2. 4. 8. Befehle für die Ein- und Ausgabe

Standard-Schnittstellen in C:

`stdin` Tastatur
`stdout` Bildschirm
`stderr` auch Bildschirm (für Fehlermeldungen)

`gets(s)` liest den String *s* von **`stdin`**

`puts(s)` schreibt den String *s* nach **`stdout`**

`getchar()` liest ein einzelnes Zeichen (Rückgabewert) von **`stdin`**
(verlangt *return*-Taste zum Abschluss;
mit Bildschirm-Echo)

`getch()` (nur in Borland-C/C++): wie **`getchar()`**, aber ohne *return*-Taste
und ohne Bildschirm-Echo

`putchar(c)` gibt das Zeichen *c* auf **`stdout`** aus

`int scanf(const char *` format { **`,`** Adresse }* **`)`**

↑ ↑ ↑

Rückgabewert: Anzahl eingelesener Werte — Formatierungs-string — Adressen der Eingabe-variablen

liest aus **`stdin`**

Varianten:

`int sscanf(char *` puffer**`, const char *`** format { **`,`** Adresse }* **`)`**

liest aus String *puffer*,

`int fscanf(FILE *`f**`, .... )`**

liest aus Datei *f*

Formatierungs-String : { *Zeichen* |
Formatierungs-Anweisung }*

Formatierungs-Anweisung :
% [*****] [*Breite*] [**F** | **N**] [**h** | **l** | **L**] *Typzeichen*

- soviele Formatierungs-Anweisungen wie Adressen von Eingabevariablen
- bei Formatierungs-Anweisung mit ***** : Unterdrückung der Zuweisung an die entsprechende Eingabevariable

Breite : optionale Angabe über die maximal zu lesende Zeichenzahl

[**F** | **N**] [**h** | **l** | **L**] : optionale Ergänzungen zum Typzeichen

Bsp.: **ld** für **long int**
Lf für **long double**

Typzeichen:

d	**int**
u	**unsigned int**
e oder **f**	Fließkommazahl (**float**)
o	Oktalzahl (**int**)
x	Hex-Zahl (**unsigned int**)
c	Zeichen (**char**)
s	String
p	Adresse
%	das Zeichen '%'

Ohne Breitenangabe wird solange eingelesen wie Zeichen zur Darstellung gehören, bei Zeichenketten bis zum Leerzeichen oder *newline*.

%**[** *Suchzeichen* **]** :

wirkt wie %**s**, aber es werden nur die angegebenen Suchzeichen eingelesen

z.B.	%**[abcd]**	nur Eingaben aus {a, b, c, d}
	%**[^abcd]**	nur Eingaben, die *nicht* {a, b, c, d} enthalten
	%**[0-9]**	akzeptiert nur die Ziffern
	%**[A-Za-zäöüÄÖÜß]**	akzeptiert nur Buchstaben

Beispiele:

scanf("%20s", name);

liest String mit bis zu 20 Zeichen von Tastatur

sscanf(s, "%d %[A-Za-z]", &zahl, name);

liest ganze Zahl und String nur aus Buchstaben aus String **s**

scanf("%3d, %f", &alter, &gehalt);

liest **int**-Zahl "alter" (max. 3 Ziffern) und **float**-Zahl "gehalt" von Tastatur, erwartet Trennung durch Komma.

Beachte: String-Bezeichner ohne Adress-Operator (**&**), Zahlen und einzelne Zeichen mit Adress-Operator!

`int printf(const char *` format { **`,`** Ausdruck }* **`)`**

↑ ↑ ↑

Rückg.: Anz. ausgeg. Bytes bzw. Fehlerwert — Formatierungs-string — Ausgabewerte

gibt nach **`stdout`** aus

Varianten:

`int sprintf(char *`s, ...**`)`** ("...." wie oben)
druckt in String **`s`**

`int fprintf(FILE *`f, ...**`)`** druckt in Datei f

Formatierungsstring enthält i.allg. Text, der unverändert ausgegeben wird, und Formatierungs-Anweisungen (vgl. **`scanf`**)

Formatierungs-Anweisung :
% [*Breite*] [**`.`***Genauigkeit*] *Typzeichen*

Breite : Minimalzahl auszugebender Zeichen
Genauigkeit : (abh. vom Typ) Maximalzahl

- der Nachkommastellen bei **`e`**, **`f`**
- der Zeichen bei **`s`**

Beispiel:

```
printf("Ergebnis: %d, %10ld, %10.2f\n",  m, n, x);
```

gibt **`int`** *m*, **`long`** *n* und **`float`** *x* auf dem Bildschirm aus
n und *x* fest formatiert

Beachte:
keine Verwendung des Adress-Operators bei **`printf`**