2.6.A Funktionen/Rückgabewert – Arbeitsblatt

Aufgabe 1: Funktionen mit Rückgabewert und ohne Parameter

Bei vielen Programmen geschieht die Eingabe von Zahlen auf die immer gleiche Art und Weise, zum Beispiel wie hier:

```
1  #include <stdio.h>
2  int main(void)
3  {
4      int x;
5      int rc;
6      x=0;
7      do{
8          printf("Wert eingeben (ganzzahlig): ");
9          rc=scanf("%i", &x);
10         if(rc!=1)
11         {
12             printf("Fehler, Wiederholung:\n");
13         }
14         while(getchar()!='\n'){}
15     }while(rc!=1);
16
17     printf("Ihre Eingabe war: %i\n", x);
18     return 0;
19 }
```

Zum Verständnis: Der Rückgabewert `rc` von `scanf()` ist die Anzahl der erfolgreich eingelesen Platzhalter (oder -1 bei Fehler). Hier wird die Eingabe solange wiederholt, bis eine Zahl eingelesen werden konnte.

Die Zeilen 4 bis 15 sollen nun in eine Funktion `liesint()` verlagert werden, so dass man in Zukunft einfach das Folgende schreiben kann und trotzdem die gleiche Funktionalität wie oben bekommt:

```
1  #include <stdio.h>
2  int main(void)
3  {
4      int x;
5      x=liesint(); // <-- Zeilen 4-15 aus dem vorigen Programmstueck
6      printf("Ihre Eingabe war: %i\n", x);
7      return 0;
8  }
```

Der Aufruf in Zeile 5 ersetzt also komplett die Zeilen 5 bis 15 aus dem oberen Programm.

a) Schreiben Sie die Funktionsdefinition zu `liesint` (vorwiegend kopieren aus dem oberen Programm!) und betten Sie die Funktionsdefinition in ein Testprogramm `liesint.c` ein, das einen Zahlenwert mit `liesint` liest und mit `printf` sofort danach wieder ausgibt (siehe unteres Programm)!

Terminal

```
schueler@debian964:~$ a.out
Wert eingeben (ganzzahlig): ABC
Fehler, Wiederholung:
Wert eingeben (ganzzahlig): 20
Ihre Eingabe war: 20
```

b) Genauso wird eine Funktion `liesdbl` benötigt, die eine Variable vom Typ `double` einliest. Schreiben Sie diese Funktion und das zugehörige Testprogramm `liesdbl.c`!

Terminal
```
schueler@debian964:~$ a.out
Wert eingeben (Kommazahl erlaubt): 20,7
Ihre Eingabe war: 20,7
```

Aufgabe 2: Mathematische Funktionen mit Rückgabewert und mit einem Parameter

In der Headerdatei `<math.h>` sind viele Funktionen beschrieben, die einen Parameter vom Typ `double` aufnehmen und den Rückgabetyp `double` zurückgeben. Beispiele sind `sin()` und `cos()`:

```
1 double sin(double x);
2 double cos(double x);
```

Ein Beispielprogramm damit sähe so aus:

```
1 #include <math.h> // compilieren mit gcc -lm
2 int main(void)
3 {
4     double x=0.0, y;
5     y=cos(x);
6     printf("%lf\n", y);
7     return 0;
8 }
```

Sie sollen jetzt weitere Funktionen dieser Art schreiben und testen.

a) Die Funktion `xquadrat()` soll den Wert x^2 zurückgeben (`xquadrat.c`).

Terminal
```
schueler@debian964:~$ a.out
Eingabe: 2,5
Ausgabe: 6,25
```

b) Die Funktion `xhoch3()` soll den Wert x^3 zurückgeben (`xhoch3.c`).

c) Die Funktion `einweggleichrichter()` soll den Wert von x zurückgeben, fall x positiv ist, und null in allen anderen Fällen (`einweg.c`).

Terminal
```
schueler@debian964:~$ a.out
Eingabe: -3
Ausgabe: 0
```

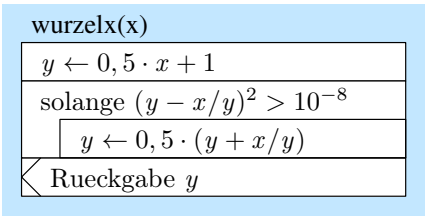
d) Die Funktion `zweiweggleichrichter()` soll den Wert von x zurückgeben, fall x positiv ist, sonst -x (`zweiweg.c`).

Terminal
```
schueler@debian964:~$ a.out
Eingabe: -3
Ausgabe: 3
```

e) Die Funktion `vorzeichen()` soll den Wert 1 zurückgeben, falls x positiv ist, den Wert -1, falls x negativ ist, und den Wert 0, falls x gleich null ist (`vorzeichen.c`).

Terminal
```
schueler@debian964:~$ a.out
Eingabe: -3
Ausgabe: -1
```

f) Die Funktion `wurzelx()` soll den Wert $\sqrt{x}$ zurückgeben (`wurzelx.c`). Das funktioniert so (Heron-Verfahren):



Hinweis: Durch den gewählten Startwert und die Abbruchbedingung funktioniert diese einfache Version nicht für alle `double`-Werte!

Terminal

```
schueler@debian964:~$ a.out
Eingabe: 2
Ausgabe: 1,41422
```

Aufgabe 3: Funktionen mit Rückgabewert und zwei Parametern

In der C-Standard-Bibliothek fehlt eine Funktion `min(a,b)`, die kleinere der beiden Zahlen a und b zurückgibt. Genauso gibt es dort keine Funktion `max(a,b)`, die die größere der beiden Zahlen zurückgibt. Deshalb kann dieses Program (`minmax.c`) nicht compiliert werden:

```c
#include <stdio.h>
int main(void)
{
    double a=4.5, b=3.2, x, y;
    x=min(a,b);   // <-- gibt es noch nicht
    y=max(a,b);   // <-- gibt es auch nicht
    printf("Kleinere Zahl:  %lg\n", x); // <-- soll 3.2 ausgeben
    printf("Groessere Zahl: %lg\n", y); // <-- soll 4.5 ausgeben
    return 0;
}
```

Sie sollen diese Funktionen nun selbst schreiben, und zwar für `double`-Zahlen.

a) Wie lauten die Prototypen für `min(a,b)` und `max(a,b)`? Beide Parameter und der Rückgabewert sollen vom Typ `double` sein.

b) Bitte schreiben Sie die Funktionsdefinitionen für `min(a,b)` und `max(a,b)`!

c) Betten Sie die beiden Funktionen in das oben angegebene Programm ein, so dass es doch noch funktioniert!

Aufgabe 4: Berechnung von X_C und X_L

In dieser Aufgabe (`zreihe1.c`) sollen Sie die Impedanz Z einer Reihenschaltung aus R, L und C berechnen (hier zunächst einmal ohne komplexe Rechnung). Dazu ist es nötig, zuerst verschiedene Funktionsbausteine zu erstellen.

a) Erstellen Sie eine Funktion `berechne_xc()`, die C und f als Parameter erhält und den Wert für X_C berechnet und zurückgibt!

b) Erstellen Sie eine Funktion `berechne_xl()`, die L und f als Parameter erhält und den Wert für X_L berechnet und zurückgibt!

c) Berechnen Sie mit Hilfe der oben erstellten Funktionen die Gesamtimpedanz Z eines Reihenschwingkreises mit $C = 100\,\text{pF}$, $L = 0{,}1\,\text{mH}$ und $R_S = 10\,\Omega$ bei $f_1 = 1\,\text{MHz}$! Hinweis: $Z = \sqrt{R^2 + (X_L - X_C)^2}$

d) Erweitern Sie das Programm aus der vorigen Aufgabe so, dass für alle Frequenzen von $f_A = 0{,}1\,\text{MHz}$ bis $f_E = 3{,}0\,\text{MHz}$ der Wert für Z berechnet und ausgegeben wird! Die Schrittweite soll $f_S = 0{,}05\,\text{MHz}$ betragen (`zreihe2.c`).

Hinweis: Die Wurzel einer Zahl `x` berechnet man in C mit der Bibliotheksfunktion `sqrt(x)`. Eingebunden wird sie über die Headerdatei `<math.h>`. Der Befehl zum Erstellen des Programms lautet: `gcc zreihe1.c -lm`

Aufgabe 5: Berechnungen zum Neubau einer Garage

Für den Neubau einer Garage (6m x 2,5m x 2,5m, Wanddicke 0,1 m) sollen überschlägig (keine Fenster, keine Türen) die folgenden Größen ermittelt werden (`garage.c`):

a) umbauter Raum (=Volumen),

b) Volumen von Boden, Wänden und Decke (zusammengerechnet),

c) Baumasse (`berechne_masse(v,rho)` nach der Formel $m = \rho V$ mit der Massendichte $\rho \approx 2400\,\text{kg/m}^3$),

d) benötigte Menge Wandfarbe (Schichtdicke $d = 0{,}5\,\text{mm}$) in Litern,

e) Dachkantenlänge für die Blitzschutzeinrichtung.

Dabei sollen zuerst folgende Funktionen erstellt werden:

- `berechne_quadervolumen()` nutzt die Formel $V = A_g h$ (A_g ist die Grundfläche, h die Höhe, V das Volumen)
- `berechne_quaderoberflaeche()` nutzt die Formel $A_o = 2A_1 + 2A_2 + 2A_3$ (A_1, A_2 und A_3 sind die Flächen links, vorne und oben)

Dazu können vorher auch diese Funktionen erstellt werden:

- `berechne_rechteckflaeche(s1,s2)` mit der Formel: $A = s_1 s_2$ (die Fläche ist A, die Seiten s_1 und s_2)
- `berechne_rechteckkantenlaenge(s1,s2)` mit der Formel: $s_k = 2s_1 + 2s_2$ (die Kantenlänge ist s_k, die Seitenlängen s_1 und s_2)

Benutzen Sie die erstellten Funktionen, um sich die Arbeit zu erleichtern!

Aufgabe 6: Berechnung des Anhaltewegs eines Fahrzeugs

Der Anhalteweg eines Fahrzeugs mit der Geschwindigkeit v setzt sich aus dem Reaktionsweg l_R und dem Bremsweg l_B zusammen. Sobald die Notwendigkeit zu bremsen erkannt ist, dauert es noch eine Zeit, bis der Bremsvorgang beginnt. Diese Zeit heißt Reaktionszeit. Während der Reaktionszeit fährt das Fahrzeug ungebremst weiter; man erhält für den Reaktionsweg $l_R = v \cdot t_R$. Für den Bremsweg gilt die Formel $l_B = v^2/(2a)$. Bei Pkws geht man von $t_R \approx 1\,\text{s}$ und $a \approx 4{,}5\,\text{m/s}^2$ aus.

Das folgende Programm soll die Berechnung ausführen:

```c
#include <stdio.h>
#define a_NORMAL    4.5
#define t_REAKTION 1.0
#include "anhalteweg_inc.c"

int main(void)
{
    double a_normal=a_NORMAL;       /* a in m/(s*s) Bremsverzoeg. */
    double t_reaktion=t_REAKTION; /* t_R in s      Reaktionszeit */
    double anhalteweg;    /* l_A in m  */
    double bremsweg;      /* l_B in m  */
    double reaktionsweg; /* l_R in m  */
    double vkmh;          /* v in km/h */
    double vms;           /* v in m/s  */
    printf("Bitte Geschwindigkeit eingeben (in km/h): ");
    scanf("%lf", &vkmh); while(getchar()!='\n'){}
    vms          = umrechnen_kmh_nach_ms(vkmh);
    reaktionsweg = berechne_reaktionsweg(vms, t_reaktion);
    bremsweg     = berechne_bremsweg(vms, a_normal);
    anhalteweg   = reaktionsweg + bremsweg;
    printf("Anhalteweg..: %lf m\n", anhalteweg);
    printf("Reaktionsweg: %lf m\n", reaktionsweg);
    printf("Bremsweg....: %lf m\n", bremsweg);
    return 0;
}
```

Dabei fehlen noch die Funktion, die die Umrechnung der Geschwindigkeit von km/h nach m/s vornimmt sowie die Funktionen zur Berechnung von Reaktions- und Bremsweg.

a) Erstellen Sie die Prototypen!

b) Erstellen Sie die genannten Funktionen dazu!

c) Testen Sie die Funktionen im Programm! Erfahrungsgemäß sind bei Pkws bei $v = 30\,\mathrm{km/h}$ Reaktions- und Bremsweg etwa gleich groß; bei höheren Geschwindigkeiten überwiegt der Bremsweg.