

UNIVERSIDADE ESTADUAL DE MS
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO
AEDI

Linguagem C - String

Prof. Nilton

String

- Uma sequência de letras, dígitos, símbolos e ou espaços em branco, conhecida também de cadeia de caracteres. A sequência deve estar entre aspas duplas:
 - “Mauro Silva”
 - “Rua Rio Grande, 1000”
 - “Curitiba, Paraná”
 - “(041) 9999-8888”
- Uma string em C é um array de caracteres que termina pelo marcador ‘\0’
- O acesso a uma string é através de ponteiro para o primeiro caractere
- Pode-se guardar uma string em uma declaração para um array de caracteres ou variável ponteiro para char:
 - char nome[12] = “Mauro Silva”; ‘M’,‘a’,‘u’,‘r’,‘o’,‘ ’,’S’,‘í’,‘l’,‘v’,‘a’,‘\0’
 - char nome[] = “Mauro Silva”; idem
 - char nome[] = {‘M’,‘a’,‘u’,‘r’,‘o’,‘ ’,’S’,‘í’,‘l’,‘v’,‘a’,‘\0’}; idem
 - Char *nome = “Mauro Silva”; cria variável ponteiro que aponta para string em algum lugar da memória
- Usando o scanf
 - char cidade[50];
 - scanf(“%s”,cidade);

int scanf (const char * format, ...);

- Lê dados de stdin e armazena no formato na memória apontada;
- Ignora espaços em branco, nova linha e tab;
- Retorna quantos itens carregados;
- O especificador de formato é

%[*][width][length]specifier

int scanf (const char * format, ...);

%[*][width][length]specifier

specifier	Description	Characters extracted
i	Integer	Any number of digits, optionally preceded by a sign (+ or -). Decimal digits assumed by default (0-9), but a 0 prefix introduces octal digits (0-7), and 0x hexadecimal digits (0-f). <i>Signed argument.</i>
d or u	Decimal integer	Any number of decimal digits (0-9), optionally preceded by a sign (+ or -). d is for a <i>signed</i> argument, and u for an <i>unsigned</i> .
o	Octal integer	Any number of octal digits (0-7), optionally preceded by a sign (+ or -). <i>Unsigned argument.</i>
x	Hexadecimal integer	Any number of hexadecimal digits (0-9, a-f, A-F), optionally preceded by 0x or 0X, and all optionally preceded by a sign (+ or -). <i>Unsigned argument.</i>
f, e, g a	Floating point number	A series of decimal digits, optionally containing a decimal point, optionally preceded by a sign (+ or -) and optionally followed by the e or E character and a decimal integer (or some of the other sequences supported by strtod). Implementations complying with C99 also support hexadecimal floating-point format when preceded by 0x or 0X.
c	Character	The next character. If a <i>width</i> other than 1 is specified, the function reads exactly <i>width</i> characters and stores them in the successive locations of the array passed as argument. No null character is appended at the end.
s	String of characters	Any number of non-whitespace characters, stopping at the first <i>whitespace</i> character found. A terminating null character is automatically added at the end of the stored sequence.
p	Pointer address	A sequence of characters representing a pointer. The particular format used depends on the system and library implementation, but it is the same as the one used to format %p in fprintf.
[characters]	Scanset	Any number of the characters specified between the brackets. A dash (-) that is not the first character may produce non-portable behavior in some library implementations.
[^characters]	Negated scanset	Any number of characters none of them specified as <i>characters</i> between the brackets.
n	Count	No input is consumed. The number of characters read so far from <i>stdin</i> is stored in the pointed location.
%	%	A % followed by another % matches a single %.

int scanf (const char * format, ...);

%[*][width][length]specifier

The *format specifier* can also contain sub-specifiers: *asterisk* (*), *width* and *length* (in that order), which are optional and follow these specifications:

sub-specifier	description
*	An optional starting asterisk indicates that the data is to be read from the stream but ignored (i.e. it is not stored in the location pointed by an argument).
width	Specifies the maximum number of characters to be read in the current reading operation (optional).
length	One of hh, h, l, ll, j, z, t, L (optional). This alters the expected type of the storage pointed by the corresponding argument (see below).

This is a chart showing the types expected for the corresponding arguments where input is stored (both with and without a *length* sub-specifier):

	specifiers					
length	d i	u o x	f e g a	c s [] [^]	p	n
(none)	int*	unsigned int*	float*	char*	void**	int*
hh	signed char*	unsigned char*				signed char*
h	short int*	unsigned short int*				short int*
l	long int*	unsigned long int*	double*	wchar_t*		long int*
ll	long long int*	unsigned long long int*				long long int*
j	intmax_t*	uintmax_t*				intmax_t*
z	size_t*	size_t*				size_t*
t	ptrdiff_t*	ptrdiff_t*				ptrdiff_t*
L			long double*			

Note: Yellow rows indicate specifiers and sub-specifiers introduced by C99.

int scanf (const char * format, ...);

/* scanf example */

#include <stdio.h>

```
int main ()
{
    char str [80];
    int i;

    printf ("Enter your family name: ");
    scanf ("%79s",str);
    printf ("Enter your age: ");
    scanf ("%d",&i);
    printf ("Mr. %s , %d years old.\n",str,i);
    printf ("Enter a hexadecimal number: ");
    scanf ("%x",&i);
    printf ("You have entered %#x (%d).\n",i,i);

    return 0;
}
```

Saída:

Enter your family name: Soulie

Enter your age: 29

Mr. Soulie , 29 years old.

Enter a hexadecimal number: ff

You have entered 0xff (255).

//entradas em linha

int x, y;

```
scanf("%d%d",&x,&y);          //20 10
printf("%d/%d=%d\n",x,y,x/y); //20/10=2
```

```
scanf("%d %d",&x,&y);          //20 10
printf("%d/%d=%d\n",x,y,x/y); //20/10=2
```

```
scanf("%d,%d",&x,&y);          //20,10
printf("%d/%d=%d\n",x,y,x/y); //20/10=2
```

//usando o *

int x, y;

```
scanf("%d%*c%d",&x,&y);        //20/10
printf("%d/%d=%d",x,y,x/y);    //20/10=2
```

//usando width

char name[5];

```
scanf("%4s",name); // "1234567890"
printf("%s",name); // "1234"
```

//usando length

short int i, j;

```
scanf("%hd",&i); //40100 -limite 32767
printf("%hd",i); // -25436
```

```
scanf("%d",&j); //40100 -limite 32767
printf("%d",j); // -25436
```

//usando scanset

char number[50];

```
scanf("%[123456789]s",number); // "89456"
printf("%s",number);           // "89456"
```

```
scanf("%[123456789]s",number); // "89056"
printf("%s",number);           // "89"
```

```
scanf("%[0-9]s",number); // "1234567890"
printf("%s",number);     // "1234567890"
```

```
scanf("%[^f]s",number); // "manifesto"
printf("%s",number);    // "mani"
```

int sscanf (const char * s, const char * format, ...);

- Lê dados de *s no formato;
- Retorna quantos itens carregados;
- O especificador de formato é igual ao do scanf

```
/* sscanf example */
```

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

```
int main ()
```

```
{
```

```
    char sentence []="Rudolph is 12 years old.";
```

```
    char str1[20], str2[20], str3[20], str4[20];
```

```
    int i;
```

```
    sscanf (sentence,"%s %s %d %s %s",str1,str2,&i,str3,str4);
```

```
    printf ("%s %s %d %s %s\n",str1,str2,i,str3,str4);
```

```
    return 0;
```

```
}
```

char * gets (char * str);

- Lê caracteres de stdin e armazena-os em str;
- Retorna a string carregada;
- Aceita espaços em branco e tab;
- Não aceita delimitar tamanho;

```
/* gets example */  
#include <stdio.h>
```

```
int main()  
{  
    char string [256];  
    printf ("Insert your full address: ");  
    gets (string);  
    printf ("Your address is: %s\n",string);  
    return 0;  
}
```


char * fgets (char * str, int num, FILE * stream);

- Lê caracteres de entrada padrão e armazena-os em str;
- Retorna a string carregada;
- Aceita espaços em branco e tab;
- Aceita delimitar tamanho e '\0' faz parte do tamanho;
- Pode armazenar um '\n';

```
/* fgets example */  
#include <stdio.h>
```

```
int main()  
{  
    char string [256];  
    printf ("Insert your full address: ");  
    fgets (string,10,stdin);  
    printf ("Your address is: %s\n",string);  
    return 0;  
}
```

int printf (const char * format, ...);

- Imprime dados no formato em stdout;
- Retorna quantos caracteres escritos;
- O especificador de formato é

%[flags][width][.precision][length]specifier

int printf (const char * format, ...);

%[flags][width][.precision][length]specifier

The *format specifier* can also contain sub-specifiers: *flags*, *width*, *.precision* and *modifiers* (in that order), which are optional and follow these specifications:

<i>flags</i>	description
-	Left-justify within the given field width; Right justification is the default (see <i>width</i> sub-specifier).
+	Forces to precede the result with a plus or minus sign (+ or -) even for positive numbers. By default, only negative numbers are preceded with a - sign.
(space)	If no sign is going to be written, a blank space is inserted before the value.
#	Used with o, x or X specifiers the value is preceeded with 0, 0x or 0X respectively for values different than zero. Used with a, A, e, E, f, F, g or G it forces the written output to contain a decimal point even if no more digits follow. By default, if no digits follow, no decimal point is written.
0	Left-pads the number with zeroes (0) instead of spaces when padding is specified (see <i>width</i> sub-specifier).

<i>width</i>	description
(number)	Minimum number of characters to be printed. If the value to be printed is shorter than this number, the result is padded with blank spaces. The value is not truncated even if the result is larger.
*	The <i>width</i> is not specified in the <i>format</i> string, but as an additional integer value argument preceding the argument that has to be formatted.

<i>.precision</i>	description
<i>.number</i>	For integer specifiers (d, i, o, u, x, X): <i>precision</i> specifies the minimum number of digits to be written. If the value to be written is shorter than this number, the result is padded with leading zeros. The value is not truncated even if the result is longer. A <i>precision</i> of 0 means that no character is written for the value 0. For a, A, e, E, f and F specifiers: this is the number of digits to be printed after the decimal point (by default, this is 6). For g and G specifiers: This is the maximum number of significant digits to be printed. For s: this is the maximum number of characters to be printed. By default all characters are printed until the ending null character is encountered. If the period is specified without an explicit value for <i>precision</i> , 0 is assumed.
,	The <i>precision</i> is not specified in the <i>format</i> string, but as an additional integer value argument preceding the argument that has to be formatted.

int printf (const char * format, ...);

```
/* printf example */  
#include <stdio.h>
```

```
int main()  
{  
    printf ("Characters: %c %c \n", 'a', 65);  
    printf ("Decimals: %d %ld\n", 1977, 650000L);  
    printf ("Preceding with blanks: %10d \n", 1977);  
    printf ("Preceding with zeros: %010d \n", 1977);  
    printf ("Some different radices: %d %x %o %#x %#o \n", 100, 100, 100, 100, 100);  
    printf ("floats: %4.2f %+.0e %E \n", 3.1416, 3.1416, 3.1416);  
    printf ("Width trick: %*d \n", 5, 10);  
    printf ("%s \n", "A string");  
    return 0;  
}
```

Saída:

Characters: a A

Decimals: 1977 650000

Preceding with blanks: 1977

Preceding with zeros: 0000001977

Some different radices: 100 64 144 0x64 0144

floats: 3.14 +3e+000 3.141600E+000

Width trick: 10

A string

int sprintf (char * str, const char * format, ...);

- Armazena dados no formato em *str;
- Retorna quantos caracteres escritos;
- O especificador de formato é
%[flags][width][.precision][length]specifier

```
/* sprintf example */  
#include <stdio.h>
```

```
int main ()  
{  
    char buffer [50];  
    int n, a=5, b=3;  
    n=sprintf (buffer, "%d plus %d is %d", a, b, a+b);  
    printf ("[%s] is a string %d chars long\n",buffer,n);  
    return 0;  
}
```

Saída:

[5 plus 3 is 8] is a string 13 chars long

int puts (const char * str);

- Imprime uma string e salta para linha seguinte;
- Retorna um valor não negativo caso nenhum erro ocorra;

```
/* puts example : hello world! */  
#include <stdio.h>
```

```
int main ()  
{  
    char string [] = "Hello world!";  
    puts (string);  
}
```

Saída:


Hello world!

Bibliotecas de manipulação de String

- Classificação
- Conversão de letras
- Conversão de valores
- Cópia
- Concatenação
- Comparação
- Pesquisa
- Outras

Bibliotecas de manipulação de String

- De classificação

isalnum	Check if character is alphanumeric (function)
isalpha	Check if character is alphabetic (function)
isblank 	Check if character is blank (function)
iscntrl	Check if character is a control character (function)
isdigit	Check if character is decimal digit (function)
isgraph	Check if character has graphical representation (function)
islower	Check if character is lowercase letter (function)
isprint	Check if character is printable (function)
ispunct	Check if character is a punctuation character (function)
isspace	Check if character is a white-space (function)
isupper	Check if character is uppercase letter (function)
isxdigit	Check if character is hexadecimal digit (function)

Bibliotecas de manipulação de String

```
/* isalnum example - Check if character is alphanumeric */  
#include <stdio.h>  
#include <ctype.h>  
int main ()  
{  
    int i;  
    char str[]="c3po...";  
    i=0;  
    while (isalnum(str[i])) i++;  
    printf ("The first %d characters are alphanumeric.\n",i);  
    return 0;  
}
```

Saída:

The first 4 characters are alphanumeric.

Bibliotecas de manipulação de String

```
/* isalpha example - Check if character is alphabetic */
#include <stdio.h>
#include <ctype.h>
int main ()
{
    int i=0;
    char str[]="C++";
    while (str[i])
    {
        if (isalpha(str[i])) printf ("character %c is alphabetic\n",str[i]);
        else printf ("character %c is not alphabetic\n",str[i]);
        i++;
    }
    return 0;
}
```

Saída:

character C is alphabetic
character + is not alphabetic
character + is not alphabetic

Bibliotecas de manipulação de String

```
/* isblank example - Check if character is blank */  
#include <stdio.h>  
#include <ctype.h>  
int main ()  
{  
    char c;  
    int i=0;  
    char str[]="Example sentence to test isblank\n";  
    while (str[i])  
    {  
        c=str[i];  
        if (isblank(c)) c='\n';  
        putchar (c);  
        i++;  
    }  
    return 0;  
}
```

Saída:
Example
sentence
to
test
isblank

Bibliotecas de manipulação de String

```
/* iscntrl example - Check if character is a control character */
#include <stdio.h>
#include <ctype.h>
int main ()
{
    int i=0;
    char str[]="first line \n second line \n";
    while (!iscntrl(str[i]))
    {
        putchar (str[i]);
        i++;
    }
    return 0;
}
```

Saída:

first line

Bibliotecas de manipulação de String

```
/* isdigit example - Check if character is decimal digit */
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
int main ()
{
    char str[]="1776ad";
    int year;
    if (isdigit(str[0]))
    {
        year = atoi (str);
        printf ("The year that followed %d was %d.\n",year,year+1);
    }
    return 0;
}
```

Saída:

The year that followed 1776 was 1777

Bibliotecas de manipulação de String

```
/* isgraph example - Check if character has graphical representation */
#include <stdio.h>
#include <ctype.h>
int main ()
{
    FILE * pFile;
    int c;
    pFile=fopen ("myfile.txt","r");
    if (pFile)
    {
        do {
            c = fgetc (pFile);
            if (isgraph(c)) putchar (c);
        } while (c != EOF);
        fclose (pFile);
    }
}
```

Saída:

Imprime o conteúdo de um arquivo sem espaços em branco e caracteres especiais

Bibliotecas de manipulação de String

```
/* islower example - Check if character is lowercase letter */
#include <stdio.h>
#include <ctype.h>
int main ()
{
    int i=0;
    char str[]="Test String.\n";
    char c;
    while (str[i])
    {
        c=str[i];
        if (islower(c)) c=toupper(c);
        putchar (c);
        i++;
    }
    return 0;
}
```

Saída:

TEST STRING.

Bibliotecas de manipulação de String

```
/* isprint example - Check if character is printable */
#include <stdio.h>
#include <ctype.h>
int main ()
{
    int i=0;
    char str[]="first line \n second line \n";
    while (isprint(str[i]))
    {
        putchar (str[i]);
        i++;
    }
    return 0;
}
```

Saída:

first line

Bibliotecas de manipulação de String

```
/* ispunct example - Check if character is a punctuation character */
#include <stdio.h>
#include <ctype.h>
int main ()
{
    int i=0;
    int cx=0;
    char str[]="Hello, welcome!";
    while (str[i])
    {
        if (ispunct(str[i])) cx++;
        i++;
    }
    printf ("Sentence contains %d punctuation characters.\n", cx);
    return 0;
}
```

Saída:

Sentence contains 2 punctuation characters.

Bibliotecas de manipulação de String

```
/* isspace example - Check if character is a white-space */
#include <stdio.h>
#include <ctype.h>
int main ()
{
    char c;
    int i=0;
    char str[]="Example sentence to test isspace\n";
    while (str[i])
    {
        c=str[i];
        if (isspace(c)) c='\n';
        putchar (c);
        i++;
    }
    return 0;
}
```

Saída:

Example
sentence
to
test
isspace

For the "C" locale, white-space characters are any of:

' '	(0x20)	space (SPC)
'\t'	(0x09)	horizontal tab (TAB)
'\n'	(0x0a)	newline (LF)
'\v'	(0x0b)	vertical tab (VT)
'\f'	(0x0c)	feed (FF)
'\r'	(0x0d)	carriage return (CR)

Bibliotecas de manipulação de String

```
/* isupper example - Check if character is uppercase letter */  
#include <stdio.h>  
#include <ctype.h>  
int main ()  
{  
    int i=0;  
    char str[]="Test String.\n";  
    char c;  
    while (str[i])  
    {  
        c=str[i];  
        if (isupper(c)) c=tolower(c);  
        putchar (c);  
        i++;  
    }  
    return 0;  
}
```

Saída:

test string.

Bibliotecas de manipulação de String

```
/* isxdigit example - Check if character is hexadecimal digit */
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
int main ()
{
    char str[]="ffff";
    long int number;
    if (isxdigit(str[0]))
    {
        number = strtol (str,NULL,16);
        printf ("The hexadecimal number %lx is %ld.\n",number,number);
    }
    return 0;
}
```

Saída:

The hexadecimal number ffff is 65535.

Bibliotecas de manipulação de String

- De conversão de letras

tolower	Convert uppercase letter to lowercase (function)
toupper	Convert lowercase letter to uppercase (function)

Bibliotecas de manipulação de String

```
/* tolower example - Convert uppercase letter to lowercase */  
#include <stdio.h>  
#include <ctype.h>  
int main ()  
{  
    int i=0;  
    char str[]="Test String.\n";  
    char c;  
    while (str[i])  
    {  
        c=str[i];  
        putchar (tolower(c));  
        i++;  
    }  
    return 0;  
}
```

Saída:
test string.

Bibliotecas de manipulação de String

```
/* toupper example - Convert lowercase letter to uppercase */
#include <stdio.h>
#include <ctype.h>
int main ()
{
    int i=0;
    char str[]="Test String.\n";
    char c;
    while (str[i])
    {
        c=str[i];
        putchar (toupper(c));
        i++;
    }
    return 0;
}
```

Saída:

TEST STRING.

For the first set, here is a map of how the original 127-character ASCII set is considered by each function (an x indicates that the function returns true on that character)

ASCII values	characters	isctrl	isblank	isspace	isupper	islower	isalpha	isdigit	isxdigit	isalnum	ispunct	isgraph	isprint
0x00 .. 0x08	NUL, (other control codes)	x											
0x09	tab ('\t')	x	x	x									
0x0A .. 0x0D	(white-space control codes: '\f', '\v', '\n', '\r')	x		x									
0x0E .. 0x1F	(other control codes)	x											
0x20	space (' ')		x	x									x
0x21 .. 0x2F	!"#\$%&'()*+,-./										x	x	x
0x30 .. 0x39	0123456789							x	x	x		x	x
0x3A .. 0x40	:;<=>?@										x	x	x
0x41 .. 0x46	ABCDEF				x		x		x	x		x	x
0x47 .. 0x5A	GHIJKLMNOPQRSTUVWXYZ				x		x			x		x	x
0x5B .. 0x60	[\] ^ _ `										x	x	x
0x61 .. 0x66	abcdef					x	x		x	x		x	x
0x67 .. 0x7A	ghijklmnopqrstuvwxyz					x	x			x		x	x
0x7B .. 0x7E	{ } ~										x	x	x
0x7F	(DEL)	x											

The characters in the extended character set (above 0x7F) may belong to diverse categories depending on the locale and the platform. As a general rule, `ispunct`, `isgraph` and `isprint` return true on these for the standard C locale on most platforms supporting extended character sets.

Bibliotecas de manipulação de String

- Mais de conversão

atof	Convert string to double (function)
atoi	Convert string to integer (function)
atol	Convert string to long integer (function)
atoll <small>C++11</small>	Convert string to long long integer (function)
strtod	Convert string to double (function)
strtof <small>C++11</small>	Convert string to float (function)
strtol	Convert string to long integer (function)
strtold <small>C++11</small>	Convert string to long double (function)
strtoll <small>C++11</small>	Convert string to long long integer (function)
strtoul	Convert string to unsigned long integer (function)
strtoull <small>C++11</small>	Convert string to unsigned long long integer (function)

Bibliotecas de manipulação de String

```
/* atof example: sine calculator - Convert string to double */
#include <stdio.h>    /* printf, fgets */
#include <stdlib.h>    /* atof */
#include <math.h>      /* sin */

int main ()
{
    double n,m;
    double pi=3.1415926535;
    char buffer[256];
    printf ("Enter degrees: ");
    fgets (buffer,256,stdin);
    n = atof (buffer);
    m = sin (n*pi/180);
    printf ("The sine of %f degrees is %f\n" , n, m);
    return 0;
}
```

Saída:

Enter degrees: 45

The sine of 45.000000 degrees is 0.707101

Bibliotecas de manipulação de String

```
/* atoi example - Convert string to integer */
#include <stdio.h>    /* printf, fgets */
#include <stdlib.h>   /* atoi */

int main ()
{
    int i;
    char buffer[256];
    printf ("Enter a number: ");
    fgets (buffer, 256, stdin);
    i = atoi (buffer);
    printf ("The value entered is %d. Its double is %d.\n",i,i*2);
    return 0;
}
```

Saída:

Enter a number: 73

The value entered is 73. Its double is 146.

Bibliotecas de manipulação de String

```
/* atoi example - Convert string to long integer */
#include <stdio.h>    /* printf, fgets */
#include <stdlib.h>   /* atoi */

int main ()
{
    long int li;
    char buffer[256];
    printf ("Enter a long number: ");
    fgets (buffer, 256, stdin);
    li = atoi(buffer);
    printf ("The value entered is %ld. Its double is %ld.\n",li,li*2);
    return 0;
}
```

Saída:

- Enter a number: 567283
- The value entered is 567283. Its double is 1134566.

Bibliotecas de manipulação de String

```
/* strtod example - Convert string to double */
#include <stdio.h>    /* printf, NULL */
#include <stdlib.h>   /* strtod */

int main ()
{
    char szOrbits[] = "365.24 29.53";
    char* pEnd;
    double d1, d2;
    d1 = strtod (szOrbits, &pEnd);
    d2 = strtod (pEnd, NULL);
    printf ("The moon completes %.2f orbits per Earth year.\n", d1/d2);
    return 0;
}
```

Saída:

The moon completes 12.37 orbits per Earth year.

Bibliotecas de manipulação de String

```
/* strtod example - Convert string to float */
#include <stdio.h>    /* printf, NULL */
#include <stdlib.h>    /* strtod */

int main ()
{
    char szOrbits[] = "686.97 365.24";
    char* pEnd;
    float f1, f2;
    f1 = strtod (szOrbits, &pEnd);
    f2 = strtod (pEnd, NULL);
    printf ("One martian year takes %.2f Earth years.\n", f1/f2);
    return 0;
}
```

Saída:

One martian year takes 1.88 Earth years.

Bibliotecas de manipulação de String

```
/* strtol example - Convert string to long integer */
#include <stdio.h>    /* printf */
#include <stdlib.h>   /* strtol */

int main ()
{
    char szNumbers[] = "2001 60c0c0 -1101110100110100100000 0x6fffff";
    char * pEnd;
    long int li1, li2, li3, li4;
    li1 = strtol (szNumbers,&pEnd,10);
    li2 = strtol (pEnd,&pEnd,16);
    li3 = strtol (pEnd,&pEnd,2);
    li4 = strtol (pEnd,NULL,0);
    printf ("The decimal equivalents are: %ld, %ld, %ld and %ld.\n", li1, li2, li3, li4);
    return 0;
}
```

Saída:

The decimal equivalents are: 2001, 6340800, -3624224 and 7340031

Bibliotecas de manipulação de String

- De cópia

memcpy	Copy block of memory (function)
memmove	Move block of memory (function)
strcpy	Copy string (function)
strncpy	Copy characters from string (function)

Bibliotecas de manipulação de String

```
/* strcpy example - Copy string */
#include <stdio.h>
#include <string.h>

int main ()
{
    char str1[]="Sample string";
    char str2[40];
    char str3[40];
    strcpy (str2,str1);
    strcpy (str3,"copy successful");
    printf ("str1: %s\nstr2: %s\nstr3: %s\n",str1,str2,str3);
    return 0;
}
```

Saída:

str1: Sample string
str2: Sample string
str3: copy successful

Bibliotecas de manipulação de String

```
/* strncpy example - Copy characters from string */
#include <stdio.h>
#include <string.h>

int main ()
{
    char str1[] = "To be or not to be";
    char str2[40];
    char str3[40];

    /* copy to sized buffer (overflow safe): */
    strncpy ( str2, str1, sizeof(str2) );

    /* partial copy (only 5 chars): */
    strncpy ( str3, str2, 5 );
    str3[5] = '\0'; /* null character manually added */

    puts (str1);
    puts (str2);
    puts (str3);

    return 0;
}
```

Saída:

To be or not to be

To be or not to be

To be

Bibliotecas de manipulação de String

- De concatenação

strcat	Concatenate strings (function)
strncat	Append characters from string (function)

Bibliotecas de manipulação de String

```
/* strcat example - Concatenate strings */
```

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

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

```
int main ()
```

```
{
```

```
    char str[80];
```

```
    strcpy (str,"these ");
```

```
    strcat (str,"strings ");
```

```
    strcat (str,"are ");
```

```
    strcat (str,"concatenated.");
```

```
    puts (str);
```

```
    return 0;
```

```
}
```

Saída:

these strings are concatenated.

Bibliotecas de manipulação de String

```
/* strncat example - Append characters from string */
```

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

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

```
int main ()
```

```
{
```

```
    char str1[20];
```

```
    char str2[20];
```

```
    strcpy (str1, "To be ");
```

```
    strcpy (str2, "or not to be");
```

```
    strncat (str1, str2, 6);
```

```
    puts (str1);
```

```
    return 0;
```

```
}
```

Saída:

To be or not

Bibliotecas de manipulação de String

- De comparação

memcmp	Compare two blocks of memory (function)
strcmp	Compare two strings (function)
strcoll	Compare two strings using locale (function)
strncmp	Compare characters of two strings (function)
strxfrm	Transform string using locale (function)

Bibliotecas de manipulação de String

```
/* memcmp example - Compare two blocks of memory */
#include <stdio.h>
#include <string.h>

int main ()
{
    char buffer1[] = "DWgaOtP12df0";
    char buffer2[] = "DWGAOTP12DF0";

    int n;

    n=memcmp ( buffer1, buffer2, sizeof(buffer1) );

    if (n>0) printf ("'%s' is greater than '%s'.\n",buffer1,buffer2);
    else if (n<0) printf ("'%s' is less than '%s'.\n",buffer1,buffer2);
    else printf ("'%s' is the same as '%s'.\n",buffer1,buffer2);

    return 0;
}
```

Saída:

'DWgaOtP12df0' is greater than 'DWGAOTP12DF0'.

Bibliotecas de manipulação de String

```
/* strcmp example - Compare two strings */
```

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

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

```
int main ()
```

```
{
```

```
    char key[] = "apple";
```

```
    char buffer[80];
```

```
    do {
```

```
        printf ("Guess my favorite fruit? ");
```

```
        fflush (stdout);
```

```
        scanf ("%79s",buffer);
```

```
    } while (strcmp (key,buffer) != 0);
```

```
    puts ("Correct answer!");
```

```
    return 0;
```

```
}
```

Return Value

Returns an integral value indicating the relationship between the strings:

return value	indicates
<0	the first character that does not match has a lower value in <i>ptr1</i> than in <i>ptr2</i>
0	the contents of both strings are equal
>0	the first character that does not match has a greater value in <i>ptr1</i> than in <i>ptr2</i>

Saída:

Guess my favourite fruit? orange

Guess my favourite fruit? apple

Correct answer!

Bibliotecas de manipulação de String

```
/* strcmp example - Compare characters of two strings */
#include <stdio.h>
#include <string.h>

int main ()
{
    char str[][5] = { "R2D2" , "C3PO" , "R2A6" };
    int n;
    puts ("Looking for R2 astromech droids...");
    for (n=0 ; n<3 ; n++)
        if (strcmp (str[n],"R2xx",2) == 0)
        {
            printf ("found %s\n",str[n]);
        }
    return 0;
}
```

Saída:

Looking for R2 astromech droids...

found R2D2

found R2A6

Bibliotecas de manipulação de String

- De pesquisa

memchr	Locate character in block of memory (function)
strchr	Locate first occurrence of character in string (function)
strcspn	Get span until character in string (function)
strpbrk	Locate characters in string (function)
strrchr	Locate last occurrence of character in string (function)
strspn	Get span of character set in string (function)
strstr	Locate substring (function)
strtok	Split string into tokens (function)

Bibliotecas de manipulação de String

```
/* memchr example - Locate character in block of memory */  
#include <stdio.h>  
#include <string.h>  
  
int main ()  
{  
    char * pch;  
    char str[] = "Example string";  
    pch = (char*) memchr (str, 'p', strlen(str));  
    if (pch!=NULL)  
        printf ("\"p' found at position %d.\n", pch-str+1);  
    else  
        printf ("\"p' not found.\n");  
    return 0;  
}
```

Saída:

'p' found at position 5.

Bibliotecas de manipulação de String

```
/* strchr example - Locate first occurrence of character in string */
#include <stdio.h>
#include <string.h>

int main ()
{
    char str[] = "This is a sample string";
    char * pch;
    printf ("Looking for the 's' character in \"%s\"...\n",str);
    pch=strchr(str,'s');
    while (pch!=NULL)
    {
        printf ("found at %d\n",pch-str+1);
        pch=strchr(pch+1,'s');
    }
    return 0;
}
```

Saída:

```
Looking for the 's' character in "This is a sample string"...
found at 4
found at 7
found at 11
found at 18
```


Bibliotecas de manipulação de String

```
/* strcspn example - Get span until character in string */
```

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

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

```
int main ()
```

```
{
```

```
    char str[] = "fcba73";
```

```
    char keys[] = "1234567890";
```

```
    int i;
```

```
    i = strcspn (str,keys);
```

```
    printf ("The first number in str is at position %d.\n",i+1);
```

```
    return 0;
```

```
}
```

Saída:

The first number in str is at position 5

Obtém a posição em str da ocorrência de keys que aparece primeiro em str.

Bibliotecas de manipulação de String

`/* strpbrk example - Locate first occurrence of characters in string */`

`#include <stdio.h>`

`#include <string.h>`

```
int main ()
{
    char str[] = "This is a sample string";
    char key[] = "aeiou";
    char * pch;
    printf ("Vowels in '%s': ",str);
    pch = strpbrk (str, key);
    while (pch != NULL)
    {
        printf ("%c ", *pch);
        pch = strpbrk (pch+1,key);
    }
    printf ("\n");
    return 0;
}
```

Saída:

Vowels in 'This is a sample string': i i a a e i

Bibliotecas de manipulação de String

```
/* strrchr example - Locate last occurrence of character in string */  
#include <stdio.h>  
#include <string.h>  
  
int main ()  
{  
    char str[] = "This is a sample string";  
    char * pch;  
    pch=strrchr(str,'s');  
    printf ("Last occurrence of 's' found at %d \n",pch-str+1);  
    return 0;  
}
```

Saída:

Last occurrence of 's' found at 18

Bibliotecas de manipulação de String

```
/* strspn example - Get span of character set in string */
#include <stdio.h>
#include <string.h>

int main ()
{
    int i;
    char strtext[] = "86t1234h";
    char cset[] = "1234567890";

    i = strspn (strtext,cset);
    printf ("The initial number has %d digits.\n",i);
    return 0;
}
```

Saída:

The initial number has 2 digits.

Obtém quantas ocorrências iniciais de strtext aparecem em cset.

Bibliotecas de manipulação de String

```
/* strstr example - Locate substring */
```

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

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

```
int main ()
```

```
{
```

```
    char str[] = "This is a simple string";
```

```
    char * pch;
```

```
    pch = strstr (str, "simple");
```

```
    strncpy (pch, "sample", 6);
```

```
    puts (str);
```

```
    return 0;
```

```
}
```

Saída:

This is a sample string

Obtém a primeira ocorrência de “simple” em str.

Bibliotecas de manipulação de String

```
/* strtok example - Split string into tokens */
#include <stdio.h>
#include <string.h>

int main ()
{
    char str[] = "- This, a sample string.";
    char * pch;
    printf ("Splitting string \"%s\" into tokens:\n",str);
    pch = strtok (str, " ,.-");
    while (pch != NULL)
    {
        printf ("%s\n",pch);
        pch = strtok (NULL, " ,.-");
    }
    return 0;
}
```

Saída:

Splitting string "- This, a sample string." into tokens:

This

a

sample

string

Bibliotecas de manipulação de String

- Outras

strerror	Get pointer to error message string (function)
strlen	Get string length (function)

Bibliotecas de manipulação de String

```
/* strerror example : error list */
```

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

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

```
#include <errno.h>
```

```
int main ()
```

```
{
```

```
    FILE * pFile;
```

```
    pFile = fopen ("unexist.ent","r");
```

```
    if (pFile == NULL)
```

```
        printf ("Error opening file unexist.ent: %s\n",strerror(errno));
```

```
    return 0;
```

```
}
```

Saída:

Error opening file unexist.ent: No such file or directory

Bibliotecas de manipulação de String

```
/* strlen example */  
#include <stdio.h>  
#include <string.h>  
  
int main ()  
{  
    char szInput[256]={'m','y',' ','s','t','u','d','y','\0'};  
    printf ("The sentence '%s' ",szInput);  
    printf ("is %d characters.\n",strlen(szInput));  
    return 0;  
}
```

Saída:

The sentence 'my study' is 8 characters.

Exercícios

- 1- Escreva um programa que receba um caractere do teclado e verifique-o com cada uma das funções manipulação de caracteres. O programa deve imprimir o valor retornado por cada função.
- 2- Escreva um programa que coloque uma linha de texto no array de caracteres `s [100]` utilizando a função `gets`. Envie a linha para o dispositivo de saída em letras maiúsculas e em letras minúsculas.
- 3- Escreva um programa que receba 4 strings que representem 4 valores de ponto flutuante, converta as strings em valores **double**, some os valores e imprima a soma dos quatro valores.
- 4- Escreva um programa que receba um número de telefone como uma string na forma **(555) 555-5555**. O programa deve usar a função `strtok` para extrair o código de área como um token, os três primeiros dígitos do número de telefone como outro token e os últimos quatro dígitos do número de telefone como mais outro token. Os sete dígitos do número de telefone devem ser concatenados e formar uma única string. O programa deve converter a string do código de área em **int** e converter a string do número de telefone em **long**. Tanto o código de área como o número de telefone devem ser impressos.

Exercícios

- 5- Escreva um programa que use a geração de números aleatórios para criar frases. O programa deve usar quatro arrays de ponteiros **char** denominados **artigo**, **substantivo**, **verbo** e **preposição**. O programa deve criar uma frase selecionando uma palavra aleatoriamente de cada array na seguinte ordem: **artigo**, **substantivo**, **verbo**, **preposição**, **artigo** e **substantivo**. A medida que cada palavra for selecionada, ela deverá ser concatenada às palavras anteriores em um array que seja suficientemente grande para conter a frase inteira. As palavras devem ser separadas por espaços. Quando a frase final for enviada para o dispositivo de saída, ela deve iniciar com uma letra maiúscula e terminar com um ponto. O programa deve gerar 20 de tais frases.

Os arrays devem ser preenchidos como se segue: o array **artigo** deve conter "o", "um", "algum", "todo" e "qualquer"; o array **substantivo** deve conter "menino", "homem", "cachorro", "carro", "gato"; o array **verbo** deve conter "passou", "pulou", "correu", "saltou", "andou"; o array **preposição** deve conter "sobre", "sob", "ante", "até" e "com".

- 6- Escreva um programa que receba uma linha de texto, divida a linha por meio da função **strtok** e imprima o resultado na ordem inversa.

Exercícios

- 7- Escreva um programa que receba do teclado uma linha de texto e uma string. Usando a função **strstr**, localize a primeira ocorrência da string de busca na linha de texto e atribua a localização daquela string à variável **buscaPtr** do tipo **char ***. Se a string de busca for encontrada, imprima o restante da linha de texto iniciando com a string de busca. A seguir, use novamente **strstr** para localizar a próxima ocorrência da string de busca na linha de texto. Se for encontrada uma segunda ocorrência, imprima o restante da linha de texto iniciando com a segunda ocorrência. Sugestão: A segunda chamada a **strstr** deve conter **buscaPtr + 1** como primeiro argumento.
- 8- Escreva um programa baseado no programa do Exercício 7 que receba várias linhas de texto e uma string de busca, e use a função **strstr** para determinar o total de ocorrências da string na linha de texto. Imprima o resultado.
- 9- Escreva um programa que receba várias linhas de texto e um caractere de busca, e use a função **strchr** para determinar o total de ocorrências do caractere nas linhas de texto.
- 10- Escreva um programa baseado no Exercício 9 que receba várias linhas de texto e use a função **strchr** para determinar o total de ocorrências de cada letra do alfabeto nas linhas de texto. As letras maiúsculas e minúsculas devem ser contadas em conjunto. Armazene o total de cada letra em um array e imprima os valores em um formato de tabela depois de os

Exercícios

- 11- A tabela ASCII mostra as representações dos códigos numéricos dos caracteres do conjunto ASCII. Estude essa tabela e depois diga se cada uma das afirmações a seguir é verdadeira ou falsa.
- a) A letra "A" vem antes da letra "B".
 - b) O dígito "9" vem antes do dígito "0".
 - c) Os símbolos usados normalmente para adição, subtração, multiplicação e divisão vem antes de qualquer um dos dígitos.
 - d) Os dígitos vem antes das letras.
 - e) Se um programa de ordenação colocar strings na ordem ascendente, o símbolo do parêntese direito virá antes do símbolo do parêntese esquerdo.
- 12- Escreva um programa que receba um código ASCII e imprima o caractere correspondente. Modifique esse programa para que ele gere todos os códigos possíveis de três dígitos no intervalo de 000 a 255 e tente imprimir os caracteres correspondentes. O que acontece quando esse programa é executado?