Brandenburgische
Technische Universität
Cottbus - Senftenberg

Brandenburgische Technische Universität Cottbus – Senftenberg
Institut für Informatik, Informations- und Medientechnik
PD Dr.-Ing. habil. Thomas Hinze

Wintersemester 2015 / 2016

Abgabe der Lösungen: bis Mittwoch, 09.12.2015

Einführung in die Programmierung
Übungsblatt 3

Hinweise

- Erstellen Sie bitte eine schriftliche Lösung der nachfolgenden Aufgaben und erzeugen Sie daraus eine pdf-Datei. Sie können auch Ihre (lesbare!) handschriftliche Lösung als pdf einscannen. Achten Sie bitte darauf, dass am Beginn der ersten Seite des pdfs Ihr *Name*, Ihre *Matrikelnummer* und der Name Ihres *Tutors* (Laborübungsgruppe) vermerkt ist. Jede Aufgabe, die durch **(Laborübung)** gekennzeichnet ist, beinhaltet als Lösung auch einen C-Quelltext. Bitte packen Sie Ihr pdf gemeinsam mit allen C-Quelltextdateien in eine zip-Datei, die Sie termingemäß im moodle hochladen.
- Sie bekommen eine Bewertung Ihrer Lösung mit der Information, ob sie ausreichend zum Bestehen des Übungsblattes ist. Im Falle einer nichtausreichenden Lösung haben Sie Gelegenheit, eine verbesserte Version zeitnah, aber spätestens bis 01.02.16 im moodle nachzureichen. Plagiate werden nicht als ausreichende Lösungen anerkannt.

Aufgabe 1

- Erläutern und vergleichen Sie die Schleifenkonstrukte `while`, `do-while` und `for`. Die drei Schleifenkonstrukte können sich gegenseitig semantisch äquivalent simulieren, zum Beispiel lässt sich jede `for`-Schleife auch als `while`-Schleife formulieren. Geben Sie alle möglichen Transformationen an.
- **(Laborübung)** Betrachten Sie folgende vier Quelltextauszüge. Welche der Schleifen terminieren? Wie oft werden die terminierenden Schleifen jeweils durchlaufen, und welche Werte haben die Variablen `i` und `j` nach dem Verlassen der Schleife?

```
/* --- (1) --- */
int i = 1;
int j = 1;
do {
    i = i + j;
    j++;
} while (i < 20);
```

```
/* --- (2) --- */
int i = 100;
int j = 27;
while (i != j) {
    i = i / 20;
    j = j / 3;
}
```

```
/* --- (3) --- */
int i = 1;
int j = 20;
while (i+j > i) {
    i = i + 2;
    j--;
}
```

```
/* --- (4) --- */
int i = 26;
int j = 24, x = 0;
for(x=0; x<1000; x++) {
    i = i/12 + 23*x;
    j = (x--)+j+5;
}
```

Aufgabe 2

(Laborübung) Die Kreiszahl π kann auf vielerlei Arten berechnet werden. Hier betrachten wir ein probabilistisches Verfahren, welches auch in einer Weihnachtsbäckerei Anwendung finden könnte. Als Utensilien genügen ein quadratisches Backblech und eine darauf gestellte kreisförmige Kuchenform, deren Durchmesser der Kantenlänge des Blechs gleicht. Nun werden aus einiger Höhe möglichst zufällig Zuckerstreusel auf das Blech verteilt. Danach zählt (oder wiegt) man die Streusel, die auf dem Blech in die Kuchenform gefallen sind. Aus dem Verhältnis zur Zahl der Streusel, die insgesamt auf das Blech gefallen sind, kann die Kreiszahl π errechnet werden:

$$\frac{\#_{\text{Kreis}}}{\#_{\text{Quadrat}}} = \frac{\pi \cdot r^2}{(2 \cdot r)^2} \qquad \text{und somit} \qquad \pi = 4 \cdot \frac{\#_{\text{Kreis}}}{\#_{\text{Quadrat}}}$$

Schreiben Sie ein C-Programm, welches das beschriebene Verfahren zur näherungsweisen Berechnung der Kreiszahl π umsetzt. Wählen Sie der Einfachheit halber ein Quadrat der Kantenlänge Eins, auf dem eine Viertelkreisscheibe mit Radius Eins liegt. Durch Einbinden der `stdlib.h` steht Ihnen die Funktion `int rand(void)` zur Verfügung, die mit jedem Aufruf eine ganzzahlige Pseudozufallszahl zwischen 0 und `RAND_MAX` zurückliefert. Die Konstante `RAND_MAX` ist ebenfalls in `stdlib.h` definiert und hat meist den Wert $2^{31} - 1$. Die x-Koordinate eines Zuckerstreusels als `float`-Wert mit $0 \leq x < 1$ ergibt sich durch:
`x = (float) rand() / (float) RAND_MAX;`
Analog lässt sich auch eine (pseudo)zufällige y-Koordinate ermitteln. x- und y-Koordinate definieren zusammen ein Zuckerstreusel, der auf das Blech fällt. Für Zuckerstreusel, die in den Viertelkreis mit Radius 1 gefallen sind, gilt: $x^2 + y^2 \leq 1$

Gestalten Sie Ihr Programm so, dass insgesamt 1 000 000 Streusel fallen, bevor die ermittelte Näherung für π ausgegeben wird. Erhöhen Sie die Genauigkeit, indem Sie 100 000 000 Streusel fallen lassen.

Aufgabe 3

Geben Sie für die Gleitkommazahlen $x = \pm m \cdot 2^e$

$0.5625 \qquad 1.5 \cdot 2^{127} \qquad -181.25 \cdot 2^{-23} \qquad 9.81$

vom Typ `float` jeweils ihre Binärdarstellung $v, z(e), z(m)$ mit $x = (-1)^v \cdot z(m) \cdot 2^{z(e)}$ im Standard IEEE 754 single an. Dort stehen insgesamt 32 Bit für die Binärdarstellung zur Verfügung, wovon 1 Bit auf das Vorzeichen v entfällt, 8 Bit auf den charakteristischen Exponenten $z(e)$ und 23 Bit auf die normalisierte Mantisse $z(m)$. Es gilt: $v = \begin{cases} 0 & \text{falls } x \geq 0 \\ 1 & \text{falls } x < 0 \end{cases}$ sowie $z(e) = (e + 127_{(10)})_{(2)}$.

Aufgabe 4

(Laborübung) Miss Marple plant einen gemütlichen Bridge-Abend und möchte als Mitspielerinnen drei ihrer Freundinnen Anita, Beate, Chris, Dunja und Evita einladen. Bei der Einladung muss sie allerdings auf die Sympathien und Antipathien der Damen achten, sonst kommt es zum Eklat!

(1) Wenn Anita eingeladen wird, dann auch Chris, sonst ist sie beleidigt.
(2) Dunja kommt nur, wenn auch Chris eingeladen wird.
(3) Beate und Dunja können sich nicht ausstehen und dürfen auf keinen Fall beide eingeladen werden.
(4) Evita und Beate kommen immer gemeinsam oder gar nicht.

Helfen Sie Miss Marple bei der Erstellung der Einladungen, und schreiben Sie ein C-Programm, das alle möglichen Kombinationen der Damen ausprobiert und nur die zulässigen Kombinationen ausgibt.

Aufgabe 5

(Laborübung) Implementieren Sie in C die Ansteuerungslogik für eine *Sieben-Segment-Anzeige*. Hierzu werden vier Eingabebits eingelesen, die zusammen eine Bitkette der Form $x_3x_2x_1x_0$ darstellen. Jede Bitkette steht für eine Hexadezimalziffer, zum Beispiel 1010 für A. Entwickeln Sie für jedes der sieben Segmente a bis g eine aussagenlogische Schaltfunktion in Abhängigkeit der Eingabebits entsprechend der Grafik.