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

TU Kaiserslautern
Fachbereich Informatik

Übungsblatt 3: Programmieren in C (WS 2019/20)

Abgabe: Montag, 25.11.19, 12:00

Aufgabe 1 *Binäre Zahlendarstellung: Gleitkommazahlen (10 Punkte)*

Abgabe: floatingpoint.txt

1. Welche Zahlen sind hier als single-precision float-Zahlen dargestellt? Erklären Sie Ihre Berechnung!
   - 01000000111000000000000000000000
   - 11000011010110000000000000000000
2. Stellen Sie die folgenden Zahlen als single-precision float-Zahlen dar! Erklären Sie Ihre Berechnung!
   - $178,625_{10}$
   - 10010110_2
   - $-152,8_{10}$

Aufgabe 2 *Binäre Zahlendarstellung: Rechnen (8 Punkte)*

Abgabe: arithmetik.txt

Lösen Sie die folgenden Rechenaufgaben in Binärarithmetik (Binär-Addition, d.h. keine Subtraktion)! Falls nötig, berechnen Sie hierzu zunächst das Zweierkomplement der gegebenen Dezimalzahlen.
Erklären Sie Ihre Berechnung!

- 17 + 19
- 16 – 10
- 18 – 23
- (-8) – (-11)

Aufgabe 3 *Wie teuer ist das Laden eines Smartphone-Akkus? (5 Punkte)*

Abgabe: Exclaim als `smartphone.c`

Die meisten Smartphone-Akkus halten gerade mal einen Tag durch. Deshalb ist der Griff zum Ladegerät am Abend für viele Smartphone-Nutzer schon Routine. Aber was kostet es, wenn wir unser Smartphone jeden Tag aufladen?

Nehmen Sie hierzu an, dass Ihr Smartphone-Akku eine Kapazität von 3000 mAh bei einer Spannung von 3.85 V aufweist und täglich zu 80% geladen werden muss. Die Verluste des Ladegeräts können vernachlässigt werden. Eine kW-Stunde Strom kostet im Durchschnitt 30 Cent.

1. Geben Sie die mathematische Formel an, um die Kosten für das Laden des Smartphones pro Tag zu berechnen.
2. Implementieren Sie Ihre Formel als C-Programm `smartphone.c`. Das Programm nimmt als Eingabe die Anzahl an Tagen, für die die Kosten berechnet werden sollen.

Die Ausgabe des Programms soll bei Eingabe `365` folgendermaßen gestaltet sein:

```
Kosten: 1.0118 EUR
```

Hinweis: Die Ausgabe des Geldbetrags muss **mit 4 Nachkommastellen** erfolgen (siehe dazu auch die nächste Aufgabe).

Aufgabe 4 *Formatierte Ausgabe (8 Punkte)*

Abgabe: tabelle.c

Die Funktion `printf` kann zur formatierten Ausgabe verwendet werden. Als ersten Parameter nimmt die Funktion einen Text/String mit Platzhaltern für Werte. Als weitere Parameter folgen dann die Werte, die für die Platzhalter eingesetzt werden. Diese Platzhalter beginnen mit einem Prozentzeichen und enden mit einem *Formatspezifizierer*, welcher beschreibt, wie der Wert umgewandelt werden soll. So steht zum Beispiel `%d` für die Dezimaldarstellung eines `int`-Wertes, `%f` für die Fließkommadarstellung einer Zahl, `%e` für die Darstellung einer Zahl in wissenschaftlicher Darstellung mit Exponent und `%s` für eine Zeichenfolge/String.

Beispiel:

```
printf("In %s ist es %d Grad warm.", "Kaiserslautern", 3);
```

Weitere Optionen können zwischen dem Prozentzeichen und dem Formatspezifizierer am Ende angegeben werden. Eine Zahl direkt nach dem Prozentzeichen gibt eine minimale Breite an. Ist eine Ausgabe des Wertes zu kurz, wird links mit Leerzeichen aufgefüllt, bis die minimale Breite erreicht ist. Wenn vor der Zahl noch ein Minus steht, dann werden die Leerzeichen rechts angehängt.

Nach der optionalen minimalen Breite kann auch noch die Präzision bei Gleitkommazahlen festgelegt werden. Dies geschieht nach einem Punkt durch eine weitere Zahl, welche die Anzahl der Nachkommastellen festlegt.

1. Was ist die Ausgabe des folgenden Programms?

   ```
   #include <stdio.h>

   int main(void)
   {
       printf("<%10d>\n", 1234);
       printf("<%-10d>\n", 1234);
       printf("<%10.5f>\n", 1234.567891011121314);
       printf("<%10.3f>\n", 1234.567891011121314);
       printf("<%.5f>\n", 12345678910.11121314);
       printf("<%14.6e>\n", 1234.567891011121314);
       return 0;
   }
   ```

   Laden Sie das Programm `format.c` von der Homepage und überprüfen Sie Ihre Antwort. (Keine Abgabe!!)

2. Schreiben Sie mit Hilfe von `printf` ein Programm `tabelle.c`, welche einen `int`-Werte n einliest und die `double`-Werte $2^{-1} = 0.5, 2^{-2} = 0.25, \ldots$ bis 2^{-n} ausgibt. Als Ausgabe soll das Programm eine Tabelle der n Werte mit Spalten erzeugen. Dabei sollen die Spalten mit | voneinander getrennt sein, jede Spalte soll mindestens 12 Zeichen breit sein und die Zahlen sollen alle 8 Nachkommastellen haben, so dass für die Eingabe von 4 die Ausgabe wie folgt aussieht:

   ```
   0.50000000 | 0.25000000 | 0.12500000 | 0.06250000
   ```