**Caracteres e Strings**

String é uma seqüência de caracteres utilizada para o armazenamento de texto. Na linguagem C strings são vetores de caracteres que possuem um caracter que indica o término de seu conteúdo, o caracter nulo ‘\0’ (contrabarra zero).

#### **Declaração de strings**

Como a string possui o caracter nulo para delimitar o final do seu conteúdo, o tamanho da string deve ser definido com um caracter a mais do que será efetivamente necessário.

|  |
| --- |
| Sintaxe: char identificador-da-string [tamanho+1]; |

Exemplo.

|  |
| --- |
| char vetc [6];  vetc é um vetor de caracteres (string) de tamanho 6. Pode receber uma palavra de no máximo 5 letras |

#### Inicialização de strings

Uma string pode ser inicializada na sua declaração com uma sequência de caracteres entre chaves e separadas por virgula.

char vetc[6]= {'T', 'e', 'x', 't', 'o', '\0'};

Lembre-se que o compilador só reconhecerá um caractere se este estiver entre aspas simple, logo usar uma atribuição do tipo*{t,e,x,t,o,\0}*ou*{texto\0}*irá gerar um erro de compilação.

Uma string pode também ser inicializada por uma seqüência de caracteres entre aspas duplas. Neste caso, não é necessário o uso de aspas simples e virgulas, o compilador C coloca automaticamente o '\0' no final.

char vetc[6] = “Texto”;

Assim como vetores e matrizes, na inicialização de uma string o seu tamanho pode ser omitido.  
  
O compilador vai verificar e considerar o tamanho declarado na inicialização.  
  
char vetc[ ] = “Texto”; /\* vetor não-dimensionado, o compilador coloca automaticamente   
o ‘\0’ no final \*/

#### **Manipulação de string**

[topo](http://www.linguagemc.xpg.com.br/strings.html" \l "topo)

### *Funções de entrada e saída (<stdio.h>)*

* **gets(s)** - Lê uma string do dispositivo de entrada padrão e armazena esta string em s. Não é uma função segura, pois o tamanho da string não é especificado.
* **fgets(s, TAM, stdin)** - Lê uma string de tamanho TAM do dispositivo de entrada padrão e armazena esta string em s.
* **puts(s)** - Imprime a string s no dispositivo de saída padrão.
* **sscanf(s, "expressão-controle", end1, end2, ...)** - Faz a leitura formatada em uma string s
* **sprintf(s, "expressão-controle", arg1, arg2, ...)** - Faz a escrita formatada em uma string s

##### trecho de código

char string[ ]= "2 20 200"; /\* inicializa a string str \*/  
sscanf(string, "%d %d %d", &i, &j, &k); /\* armazena 2 em i, 20 em j e 200 em k \*/

Observe que foram usados exemplos numérico como valores, neste caso o compilador interpreta os numeros como sendo uma string. Obcerve as notações abaixo.

* 1 para o compilador poder ser um numero inteiro ou real.
* '1' para o compilador é o alfanumerico (caracter) char.
* "11..." para o compilador é uma string de conteudo 11...
* logo deve-ser ter muita atenção ao uso de aspas para manipular strings.

##### trecho de código

char string[10];  
int i=2;  
sprintf(string,"%d", i); /\*armazena 2 em str \*/

##### Programa 17. manipulação de string

#include <stdio.h>

#include <stdlib.h>

int main ()

{

char nome[] = "fulano";

char sobrenome[] = "de tal";

char nomeCompleto[] = nome + sobrenome;

int i=0;

while(nomeCompleto[i] != '\0'){

printf("%c",nomeCompleto[i]);

nomeCompleto[i++];

}

return(0);

system("pause");

}

##### ERRO DE COMPILAÇÃO

|  |
| --- |
| PORQUE O PROGRAMA ACIMA NÃO COMPILA?  Vamos entender o que o programa faz. Declara uma string contedo um nome, uma segunda string que é inicializada com o sobrenome, e uma terceira string que recebe a string nome concatenada a string sobrenome. Em seguido uma estrutura de fluxo while controla impressão do conteúdo da string nomeCompleto  Um erro muito comum no uso de string em C esta sendo cometido na linha 7 do prog. 17. char nomeCompleto[] = nome + sobrenome;. String não poder ser concatenadas utilizando o operador +. Existe uma diretiva em C que implementa diversas funções de manipulação de valores em string. |

[topo](http://www.linguagemc.xpg.com.br/strings.html" \l "topo)

### *Funções de manipulação de strings <string.h>*

Strings não podem ser comparadas com o operador de comparação padrão (==), neste caso deve-se usar função strcmp() ou a função stricmp().

* **strcmp(s1,s2)** – Retorna 0 se s1 e s2 são iguais; menor que 0 se s1<s2; maior que 0 se s1>s2 (comparação alfabética).
* **stricmp(s1,s2)** – Retorna 0 se s1 e s2 são iguais; menor que 0 se s1<s2; maior que 0 se s1>s2 (comparação alfabética). Essa função considera letras maiúsculas ou minúsculas como símbolos iguais.

Strings não podem ser atribuídas com o operador de atribuição (=), para uma atribuição usa-se a função strcpy( ).

* **strcpy(s1,s2)**– Copia s2 em s1.

Strings não podem ser concatenadas com o operador (+), para tal usa-se a função strcat().

* **strcat(s1,s2)** – Concatena s2 ao final de s1.
* **strlen(s)** – Retorna o número de caracteres em s (sem contar o caracter nulo (/0)).
* **strchr(s,c)** – Retorna um ponteiro na primeira ocorrência do caracter c na string s.
* **strstr(s1,s2)** - Retorna um ponteiro na primeira ocorrência se s2 em s1.
* **strrev(s)** – Inverte a string s sobre ela mesma.

##### Programa 17. manipulação de string

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int main (){

char nome[] = "fulano";

char sobrenome[] = " de tal";

char nomeCompleto[15];

strcat(nome, sobrenome);

strcpy(nomeCompleto, nome);

puts(nomeCompleto);

return(0);

system("pause");

}

##### fulano de tal

##### Programa 18. manipulação de string

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

void main(){

char f1[50], f2[50],f3[100],c;

int tam1, tam2, i,cont1,cont2;

printf("Digite uma frase: ");

gets(f1);

printf("Digite uma frase: ");

gets(f2);

tam1=strlen(f1);

tam2=strlen(f2);

c='a';

cont1=cont2=0;

for(i=0;i<=tam1;i++)

if(f1[i]==c) cont1++;

for(i=0; i<=tam2;i++)

if(f2[i]==c) cont2++;

printf("\nA letra %c aparece %d vez(es) na primeira e %d

vez(es) na segunda frase.\n", c, cont1, cont2);

if(!strcmp(f1,f2)) printf("\nAs frases sao iguais\n");

else{

strcat(f3,f1);

strcat(f3," ");

strcat(f3,f2);

printf("Nova frase: ");

puts(f3);

if(strchr(f3,'i')) printf("A letra 'i' aparece nesta frase\n\n");

if(strstr(f3,"string"))printf("A sequencia \"string\" tambem aparece nesta frase\n\n");

printf("Frase invertida: ");

strrev(f3);

puts(f3);

}

}

##### frase 1 = testando o uso frase 2 = de string.

**Limitações de C em trabalho com caracteres e strings**

C não possui um tipo string explícito. Não existe palavra reservada que declare uma variável para armazenar uma string. Em vez disto, você aloca espaço na memória para uma string definindo uma matriz de caracteres.

Uma matriz de caracteres algumas vezes se parece com qualquer outra variável. Por exemplo você pode mostrar uma string inteira na tela numa única operação printf e usar uma variedade de funções para ler uma string do teclado e armazenar numa variável de entrada.

Exemplo:

printf ("%s\n", strings);

Mas algumas operações essenciais de strings estão ausentes da linguagem C propriamente dita. Essas operações estão disponíveis nas bibliotecas de funções no Borland C:

As operações de

\* atribuição de uma string a uma variável  
\* adição ou concatenação de uma string a outra string  
\* comparação de strings

que são operações simples em outras linguagens, não são possíveis de realizar diretamente, tem que ser utilizadas funções específicas de biblioteca, respectivamente:

\* strcpy  
\* strcmp ou stricmp  
\* strcat

**Cuidados com as limitações nas atribuições de string**

**Atribuições de string**- atribuir um valor literal string a uma variável string. Uma tentativa de execução da operação resulta num erro de compilação. Em vez disto, você utiliza uma função da biblioteca < string.h> chamada **strcpy**

Exemplos errado e correto:

char linguagem[20];  
linguagem = "linguagem C"; **/\* ilegal em C \*/**  
strcpy( linguagem, "linguagem C"); **/\* legal \*/**

**Comparações de string**- determinar quando uma string é alfabeticamente maior ou menor que outra string. Em vez disto, você utiliza uma função da biblioteca< string.h> chamada **strcmp**.

Exemplos errado e correto:

string1 < string2 **/\* ilegal em C \*/**  
strcomp( string1, string2); **/\* legal \*/**

**Concatenação de strings** - juntar uma string no final de outra para produzir uma nova string combinada. Em vez disto, você utiliza uma função da biblioteca< string.h> chamada **strcat**.

Exemplos errado e correto:

string3 = string1 + string2; **/\* ilegal em C \*/**  
strcat( string1, string2); **/\* legal \*/**

**Examinando as funções de string da biblioteca < string.h>**

Um arquivo de cabeçalho chamado string.h contém as declarações de um número de funções da biblioteca que você usará freqüentemente em tarefas de tratamento de strings. A seguinte diretiva lê este arquivo de cabeçalho para usar em um programa.

#include < string.h>

Convenientemente, muitas funções declaradas em string.h possuem nomes que começam com str - por exempo, strcpy, strcmp e strcat, as funções que executam atribuições, comparações e concatenações.

**Copiando uma string: a função strcpy**

Para armazenar uma string literal numa variável string - ou copiar o conteúdo de uma variável string para outra - você usa a função strcpy.

Sintaxe:

strcpy (variável, string);

Essa função copia o valor da string na segunda posição do argumento para uma variável string na primeira posição. O segundo argumento pode ser uma variável, uma expressão string ou um valor literal string contido entre aspas.

/\* ex48.c utilização da função strcpy \*/

#include < stdio.h> /\* diretiva \*/

#include < string.h>

#define LENGHT 17

main() /\* função principal \*/

{

char origem[LENGHT} = "Aqui vamos de novo!",

destino[LENGHT];

**strcpy (destino, "constante string");**

**printf ("%s\n", destino);**

**strcpy (destino, origem);**

**printf ("%s\n", destino);**

}

**Comparando duas strings: a função strcmp**

A função strcmp compara duas strings e devolve um valor inteiro que lhe diz qual das strings vem antes no código ASCII:

Sintaxe:

strcmp (string1, string2);

Nesse contexto, string1 é dita ser "maior que" string2 se o primeiro caracter não-coincidente em string1 tiver um código ASCII maior que o caracter correspondente na string2.

A função strcmp devolve um valor inteiro que indica o relacionamento entre string1 e string2:

Um valor menor que zero significa que string1 é menor que string2.

Um valor zero significa que ambas as strings são iguais.

Um valor maior que zero significa que string1 é maior que string2.

Exemplo:

Se string1 é "ABC" e string2 é "DEF", strcmp fornece um valor negativo, indicando que string1 é menor que string2.

As letras minúsculas possuem um código ASCII maior que as letras maiúsculas. Se string1 é "abc" e string2 é "DEF", strcmp fornece uma valor positivo, indicando que string1 é maior que string2. Borland C também possui uma função de comparação de strings chamada stricmp que ignora as diferenças das letras maiúsculas e minúsculas nas comparações.

/\* ex49.c utilização da função strcmp \*/

#include < stdio.h> /\* diretivas \*/

#include < string.h>

main() /\* função principal \*/

{

char string1[] = "um", string2[] = "um";

int menor, result = 0;

if (strlen (string2) > = strlen (string1))

result = **strcmp(string1, string2);**

printf ("A string %s encontrada.", result = = 0? "foi :

"não foi");

}

**Concatenado duas strings: a função strcat**

A função strcat junta uma string ao final de outra.

Sintaxe:

strcat (string1, string2);

Como resultado desse comando, o valor de string2 é anexado ao final de string1. Uma vez que o valor concatenado é realmente armazenado em string1, essa variável deve estar definida com o tamanho adequado para armazenar a string combinada.

/\* ex50.c utilização da função strcat \*/

#include < stdio.h> /\* diretivas \*/

#include < string.h>

main() /\* função principal \*/

{

char string1[20] = "Faturamento ";

char string2[20] = "Horas";

**strcat (string1, string2);**

printf ("\n\"%s\"contém %d caracteres.\n", string1, strlen(string1));

}

**Convertendo maiúsculas e minúsculas: strupr e strlwr ou toupper e tolower**

A função **strupr** converte uma string para maiúsculas e **strlwr** para minúsculas. Cada função toma uma única string como argumento:

strupr (string)  
strlwr (string)

Essas funções afetam apenas os caracteres alfabéticos contidos na string, excluindo dígitos ou caracteres de pontuação. Cada função devolve um valor string especificado.

Borland C também suporta funções que convertem o caso alfabético para caracteres individuais. Essas funções, chamadas **toupper** e **tolower**, são declaradas no arquivo de cabeçalho **ctype.h**. Ambas pegam um único caracter como argumento e devolvem o equivalente maiúsculo ou minúsculo:

toupper (c)  
tolower (c)

/\* ex53.c utilização da função strupr \*/

#include < stdio.h>

#include < string.h>

main()

{

char s[80];

printf ("Digite uma string: ");

printf ("\nA string em maiúscula é: %s\n",

**strupr(gets(s)));**

}

/\* ex54.c utilização da função toupper e tolower \*/

#include < stdio.h> /\* diretivas \*/

#include < ctype.h>

main ()

{

printf ("tolower(A) = %c\n, **tolower('A')**);

printf ("toupper(A) = %c, **toupper('A')**);

}

**resultado:**

tolower(A) = a

touper(A) = A

/\* ex56.c utilização da função strlwr \*/

#include< stdio.h>

main()

{

char s1[] = "ABC";

**strlwr(s2);**

printf ("%s\n", s2);

}

**Exercícios sobre String**

1. Qual a diferença entre "mno" e "m\no. Qual a diferença entre "ABC\0abc" e "ABC0abc"?
2. Qual a diferença entre "A" e 'A'?
3. Escreva um programa que receba um caractere c e devolve uma string cujo único caracter é c.
4. Escreva uma função que calcule o comprimento da mais longa sequência de espaços em branco em uma string.
5. Escreva uma função que receba uma string e substitua cada segmento de dois ou mais espaços em branco por um só caracter ' '.
6. Escreva uma função que receba uma string de 0s e 1s, interprete essa string como um número em notação binária e devolva o valor desse número.
7. Fazer um programa que inverte um string. “Parabens a voce'' vira “ecov a snebaraP''.
8. Faça um programa capaz de contar a quantidade de vogais (maiúsculas e minúsculas) de uma string recebida pelo teclado.

Considere o trecho de código abaixo:

char s[20];

strcpy (s, "EXEMPLO");

//s, passa a ter como valor [EXEMPLO\0]

onde strcpy é uma função que discutida adiante, capaz de copiar o conteúdo de uma string para outra. O segundo argumento de strcpy no exemplo é uma string constante. Note que strings constantes são indicadas por aspas duplas (e o caracter nulo final é subentendido).

1. Faça um programa capaz de receber uma string e copiar seu conteúdo para outra.