscula, a primeira letra de cada palavra. Veja um exemplo abaixo :

   Frase digitada :

   "que não seja para sempre, mas que seja eterno enquanto dure"

   Frase convertida :

   "Que Não Seja Para Sempre, Mas Que Seja Eterno Enquanto Dure"

   Dica : as letras minúsculas estão no intervalo de 97 a 122 na tabela ASCII, enquanto que as maiúsculas estão no intervalo de 65 a 90. Portanto, para converter um caracter 'a' (minúsculo) em um 'A' (maiúsculo), basta subtrair 32 de seu código ASCII. Veja um exemplo abaixo :

```
char c,d;
. . .
c = 'a';
d = c – 32;
printf ("%c",d);  /* vai exibir A maiúsculo */
```

4) Ler pelo teclado o nome de uma pessoa. Supondo que esse nome possui :
   - O primeiro nome
   - Zero ou mais nomes intermediários
   - O último sobrenome

Coloque o nome digitado no formato utilizado em listas telefônicas, e mostre-o na tela. Veja o exemplo abaixo :

JOSÉ CARLOS DA SILVA $\rightarrow$ SILVA, JOSÉ CARLOS DA

5) Ler uma palavra qualquer através do teclado. Em seguida, verifique se essa palavra é palíndrome, ou seja, pode ser lida nos dois sentidos, com o mesmo efeito. Veja alguns exemplos de palavras palíndromas :

   RADAR, ATA, MATAM