Zum Umgang mit Dateien in C:

- Bei der Ausgabe mit **`fprintf`** (allgemeiner: **`printf`**) können auch (minimale) Feldbreite und/oder (maximale) Genauigkeit, letztere angeführt durch einen Punkt, angegeben werden. Bei Verwendung nur einer der beiden Angaben entscheidet die Anwesenheit bzw. Lage des Punktes über ihre Bedeutung. Zur variablen Gestaltung des Formats kann jeweils auch ein Sternchen (**`*`**) angegeben werden; dann wird der Wert durch Umwandlung des nächsten Arguments festgelegt (das ein **`int`**-Wert sein muß). Ein vorangestelltes Minuszeichen (**`-`**) bewirkt jeweils linksbündige Ausgabe:

  `fprintf (filnam,"%*d\n", width, x);`

  bedeutet also, daß in die Datei (Datenstrom) **`filnam`** der Inhalt der **`int`**-Variablen **`x`** mit so vielen Stellen übertragen werden soll, wie die **`int`**-Variable **`width`** angibt. Unzureichende Zahlen-Formate werden ggf. angepaßt. Als konkretes Beispiel betrachte man die folgende Sequenz von Anweisungen.

```
#define SWIDTH     6
#define SPRECISION 5
#define IWIDTH     2
#define IPRECISION 0 /*keine Ausgabe von '0'-ansonsten Anpassung*/
#define MEMBERS  001
fprintf(filnam,"_|%sfor%d|_\n", "Diners", MEMBERS);
fprintf(filnam,"_|%-*.*sfor%*.*d|_\n",
       SWIDTH, SPRECISION, "Diners", IWIDTH, IPRECISION, MEMBERS);
```

  Das Ergebnis ist folgende Ausgabe:

```
_|Dinersfor1|_
_|Diner for 1|_
```

  [Hinweis: Das Format `%<width>.<precision><Type>` bewirkt, falls `<Type>` `f` oder `lf` ist, daß eine `<width>`stellige Zahl mit `<precision>` Nachkommastellen ausgegeben wird. Bei `d` erfolgt die Ausgabe einer `<width>`stelligen ganzen Zahl mit mindestens `<precision>` (notfalls mit Nullen aufgefüllten) Stellen, bei `s` die Ausgabe von mindestens `<width>` Schriftzeichen, darunter `<precision>` aus der spezifizierten Zeichenkette.]

- Die Formatbezeichnungen für **`fscanf()`** sind größtenteils die gleichen wie jene für **`fprintf()`**. Interessant für manche Aufgaben ist, daß ein Sternchen (**`*`**) zwischen dem %-Zeichen und dem Datentyp (vergleichbar der variablen Feldbreite bei **`fprintf()`**) bei **`scanf()`** bewirkt, daß die gelesenen Daten nicht in die angebotenen Variablen übertragen und dort gespeichert werden. Zudem kann man über die Eingabe eines sog. **scanset** (Suchmenge) eine Gruppe oder einen Bereich von Zeichen (in eckigen Klammern, z.B.: **`["ABC"]`**) definieren, die als einzige zu berücksichtigen – bei vorangestelltem Dachzeichen (**`^`**) nicht zu berücksichtigen – sind; d.h.:

  `fscanf (filnam,"%*[^A-Z]s", string);`

  bedeutet, daß aus der Datei (Datenstrom) **`filnam`** Zeichen (Zeichenkette **`s`**, passend zur Definition von **`string`** – z.B.: **`"abcABCdefDEF"`**) solange ausgelesen werden sollen, bis der erste Großbuchstabe (A-Z) angetroffen wird (aber ohne diesen auszulesen, wegen **`^`** – z.B. hier: `"abc"`); die ausgelesenen Zeichen sollen dann aber

nicht in **string** gespeichert werden (wie durch * angezeigt wird); nur der Dateizeiger wird weiterpositioniert.

- Die Anweisung

```
fgets (string, LENGTH, filnam);
```

liest aus der formatierten Datei eine ganze Zeile aus (beendet mit **"\n")** bzw. maximal die Anzahl von Zeichen, die durch **LENGTH** (als **#define** oder als Zahl) angegeben wird.

- Die Datenübertragung zwischen Arbeitsspeicher und Datei erfolgt über einen Puffer (Zwischenspeicher); dessen physische Leerung wird vom Betriebssystem gesteuert (und ausgelöst, wenn der Puffer voll ist, das Programm beendet werden soll u.s.w.).

  fflush() sorgt nach **fprintf()** für eine Leerung des Datenpuffers zu einer vom Programmablauf bestimmten Zeit; dies ist z.B. wichtig, wenn Geschriebenes sofort wieder gelesen werden soll. Ein Aufruf von **fflush()** nach **fscanf()** ist nicht vorgesehen, sorgt aber (Compiler-abhängig, z.B. bei DevStudio) für korrekte Positionierung des Dateizeigers.

- Es ist zu beachten, daß beim Wechsel zwischen Datei-Lesen und -Schreiben die Plazierung des Dateizeigers nicht mehr zur Verfügung steht; sie kann wiedergewonnen werden durch Aufruf von **fseek()** oder **rewind()**. (Man beachte dabei die Warnung der DevStudio-Hilfe: „For streams opened in text mode, **fseek** has limited use, because carriage return–linefeed translations can cause **fseek** to produce unexpected results.“.)

- Nähreres zu allen obigen Anweisungen findet sich in der einschlägigen Literatur und in der On-Line-Hilfe.