stdio.h

fopen

```
FILE *fopen (const char *filename, const char *mode)
```

Datei öffnen. Zulässige Werte für „mode“ sind:

„r“ zum Lesen öffnen
„w“ zum Schreiben erzeugen
„a“ zum Anfügen öffnen oder erzeugen
„r+“ zum Ändern öffnen
„w+“ zum Ändern erzeugen
„a+“ zum Ändern öffnen oder erzeugen

Liefert Datenstrom bei fehlerfreier Ausführung und NULL bei Fehler.

fflush

```
int fflush (FILE *stream)
```

Datenpuffer zurückschreiben. Liefert EOF bei Schreibfehler und 0 bei fehlerfreier Ausführung.

fclose

```
int fclose (FILE *stream)
```

Datei schließen. Liefert EOF beo Fehlern und 0 bei fehlerfreier Ausführung.

remove

```
int remove (const char *filename)
```

Datei löschen. Liefert bei Fehlern einen von 0 verschiedenen Wert.

rename

```
int rename (const char *oldname, const char *newname)
```

Datei umbenennen. Liefert bei Fehlern einen von 0 verschiedenen Wert.

fseek

```
int fseek (FILE *stream, long offset, int origin)
```

Dateiposition setzen. Liefert von 0 verschiedenen Wert bei Fehler.

rewind

```
void rewind (FILE *stream)
```

Dateiposition auf Anfang setzen.

fgetpos

```
int fgetpos (FILE *stream, fpos_t *ptr)
```

Dateiposition auslesen. Liefert einen von 0 verschiedenen Wert. bei Fehler.

fsetpos

```
int fsetpos (FILE *stream, const fpos_t *ptr)
```

Dateiposition setzen. Liefert einen von 0 verschiedenen Wert bei Fehler.

feof

```
int feof (FILE *stream)
```

Liefert von 0 verschiedenen Wert, wenn für stream ein Fehler notiert ist.

printf, fprintf

```
int printf (const char *format, ...)
int fprintf (FILE *stream, const char *format, ...)
```

Daten formatiert ausgeben. Liefert die Anzahl der geschriebenen Zeichen oder negative Zahl bei Fehler. Formatangaben beginnen mit % und enden mit einem Umwandlungszeichen. Dazwischen sind (unter anderem) folgende Angaben möglich:

- Steuerzeichen (flags)
  - \- linksbündige Ausrichtung
  - \+ Zahlen mit Vorzeichen ausgeben
  - 0 führende Nullen bei numerischen Angaben
- minimale Feldbreite
- „.“ trennt Feldbreite von Genauigkeit
- Genauigkeit
- Buchstabe als Längenangabe
- Umwandlungszeichen

d, i	int; dezimal mitVorzeichen
o	int; oktal ohne Vorzeichen
x, X	int; hexadezimal ohne Vorzeichen
u	int; dezimal ohne Vorzeichen
c	int; einzelnes Zeichen
s	char *; Zeichenkette
f	double; dezimal (bei scanf jedoch float)
e, E	double; dezimal in wissenschaftlicher Schreibweise
g, G	double; dezimal je nach Exponent
p	void *; Zeiger
n	int *; Anzahl der ausgegebenen Zeichen

Hinweis: Um ein Prozentzeichen auszugeben wird „%%“ verwendet.

scanf, fscanf

```
int scanf (const char *format, ...)
int fscanf (FILE *stream, const char *format, ...)
```

Daten formatiert einlesen. Alle weiteren Parameter müssen Zeiger sein. Liefert EOF bei Fehler und die Anzahl der umgewandelten und abgelegten Eingaben. Für den Formatstring vgl. printf, fprintf.

getc, fgetc

```
int getc (FILE *stream)
int fgetc (FILE *stream)
```

Nächstes Zeichen als unsigned char z.B. von stdin lesen. Liefert das Zeichen oder EOF bei Fehler.

getchar

```
int getchar (void)
```

Äquivalent zu getc(stdin).

gets

```
char *gets (char *s)
```

Nächste Zeile von stdin auslesen. Liefert s oder NULL bei Fehler.

fgets

```
char *fgets (char *s, int n, FILE *stream)
```

Höchstens die nächsten n-1 Zeichen in s einlesen. Liefert s oder NULL bei Fehler.

putc, fputc

```
int putc (int c, FILE *stream)
int fputc (int c, FILE *stream)
```

Zeichen schreiben. Liefert das Zeichen oder EOF bei Fehler.

putchar

```
int putchar (int c)
```

Äquivalent zu outc (c, stdout)

puts

```
int puts (const char *s)
```

Zeichenkette auf stdout ausgeben. Liefert EOF bei Fehler oder nichtnegativen Wert.

fputs

```
int fputs (const char *s, FILE *stream)
```

Zeichenkette schreiben. Liefert nicht-negativen Wert oder EOF bei Fehler.

ctype.h

Alle Funktionen erwarten als Parameter ein int-Argument und liefern einen von 0 verschiedenen Wert zurück, wenn das Ergebnis wahr ist. Andernfalls wird 0 zurückgegeben.

`isalnum`	wahr, wenn isalpha oder isdigit wahr ist
`isalpha`	wahr, wenn isupper oder islower wahr ist
`iscntrl`	wahr, wenn Steuerzeichen
`isdigit`	wahr, wenn dezimale Ziffer
`isgraph`	wahr, wenn sichtbares Zeichen (kein Leerzeichen)
`islower`	wahr, wenn Kleinbuchstabe (kein Umlaut, kein ß)
`isprint`	wahr, wenn sichtbares Zeichen (auch Leerzeichen)
`ispunct`	wahr, wenn sichtbares Zeichen, außer Leerzeichen, Buchstabe oder Ziffer

`isspace` wahr, wenn Leerzeichen, Seitenvorschub (\f), Zeilentrenner (\n), Wagenrücklauf (\r), Tabulator (\t) oder Vertikaltabulator (\v)
`isupper` wahr, wenn Großbuchstabe (kein Umlaut)
`isxdigit` wahr, wenn hexadezimale Ziffer

tolower
```
int tolower (int c)
```
Unwandlung in Kleinbuchstaben. Liefert das umgewandelte Zeichen.

toupper
```
int toupper (int c)
```
Unwandlung in Großbuchstaben. Liefert das umgewandelte Zeichen.

string.h

strcpy
```
char *strcpy (char *s, const char *ct)
```
Zeichenkette ct in Vektor s kopieren.

strcat
```
char *strcat (char *s, const char *ct)
```
Zeichenkette ct an Zeichenkette s anhängen.

strcmp
```
int strcmp (const char *cs, const char *ct, size_t n)
```
Zeichenketten cs und ct vergleichen.

strchr, strrchr
```
char *strchr (const char *cs, int c)
char *strrchr (const char *cs, int c)
```
Zeichen c in Zeichenkette cs suchen. Liefert Zeiger auf erstes/letztes Zeichen c oder NULL, wenn nicht gefunden.

strstr
```
char *strstr (const char *cs, const char *ct)
```
Zeiger auf erste Kopie von ct in cs oder NULL, wenn nicht gefunden.

strlen
```
size_t strlen (const char *cs)
```
Länge der Zeichenkette cs.

math.h

Diverse mathematische Funktionen wie
`sin (x), cos (x), tan (x), sinh (x), log (x), exp (x), pow (x, y), sqrt (x)`

stdlib.h

atof
```
double atof (const char *s)
```
Zeichenkette in double umwandeln.

atoi
```
int atoi (const char *s)
```
Zeichenkette in int umwandeln.

atol
```
long atol (const char *s)
```
Zeichenkette in long umwandeln.

rand
```
int rand (void)
```
Zufallszahl zwischen 0 und RAND_MAX (>= 32767).

malloc
```
void *malloc (size_t size)
```
Speicherplatz reservieren. Liefert Zeiger auf Speicherplatz oder NULL bei Fehler.

free

```
void free (void *p)
```

Speicherplatz freigeben.

abs

```
int abs (int n)
```

Absolutwert von n

time.h

clock

```
clock_t clock (void)
```

Rechenkernzeit seit Programmausführung.

time

```
time_t time (time_t *tp)
```

Kalenderzeit.

difftime

```
double difftime (time_t time2, time_t time1)
```

time2 – time1 in Sekunden.

limits.h und float.h

Grenzwerte einer Implementierung