1.4 Programmstrukturen/Eingabe

1.4.1 Situation

Die bisherige Version von `xc.c`, die aus Frequenz und Kapazität den Blindwiderstand eines Kondensators berechnet, kann nur mit Werten arbeiten, die *vor* dem Compilieren eingegeben wurden. Schön wäre es aber, wenn man die Werte während des Programmlaufs (zur sogenannten *Laufzeit*) auf der Tastatur eingeben könnte.

1.4.2 Funktion `scanf`

Das Programm `xc2.c` setzt den genannten Wunsch um:

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      double c;
6      double f;
7      double xc;
8                                  /* Eingabe */
9      printf("Bitte C eingeben:");
10     scanf("%lf", &c);
11     printf("Bitte f eingeben:");
12     scanf("%lf", &f);
13                                 /* Verarbeitung */
14     xc = 1.0/(2*3.14*f*c);
15                                 /* Ausgabe */
16     printf("C=%lf Farad, ", c);
17     printf("f=%lf Hertz, ", f);
18     printf("Xc=%lf Ohm\n", xc);
19     return 0;
20 }
```

Wenn man das Programm startet, kann es so aussehen (Benutzereingaben in Fettdruck!):

Terminal

```
schueler@debian964:~$ ./a.out
Bitte C eingeben:0.001
Bitte f eingeben:1500
C=0.001000 Farad, f=1500.000000 Hertz, Xc=0.106157 Ohm
```

In Zeile 9 des Quelltextes findet man einen Aufruf der Funktion `scanf`, die für Tastatureingaben zuständig ist. Der erste Parameter enthält den Formatstring `"%lf"`. Das ist fast genauso wie bei der `printf`-Funktion, `%lf` ist ein Platzhalter für eine Gleitkommazahl. Nur ist es hier so, dass außer dem Platzhalter in dem Formatstring möglichst nichts vorkommen sollte (auch kein `\n`-Zeichen!).

Der zweite Parameter enthält den Namen der Variablen, in die das Ergebnis der Tastatureingabe geschrieben werden soll. Hier ist ein bedeutender Unterschied zu `printf`: Vor die Variable muss der sogenannte Adress-Operator `&` (Kaufmanns-Und oder *ampersand* genannt) gestellt werden. Vergisst man ihn, so stürzt das Programm in den meisten Fällen direkt nach der Tastatureingabe ab.

Die Eingabe muss mit der Return-Taste (⏎) abgeschlossen werden; dann wird die eingegebene Tastenfolge ausgewertet und in die vorgegebene Variable geschrieben – vorher passiert nichts. Erst dann läuft das Programm weiter.

Die Tabelle 1 zeigt die Unterschiede bei den Platzhaltern für `printf` und `scanf`.

Datentyp	Platzhalter printf	Platzhalter scanf
int	%i,%d (dezimal)	%i (dez.,okt.,hex.), %d (nur dez.)
float	%f	%f
double	%f oder %lf	%lf
long double	%Lf	%Lf

Tabelle 1: Unterschiede `printf` und `scanf`

1.4.3 Lokalisierung des Programmes

Bisher muss der Benutzer bei seinen Eingaben beachten, dass er statt des Dezimalkommas einen Punkt setzt. Man kann die Funktionen `scanf` und `printf` aber so anpassen, dass sie statt des Punktes den Dezimaltrenner verwenden, der ortsüblich ist. Dazu gibt es die Funktion `setlocale`, die im Programm `src/ausgabe_locale.c` benutzt wird:

```
1  #include <stdio.h>  /* fuer printf()     */
2  #include <stdlib.h> /* fuer getenv()     */
3  #include <locale.h> /* fuer setlocale() */
4  int main(void)
5  {
6      printf("Ausgabe ohne setlocale():\n");
7      printf("%f\n", 1234567.89);
8      printf("Rufe setlocale(%s) auf...\n", getenv("LANG"));
9      setlocale(LC_ALL, getenv("LANG"));
10     printf("Ausgabe mit setlocale():\n");
11     printf("%f\n", 1234567.89);
12     printf("Ausgabe mit setlocale() - alternative Form:\n");
13     printf("%'f\n", 1234567.89);
14     return 0;
15 }
```

Außer der Einbindung der Headerdatei `locale.h` ist nur der Aufruf von `setlocale` notwendig.

Der erste Parameter gibt an, welche Programmeigenschaften örtlich angepasst werden sollen: Zeit- und Datumsformat, Zahlenformat, Sortierreihenfolge usw. Mit `LC_ALL` gibt man an, dass man alle Eigenschaften anpassen möchte.

Der zweite Parameter gibt an, an welche Sprache und welches Land man anpassen möchte:

- `"C"` – Grundeinstellung der Sprache C, also Sprache englisch, Land USA, Zeichensatz ASCII (nicht UTF-8)
- `"de_AT.utf8"` – Das bedeutet: Sprache deutsch, Land Österreich, Zeichensatz UTF-8
- `getenv("LANG")` – Auf diese Weise wird die Einstellung übernommen, die in der Umgebungsvariablen `LANG` (=*language*) steht
- `""` – Wenn man eine leere Zeichenfolge angibt, wird die momentane Systemeinstellung verwendet

In unserem Fall wird die momentane Systemeinstellung genommen.

Bei einem Linux-System kann man übrigens mit folgendem Befehl auf der Konsole herausfinden, welche Lokalisationen (*locales*) verfügbar sind:

Terminal

```
schueler@debian964:~$ locale -a
C
de_DE.utf8
POSIX
```

Und man kann weitere Lokalisationen verfügbar machen (mit der Installation des Paketes `locales-all`).

Mit der lokalisierten Version von `xc2.c` gestaltet sich der Dialog so:

Terminal

```
schueler@debian964:~$ ./a.out
Bitte C eingeben:0,001
Bitte f eingeben:1500
C=0,001000 Farad, f=1500,000000 Hertz, Xc=0,106157 Ohm
```

Im Quelltext dagegen hat `setlocale` keine Auswirkung; bei der Programmierung muss man Gleitkommazahlen in C immer mit Punkt schreiben.