Beispiel 1[1]

(3 Punkte)

Der elektrische Gesamtwiderstand von n parallel geschalteten Widerständen ist:

$$\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2} \cdots + \frac{1}{R_n}$$

Angenommen wir haben zwei Widerstände mit den Werten 400 Ω und 200 Ω. Dann schaut unsere Gleichung so aus:

$$\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2}$$

mit eingesetzten Werten:

$$\frac{1}{R} = \frac{1}{400} + \frac{1}{200} = \frac{3}{400} \quad \rightarrow \quad R = \frac{400}{3} = 133{,}3$$

Der Gesamtwiderstand der beiden Widerstände beträgt 133,3 Ω.

Schreibe ein Programm, das den Gesamtwiderstand von einer beliebigen Anzahl parallel geschalteter Widerstände berechnet.

Tipps zur Vorgangsweise:
Lies als erstes die Anzahl (n) der Widerstände ein. Anschließend frage in einer `for`-Schleife alle Widerstände (n-mal) ab und berechne gleich 1 durch den zuletzt eingelesenen Widerstand und addiere diesen Betrag zum Gesamtwiderstand.
Nach der Schleife muss dann nur mehr der Kehrwert der Summe (= 1 / Summe) gebildet werden.
Speichere das Programm mit dem Namen `resistance.c`.

[1] Beispiel aus Oualline, Steve: Practical C Programming, 2nd ed., Sebastopol, 1993

Beispiel 2

(3 Punkte)

Erstelle ein Programm zur Berechnung von Wurzeln.

Das Programm soll in einer Schleife laufen, bei Eingabe der Zahl 0 soll das Programm beendet werden (vgl. gestriges Beispiele 10).

Die Wurzel einer Zahl kann mit der Funktion `sqrt(Zahl)` berechnet werden. Dazu muss die Header-Datei `<math.h>` eingebunden werden.

Prüfe nach dem Einlesen der Zahl, ob die Zahl eine positive Zahl ist. Wenn nicht, gib eine Meldung aus, dass die Wurzel einer negativen Zahl nicht berechnet werden kann.

Speichere das Programm unter dem Namen `sqrt.c`.

Anmerkung: Für den in den Rechenräumen der Informatik verwendeten C-Compiler muss der Schalter `–lm` beim Kompilieren verwendet werden.

Weiters verlangt dieser Compiler die Inkludierung der Headerdatei `<stdlib.h>` um die Funktionen `rand()` und `srand()` ohne Warnmeldungen verwenden zu können.

Beispiel 3
(2 Punkte)

Schreibe ein Programm, das die Anzahl der Zeilen eines eingegebenen Textes zählt. Das Einlesen soll beendet werden, wenn EOF (end of file) erreicht ist. EOF bezeichnet das Ende einer Datei und kann auf der Tastatur unter Linux mit Strg + D und unter DOS mit Strg + Z eingegeben werden.

Verwende zum Einlesen nicht die formatiert Eingabe mit `scanf()` sondern die Funktion `getchar()` zum Einlesen einzelner Zeichen. `putchar()` ist das Gegenstück dazu, es dient zum Ausgeben einzelner Zeichen.

Das Zählen aller Zeichen könnte so aussehen:

```
double nc;
for (nc = 0; getchar() =! EOF; ++nc)
  ;  //eine leere Anweisung
printf("%.0f\n", nc);
```

Beim Zählen der Zeilen zählt man nur das Zeilenendezeichen newline:

```
int c, nl; //ein Zeichen kann auch als int eingelesen werden
while ( ( c = getchar() ) != EOF)
  if ( c ==’\n’)
    ++nl;
printf("%d\n", nl);
```

Beispiel 4
(3 Punkte)

Erweitere das vorige Beispiel dahingehend, dass alle eingegebenen Wörter gezählt werden. Dabei sollen aber nicht wirklich alle Wörter gezählt werden, sondern die Räume zwischen den Wörtern. Man prüft also, ähnlich wie im vorigen Beispiel, auf die Zeichen `' '` (Leerraum), `'\n'` (newline) oder `'\t'` (Tabulator). Die Texteingabe soll wieder mit dem EOF-Zeichen beendet werden.

Beispiel 5

(2 Punkte)

Zufallszahlen erzeugen

Erzeuge 10 Zufallszahlen zwischen 15 und 30 und gib sie aus. Achte darauf, dass es bei jedem Programmlauf andere Zahlen sind.

Beispiel 6

(5 Punkte)

Lottozahlengenerator

Erzeuge 6 Zufallszahlen zwischen 1 und 45 mittels Zufallsgenerator und gib sie anschließend mittels einer `for`-Schleife in einem Feld von 9 (breit) x 5 (hoch) aus.

Beispiel:

Die generierten Lottozahlen sind:

1	2	=3=	4	5	6	7	8	=9=
10	11	12	13	14	15	16	=17=	18
19	20	21	22	23	=24=	25	26	27
28	29	30	31	32	=33=	34	35	36
37	=38=	39	40	41	42	43	44	45

Kennzeichne die Gewinnzahlen mit = vor und nach der Zahl.

Speichere die Gewinnzahlen in jeweils einer Variablen. Überprüfe beim Hochzählen der Schleifenvariable ob die Variable eine Gewinnzahl ist (mit `if` oder `switch`) und gib die =-Zeichen mit aus. Beachte, dass bei Zahlen unter 10 vor der Zahl noch ein Leerzeichen ausgegeben werden muss.

Den Zeilenumbruch nach jeder neunten (in diesem Falle auch durch neun teilbaren) Zahl berechne mit dem Modulo-Operator.

Beispiel 7

(2 Punkte)

Folgendes Beispiel demonstriert die Verwendung eines Feldes. Allerdings haben sich im Quelltext ein paar Fehler eingeschlichen. Kompiliere das Programm und korrigiere die Fehler.

```
// programme:  average.c
// author:     Josef Strasser-Leitner
// date:       02.10.2005
// version:    1.00
// purpose:    averages 10 numbers

#include <stdio.h>;


// purpose: main function
// parameters: none
// return value: 0
int main()
{
  double values[10];
  double sum=0.0;

  printf("Dieses kleine Programm fragt 10 Werte ab und speichert ");
  printf("sie in einem Feld und berechnet den Mittelwert."\n);
  for (int loop = 0 ; loop < 10 ; ++loop );
  {
    printf("Geben Sie einen Wert ein: )";
    scanf("%",values[loop]);
  }

  for ( loop = 1 ; loop < 10 ; ++loop )
  {
    sum += values[loop];

  printf("Mittelwert: " sum / 10.0 \n);

  return 0;
} // end main
```

Beispiel 8

(7 Punkte)

Die Auswertung einer Klausur ergab folgendes Ergebnis (fiktiv):

Note	Häufigkeit
1	34
2	79
3	228
4	57
5	20

Erstelle ein Programm, das

- die Häufigkeit der Noten 1 bis 5 in *n*[*0*] bis *n*[*4*] einliest (n für note)
- die Anzahl der Schüler aus *n*[*0*] bis *n[4]* berechnet (Variable *students*)
- die Summe (*sum*) aus *1* * *n*[0] + *2* * *n*[*1*] ... bis 5 * *n*[*4*] für den Notendurchschnitt berechnet
- den Notendurchschnitt berechnet und ausgibt: *sum / students*
- den prozentualen Anteil für alle Noten berechnet und ausgibt

```
percentage = n[0] * 100 / students  //  für n[0] bis n[4]
```

Die einzelnen Aufgaben sollen jeweils in einer Funktion gelöst werden.

Da ein Array eigentlich ein Zeiger auf das erste Feld ist, also eine Speicheradresse, wird ein Feld an eine Funktion "by reference", also das Original, und nicht wie bei anderen Variablen üblich als Kopie ("by value") übergeben.

Übergabe eines Feldes an eine Funktion:

```
void input( double values[10] )
  {
  ...
  }


int main()
  {
  double values[10];
  ...
  input (values);  // Übergabe des Feldes an die Funktion input
  ...
  return 0;
 }
```

Beispiel 9

(4 Punkte)

Texte

Sie sollen ein Programm schreiben, in das Personendaten eingegeben werden können. Dabei sind folgende Daten einzugeben:
→ Vorname
→ Nachname
→ Postleitzahl
→ Wohnort
→ Straße
→ Hausnummer

Beispiel für einen Programmlauf (die Eingaben während des Programmlaufes sind *kursiv* geschrieben):

```
Bitte geben Sie Ihren Vornamen ein     :    Willi
Wie heißen Sie mit Nachnamen           :    Maier
Die Postleitzahl Ihres Wohnortes       :    6020
Und der Name Ihres Wohnortes           :    Innsbruck
In welcher Straße wohnen Sie           :    Höttingergasse
Geben Sie Ihre Hausnummer ein          :    12


Ihre Adresse lautet: Willi Maier
                     Höttingergasse 12
                     6020 Innsbruck

Vielen Dank für Ihren Besuch! Auf Wiedersehen und einen schönen Tag noch!
```

Verwenden Sie `scanf()` und `printf()` zum Einlesen und Ausgeben der Daten. Ein Tabsprung kann – zur Erinnerung – mit `\t` ausgegeben werden.