Merkblatt zur Programmierung in C

Ein- und Ausgabefunktionen deklariert in <stdio.h>

	Funktion	Beschreibung	Beispiel
Dateibehandlungsfunktionen			`FILE *fp;`
	`fopen()`	Öffnet Datenstrom `fp` zu Datei mit Namen `fname`. Die zweite Zeichenkette (`mode`) beschreibt die Dateizugriffsart (siehe unten).	`fp = fopen(fname, "w+");`
	`fclose()`	Schließt Datei zu Datenstrom `fp`.	`err = fclose(fp);`
	`fputc()`	Schreibt Zeichen `ch` in den Datenstrom `fp`, gibt `EOF` bei Fehler zurück.	`re = fputc(ch, fp);`
	`putc()`	Schreibt Zeichen `ch` in den Datenstrom `fp`, gibt `EOF` bei Fehler zurück.	`re = putc(ch, fp);`
	`fgetc()`	Liest ein Zeichen mit Echo vom Datenstrom `fp`.	`ch = fgetc(fp);`
	`getc()`	Liest ein Zeichen mit Echo (als Macro definiert).	`ch = getc(fp);`
	`ungetc()`	Schreibt Zeichen `ch` zurück in den Datenstrom `fp`.	`re = ungetc(ch, fp);`
	Datentyp für Ein- und Ausgabefunktionen einzelner Zeichen ist `int`, um mit der `EOF` Konstante vergleichen zu können.		
	`fputs()`	Schreibt Zeichenkette `str` in Datenstrom `fp`.	`rstr = fputs(str, fp);`
	`fgets()`	Liest **n-2** Zeichen vom Datenstrom `fp`, speichert diese inkl. `'\n'+'\0'`(!) am Ende in `str` ab. Die Zeichenkette oder `NULL` wird zurückgeben.	`rstr = fgets(str, n, fp);`
	`fprintf()`	Formatierte Ausgabe (siehe separates Merkblatt dazu).	`fprintf(fp, "x = %f", x);`
	`fscanf()`	Formatierte Eingabe (siehe separates Merkblatt dazu).	`n = fscanf(fp, "%f", &x);`
	`fwrite()`	Schreibt `n` Elemente der Bytegröße `size` ab Zeiger p in Datenstrom `fp`.	`fwrite(p, size, n, fp);`
	`fread()`	Liest `n` Elemente der Größe `size` in Bytes vom Datenstrom `fp` in Speicher ab Zeiger `p`.	`m = fread(p, size, n, fp);`
	`ftell()`	Liefert die aktuelle Schreib- Leseposition zurück.	`pos = ftell(fp);`
	`fseek()`	Versetzt Schreib- Leseposition um `offset` bezüglich Reference `ref` (s.u).	`fseek(fp, offset, ref);`
	`feof()`	Liefert 0 solange Schreib- Leseposition noch nicht am Dateiende (`EOF`), ansonsten != 0.	`while (!feof(fp)) { … }`
	`ferror()`	Liefert != 0 falls Fehler beim Lesen oder Schreiben eingetreten sind, ansonsten 0.	`if (ferror(fp)) { … }`
	`fflush()`	Ausgabepuffer für Ausgabedatenstrom `fp` leeren.	`fflush(fp);`
	`rewind()`	Setzt Schreib- Leseposition zurück auf Dateianfang.	`rewind(fp);`
	`remove()`	Löscht die Datei mit Namen `fname`.	`err = remove(fname);`
Standard Ein- und Ausgabe von Zeichen			
	`putchar()`	Äquivalent zu `putc(ch, stdin)`.	`putchar('\n');`
	`getchar()`	Äquivalent zu `getc(stdin)`.	`ch = getchar();`
	`getch()`	Liest ein Zeichen direkt von der Tastatur, ohne Echo (nur in Windows OS). Ab Windows 10 im non-Legacymode müssen 2 Bytes (2x) pro Tastendruck ausgelesen werden! Deklariert in Header Datei `<conio.h>`.	`ch = getch();` `ch = _getwch();` (besser)
Standard Ein- und Ausgabe von Zeichenketten			
	`puts()`	Schreibt Zeichenkette `str` in `stdout`. Wie `fputs()` aber **mit** `'\n'`.	`puts(str);`
	`gets()`	Liest eine Zeichenkette von `stdin` in `str`. Wie `fgets()` nur **ohne** Anzahlbegrenzung `n` und **ohne** `'\n'` in `str`.	`gets(str);`

Die Variablendefinitionen zu den obigen Beispielen:

```
FILE *fp;    // Zeiger auf Dateideskriptor
int err;     // Fehlercode oder 0
int ch, re; // Zeichen (als int zum Vgl. mit EOF)
char *rstr; // Rückgabe-Pointer
float x;     // float-Wert
void *p;     // Zeiger auf Speicherbereich
int size, n, m; // Größe, Anzahl, Rückgabewert
long pos;    // Schreib- Leseposition
long offset;// Versatz in Bytes
int ref;     // (0=SEEK_SET, 1=SEEK_CUR, 2=SEEK_END)
```

```
char fname[N], str[N]; // Zeichenkette
char *mode; // Dateiöffnungsmodus:
```

r	read	Nur lesen	return NULL falls nicht gefunden
r+	read Update	+ Aktualisierung	
w	write	Löscht Inhalt vorhandener Datei!	Erzeugt nicht vorhandene Datei
w+	write Update		
a	append	Nur Anhängen	
a+	append Update	+ Aktualisierung	