Dr. habil. Bernd Schürmann
Dr. Annette Bieniusa
pinc-support@cs.uni-kl.de

TU Kaiserslautern
Fachbereich Informatik

Lösungshinweise zum Übungsblatt 4: Programmieren in C (WS 2018/19)

1. Zur Beantwortung von Fragen und Hilfe bei Problemen stehen wir Ihnen immer in der Praktische Übung, Mittwoch 15:30, Raum 32-410-PC zur Verfügung sowie per Email an `pinc-support@cs.uni-kl.de`!
2. Sie können sich im Exclaim System unter `https://softech.cs.uni-kl.de/exclaim` mit Ihrem persönlichen STATS-Account einloggen und Dateien zu den einzelnen Übungen hochladen.
3. Die Aufgaben auf diesem Übungsblatt sind alle freiwillig zu bearbeiten.

Aufgabe 1: Rechnen mit Gleitkommazahlen (keine Abgabe)

- Runden Sie folgende Gleitkommazahlen auf zwei Nachkommastellen nach den vier in der Vorlesung vorgestellten Rundungsarten plus kaufmännisches Runden!
  (a) +01,1110
  (b) +11,1011
  (c) –10,1110

	up	down	truncate	nearest even	nearest
+01,1110	10,00	01,11	01,11	10,00	10,00
+11,1011	11,11	11,10	11,10	11,11	11,11
-10,1110	-10,11	-11,00	-10,11	-11,00	-11,00

- Addieren Sie folgende Gleitkommazahlen binär!
  (a) $10,1_2 * 2^{-2} + -111_2 * 2^{-4}$
  - Umwandeln -00000111_2 in 11111001_2 als 2K-Zahl
  - Denormalisieren durch Anpassen der Exponenten

    Exponent 2^{-2}:
    ```
    00001010,0
    11111001,0
    ----------
    00000011,0
    ```
    Exponent 2^{-4}:
    ```
    00000010,10
    11111110,01
    -----------
    00000000,11
    ```

    Check: $11,0_2 * 2^{-4} = 1,1_2 * 2^{-3} = 0,11_2 * 2^{-2}$

  (b) $7,53125 + 3,375$
  - Umwandeln: $7,53125 + 3,375 = 111,10001_2 + 11,011_2$
    ```
      111,10001
      011,01100
    -----------
    1 010,11101
    ```

– Umwandeln: $1010,11101_2 = 10,90625$

Aufgabe 2: Rechnen mit Zahlen (Abgabe in Exclaim als `unsigned.c`)

Gegeben sei das folgende Programm:

```
#include <stdio.h>
int main (void)  {
   signed char a = 119;
   signed char b = 19;
   signed char c;

   c = a + b;

   printf("%i\n", c);
   return 0;
}
```

1. Überlegen Sie zuerst, was das Programm ausgibt!
2. Implementieren Sie das Programm! Erklären Sie das Ergebnis.

   Der Zahlenbereich für Werte vom Typ `signed char` umfasst die ganzen Zahlen im Interval $[-128, 127]$. Dabei werden die negativen Zahlen als Komplement gegen 2^7 gebildet. Bei der Addition von $119_{10} = 1110111_2$ und $19_{10} = 10011_2$ ist das Ergebnis 10001010_2. Als Komplement gegen 2^7 entspricht dies dem Wert $-1 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = -118$.

3. Ändern Sie den Datentyp der Variablen `a`, `b` und `c` in `unsigned char`. Implementieren Sie das Programm erneut als `unsigned.c`! Erklären Sie das neue Ergebnis.

   Der Zahlenbereich für Werte vom Typ `unsigned char` umfasst die ganzen Zahlen im Interval $[0, 255]$. Bei der Addition von $119_{10} = 1110111_2$ und $19_{10} = 10011_2$ ist das Ergebnis $10001010_2 = 138_{10}$.

Aufgabe 3: Umwandlung von Zeichen (Abgabe als `klein.c`)

Schreiben Sie ein Programm `klein.c`, das ein einzelnes ASCII-Zeichen einliest und folgendes ausgibt:

- Falls das Zeichen ein Kleinbuchstabe (aus dem lateinischen Alphabet) ist, soll der entsprechende Großbuchstabe ausgegeben werden.
- Falls das Zeichen ein Großbuchstabe ist, soll der gleiche Großbuchstabe ausgegeben werden.
- Andernfalls soll ein Fehler ausgegeben werden.

Bitte verwenden Sie zum Einlesen des Zeichens die Funktion `getchar()` wie folgt:

```
char c = getchar();
// danach ist das eingelesene Zeichen ueber c verfuegbar
```

Beispiele:

- Bei Eingabe von `t`, soll `T` ausgegeben werden.
- Bei Eingabe von `T`, soll `T` ausgegeben werden.
- Bei Eingabe von `7`, soll `Fehler: Kein Buchstabe` ausgegeben werden.

```
// Umwandlung von Klein- in Grossbuchstaben

#include <stdio.h>

int main(void)
```

```
{
    char c;
    c = getchar();

    if (c >= 'a' && c <= 'z')
        printf("%c\n", c - 32);

    else if (c >= 'A' && c <= 'Z')
        printf("%c\n", c);

    else
        printf("Fehler: Kein Buchstabe\n");

    return 0;
}
```

Aufgabe 4: Berechnung des Mittelwerts (Abgabe als `mittelwert.c`)

Schreiben Sie ein Programm, das als erstes eine positive ganze Zahl `n` einliest, dann `n` Gleitkommazahlen (`double`) einliest und als Ergebnis den Mittelwert der eingegebenen Gleitkommazahlen ausgibt.

Bitte verwenden Sie zum Einlesen eines `double`-Wertes die Funktion `scanf()` wie folgt:

```
double d;
scanf("%lf",&d);
// danach ist der double-Wert ueber d verfuegbar
```

Beispiel: Bei der Eingabe von:

```
2
1.0
2.0
```

soll der Mittelwert folgendermaßen ausgegeben werden (die Ausgabe soll immer mit genau 4 Nachkommastellen erfolgen):

```
1.5000
```

```
// Berechnung des Mittelwerts von n Gleitkommazahlen

#include <stdio.h>

int main(void)
{
    int n,i;
    double d, sum;
    scanf("%d", &n);

    i = n;
    while (i > 0) {
        scanf("%lf", &d);
        sum += d;
        --i;
    }

    printf("%4.4f\n", sum / n);

    return 0;
}
```

Hinweis: Für die Berechnung des Mittelwerts ist es nicht notwendig die Eingaben in einem Array abzuspeichern. Das Allozieren eines Arrays auf dem Stack ist außerdem problematisch, da dessen Größe dynamisch vom Benutzer festgelegt wird. Dies kann zu einem Sicherheitsproblem bei der Ausführung führen.