Capitolo 8 – Caratteri e Stringhe

Outline

Obiettivi

- In questo capitolo, impareremo a:
  - Utilizzare le funzioni della libreria per la manipolazione di caratteri (`ctype`).
  - Utilizzare le funzioni di input/output di caratteri e stringhe e le funzioni della libreria standard di input/output (`stdio`).
  - Utilizzare le funzioni di conversione di stringhe della libreria di utility generali (`stdlib`).
  - Utilizzare le funzioni di elaborazione di stringhe della libreria per la manipolazione di stringhe (`string`).
  - Apprezzare il potere delle funzioni delle librerie come mezzo per ottenere la riusabilità del software.

Introduzione

- Introduzione di alcune funzioni della libreria standard
  - Facile elaborazione di caratteri e stringhe
  - I programmi possono elaborare caratteri, stringhe, linee di testo, e blocchi di memoria
- Tali tecniche vengono utilizzate per
  - Word processors
  - Page layout software
  - Typesetting programs

Concetti fondamentali di stringhe e caratteri

- Caratteri
  - Scrivere blocchi di programmi
    - Ogni programma è una sequenza di caratteri raggruppati significativamente
  - Carattere costante
    - Un valore `int` rappresentato come un carattere fra apici
    - `'z'` rappresenta il valore intero di `z`
- Stringhe
  - Serie di caratteri trattati come singola unità
    - Possono includere lettere, cifre e caratteri speciali `(*, /, $)`
  - Il letterale stringa (string constant) – scritta fra virgolette
    - `"Hello"`
  - Le stringhe sono array di caratteri
    - La stringa è un puntatore al primo carattere
    - Il valore di una stringa è l'indirizzo del primo carattere

Concetti fondamentali di stringhe e caratteri

- Definizioni di stringhe
  - Definita come un array di caratteri o come una variabile di tipo `char *`

```
char color[] = "blue";
char *colorPtr = "blue";
```

  - Ricordare che le stringhe rappresentate come array di caratteri terminano con `'\0'`
    - `color` ha `5 elementi`
- Acquisizione di stringhe
  - Utilizzo di `scanf`

```
scanf("%s", word);
```

    - Copia l'input in `word[]`
    - Non è necessario il `&` (poiché una stringa è già un puntatore)
  - Ricordare di lasciare spazio nell'array per `'\0'`

Libreria per la manipolazione di caratteri

- Libreria per la manipolazine di caratteri
  - Contiene le funzioni per effettuare utili test e manipolazioni su caratteri
  - Ogni funzione riceve un carattere (un `int`) o `EOF` come argomento
- La seguente diapositiva contiene una tabella di tutte le funzioni presenti in `<ctype.h>`

Libreria per la manipolazione di caratteri

Prototype	Description
`int isdigit( int c );`	Returns `true` if `c` is a digit and `false` otherwise.
`int isalpha( int c );`	Returns `true` if `c` is a letter and `false` otherwise.
`int isalnum( int c );`	Returns `true` if `c` is a digit or a letter and `false` otherwise.
`int isxdigit( int c );`	Returns `true` if `c` is a hexadecimal digit character and `false` otherwise.
`int islower( int c );`	Returns `true` if `c` is a lowercase letter and `false` otherwise.
`int isupper( int c );`	Returns `true` if `c` is an uppercase letter; `false` otherwise.
`int tolower( int c );`	If `c` is an uppercase letter, `tolower` returns `c` as a lowercase letter. Otherwise, `tolower` returns the argument unchanged.
`int toupper( int c );`	If `c` is a lowercase letter, `toupper` returns `c` as an uppercase letter. Otherwise, `toupper` returns the argument unchanged.
`int isspace( int c );`	Returns `true` if `c` is a white-space character—newline (`'\n'`), space (`' '`), form feed (`'\f'`), carriage return (`'\r'`), horizontal tab (`'\t'`), or vertical tab (`'\v'`)—and `false` otherwise
`int iscntrl( int c );`	Returns `true` if `c` is a control character and `false` otherwise.
`int ispunct( int c );`	Returns `true` if `c` is a printing character other than a space, a digit, or a letter and `false` otherwise.
`int isprint( int c );`	Returns `true` value if `c` is a printing character including space (`' '`) and `false` otherwise.
`int isgraph( int c );`	Returns `true` if `c` is a printing character other than space (`' '`) and `false` otherwise.


Outline

fig08_02.c (Part 1 of 2)

```
1  /* Fig. 8.2: fig08_02.c
2     Using functions isdigit, isalpha, isalnum, and isxdigit */
3  #include <stdio.h>
4  #include <ctype.h>
5
6  int main()
7  {
8     printf( "%s\n%s%s\n%s%s\n\n", "According to isdigit: ",
9        isdigit( '8' ) ? "8 is a " : "8 is not a ", "digit",
10       isdigit( '#' ) ? "# is a " : "# is not a ", "digit" );
11
12    printf( "%s\n%s%s\n%s%s\n%s%s\n%s%s\n\n",
13       "According to isalpha:",
14       isalpha( 'A' ) ? "A is a " : "A is not a ", "letter",
15       isalpha( 'b' ) ? "b is a " : "b is not a ", "letter",
16       isalpha( '&' ) ? "& is a " : "& is not a ", "letter",
17       isalpha( '4' ) ? "4 is a " : "4 is not a ", "letter" );
18
19    printf( "%s\n%s%s\n%s%s\n%s%s\n\n",
20       "According to isalnum:",
21       isalnum( 'A' ) ? "A is a " : "A is not a ",
22       "digit or a letter",
23       isalnum( '8' ) ? "8 is a " : "8 is not a ",
24       "digit or a letter",
25       isalnum( '#' ) ? "# is a " : "# is not a ",
26       "digit or a letter" );
27
```

```
28    printf( "%s\n%s%s\n%s%s\n%s%s\n%s%s\n%s%s\n",
29       "According to isxdigit:",
30       isxdigit( 'F' ) ? "F is a " : "F is not a ",
31       "hexadecimal digit",
32       isxdigit( 'J' ) ? "J is a " : "J is not a ",
33       "hexadecimal digit",
34       isxdigit( '7' ) ? "7 is a " : "7 is not a ",
35       "hexadecimal digit",
36       isxdigit( '$' ) ? "$ is a " : "$ is not a ",
37       "hexadecimal digit",
38       isxdigit( 'f' ) ? "f is a " : "f is not a ",
39       "hexadecimal digit" );
40
41    return 0; /* indicates successful termination */
42
43 } /* end main */
```

Outline

fig08_02.c (Part 2 of 2)


```
According to isdigit:
8 is a digit
# is not a digit

According to isalpha:
A is a letter
b is a letter
& is not a letter
4 is not a letter

According to isalnum:
A is a digit or a letter
8 is a digit or a letter
# is not a digit or a letter

According to isxdigit:
F is a hexadecimal digit
J is not a hexadecimal digit
7 is a hexadecimal digit
$ is not a hexadecimal digit
f is a hexadecimal digit
```

Outline

Program Output


```
1  /* Fig. 8.3: fig08_03.c
2     Using functions islower, isupper, tolower, toupper */
3  #include <stdio.h>
4  #include <ctype.h>
5
6  int main()
7  {
8     printf( "%s\n%s%s\n%s%s\n%s%s\n\n",
9             "According to islower:",
10            islower( 'p' ) ? "p is a " : "p is not a ",
11            "lowercase letter",
12            islower( 'P' ) ? "P is a " : "P is not a ",
13            "lowercase letter",
14            islower( '5' ) ? "5 is a " : "5 is not a ",
15            "lowercase letter",
16            islower( '!' ) ? "! is a " : "! is not a ",
17            "lowercase letter" );
18
19    printf( "%s\n%s%s\n%s%s\n%s%s\n\n",
20            "According to isupper:",
21            isupper( 'D' ) ? "D is an " : "D is not an ",
22            "uppercase letter",
23            isupper( 'd' ) ? "d is an " : "d is not an ",
24            "uppercase letter",
25            isupper( '8' ) ? "8 is an " : "8 is not an ",
26            "uppercase letter",
27            isupper( '$' ) ? "$ is an " : "$ is not an ",
28            "uppercase letter" );
29
```

Outline

fig08_03.c (Part 1 of 2)


```
30    printf( "%s%c\n%s%c\n%s%c\n%s%c\n",
31            "u converted to uppercase is ", toupper( 'u' ),
32            "7 converted to uppercase is ", toupper( '7' ),
33            "$ converted to uppercase is ", toupper( '$' ),
34            "L converted to lowercase is ", tolower( 'L' ) );
35
36    return 0; /* indicates successful termination */
37
38 } /* end main */
```

```
According to islower:
p is a lowercase letter
P is not a lowercase letter
5 is not a lowercase letter
! is not a lowercase letter

According to isupper:
D is an uppercase letter
d is not an uppercase letter
8 is not an uppercase letter
$ is not an uppercase letter

u converted to uppercase is U
7 converted to uppercase is 7
$ converted to uppercase is $
L converted to lowercase is l
```

Outline

fig08_03.c (Part 2 of 2)

Program Output

Outline

fig08_04.c (Part 1 of 2)

```
1  /* Fig. 8.4: fig08_04.c
2     Using functions isspace, iscntrl, ispunct, isprint, isgraph */
3  #include <stdio.h>
4  #include <ctype.h>
5
6  int main()
7  {
8     printf( "%s\n%s%s%s\n%s%s%s%s\n%s%s%s\n\n",
9        "According to isspace:",
10       "Newline", isspace( '\n' ) ? " is a " : " is not a ",
11       "whitespace character", "Horizontal tab",
12       isspace( '\t' ) ? " is a " : " is not a ",
13       "whitespace character",
14       isspace( '%' ) ? "% is a " : "% is not a ",
15       "whitespace character" );
16
17    printf( "%s\n%s%s%s\n%s%s%s\n\n", "According to iscntrl:",
18       "Newline", iscntrl( '\n' ) ? " is a " : " is not a ",
19       "control character", iscntrl( '$' ) ? "$ is a " :
20       "$ is not a ", "control character" );
21
```


Outline

fig08_04.c (Part 2 of 2)

```
22    printf( "%s\n%s%s\n%s%s\n%s%s\n\n",
23       "According to ispunct:",
24       ispunct( ';' ) ? "; is a " : "; is not a ",
25       "punctuation character",
26       ispunct( 'Y' ) ? "Y is a " : "Y is not a ",
27       "punctuation character",
28       ispunct( '#' ) ? "# is a " : "# is not a ",
29       "punctuation character" );
30
31    printf( "%s\n%s%s\n%s%s%s\n\n", "According to isprint:",
32       isprint( '$' ) ? "$ is a " : "$ is not a ",
33       "printing character",
34       "Alert", isprint( '\a' ) ? " is a " : " is not a ",
35       "printing character" );
36
37    printf( "%s\n%s%s\n%s%s%s\n",  "According to isgraph:",
38       isgraph( 'Q' ) ? "Q is a " : "Q is not a ",
39       "printing character other than a space",
40       "Space", isgraph( ' ' ) ? " is a " : " is not a ",
41       "printing character other than a space" );
42
43    return 0; /* indicates successful termination */
44
45 } /* end main */
```


Outline

Program Output

```
According to isspace:
Newline is a whitespace character
Horizontal tab is a whitespace character
% is not a whitespace character

According to iscntrl:
Newline is a control character
$ is not a control character

According to ispunct:
; is a punctuation character
Y is not a punctuation character
# is a punctuation character

According to isprint:
$ is a printing character
Alert is not a printing character

According to isgraph:
Q is a printing character other than a space
Space is not a printing character other than a space
```


Funzioni per la conversione di stringhe

- Funzioni di conversione
  - In `<stdlib.h>` (libreria di utility generali)
- Converte stringhe di cifre in valori interi e floating-point

Function prototype	Function description
`double atof( const char *nPtr );`	Converts the string `nPtr` to `double`.
`int atoi( const char *nPtr );`	Converts the string `nPtr` to `int`.
`long atol( const char *nPtr );`	Converts the string `nPtr` to `long int`.
`double strtod( const char *nPtr, char **endPtr );`	Converts the string `nPtr` to `double`.
`long strtol( const char *nPtr, char **endPtr, int base );`	Converts the string `nPtr` to `long`.
`unsigned long strtoul( const char *nPtr, char **endPtr, int base );`	Converts the string `nPtr` to `unsigned long`.

Outline

fig 08_06.c

```
1  /* Fig. 8.6: fig08_06.c
2     Using atof */
3  #include <stdio.h>
4  #include <stdlib.h>
5
6  int main()
7  {
8     double d; /* variable to hold converted string */
9
10    d = atof( "99.0" );
11
12    printf( "%s%.3f\n%s%.3f\n",
13            "The string \"99.0\" converted to double is ", d,
14            "The converted value divided by 2 is ",
15            d / 2.0 );
16
17    return 0; /* indicates successful termination */
18
19 } /* end main */
```

Program Output

```
The string "99.0" converted to double is 99.000
The converted value divided by 2 is 49.500
```


Outline

fig08_07.c

```
1  /* Fig. 8.7: fig08_07.c
2     Using atoi */
3  #include <stdio.h>
4  #include <stdlib.h>
5
6  int main()
7  {
8     int i; /* variable to hold converted string */
9
10    i = atoi( "2593" );
11
12    printf( "%s%d\n%s%d\n",
13            "The string \"2593\" converted to int is ", i,
14            "The converted value minus 593 is ", i - 593 );
15
16    return 0; /* indicates successful termination */
17
18 } /* end main */
```

Program Output

```
The string "2593" converted to int is 2593
The converted value minus 593 is 2000
```


Outline

fig08_08.c

```
1  /* Fig. 8.8: fig08_08.c
2     Using atol */
3  #include <stdio.h>
4  #include <stdlib.h>
5
6  int main()
7  {
8     long l; /* variable to hold converted string */
9
10    l = atol( "1000000" );
11
12    printf( "%s%ld\n%s%ld\n",
13            "The string \"1000000\" converted to long int is ", l,
14            "The converted value divided by 2 is ", l / 2 );
15
16    return 0; /* indicates successful termination */
17
18 } /* end main */
```

Program Output

```
The string "1000000" converted to long int is 1000000
The converted value divided by 2 is 500000
```


Outline

fig08_09.c

```
1  /* Fig. 8.9: fig08_09.c
2     Using strtod */
3  #include <stdio.h>
4  #include <stdlib.h>
5
6  int main()
7  {
8     /* initialize string pointer */
9     const char *string = "51.2% are admitted";
10
11    double d;        /* variable to hold converted sequence */
12    char *stringPtr; /* create char pointer */
13
14    d = strtod( string, &stringPtr );
15
16    printf( "The string \"%s\" is converted to the\n", string );
17    printf( "double value %.2f and the string \"%s\"\n", d, stringPtr );
18
19    return 0; /* indicates successful termination */
20
21 } /* end main */
```

Program Output

```
The string "51.2% are admitted" is converted to the
double value 51.20 and the string "% are admitted"
```

Outline

fig08_10.c

```
1  /* Fig. 8.10: fig08_10.c
2     Using strtol */
3  #include <stdio.h>
4  #include <stdlib.h>
5
6  int main()
7  {
8     const char *string = "-1234567abc"; /* initialize string pointer */
9
10    char *remainderPtr; /* create char pointer */
11    long x;             /* variable to hold converted sequence */
12
13    x = strtol( string, &remainderPtr, 0 );
14
15    printf( "%s\"%s\"\n%s%ld\n%s\"%s\"\n%s%ld\n",
16            "The original string is ", string,
17            "The converted value is ", x,
18            "The remainder of the original string is ",
19            remainderPtr,
20            "The converted value plus 567 is ", x + 567 );
21
22    return 0; /* indicates successful termination */
23
24 } /* end main */
```

Program Output

```
The original string is "-1234567abc"
The converted value is -1234567
The remainder of the original string is "abc"
The converted value plus 567 is -1234000
```


Outline

fig08_11.c

```
1  /* Fig. 8.11: fig08_11.c
2     Using strtoul */
3  #include <stdio.h>
4  #include <stdlib.h>
5
6  int main()
7  {
8     const char *string = "1234567abc"; /* initialize string pointer */
9     unsigned long x;    /* variable to hold converted sequence */
10    char *remainderPtr; /* create char pointer */
11
12    x = strtoul( string, &remainderPtr, 0 );
13
14    printf( "%s\"%s\"\n%s%lu\n%s\"%s\"\n%s%lu\n",
15            "The original string is ", string,
16            "The converted value is ", x,
17            "The remainder of the original string is ",
18            remainderPtr,
19            "The converted value minus 567 is ", x - 567 );
20
21    return 0; /* indicates successful termination */
22
23 } /* end main */
```

Program Output

```
The original string is "1234567abc"
The converted value is 1234567
The remainder of the original string is "abc"
The converted value minus 567 is 1234000
```


Funzioni della libreria Input/Output

- Funzioni in `<stdio.h>`
- Usate per manipolare caratteri e stringhe

Function prototype	Function description
`int getchar( void );`	Inputs the next character from the standard input and re turns it as an integer.
`char *gets( char *s );`	Inputs characters from the standard input into the array `s` until a newline or end-of-file character is encountered. A terminating null character is appended to the array.
`int putchar( int c );`	Prints the character stored in `c`.
`int puts( const char *s );`	Prints the string `s` followed by a newline character.
`int sprintf( char *s, const char *format, ... );`	Equivalent to `printf`, except the output is stored in the array `s` instead of printing it on the screen.
`int sscanf( char *s, const char *format, ... );`	Equivalent to `scanf`, except the input is read from the array `s` instead of reading it from the keyboard.

Outline

fig08_13.c (Part 1 of 2)

```
1  /* Fig. 8.13: fig08_13.c
2     Using gets and putchar */
3  #include <stdio.h>
4
5  int main()
6  {
7     char sentence[ 80 ]; /* create char array */
8
9     void reverse( const char * const sPtr ); /* prototype */
10
11    printf( "Enter a line of text:\n" );
12
13    /* use gets to read line of text */
14    gets( sentence );
15
16    printf( "\nThe line printed backwards is:\n" );
17    reverse( sentence );
18
19    return 0; /* indicates successful termination */
20
21 } /* end main */
22
```

```
23 /* recursively outputs characters in string in reverse order */
24 void reverse( const char * const sPtr )
25 {
26    /* if end of the string */
27    if ( sPtr[ 0 ] == '\0' ) {
28       return;
29    } /* end if */
30    else { /* if not end of the string */
31       reverse( &sPtr[ 1 ] );
32
33       putchar( sPtr[ 0 ] ); /* use putchar to display character */
34    } /* end else */
35
36 } /* end function reverse */
```

```
Enter a line of text:
Characters and Strings

The line printed backwards is:
sgnirtS dna sretcarahC
Enter a line of text:
able was I ere I saw elba

The line printed backwards is:
able was I ere I saw elba
```

Outline

fig08_13.c (Part 1 of 2)

Program Output


```
1  /* Fig. 8.14: fig08_14.c
2     Using getchar and puts */
3  #include <stdio.h>
4
5  int main()
6  {
7     char c;              /* variable to hold character input by user */
8     char sentence[ 80 ]; /* create char array */
9     int i = 0;           /* initialize counter i */
10
11    /* prompt user to enter line of text */
12    puts( "Enter a line of text:" );
13
14    /* use getchar to read each character */
15    while ( ( c = getchar() ) != '\n') {
16       sentence[ i++ ] = c;
17    } /* end while */
18
19    sentence[ i ] = '\0';
20
21    /* use puts to display sentence */
22    puts( "\nThe line entered was:" );
23    puts( sentence );
24
25    return 0; /* indicates successful termination */
26
27 } /* end main */
```

Outline

fig8_14.c


```
Enter a line of text:
This is a test.

The line entered was:
This is a test.
```

Outline

Program Output


```
1  /* Fig. 8.15: fig08_15.c
2     Using sprintf */
3  #include <stdio.h>
4
5  int main()
6  {
7     char s[ 80 ]; /* create char array */
8     int x;        /* define x */
9     double y;     /* define y */
10
11    printf( "Enter an integer and a double:\n" );
12    scanf( "%d%lf", &x, &y );
13
14    sprintf( s, "integer:%6d\ndouble:%8.2f", x, y );
15
16    printf( "%s\n%s\n",
17            "The formatted output stored in array s is:", s );
18
19    return 0; /* indicates successful termination */
20
21 } /* end main */
```

```
Enter an integer and a double:
298 87.375
The formatted output stored in array s is:
integer:   298
double:   87.38
```

Outline

fig08_15.c

Program Output

```c
/* Fig. 8.16: fig08_16.c
   Using sscanf */
#include <stdio.h>

int main()
{
   char s[] = "31298 87.375"; /* initialize array s */
   int x;                     /* define x */
   double y;                  /* define y */

   sscanf( s, "%d%lf", &x, &y );

   printf( "%s\n%s%6d\n%s%8.3f\n",
           "The values stored in character array s are:",
           "integer:", x, "double:", y );

   return 0; /* indicates successful termination */

} /* end main */
```

Outline

fig08_16.c

Program Output

```
The values stored in character array s are:
integer: 31298
double:  87.375
```


Funzioni per la manipolazione di stringhe

- La libreria delle funzioni per la manipolazione di stringhe contiene funzioni per
  - Manipolare stringhe
  - Ricercare stringhe
  - Tokenizzare stringhe
  - Determinare la lunghezza di una stringa

Function prototype	Function description
`char *strcpy( char *s1, const char *s2 )`	Copies string `s2` into array `s1`. The value of `s1` is returned.
`char *strncpy( char *s1, const char *s2, size_t n )`	Copies at most `n` characters of string `s2` into array `s1`. The value of `s1` is returned.
`char *strcat( char *s1, const char *s2 )`	Appends string `s2` to array `s1`. The first character of `s2` overwrites the terminating null character of `s1`. The value of `s1` is returned.
`char *strncat( char *s1, const char *s2, size_t n )`	Appends at most `n` characters of string `s2` to array `s1`. The first character of `s2` overwrites the terminating null character of `s1`. The value of `s1` is returned.


```c
/* Fig. 8.18: fig08_18.c
   Using strcpy and strncpy */
#include <stdio.h>
#include <string.h>

int main()
{
   char x[] = "Happy Birthday to You"; /* initialize char array x */
   char y[ 25 ];                       /* create char array y */
   char z[ 15 ];                       /* create char array z */

   /* copy contents of x into y */
   printf( "%s%s\n%s%s\n",
           "The string in array x is: ", x,
           "The string in array y is: ", strcpy( y, x ) );

   /* copy first 14 characters of x into z. Does not copy null
      character */
   strncpy( z, x, 14 );

   z[ 14 ] = '\0'; /* append '\0' to z's contents */
   printf( "The string in array z is: %s\n", z );

   return 0; /* indicates successful termination */

} /* end main */
```

Outline

fig08_18.c

Program Output

```
The string in array y is: Happy Birthday to You
The string in array z is: Happy Birthday
```



```c
/* Fig. 8.19: fig08_19.c
   Using strcat and strncat */
#include <stdio.h>
#include <string.h>

int main()
{
   char s1[ 20 ] = "Happy "; /* initialize char array s1 */
   char s2[] = "New Year ";  /* initialize char array s2 */
   char s3[ 40 ] = "";       /* initialize char array s3 */

   printf( "s1 = %s\ns2 = %s\n", s1, s2 );

   /* concatenate s2 to s1 */
   printf( "strcat( s1, s2 ) = %s\n", strcat( s1, s2 ) );

   /* concatenate first 6 characters of s1 to s3. Place '\0'
      after last character */
   printf( "strncat( s3, s1, 6 ) = %s\n", strncat( s3, s1, 6 ) );

   /* concatenate s1 to s3 */
   printf( "strcat( s3, s1 ) = %s\n", strcat( s3, s1 ) );

   return 0; /* indicates successful termination */

} /* end main */
```

Outline

fig08_19.c

```
s1 = Happy
s2 = New Year
strcat( s1, s2 ) = Happy New Year
strncat( s3, s1, 6 ) = Happy
strcat( s3, s1 ) = Happy Happy New Year
```


Program Output

Funzioni di confronto fra stringhe

- Confrontare le stringhe
  - Il computer confronta i codici numerici ASCII dei caratteri delle stringhe

```
int strcmp( const char *s1, const char *s2 );
```

  - Confronta la stringa `s1` con `s2`
  - Restituisce un numero negativo se `s1 < s2`, zero se `s1 == s2` o un numero positivo se `s1 > s2`

```
int strncmp( const char *s1, const char *s2,
      size_t n );
```

  - Confronta `n` caratteri della stringa `s1` con `s2`
  - Restituisce gli stessi valori come sopra


```
1  /* Fig. 8.21: fig08_21.c
2     Using strcmp and strncmp */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8     const char *s1 = "Happy New Year"; /* initialize char pointer */
9     const char *s2 = "Happy New Year"; /* initialize char pointer */
10    const char *s3 = "Happy Holidays"; /* initialize char pointer */
11
12    printf("%s%s\n%s%s\n%s%s\n\n%s%2d\n%s%2d\n%s%2d\n\n",
13           "s1 = ", s1, "s2 = ", s2, "s3 = ", s3,
14           "strcmp(s1, s2) = ", strcmp( s1, s2 ),
15           "strcmp(s1, s3) = ", strcmp( s1, s3 ),
16           "strcmp(s3, s1) = ", strcmp( s3, s1 ) );
17
18    printf("%s%2d\n%s%2d\n%s%2d\n",
19           "strncmp(s1, s3, 6) = ", strncmp( s1, s3, 6 ),
20           "strncmp(s1, s3, 7) = ", strncmp( s1, s3, 7 ),
21           "strncmp(s3, s1, 7) = ", strncmp( s3, s1, 7 ) );
22
23    return 0; /* indicates successful termination */
24
25 } /* end main */
```


fig08_21.c


```
s1 = Happy New Year
s2 = Happy New Year
s3 = Happy Holidays

strcmp(s1, s2) =  0
strcmp(s1, s3) =  1
strcmp(s3, s1) = -1

strncmp(s1, s3, 6) =  0
strncmp(s1, s3, 7) =  1
strncmp(s3, s1, 7) = -1
```


Program Output

Funzioni di ricerca di stringhe

Function prototype	Function description
`char *strchr( const char *s, int c );`	Locates the first occurrence of character `c` in string `s`. If `c` is found, a pointer to `c` in `s` is returned. Otherwise, a `NULL` pointer is returned.
`size_t strcspn( const char *s1, const char *s2 );`	Determines and returns the length of the initial segment of string `s1` consisting of characters not contained in string `s2`.
`size_t strspn( const char *s1, const char *s2 );`	Determines and returns the length of the initial segment of string `s1` consisting only of characters contained in string `s2`.
`char *strpbrk( const char *s1, const char *s2 );`	Locates the first occurrence in string `s1` of any character in string `s2`. If a character from string `s2` is found, a pointer to the character in string `s1` is returned. Otherwise, a `NULL` pointer is returned.
`char *strrchr( const char *s, int c );`	Locates the last occurrence of `c` in string `s`. If `c` is found, a pointer to `c` in string `s` is returned. Otherwise, a `NULL` pointer is returned.
`char *strstr( const char *s1, const char *s2 );`	Locates the first occurrence in string `s1` of string `s2`. If the string is found, a pointer to the string in `s1` is returned. Otherwise, a `NULL` pointer is returned.
`char *strtok( char *s1, const char *s2 );`	A sequence of calls to `strtok` breaks string `s1` into "tokens"—logical pieces such as words in a line of text—separated by characters contained in string `s2`. The first call contains `s1` as the first argument, and subsequent calls to continue tokenizing the same string contain `NULL` as the first argument. A pointer to the current token is returned by each call. If there are no more tokens when the function is called, `NULL` is returned.

Outline

fig08_23.c (Part 1 of 2)

```
1  /* Fig. 8.23: fig08_23.c
2     Using strchr */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8     const char *string = "This is a test"; /* initialize char pointer */
9     char character1 = 'a';                 /* initialize character1 */
10    char character2 = 'z';                 /* initialize character2 */
11
12    /* if character1 was found in string */
13    if ( strchr( string, character1 ) != NULL ) {
14       printf( "\'%c\' was found in \"%s\".\n",
15               character1, string );
16    } /* end if */
17    else { /* if character1 was not found */
18       printf( "\'%c\' was not found in \"%s\".\n",
19               character1, string );
20    } /* end else */
21
```


Outline

fig08_23.c (Part 2 of 2)

```
22    /* if character2 was found in string */
23    if ( strchr( string, character2 ) != NULL ) {
24       printf( "\'%c\' was found in \"%s\".\n",
25               character2, string );
26    } /* end if */
27    else { /* if character2 was not found */
28       printf( "\'%c\' was not found in \"%s\".\n",
29               character2, string );
30    } /* end else */
31
32    return 0; /* indicates successful termination */
33
34 } /* end main */
```

Program Output

```
'a' was found in "This is a test".
'z' was not found in "This is a test".
```


Outline

fig08_24.c

```
1  /* Fig. 8.24: fig08_24.c
2     Using strcspn */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8     /* initialize two char pointers */
9     const char *string1 = "The value is 3.14159";
10    const char *string2 = "1234567890";
11
12    printf( "%s%s\n%s%s\n\n%s\n%s%u",
13            "string1 = ", string1, "string2 = ", string2,
14            "The length of the initial segment of string1",
15            "containing no characters from string2 = ",
16            strcspn( string1, string2 ) );
17
18    return 0; /* indicates successful termination */
19
20 } /* end main */
```

Program Output

```
string1 = The value is 3.14159
string2 = 1234567890

The length of the initial segment of string1
containing no characters from string2 = 13
```

Outline

fig08_25.c

```
1  /* Fig. 8.25: fig08_25.c
2     Using strpbrk */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8     const char *string1 = "This is a test"; /* initialize char pointer */
9     const char *string2 = "beware";         /* initialize char pointer */
10
11    printf( "%s\"%s\"\n'%c'%s\n\"%s\"\n",
12            "Of the characters in ", string2,
13            *strpbrk( string1, string2 ),
14            " is the first character to appear in ", string1 );
15
16    return 0; /* indicates successful termination */
17
18 } /* end main */
```

Program Output

```
Of the characters in "beware"
'a' is the first character to appear in
"This is a test"
```


Outline

fig08_26.c

```
1  /* Fig. 8.26: fig08_26.c
2     Using strrchr */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8     /* initialize char pointer */
9     const char *string1 = "A zoo has many animals "
10                          "including zebras";
11    int c = 'z'; /* initialize c */
12
13    printf( "%s\n%s'%c'%s\"%s\"\n",
14            "The remainder of string1 beginning with the",
15            "last occurrence of character ", c,
16            " is: ", strrchr( string1, c ) );
17
18    return 0; /* indicates successful termination */
19
20 } /* end main */
```

Program Output

```
The remainder of string1 beginning with the
last occurrence of character 'z' is: "zebras"
```


Outline

fig08_27.c

```
1  /* Fig. 8.27: fig08_27.c
2     Using strspn */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8     /* initialize two char pointers */
9     const char *string1 = "The value is 3.14159";
10    const char *string2 = "aehi lsTuv";
11
12    printf( "%s%s\n%s%s\n\n%s\n%s%u\n",
13            "string1 = ", string1, "string2 = ", string2,
14            "The length of the initial segment of string1",
15            "containing only characters from string2 = ",
16            strspn( string1, string2 ) );
17
18    return 0; /* indicates successful termination */
19
20 } /* end main */
```

Program Output

```
string1 = The value is 3.14159
string2 = aehi lsTuv

The length of the initial segment of string1
containing only characters from string2 = 13
```


Outline

fig08_28.c

```
1  /* Fig. 8.28: fig08_28.c
2     Using strstr */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8     const char *string1 = "abcdefabcdef"; /* initialize char pointer */
9     const char *string2 = "def";          /* initialize char pointer */
10
11    printf( "%s%s\n%s%s\n\n%s\n%s%s\n",
12            "string1 = ", string1, "string2 = ", string2,
13            "The remainder of string1 beginning with the",
14            "first occurrence of string2 is: ",
15            strstr( string1, string2 ) );
16
17    return 0; /* indicates successful termination */
18
19 } /* end main */
```

Program Output

```
string1 = abcdefabcdef
string2 = def

The remainder of string1 beginning with the
first occurrence of string2 is: defabcdef
```

Outline

fig08_29.c

```
1  /* Fig. 8.29: fig08_29.c
2     Using strtok */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8     /* initialize array string */
9     char string[] = "This is a sentence with 7 tokens";
10    char *tokenPtr; /* create char pointer */
11
12    printf( "%s\n%s\n\n%s\n",
13            "The string to be tokenized is:", string,
14            "The tokens are:" );
15
16    tokenPtr = strtok( string, " " ); /* begin tokenizing sentence */
17
18    /* continue tokenizing sentence until tokenPtr becomes NULL */
19    while ( tokenPtr != NULL ) {
20       printf( "%s\n", tokenPtr );
21       tokenPtr = strtok( NULL, " " ); /* get next token */
22    } /* end while */
23
24    return 0; /* indicates successful termination */
25
26 } /* end main */
```


Outline

Program Output

```
The string to be tokenized is:
This is a sentence with 7 tokens

The tokens are:
This
is
a
sentence
with
7
tokens
```


Funzioni per la manipolazione della memoria

- Funzioni per la manipolazione della memoria
  - In `<stdlib.h>`
  - Manipolano, confrontano, e ricercano blocchi di memoria
  - Possono manipolare qualsiasi blocco di dati
- I parametri puntatore sono `void *`
  - Ogni puntatore può esere assegnato a `void *`, e viceversa
  - `void *` non può essere dereferenziato
    - Ogni funzione riceve un argomento size che specifica il numero di byte (caratteri) da elaborare

Funzioni per la manipolazione della memoria

Function prototype	Function description
`void *memcpy( void *s1, const void *s2, size_t n );`	Copies `n` characters from the object pointed to by `s2` into the object pointed to by `s1`. A pointer to the resulting object is returned.
`void *memmove( void *s1, const void *s2, size_t n );`	Copies `n` characters from the object pointed to by `s2` into the object pointed to by `s1`. The copy is performed as if the characters were first copied from the object pointed to by `s2` into a temporary array and then from the temporary array into the object pointed to by `s1`. A pointer to the resulting object is returned.
`int memcmp( const void *s1, const void *s2, size_t n );`	Compares the first `n` characters of the objects pointed to by `s1` and `s2`. The function returns `0`, less than `0` or greater than `0` if `s1` is equal to, less than or greater than `s2`.
`void *memchr( const void *s, int c, size_t n );`	Locates the first occurrence of `c` (converted to `unsigned char`) in the first `n` characters of the object pointed to by `s`. If `c` is found, a pointer to `c` in the object is returned. Otherwise, `NULL` is returned.
`void *memset( void *s, int c, size_t n );`	Copies `c` (converted to `unsigned char`) into the first `n` characters of the object pointed to by `s`. A pointer to the result is returned.

```c
/* Fig. 8.31: fig08_31.c
   Using memcpy */
#include <stdio.h>
#include <string.h>

int main()
{
   char s1[ 17 ];                  /* create char array s1 */
   char s2[]  = "Copy this string"; /* initialize char array s2 */

   memcpy( s1, s2, 17 );
   printf( "%s\n%s\"%s\"\n",
           "After s2 is copied into s1 with memcpy,",
           "s1 contains ", s1 );

   return 0; /* indicates successful termination */

} /* end main */
```

Outline

fig08_31.c

Program Output

```
After s2 is copied into s1 with memcpy,
s1 contains "Copy this string"
```



```c
/* Fig. 8.32: fig08_32.c
   Using memmove */
#include <stdio.h>
#include <string.h>

int main()
{
   char x[] = "Home Sweet Home"; /* initialize char array x */

   printf( "%s%s\n", "The string in array x before memmove is: ", x );
   printf( "%s%s\n", "The string in array x after memmove is: ",
           memmove( x, &x[ 5 ], 10 ) );

   return 0; /* indicates successful termination */

} /* end main */
```

Outline

fig08_32.c

Program Output

```
The string in array x before memmove is: Home Sweet Home
The string in array x after memmove is: Sweet Home Home
```



```c
/* Fig. 8.33: fig08_33.c
   Using memcmp */
#include <stdio.h>
#include <string.h>

int main()
{
   char s1[] = "ABCDEFG"; /* initialize char array s1 */
   char s2[] = "ABCDXYZ"; /* initialize char array s2 */

   printf( "%s%s\n%s%s\n\n%s%2d\n%s%2d\n%s%2d\n",
           "s1 = ", s1, "s2 = ", s2,
           "memcmp( s1, s2, 4 ) = ", memcmp( s1, s2, 4 ),
           "memcmp( s1, s2, 7 ) = ", memcmp( s1, s2, 7 ),
           "memcmp( s2, s1, 7 ) = ", memcmp( s2, s1, 7 ) );

   return 0; /* indicate successful termination */

} /* end main */
```

Outline

fig08_33.c

Program Output

```
s1 = ABCDEFG
s2 = ABCDXYZ

memcmp( s1, s2, 4 ) =  0
memcmp( s1, s2, 7 ) = -1
memcmp( s2, s1, 7 ) =  1
```



```c
/* Fig. 8.34: fig08_34.c
   Using memchr */
#include <stdio.h>
#include <string.h>

int main()
{
   const char *s = "This is a string"; /* initialize char pointer */

   printf( "%s\'%c\'%s\"%s\"\n",
           "The remainder of s after character ", 'r',
           " is found is ", memchr( s, 'r', 16 ) );

   return 0; /* indicates successful termination */

} /* end main */
```

Outline

Fig8_34.c

Program Output

```
The remainder of s after character 'r' is found is "ring"
```

Ulteriori funzioni

- `char *strerror( int errornum );`
  - Crea un messaggio di errore *system-dependent* o `errornum`
  - Restituisce un puntatore alla stringa
- `size_t strlen( const char *s );`
  - Restituisce il numero di caratteri (primo `NULL`) della stringa `s`

Outline

fig08_35.c

```
1  /* Fig. 8.35: fig08_35.c
2     Using memset */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8     char string1[ 15 ] = "BBBBBBBBBBBBBB"; /* initialize string1 */
9
10    printf( "string1 = %s\n", string1 );
11    printf( "string1 after memset = %s\n", memset( string1, 'b', 7 ) );
12
13    return 0; /* indicates successful termination */
14
15 } /* end main */
```

Program Output

```
string1 = BBBBBBBBBBBBBB
string1 after memset = bbbbbbbBBBBBBB
```


Outline

fig08_37.c

```
1  /* Fig. 8.37: fig08_37.c
2     Using strerror */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8     printf( "%s\n", strerror( 2 ) );
9
10    return 0; /* indicates successful termination */
11
12 } /* end main */
```

Program Output

```
No such file or directory
```


Outline

fig08_38.c

```
1  /* Fig. 8.38: fig08_38.c
2     Using strlen */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8     /* initialize 3 char pointers */
9     const char *string1 = "abcdefghijklmnopqrstuvwxyz";
10    const char *string2 = "four";
11    const char *string3 = "Boston";
12
13    printf("%s\"%s\"%s%lu\n%s\"%s\"%s%lu\n%s\"%s\"%s%lu\n",
14       "The length of ", string1, " is ",
15       ( unsigned long ) strlen( string1 ),
16       "The length of ", string2, " is ",
17       ( unsigned long ) strlen( string2 ),
18       "The length of ", string3, " is ",
19       ( unsigned long ) strlen( string3 ) );
20
21    return 0; /* indicates successful termination */
22
23 } /* end main */
```

Program Output

```
The length of "abcdefghijklmnopqrstuvwxyz" is 26
The length of "four" is 4
The length of "Boston" is 6
```