

Dateneingabe und Datenausgabe

1. Dateneingabe von der Tastatur

Mit der Funktion **scanf()** können Daten von der Tastatur eingelesen werden. Die Eingabe wird zeilenweise gepuffert und beim Drücken der *Enter*-Taste übernommen. Das f in scanf steht für formatiert. Dies bedeutet, dass die Eingabe in einer formatierten Form übernommen werden muss. Allgemein gilt folgende Syntax für das Einlesen:

scanf (Formatstring, Variable);

← Wobei mehrere Variablen und zugehörige Formatstrings eingelesen werden können

Konkret für das Einlesen eines ganzzahligen Wertes:

```
scanf ("%d", &Zahl);
```

oder gleichzeitiges Einlesen mehrere Zahlen:

```
scanf ("%d %d %d", &Tag, &Monat, &Jahr);
```

Besonders zu beachten ist hierbei der &-Operator. scanf() erwartet als Übergabewert einen Zeiger, um nun die Adresse der Variablen zu erhalten benötigt es den & Operator (zwingend).

Selbstverständlich können auch andere Werte ausser Ganzzahlwerten eingelesen werden. Folgende Formatelemente stehen zur Verfügung und gelten sowohl für die Eingabe, wie auch für die Ausgabe:

:

Formatelement	Typ	Ausgabe
%d	int	Dezimalzahl
%u	unsigned int	Natürliche (Dezimal-)Zahl (pos. Zahl)
%x, %X	unsigned int	Hexadezimalzahl
%o	unsigned int	Oktalzahl
%c	char	Einzelnes Zeichen (ASCII-Zeichen)
%s	char *	Zeichenkette („String“)
%f	float, double	Gleitkommazahl Einlesen: float => %f double => %lf
%e, %E	float, double	Gleitkommazahl in Exponential-Darstellung
%g, %G	float, double	Normale od. Exponential-Darstellung (was günstiger ist)

Wie bereits erwähnt, erfolgt die Eingabe gepuffert. Es wird aus dem Eingabepuffer gelesen und Eingaben sind erst dann zu Ende, wenn *Enter* (entspricht **\n**) gedrückt wird. Wenn Sie nun mehrere Werte hintereinander einlesen, kann es passieren, dass von der vorigen Eingabe noch Daten im Eingabepuffer sind, und die nächste Eingabe einfach übersprungen wird. Meistens bleibt ein **\n** im Puffer zurück. Wir müssen uns also von den Daten im Puffer trennen.

Ein häufig gegangener Weg ist, den Eingabestrom (**standard input stream**) **stdin** mit **fflush(stdin)** zu löschen. Das Problem: Das Verhalten von **fflush(stdin)** ist nicht standardisiert! Das heisst: Kann funktionieren, kann nicht funktionieren. In der Regel funktioniert es unter Windows, unter Linux nicht.

Hinweis:

Das ist aber natürlich kein Ansatz, um plattformunabhängigen Code zu schreiben!

Folgender Code funktioniert für Windows, wie auch für Linux:

```
int c; // int c, nicht char c.  
while ((c = getchar()) != EOF && c != '\n'); // kein {} block,
```

2. Datenausgabe

Daten können mittels der Funktion `printf` ausgegeben werden. Der Wert der Variablen kann hier mittels Formatstring ausgegeben werden.

Beispiel:

```
Tag = 12;  
Monat = 11;  
Jahr = 2013;
```

```
printf ("Die Standortbestimmung findet am %d.%d.%d statt!", Tag,  
Monat, Jahr);
```

Ausgabe: „Die Standortbestimmung findet am 12.11.2013 statt!“

Zu beachten:

- Die Textausgabe muss immer zwischen “ “ stehen.
- Bei der `printf` Funktion darf der `&`-Operator nicht verwendet werden!
- Es dürfen mehrere Formatelemente ausgegeben werden. Die Reihenfolge der Formatelemente muss dabei mit der Reihenfolge der Variablen übereinstimmen.

2.1 Kennzeichner (Flags)

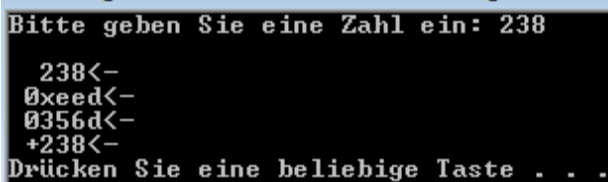
Um die Ausgabe in eine definierte Form zu bringen stehen weitere Kennzeichnungen (Flags) zur Verfügung. Diese können mit den Formatelementen kombiniert werden.

Flag	Wirkung
-	Linksbündig
0	Felder mit 0 ausfüllen (an Stelle von Leerzeichen).
+	Vorzeichen einer Zahl immer ausgeben.
blank	positive Zahlen mit Leerzeichen beginnen.
#o	Oktal Zeichen (0 Präfix wird eingeführt).
#x, #X	Hexadezimalzeichen (0x oder 0X Präfix wird eingeführt).

- Die Flags müssen nach dem % stehen.
- Es können mehrere Flags kombiniert werden.

Beispiel:

```
printf(" % d<- \n", Zahl); //<-zeigt wo der Text weiter geht
printf(" %#xd<- \n", Zahl);
printf(" %#od<- \n", Zahl);
printf(" %+d<- \n", Zahl);
```

Ausgabe;

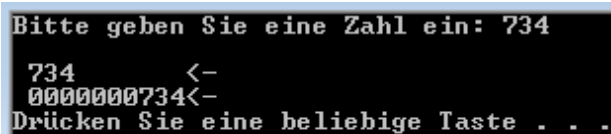
```
Bitte geben Sie eine Zahl ein: 238
238<-
0xfeed<-
0356d<-
+238<-
Drücken Sie eine beliebige Taste . . .
```

2.2 Minimale Anzahl Felder

Wenn eine Ausgabe erfolgt wird Standardmässig die minimale Anzahl Felder ausgegeben. Es besteht die Möglichkeit die Anzahl der Felder zu wählen indem eine Zahl vor dem Formatelement eingegeben wird.

Beispiel:

```
printf(" %-10d<- \n", Zahl); //<-zeigt wo der Text weiter geht
printf(" %010d<- \n", Zahl);
```



```
Bitte geben Sie eine Zahl ein: 734
734      <-
00000000734<-
Drücken Sie eine beliebige Taste . . .
```

2.3 Genauigkeit

Mit der Angabe von einem Punkt gefolgt von einer Zahl kann die Anzahl der Nachkommastellen definiert werden.

Beispiel:

```
printf(" %08.3f<- \n", Zahl2); //<-zeigt wo der Text weiter geht
printf(" %02.3f<- \n", Zahl2);
printf(" %012.3f<- \n", Zahl2);
```

```
Bitte geben Sie eine Zahl ein: 456.80234675
```

```
0456.802<-
```

```
456.802<-
```

```
00000456.802<-
```

```
Drücken Sie eine beliebige Taste . . .
```

2.4 Escape-Sequenzen

Bei unseren Programmen haben wir bereits einige Male das `\n` verwendet. Weitere Steuerzeichen sind in der untenstehenden Tabelle ersichtlich:

Escape-Sequenz	Beschreibung
<code>\a</code>	Bell (Standardton); auch Escape-Sequenz <code>\007</code> möglich
<code>\b</code>	Backspace (letztes Zeichen löschen)
<code>\f</code>	Seitenvorschub (formfeed)
<code>\n</code>	New Line (neue Zeile)
<code>\r</code>	Wagenrücklauf (an den Anfang der Zeile)
<code>\t</code>	Tabulator (Sprung um einen Standardwert)
<code>\v</code>	Vertikaler Tabulator (Sprung um einen Standardwert)
<code>\'</code>	Hochkomma ausgeben
<code>\"</code>	Anführungszeichen ausgeben
<code>\\</code>	Backslash (verkehrter Schrägstrich) ausgeben
<code>\ooo</code>	Oktalzahl mit max. 3 Stellen ausgeben, z.B. oktal <code>\077</code> entspricht dezimal 63. Ersetzen Sie <code>o</code> durch eine Oktalzahl. Ausgegeben wird das entsprechende ASCII-Zeichen.
<code>\xhh</code>	Hexadezimalzahl mit max. 2 Stellen ausgeben. Ersetzen Sie <code>h</code> durch eine Hexadezimalzahl. Ausgegeben wird das entsprechende ASCII-Zeichen.

Beispiel:

```
printf ("Sie hören einen Ton ...\a\n");
printf ("Dem folgenden \"xyz\" wird das letzte Zeichen entfernt
und mit einem \"!\" überschrieben: xyz\b!\n");
printf ("XXXZurück an den Anfang der Seite und 'XXX' überschrei-
ben.\r...\n");
printf ("\t\t2 Tabulatorsprünge; der Backslash: \\ \n");
```

```
Sie hören einen Ton ...
Dem folgenden "xyz" wird das letzte Zeichen entfernt und mit einem "?" überschrieben: xy!
...Zurück an den Anfang der Seite und 'XXX' überschreiben.
                2 Tabulatorsprünge; der Backslash: \
Drücken Sie eine beliebige Taste . . . _
```

2.5 Ausgabe von Umlauten

Um Umlaute auszugeben, kann der ANSI Wert des entsprechenden Umlauts als char Formatelement ausgegeben werden.

Beispiel:

```
printf ("Umlaute ohne spezielle Behandlung, z.B.: ä ö ü\n ");
printf ("Umlaute als char Ausgabe, z.B.: %c %c %c \n",132, 148,
129);
printf("\n");
printf("\n");

Zaehler = 128;
while (Zaehler <= 190){

    printf ("%i:%c    %i:%c    %i:%c\n",Zaehler,Zaehler, Zaehler+1,
    Zaehler+1, Zaehler+2,Zaehler+2);

    Zaehler = Zaehler + 3;
}
```

```
Umlaute ohne spezielle Behandlung, z.B.: ö ÷ ³
Umlaute als char Ausgabe, z.B.: ä ö ü
```

```
128:ç    129:ü    130:é
131:ä    132:ä    133:à
134:â    135:ç    136:ê
137:è    138:è    139:ï
140:î    141:ì    142:ñ
143:ð    144:é    145:æ
146:œ    147:ô    148:ö
149:ð    150:û    151:ù
152:ÿ    153:ö    154:Ü
155:ø    156:£    157:Ø
158:×    159:f    160:á
161:í    162:ó    163:ú
164:ñ    165:Ñ    166:ª
167:º    168:¿    169:®
170:¬    171:½    172:¼
173:¡    174:«    175:»
176:¶    177:§    178:§
179:¸    180:¸    181:À
182:Á    183:Â    184:Ã
185:Ä    186:Å    187:Ç
188:È    189:É    190:Ê
Drücken Sie eine beliebige Taste . . . _
```