

```
#include <stdio.h>
FILE *fopen(char *filename, char * mode) -
Apertura di un file (mode: "r" lettura - "w"
scrittura - "a" append)
FILE *freopen(char *filename, char * mode,
FILE *file_pointer) - Riassegna un file
puntatore ad un file diverso.
int fclose(FILE *file_pointer) - Chiude un file
int feof(FILE *file_pointer) - Controlla se e'
stato incontrato un end-of-file in un file.
int fflush(FILE *file_pointer) - Svuota il buffer
di un file.
int getchar(void) - Legge un carattere da
"stdin" (tastiera)
int fgetc(FILE *file_pointer) - Prende un
carattere da un file
char *gets(char *buffer) - Legge una riga da
"stdin" (tastiera)
char *fgets(char *string, int maxchar, FILE
*file_pointer) - Legge una riga da un file.
int printf(char *format_string, ...) - Scrive
output formattato su "stdout" (schermo)
int fprintf(FILE *file_pointer, char
*format_string, ...) - Scrive output
formattato in un file.
int sprintf(char *string, char *format_string,
...) - Scrive output formattato su una stringa
int fputc(int c, FILE *file_pointer) - Scrive un
carattere in un file
int putchar(int c) - Scrive un carattere su
"stdout" (schermo)
int puts(char *string) - Scrive una stringa su
"stdout" (schermo)
int fputs(char *string, FILE *file_pointer) -
Scriva una stringa in un file.
int scanf(char *format_string, args) - Legge
input formattato da "stdin" (tastiera)
int fscanf(FILE *file_pointer, char *format
string, args) - Legge input formattato da file
int sscanf(char *buffer, char *format_string,
args) - Legge input formattato da una stringa
EOF - end of file (costante a valore negativo)
NULL - puntatore nullo (valore 0)
```

```
#include <stdlib.h>
double atof(char *string) - Converte una
stringa in un valore in floating point.
int atoi(char *string) - Converte una stringa
in un valore integer.
int atol(char *string) - Converte una stringa
in un valore long integer.
void exit(int val) - Termina il programma,
restituendo il valore 'val'.
EXIT_FAILURE - costante per segnalare
terminazione senza successo del programma
con exit(); valore diverso da zero
EXIT_SUCCESS - segnala terminazione con
successo del programma con exit(); vale 0
```

```
#include <string.h>
char *strcpy(char *dest, char *src) - Copia
una stringa in un'altra. Restituisce dest
char *strncpy(char *s1, char *s2, size_t n) -
Copia i primi "n" caratteri di s2 in s1.
Restituisce s1
int strcmp(char *s1, char *s2) - Confronta s1
e s2 per determinare l'ordine alfabetico (<0,
s1 prima di s2, 0 uguali, >0 s1 dopo s2)
int strncmp(char *s1, char *s2, size_t n) -
Confronta i primi "n" caratteri di due
stringhe.
char *strncpy(char *s1, char *s2) - Copia s2 in
s1. Restituisce s1
int strlen(char *string) - Determina la
lunghezza di una stringa.
char *strcat(char *s1, char *s2, size_t n) -
Aggiunge s2 a s1. Ritorna s1
char *strncat(char *s1, char *s2, size_t n) -
Aggiunge "n" caratteri di s2 a s1. Ritorna s1
char *strchr(char *string, int c) - Cerca la
prima occorrenza del carattere 'c' in string;
restituisce un puntatore alla prima
occorrenza di c in s, NULL se non presente
char *strrchr(char *string, int c) - Cerca
l'ultima occorrenza del carattere 'c' in string
char* strstr(char* s, char* t) - Restituisce un
puntatore alla prima occorrenza di t
```

all'interno di s. Restituisce NULL se t non è presente in s.

char* strtok(char* s, const char* t) - scompone s in token, i caratteri che delimitano i token sono contenuti in t. Restituisce il puntatore al token (NULL se non ne trova nessuno). Alla prima chiamata in s va inserita la stringa da scomporre e in t i caratteri che delimitano i vari token. Per operare sulla stessa stringa, alle successive chiamate al posto di s si deve passare NULL

```
#include <ctype.h>
int isalnum(int c) - Vero se 'c' e'
alfanumerico.
int isalpha(int c) - Vero se 'c' e' una lettera
dell'alfabeto.
int iscntrl(int c) - Vero se 'c' e' un carattere di
controllo.
int isdigit(int c) - Vero se 'c' e' un numero
decimale.
int islower(int c) - Vero se 'c' e' una lettera
minuscola.
int isprint(int c) - Vero se 'c' e' un carattere
stampabile.
int ispunct(int c) - Vero se 'c' e' un carattere
di punteggiatura.
int isspace(int c) - Vero se 'c' e' un carattere
spazio.
int isupper(int c) - Vero se 'c' e' una lettera
maiuscola.
tolower(int c) - Converte 'c' in minuscolo.
int toupper(int c) - Converte 'c' in maiuscolo.
```

```
#include <math.h>
int abs(int n) - valore assoluto intero
long labs(long n) - valore assoluto long
double fabs(double x) - valore assoluto di x
double acos(double x) - arcocoseno
double asin(double x) - arcseno
double atan(double x) - arcotangente
double atan2(double y, double x) -
arcotangente di y/x.
double ceil(double x) - intero superiore a x
```

double floor(double x) - intero inferiore a x.

double cos(double x) - x in radianti

double sin(double x) - x in radianti

double tan(double x) - x in radianti

double cosh(double x) - coseno iperbolico

double sinh(double x) - seno iperbolico

double tanh(double x) - tangente iperbolica

double exp(double x) - e^x

double log(double x) - log(x).

double log10(double x) - logaritmo base 10

double pow(double x, double y) - x^y

int rand(void) - intero casuale tra 0 e RND_MAX.

int random(int max_num) - valore casuale tra 0 e max_num.

void srand(unsigned seed) - inizializza la sequenza di valori casuali

double sqrt(double x) - radice quadrata

```
#include <limits.h>
INT_MAX - Indica il più grande valore che è
possibile rappresentare con un int.
INT_MIN - Indica il più piccolo valore che è
possibile rappresentare con un int.
LONG_MAX - Indica il più grande valore che è
possibile rappresentare con un long.
LONG_MIN - Indica il più piccolo valore che è
possibile rappresentare con un long.
```

```
#include <float.h>
FLT_MAX, DBL_MAX - Indica il più grande
valore che è possibile rappresentare con un
float (o double)
FLT_MIN, DBL_MIN - Indica il più piccolo
valore che è possibile rappresentare con un
float (o double)
```