

String em C

Roberto Rocha

Trabalhando com string

Strings

A linguagem C, como visto anteriormente, não determina um tipo específico para a manipulação de strings (que são vetores ou cadeias de caracteres, terminados pelo caractere NULL).

Para isso, fornece uma completa biblioteca de funções específicas (string.h).

Funções que manipulam strings a percorrem até encontrar o caractere NULL, quando saberão que ela terminou.

Utilizamos, portanto, o caractere zero da tabela ASCII ('\0') para encerrar a string, e este ocupa um espaço que deve ser previsto pelo programador.

A relação entre strings e vetores é, dessa forma, direta.

Uma string é um vetor de caracteres, mas nem todo vetor de caracteres é uma string.

Declarando e inicializando strings

Vamos declarar uma string que poderá armazenar até 6 caracteres (incluindo o NULL).

`char frase[6] = "ABCDE";` // usando 5+1 posições

Eis um programa que pode ilustrar essa questão:

```
#include <stdlib.h>
#include <stdio.h>
#include <locale.h>

int main()
{
    setlocale(LC_ALL,"portuguese");
    char frase[6] = "ABCDE"; // usando 5+1 posições
    int i;
    printf("Valor de frase = %s\n",frase);
    // vamos ver posição a posição
    for(i=0; i < 6; i++)
    {
        printf("%d - letra = %c código ascii = %d\n",i,frase[i],frase[i]);
    }
    system("pause");
}
```

0	1	2	3	4	5	Posição
A	B	C	D	E	\0	Conteúdo

```
Valor de frase = ABCDE
0 - letra = A código ascii = 65
1 - letra = B código ascii = 66
2 - letra = C código ascii = 67
3 - letra = D código ascii = 68
4 - letra = E código ascii = 69
5 - letra = código ascii = 0
Pressione qualquer tecla para continuar. . .
```

Declarando e inicializando strings

Vamos declarar uma string que poderá armazenar até 20 caracteres (incluindo o NULL).

Eis um programa que pode ilustrar essa questão:

```
#include <stdio.h>
#include <stdlib.h>
main()
{
    char frase[50];
    int i;
    for(i=0; i < 30; i++)
    {
        frase[i] = 'A' + i; /* a variável 'i' incrementa a posição do caractere na Tabela ASCII */
    }
    frase[i] = NULL;
    printf("A string contém %s \n", frase);
    system("pause");
}
```

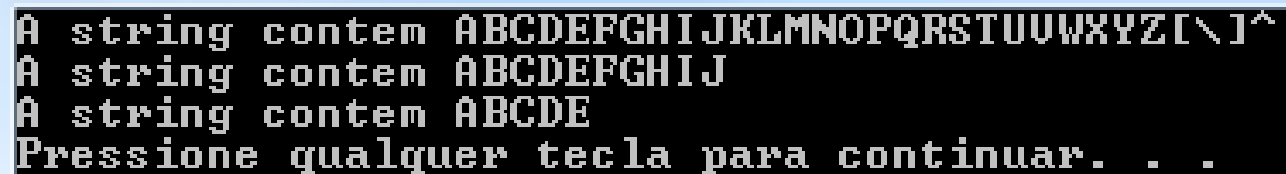
Manipulando strings

Se o mesmo programa forçasse o elemento `frase[10]` a ser `NULL`, a string pararia na letra J ou se colocássemos o `\0` na posição 5 pararia na letra E.

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

int main()
{
    char frase[50];
    int i;
    for(i=0; i < 30; i++)
    {
        frase[i] = 'A' + i; /* a variável 'i' incrementa a posição do caractere na Tabela ASCII */
    }
    frase[i] = NULL;
    printf("A string contém %s \n", frase);
    frase[10] = NULL;
    printf("A string contém %s \n", frase);
    frase[5] = '\0';
    printf("A string contém %s \n", frase);

    system("pause");
}
```



```
A string contém ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^
A string contém ABCDEFGHIJ
A string contém ABCDE
Pressione qualquer tecla para continuar. . .
```

Manipulando strings

Podemos, agora, diferenciar **caracteres** de **strings**, ou seja, **'A'** é diferente de **"A"**.

'A' é o **caractere simples**, enquanto **"A"** significa o **caractere simples mais \0** (NULL).

Aspas simples indicam um caractere. Aspas duplas indicam uma cadeia de caracteres (string).

A seguir temos um vetor de caracteres que não é uma string:

```
char nome[ ] = {'A', 'n', 'a'}; /* não possui o marcador de final de string; estes caracteres serão utilizados de forma separada */
```

Determinando o comprimento da string – percorrer a string até achar o NULL ('\0')

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    char string[50];
    int i;
    printf("Digite um conjunto de caracteres: \n");
    gets(string);
    for(i=0; string[i] != NULL ; i++) printf("%c",string[i]);
    printf("\n");
    printf("O numero de caracteres e %d \n", i);
    system("pause");
}
```

A função de biblioteca **strlen()**, da mesma forma, conta o numero de caracteres da string. Estas funções estão definidas em **string.h**.

Exemplo:

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

int main()
{
    int i;
    char instituto[ ] = "Instituto de Ciencias Exatas e Informatica";
    printf("%s contem %d caracteres. \n", instituto, strlen(instituto));
    for(i=0; instituto[i] != NULL ; i++) printf("%c",instituto[i]);
    printf("\n");
    printf("O numero de caracteres e %d \n", i);

    printf("%s contem %d caracteres. \n", instituto, mystrlen(instituto));

    return 0;
}

int mystrlen(char *string)
{
    int i = 0;
    while (string[i]) i++;
    return(i);
}
```

Repare que o caractere NULL, de final de string, não foi contado.

Funções de manipulação de strings

strcpy(): copia uma string em outra;

strcat(): adiciona o conteúdo de uma string em outra;

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

```
int main ()
{
    char s1[50], s2[50];

    strcpy (s1, "Inicio");
    strcpy (s2, "Fim");
```

```
    strcat (s2, s1);
```

```
    printf("Resultado : |%s|\n\n", s2);
}
```

```
Resultado : !FimInicio!
```

Funções de manipulação de strings

strcmp(): compara duas strings (-1 >, 0 =, 1 >) – case sensitive

strcasecmp(): compara duas strings (-1 >, 0 =, 1 >) – no case sensitive

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <locale.h>
int main()
{
    setlocale(LC_ALL, "portuguese");
    char a[] = "CASA";
    char b[] = "casa";
    /* Utilizando strcmp */
    printf ("Utilizando 'strcmp':\n");
    printf ("%s e %s são ", a, b);
    if (strcmp (a, b) == 0)
    {
        printf ("iguais\n");
    }
    else
    {
        printf ("diferentes\n");
    }
    printf ("Utilizando 'strcasecmp':\n");
    printf ("%s e %s são ", a, b);
    if (strcasecmp (a, b) == 0)
    {
        printf ("iguais\n");
    }
    else
    {
        printf ("diferentes\n");
    }
    return 0;
}
```

```
Utilizando 'strcmp':
CASA e casa são diferentes
Utilizando 'strcasecmp':
CASA e casa são iguais

Process returned 0 (0x0)    execution time : 0.096 s
Press any key to continue.
```

Funções de manipulação de strings

strlwr(): converte conteúdo para minúsculas;
strupr(): converte conteúdo para maiúsculas;
strcmp(): compara duas strings (-1 >, 0 = , 1 >)

// Comparando duas cadeias de caracteres

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{ char x[10], y[10];
  printf("Entre a primeira string: \n");
  gets(x);
  printf("Entre a segunda string: \n");
  gets(y);
  if (strcmp(strlwr(x), strlwr(y)) < 0) printf("%s < %s\n", x, y);
  if (strcmp(strlwr(x), strlwr(y)) == 0) printf("%s = %s\n", x, y);
  if (strcmp(strlwr(x), strlwr(y)) > 0) printf("%s > %s\n", x, y);
  system("pause");
}
```

É possível converter caracteres de uma string para o formato de numero:

atof(): converte cadeia de caracteres para um valor real;

atoi(): converte cadeia de caracteres para um valor inteiro;

sprintf(): converte a cadeia de saída em uma string;

// Comparando duas cadeias de caracteres

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

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

```
#include <stdlib.h>
```

```
int main()
```

```
{ char x[10], y[10],z[20];
```

```
  int a,b;
```

```
  double c,d;
```

```
  strcpy(x,"123");
```

```
  strcpy(y,"23.67");
```

```
  a=atoi(x);
```

```
  b=a+1;
```

```
  c=atof(y);
```

```
  d=c+1;
```

```
  printf("\nValor de x=%s, a=%d,b=%d\n",x,a,b);
```

```
  printf("\nValor de y=%s, c=%3.2lf,d=%3.2lf\n",y,c,d);
```

```
  strcpy(z,x);
```

```
  strcat(z,y);
```

```
  printf("\nVvalor de z=%s\n\n",z);
```

```
  sprintf(z,"%d%3.2f",a,c);
```

```
  printf("\nVvalor de z=%s\n\n",z);
```

```
  system("pause");
```

```
}
```

```
Valor de x=123, a=123,b=124
```

```
Valor de y=23.67, c=23.67,d=24.67
```

```
Uvalor de z=12323.67
```

```
Uvalor de z=12323.67
```

Exercício de fixação:

- 1 - Leia um conjunto indeterminado de palavras e ao final (estipule você um flag) informe qual foi a maior palavra e a menor palavra digitada, em tamanho e lexicograficamente.
- 2 – Ler um string de no máximo 50 caracteres e contar quantas letras A essa palavra possui.
- 3 – Ler uma string de no máximo 50 caracteres e indicar quais as posições da letra A nessa palavra.
- 4 – Ler uma string de no máximo 50 caracteres e em seguida um caractere, mostrar quais as posições esse caractere aparece na string lida e quantas vezes ele apareceu.
- 5 - Ler uma string de no máximo 50 caracteres e em seguida um caractere (entre A e z - consista se o caracter esta nesse intervalo), mostrar quais as posições esse caractere (maiúscula ou minúscula) na string lida e quantas vezes ele apareceu.
- 6 - Ler uma string de no máximo 50 caracteres e mostrar quantas letras possui e quantos caracteres são números e quantos não são nem números nem letras.
- 7 – Ler uma string de no máximo 50 caracteres e criar uma nova string com seu inverso, isso é a ultima letra da primeira string será a primeira na nova string e assim sucessivamente.
- 8 – Ler uma string de no máximo 50 caracteres e retire dessa string todos os espaços em branco. Utilize uma string auxiliar.
- 9 – Ler uma string de no máximo 50 caracteres e retire dessa string todos os espaços em branco. Sem utilize string auxiliar.
- 10 - Ler uma string de no máximo 50 caracteres em seguida leia outra string de no máximo 3 caracteres, informe quantas vezes a segunda string aparece na primeira string, e diga as posições iniciais em que ela aparece.



PUC Minas
Virtual