

Capítulo 7

Manipulação de Strings e Caracteres

7.1 Introdução

Como já comentado anteriormente em nosso curso de Algoritmos, a linguagem C não apresenta um tipo “string”. Sabemos também desde o início do nosso curso que uma string nada mais é do que uma cadeia (conjunto) de caracteres. As strings são representadas usando aspas (“”), enquanto que os caracteres são representados usando aspas simples. Dessa forma temos:

Exemplos de strings:

“Maria”

“São Paulo”

“Faculdade de Informática”

Exemplos de caracteres:

‘q’

‘s’;

Cada caractere em C ocupa um byte na memória. Dessa forma, ‘C’ é o caractere C (ocupa apenas 1 byte na memória) enquanto que, “C” é um vetor de caracteres (ocupa 2 bytes na memória onde 1 byte é reservado para o finalizador de strings).

Para manipularmos uma cadeia de caracteres em C devemos trabalhar com vetores unidimensionais, ou seja, **uma string em C é uma cadeia de caracteres armazenadas em um vetor**. A utilização mais comum de um array unidimensional em C é na definição de uma string. Portanto, strings são vetores do tipo *char*. Devemos estar atentos ao fato de que as strings têm como último elemento um ‘\0’, que é adicionado automaticamente ao final da string. **Vale reforçar que uma string é um vetor de caracteres, mas o inverso é falso, isto é, um vetor de caracteres pode não ser uma string.**

A forma geral para declararmos uma string segue o mesmo padrão para a declaração de um array unidimensional:

char nome_da_string[tamanho];

onde *tamanho* representa o número máximo de caracteres que a string irá armazenar. Devemos incluir neste valor o finalizador de strings.

Exemplo de declaração: `char nome[20];`

No exemplo acima, temos uma variável vetor que armazena no máximo 20 caracteres. Lembre-se que o tamanho da string deve incluir o finalizador de strings (‘\0’). A função do finalizador de strings é única e estritamente para definir quais são as posições preenchidas dentro de vetor de caracteres das posições que não foram preenchidas. Por exemplo, se declararmos um vetor chamado

nome com 200 posições e preencheremos com a string “Ana”, o que realmente acontece na memória?

nome[0]	nome[1]	nome[2]	nome[3]	nome[4]	nome[5]	...	nome[199]
A	n	a	\0	<lixo>	<lixo>	...	<lixo>

As variáveis strings podem ser inicializadas no momento da declaração da mesma forma que as variáveis de outros tipos. Não se esqueça que uma string sempre vem entre aspas (“”). Exemplo:

```
char nome[20] = “Roberto”;  
char nome[20] = { ‘R’, ‘o’, ‘b’, ‘e’, ‘r’, ‘t’, ‘o’ }; //o ‘\0’ é colocado automaticamente  
char nome[] = “Roberto”; //é equivalente a nome[7+1]  
char *nome = “Roberto”; //declaração via ponteiro. É equivalente ao exemplo logo acima.
```

7.2 Funções para manipulação de strings

A linguagem C apesar de não apresentar um tipo específico “string”, apresenta uma série de funções que podem ser aplicadas a uma variável do tipo vetor de caracteres. A maioria das funções de manipulação de string apresenta seus cabeçalhos em **string.h**. Dentre as principais funções destacam-se:

Função printf() e puts()

A função **printf()** é utilizada para escrever uma string no vídeo. A função recebe como parâmetro uma string que pode ser escrita diretamente no vídeo, por exemplo: **printf**(“Testando strings”). A função **printf()** também pode imprimir o conteúdo de uma string que está armazenada em um vetor de caracteres. Nesse caso usa-se o formato próprio de string (%s). Por exemplo:

```
char nome[] = “Maria Cristina”;  
char sobrenome[] = “Silva”;  
printf(“%s %s”, nome, sobrenome);
```

Outra função utilizada para escrever uma string no vídeo é **puts()** (biblioteca stdio.h). A função **puts()** é utilizada exclusivamente para a impressão de strings, sejam constantes ou valores armazenados em variáveis. A função **puts()** escreve a string no vídeo e automaticamente muda de linha. Exemplo:

```
puts(“Testando strings”); //é equivalente a printf(“Testando strings\n”);
```

Função scanf() e gets()

A função **scanf()** permite a leitura de strings usando o formato %s. Para a leitura de strings a função permite ler apenas uma palavra, ou seja, a função lê todos os caracteres até encontrar um espaço em branco ou um TAB ou um ENTER. Para a leitura de strings a melhor a função é **gets()** da biblioteca stdio.h. A sintaxe para a função **gets()** é:

```
gets( nome_da_string );
```

Exemplo de utilização da função `gets()`:

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    char string[100];
    printf ("Digite o seu nome: ");
    gets (string);
    printf ("\n\n Ola %s",string);
    system("pause");
    return(0);
}
```

Função `strlen()`

Uma função bastante útil na manipulação de strings é a função **`strlen()`** da biblioteca **`string.h`**, utilizada para retornar o comprimento da string. No comprimento retornado não é incluído o finalizador de strings. A forma geral para a função é:

`strlen(nome_da_string);`

A função **`strlen()`** retorna um valor do tipo inteiro. Exemplo:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int main()
{
    int tamanho;
    char str[100];
    printf("Entre com uma string: ");
    gets(str);
    tamanho = strlen(str);
    printf("\n\nTamanho da string digitada %d", tamanho);
    system("pause");
    return(0);
}
```

Função `strcpy()`

Em C não é permitido usar o operador de atribuição para copiar o conteúdo de uma variável string para outra. Para realizar essa operação existe a função **`strcpy()`** da biblioteca **`string.h`**. Esta função é utilizada para copiar o conteúdo de uma string para outra. A sua forma geral é:

`strcpy(string_destino, string_origem);`

A *string_origem* é copiada na *string_destino*. Exemplo:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int main()
{
    char str1[100], str2[100], str3[100];
    printf ("Entre com uma string: ");
    gets (str1);
    strcpy (str2, str1); /* Copia str1 em str2 */
    strcpy (str3, "Voce digitou a string ");
    printf ("\n\n%s%s",str3,str2);
    system("pause");
    return(0);
}
```

No programa exemplo acima, a string informada via teclado é armazenada na variável *str1*. Em seguida, o conteúdo de *str1* é copiado para a variável *str2* usando a função **strcpy()**. Perceba que neste caso teremos duas variáveis com o mesmo conteúdo. Caso a variável *str2* tenha um conteúdo inicial, este será substituído.

Função **strcat()**

Esta função é utilizada para concatenar duas strings, ou seja, o conteúdo de uma string é adicionado ao final de outra string. A função **strcat()** está na biblioteca **string.h**. A forma geral da função é:

$$\text{strcat}(\text{string_destino}, \text{string_origem});$$

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

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int main()
{
    char str1[100], str2[100];
    printf ("Entre com uma string: ");
    gets (str1);
    strcpy (str2,"Voce digitou a string ");
    strcat (str2, str1);
    printf ("\n\n%s",str2);
    system("pause");
    return(0);
}
```

Função strcmp()

Esta função é utilizada para comparar alfabeticamente duas strings, uma vez que o operador relacional de igualdade não pode ser aplicado a variáveis do tipo string. O retorno desta função é um valor inteiro tendo o seguinte significado: < 0 (se a primeira string é alfabeticamente menor do que a segunda string), 0 (se as string são alfabeticamente iguais) ou > 0 (se a primeira string é alfabeticamente maior do que a segunda string).

A forma geral da função **strcmp()** é:

strcmp(string1, string2);

A tabela abaixo mostra exemplos da utilização da função strcmp() (**Exemplo tirado do livro Linguagem C – Luis Damas – Editora LTC – 10ª Edição 2007**):

Invocação	Resultado	Observações
strcmp("abc", "abxpo")	< 0	'c' $<$ 'x'
strcmp("beatriz", "carlos")	< 0	'b' $<$ 'c'
strcmp("carlos", "carla")	> 0	'o' $>$ 'a'
strcmp("carlos", "beatriz")	> 0	'c' $>$ 'b'
strcmp("mario", "maria")	> 0	'o' $>$ 'a'
strcmp("maria", "mariana")	< 0	A 1ª string é alfabeticamente menor do que a 2ª
strcmp("", "")	0	As string são iguais
strcmp("ola", "ola")	0	As string são iguais

A função **strcmp()** distingue os caracteres entre maiúsculo e minúsculo retornando valores diferentes caso isso aconteça. Uma opção é utilizar a função **strcmpi()** que executa a mesma função mas ignorando caracteres maiúsculos e minúsculos.

```
#include <stdio.h>
#include <string.h>
#include <string.h>
int main()
{
    char str1[100], str2[100];
    printf("Entre com uma string: ");
    gets( str1 );
    printf("\n\nEntre com outra string: ");
    gets( str2 );
    if(strcmp(str1,str2))
        printf("\n\nAs duas strings são diferentes.");
    else
        printf("\n\nAs duas strings são iguais.");
    system("pause");
    return(0);
}
```

Outras funções de manipulação de string e caracteres

strchr(): função utilizada para procurar por um caractere em uma string. A função recebe como parâmetro uma string e um caractere. Essa função retorna um ponteiro para a primeira ocorrência do caractere na string. Caso o caractere não esteja na string é retornado o valor NULL (0). Exemplo:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{
    char nome[100], ch, *ch2;
    printf("Qual o nome? -> ");
    gets(nome);
    printf("Qual o caractere? -> ");
    ch = getchar();
    ch2 = strchr(nome, ch);
    if(ch2)
        puts("o caractere está na string");
    else
        puts("o caractere não está na string");
    system("pause");
    return(0);
}
```

strstr(): função utilizada para localizar uma string (substring) dentro de outra string. A função recebe como parâmetro duas string e retorna um ponteiro para o **início da ocorrência** da segunda string dentro da primeira, passada como parâmetro. Exemplo:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{
    char s1[] = "linguagem de desenvolvimento c";
    char s2[] = "agem", *pt;
    pt = strstr(s1, s2);
    if(pt)
        printf("%s esta na string", pt);
    else
        puts("a substring nao foi localizada");
    system("pause");
    return(0);
}
```

strlwr(): converte todos os caracteres da string para minúsculo. Exemplo:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{
    char s1[] = "Linguagem de Desenvolvimento C";
    strlwr(s1);
    puts(s1);
    system("pause");
    return(0);
}
```

strupr(): converte todos os caracteres da string para maiúsculo. Exemplo:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{
    char s1[] = "Linguagem de Desenvolvimento C";
    strupr(s1);
    puts(s1);
    system("pause");
    return(0);
}
```

7.3 Exercícios em Classe

1. Escreva um programa em C que leia uma string (no máximo 50 caracteres) via teclado e informe a quantidade de letras “a” presentes na string (podendo ser maiúscula ou minúscula).
2. Escreva um programa em C que conte o número de vogais presentes em uma string informada via teclado por um usuário. Observação: as letras podem estar grafadas em caixa alta ou caixa baixa.
3. Escreva um programa em C que leia uma string (no máximo 30 caracteres) via teclado. Inverta a string e armazene em outra variável. Imprima as duas strings no vídeo.
4. Usando a tabela de códigos ASCII, responda o que está escrito na propaganda abaixo:

```
char msg[] = {69,83,84,65,77,79,83,32,67,79,78,84,82,65,84,65,78,68,79,'\\0'};
```

Tabela ASCII – American Standard Code for Interchange Information

Char	Dec	Char	Dec	Char	Dec	Char	Dec
NUL ^@	0	SPACE	32	@	64	‘	96
^A	1	!	33	A	65	a	97
^B	2	“	34	B	66	b	98
^C	3	#	35	C	67	c	99
^D	4	\$	36	D	68	d	100
^E	5	%	37	E	69	e	101
^F	6	&	38	F	70	f	102
^G	7	‘	39	G	71	g	103
^H	8	(40	H	72	h	104
TAB ^I	9)	41	I	73	i	105
^J	10	*	42	J	74	j	106
^K	11	+	43	K	75	k	107
^L	12	,	44	L	76	l	108
^M	13	-	45	M	77	m	109
^N	14	.	46	N	78	n	110
^O	15	/	47	O	79	o	111
^P	16	0	48	P	80	p	112
^Q	17	1	49	Q	81	q	113
^R	18	2	50	R	82	r	114
^S	19	3	51	S	83	s	115
^T	20	4	52	T	84	t	116
^U	21	5	53	U	85	u	117
^V	22	6	54	V	86	v	118
^W	23	7	55	W	87	w	119
^X	24	8	56	X	88	x	120
^Y	25	9	57	Y	89	y	121
^Z	26	:	58	Z	90	z	122
^[27	;	59	[91	{	123
^	28	<	60	\	92		124
^]	29	=	61]	93	}	125
^^	30	>	62	^	94	~	126
^_	31	?	63	_	95	del	127

5. Faça um programa em C que leia uma palavra pelo teclado e faça a impressão conforme o exemplo a seguir para a palavra AMOR

AMOR
AMO
AM
A

6. Faça um programa em C que leia uma palavra pelo teclado e faça a impressão conforme o exemplo a seguir para a palavra AMOR

A
AM
AMO
AMOR

7. Escreva um programa em C que leia uma string e também um caractere. O seu programa deverá apagar todas as ocorrências do caractere informado na string. **Observação: apagar não é substituir.**

8. Escreva um programa em C que leia apenas uma palavra (dica: use a função **scanf()** para ler a string). Gerar uma nova string que será a string lida contendo um espaço em branco após cada caractere.

9. Uma string é considerada um palíndromo se quando lida da esquerda para a direita e da direita para a esquerda apresentam o mesmo valor. Por exemplo: ANA. Escreva um programa em C em que o usuário possa digitar várias palavras e informe se são palíndromos ou não. A cada palavra informada pergunte ao usuário se ele deseja continuar a execução do programa.

10. Escreva um programa em C que leia uma string via teclado. Converta os caracteres para minúsculo e informe a quantidade de caracteres que tem código ASCII par.

11. Escreva um programa em C que solicite o nome do usuário do sistema. O seu programa irá gerar uma senha que é formada apenas pelos caracteres das posições ímpares mais a soma do código ASCII dos caracteres nas posições pares.

7.4 Exercícios Complementares

1. Faça um programa em C que leia quatro strings pelo teclado. Depois, concatene todas as strings lidas em uma única variável. Imprima as strings concatenadas no vídeo.

2. Faça um programa em C que leia uma palavra pelo teclado e informe a quantidade de caracteres que não são vogais.

3. Escreva um programa em C que leia uma palavra fornecida pelo teclado e em seguida imprima o caractere presente no meio da palavra, caso esta tenha um número ímpar de caracteres. Como exemplo, considere a palavra SONHO. O caractere a ser impresso será o N.

4. Escreva um programa em C que leia uma string e também um caractere. O seu programa deverá contar o número de ocorrências do caractere lido na string.

5. Escreva um programa em C que leia uma string. O seu programa deverá contar o número de dígitos (0 – 9) presentes na string. Para resolver esse exercício utilize a função **isdigit(char c)** da biblioteca **ctype.h**. Se o caractere passado para a função **isdigit()** for um dígito a função retornará um valor diferente de zero (valor verdadeiro em C).

6. Escreva um programa em C que leia uma string. O seu programa deverá contar o número de caracteres alfabéticos da string. Dica: usar a função **isalpha(char c)** da biblioteca **ctype.h**. A função retornará um valor diferente de zero (valor verdadeiro em C) caso o caractere passado para a função **isalpha()** for uma letra (A até Z ou a até z).

7. Escreva um programa em C que leia uma string via teclado. Em seguida, transforme a string lida em vazia, ou seja, a string deverá ficar em branco (sem nenhum caractere).

8. Escreva um programa em C que leia uma string e também um caractere. O programa deverá imprimir no vídeo a posição em que se encontra a última ocorrência do caractere na string. Caso o caractere não esteja na string deverá ser impresso no vídeo o valor -1.

9. Escreva um programa em C que leia uma string via teclado e retorne o valor do maior caractere ASCII presente na string. Caso a string esteja vazia retornar o caractere 0 ('\0').

10. Escreva um programa em C que leia nomes completos do teclado e os escreva na tela no formato **Sobrenome, Nome sem Sobrenome**. Exemplos:

Nome → Antônio Marcos Selmini

Saída no vídeo → Selmini, Antônio Marcos

Nome → Catarina Silva

Saída no vídeo → Silva, Catarina

Observações: i) a primeira letra de cada palavra deve estar grafada em maiúscula; ii) o processamento termina quando o usuário digitar Sair.

11. Escreva um programa em C que leia uma string (no máximo 100 caracteres) via teclado e informe a quantidade de palavras presentes na string.