Universität Augsburg
Institut für Informatik

Übung zur Vorlesung Informatik 1 WS 2017/18
Fakultät für Angewandte Informatik
Lehrprofessur für Informatik
Prof. Dr. Lorenz, Lev Sorokin, Johannes Metzger 25.10.2017

Übungsblatt 1

Abgabe: Donnerstag 02.11.2017, 12:00 Uhr, da Mittwoch Feiertag (Postkasten der Veranstaltung und E-Mail an Tutor)

- Dieses Übungsblatt muss im Team abgegeben werden (Einzelabgaben sind nicht erlaubt!).
- Die Einteilung in **Teams** erfolgt in den zugewiesenen Übungsgruppen ab Montag, den 23.10., bis Freitag, den 27.10.
- Bitte zur Angabe von Namen, Übungsgruppe und Teamnummer das **Deckblatt** verwenden!
- Die Übungsaufgaben haben jeweils eine Schwierigkeitsangabe: * leichte Aufgabe / ** mittelschwere Aufgabe / *** schwere Aufgabe
- Die **Zeitangaben** geben zur Orientierung an, wie viel Zeit für eine Aufgabe später in der Klausur vorgesehen wäre; gehen Sie davon aus, dass Sie zum jetzigen Zeitpunkt wesentlich länger brauchen und die angegebene Zeit erst nach ausreichender Übung erreichen.
- Alle Aufgaben dieses Übungsblattes sind **klausurrelevant**.
- Das Übungsblatt wird vom **6.11.2017** bis zum **10.11.2017** in den Übungsgruppen besprochen.

Allgemeine Hinweise zur Abgabe der Übungsblätter

1. Abgabeort: Briefkästen Informatik 1 (Erdgeschoss, hinter 2045 N)
2. Abgabe schriftlich: Alle Aufgaben + Quellcode; Deckblatt verwenden; Links oben tackern (z.B. im Raum der Fachschaft, 1007 N)
3. Abgabe per E-Mail an den Tutor: Nur Quellcode (andere Aufgaben optional); Quellcode als `*.c` / `*.h` - Dateien!

Allgemeine Hinweise zu den Programmieraufgaben:

- Achten Sie, wie auf Übungsblatt 0 erwähnt, bei allen zukünftigen Programmieraufgaben auf *Kompilierbarkeit* und *Einhaltung der Coding Conventions*; auch dann, wenn es nicht explizit im Aufgabentext gefordert ist.
- Gehen Sie davon aus, dass alle Programme in der Programmiersprache C zu erstellen sind.
- Kompilieren Sie alle Ihre Programme mit den folgenden *Compiler-Schaltern*:

  ```
  -Wall -Wextra -ansi -pedantic
  ```

  Achten Sie darauf, dass trotz Verwendung dieser Schalter keine Fehler-/Warnmeldungen erzeugt werden.

Aufgabe 1 * *(Wissensfragen, 1 Minute pro Frage, insgesamt 17 Minuten)*

Beantworten Sie möglichst knapp und genau in einem Satz die folgenden Fragen:

a) (Quellcode)

1. Wie lautet die Anweisung, um zwei Variablen `i` und `j` vom Datentyp `int` in einer Zeile zu deklarieren?
2. Wie lautet die Anweisung, um einer Zeichen-Variable `a` das ASCII-Zeichen `<` als Wert zuzuweisen?
3. Geben Sie die Konstante `0.000314e4` in Festkommaschreibweise an.
4. Welchen Datentyp hat die Konstante `+12000`?
5. Ist `-1e-12` eine gültige Konstante? Wenn ja, von welchem Datentyp? Wenn nein, warum?
6. Ist `'--1'` eine gültige Konstante? Wenn ja, von welchem Datentyp? Wenn nein, warum?
7. Ist `'\\'` eine gültige Konstante? Wenn ja, von welchem Datentyp? Wenn nein, warum?
8. Welchen Wert hat der arithmetische Ausdruck `1 + 2 * 1 - 1 / 1`?
9. Mit welcher Wertzuweisung quadriert man den Wert einer positiven Zahl-Variable `x`?

b) (Fragen zu http://www2.hs-fulda.de/~klingebiel/c-stdlib/stdio.htm)

1. Was bewirkt die Umwandlungsangabe `%3i` in der Formatzeichenkette von `printf`?
2. Was bewirkt die Umwandlungsangabe `%.4f` in der Formatzeichenkette von `printf`?
3. Ist `%+07i` eine gültige Umwandlungsangabe in der Formatzeichenkette von `printf`? Wenn nein, warum?
4. Ist `%p` eine gültige Umwandlungsangabe in der Formatzeichenkette von `printf`? Wenn nein, warum?
5. Welche Ausgabe bewirkt die Anweisung `printf("%03.1f",1.0);` ?
6. Welche Ausgabe bewirkt die Anweisung `printf("%05.1f",1.111);` ?

c) (Programme kompilieren und ausführen)

1. Mit welchem Kommandozeilenbefehl führt man ein Programm `a` mit den Kommandozeilenparametern `-x` und `test` aus?
2. Mit welchem Kommandozeilenbefehl kompiliert man die Quellcode-Datei `a.c` und speichert den Maschinencode in der Datei `Programm`?

Aufgabe 2 * *(Fehler finden und verbessern, insgesamt 20 Minuten)*

Die folgenden C-Programme sollten direkt nach dem Kompilieren auf der Konsole ausführbar sein und die angegebene Ausgabe haben. Leider haben sich einige Fehler eingeschlichen! Nennen Sie die jeweiligen Zeilennummern und geben Sie zu jedem Fehler eine genaue Beschreibung an. Schreiben Sie jeweils eine fehlerbereinigte Version des Programms mit der gewünschten Ausgabe.
a) (*, 5 Minuten)

```
1: Main{void}
2: (
3: Return (0)
4: )
```

b) (*, 5 Minuten)
Ausgabe: `Hallo`

```
1: ##include <stdio.h>
2:
3: char main(void)
4: {
5: printf Hallo;
6: }
```

c) (*, 5 Minuten)
Ausgabe: `5.50`

```
1: int main
2: {
3: double d = 5.50;
4: print("%e",d);
5: return 0;
6: }
```

d) (*, 5 Minuten)
Ausgabe: `a`

```
1: #include "stdio.h"
2:
3: int main()
4: {
5: char d = 'a';
6: printf(d);
7: return 0;
```

Aufgabe 3 ** *(Einfache Programme mit der Funktion `printf`, 20 Minuten)*

a) (*, 2 Minuten)
Schreiben Sie ein C-Programm, das das Wort `line` und danach zwei leere Zeilen (Zeilenumbrüche) ausgibt.

b) (*, 2 Minuten)
Schreiben Sie ein C-Programm mit folgender Ausgabe: `%\\n`

c) (**, 3 Minuten)
Schreiben Sie ein C-Programm, das die Anzahl der bei Programm-Aufruf übergebenen Kommandozeilenparameter und danach in einer neuen Zeile den Programmnamen ausgibt.

Hinweis: Benutzen Sie für die Ausgabe des Programmnamens den Platzhalter `%s`.

d) (*, 3 Minuten)
Schreiben Sie ein C-Programm, das das Vierfache der Anzahl der bei Programm-Aufruf übergebenen Kommandozeilenparameter ausgibt.

Hinweis: Die Warnung, dass die Variable `argv` nicht verwendet wird, darf ignoriert werden.

e) (**, 3 Minuten)
Schreiben Sie ein C-Programm, das die Quadratwurzel der verdoppelten Anzahl der bei Programm-Aufruf übergebenen Kommandozeilenparameter ausgibt.

Hinweise:

- Benutzen Sie eine geeignete Funktion aus der Bibliothek `math.h`. (auf manchen Systemen muss der Compiler-Schalter `-lm` verwendet werden, um die `math.h`-Bibliothek zu benutzen)
- Die Warnung, dass die Variable `argv` nicht verwendet wird, darf ignoriert werden.

f) (**, 3 Minuten)
Schreiben Sie ein C-Programm, das den Programmnamen und die ersten zwei Kommandozeilenparameter jeweils getrennt durch ein Leerzeichen ausgibt. Nehmen Sie dabei an, dass bei Programmaufruf mindestens zwei Kommandozeilenparameter übergeben werden.

Hinweise:

- Die Warnung, dass die Variable `argc` nicht verwendet wird, darf ignoriert werden.
- Benutzen Sie für die Ausgabe des Programmnamens und der Kommandozeilenparameter den Platzhalter `%s`.

g) (**, 4 Minuten)
Schreiben Sie ein C-Programm, das mit `scanf` eine Dezimalzahl einliest und dann den Rückgabewert des `scanf`-Aufrufs ausgibt.

Aufgabe 4 ** *(`scanf` und eigene Funktionen, 17 Minuten)*

a) (*, 1 Minute)
Entwerfen Sie den Kopf einer Funktion, die das arithmetische Mittel zweier ganzer Zahlen berechnet.

b) (*, 2 Minuten)
Implementieren Sie eine Funktion mit dem Kopf
`double calc_circumference_square(double length, double width)`,
die den Umfang eines Rechtecks mit Länge `length` und Breite `width` berechnet und zurückgibt.
Benutzen Sie dazu folgende Vorlage:

```c
#include <stdio.h>
#include <math.h>

double calc_circumference_square(double length, double width)
{
        /* Hier kommt Ihre Lösung hin */
}

int main(void)
{
        double length, width;
        scanf("%lf %lf", &length, &width);
        printf("Der Umfang eines Rechtecks mit Länge %f und Breite %f ist %f",
                        length,
                        width,
                        calc_circumference_square(length, width));
        return 0;
}
```

c) (**, 4 Minuten)
Schreiben Sie ein C-Programm, das vom Benutzer ein ASCII-Zeichen einliest und das in der ASCII-Tabelle vorhergehende Zeichen ausgibt.

d) (**, 4 Minuten)
Schreiben Sie ein C-Programm, das vom Benutzer ein ASCII-Zeichen einliest und dann ausgibt, wie viele Zeichen in der ASCII-Tabelle das Zeichen vom Zeichen 'A' entfernt sind.
Hinweis: In `stdlib.h` gibt es eine Funktion zur Berechnung des Absolutwerts einer ganzen Zahl (eigene Recherche).

e) (***, 6 Minuten)
Schreiben Sie ein C-Programm, das vom Benutzer eine ganzzahlige Zeitdauer in Millisekunden einliest und diese dann in Stunden umrechnet und ausgibt. Die Umrechnung von Millisekunden in Stunden soll in einer eigenen Funktion erfolgen.

Es gibt folgende Arten von **Zusatzaufgaben**:

- Aufgaben zu **Schulstoff**, der Voraussetzung zum Verständnis der Vorlesung ist. Es wird davon ausgegangen, dass Sie diesen (Schul-)Stoff bereits beherrschen.
- **Alte Klausuraufgaben** zu Themen des Übungsblatts.

Besprechungen / Lösungen zu Zusatzaufgaben:

- Alle Zusatzaufgaben sind **unbewertet** und dementsprechend nicht abzugeben.
- Alle Zusatzaufgaben werden in der **Globalübung** besprochen.

Zusatzaufgabe A *(Wiederholung von Schulstoff: Potenzen und Einheiten)*

Die erwähnten Einheiten müssen bei Bedarf selbst recherchiert werden.

a) Wie viel Kilogramm (kg) entsprechen 13 Gramm (g)?
b) Wie viele Millisekunden (ms) haben 17.51 Sekunden (s)?
c) Schreiben Sie die Zahl 100000 in der Form 10^x mit einer passenden ganzen Zahl x.
d) Schreiben Sie die Zahl 0.0000001 in der Form 10^x mit einer passenden ganzen Zahl x.
e) Schreiben Sie die Zahl 1024 in der Form 2^x mit einer passenden ganzen Zahl x.
f) Schreiben Sie die Zahl 0.125 in der Form 2^x mit einer passenden ganzen Zahl x.
g) Wie viel Mega-Hertz (MHz) entsprechen 2.4 Giga-Hertz (GHz)?
h) Wie viel Giga-Byte (GB) entsprechen 54 Kilo-Byte (KB)?
i) Wie viele Mega-Byte (MB) ergeben 30 Byte, 1.2 GB und 400 Kilo-Byte zusammenaddiert?
j) Stellen Sie die Zahl 87 in der Form $m \cdot 10^n$ dar mit einer ganzen Zahl n und einer Zahl m mit $1 \leq m < 10$.
k) Stellen Sie die Zahl 0.12 in der Form $m \cdot 10^n$ dar mit einer ganzen Zahl n und einer Zahl m mit $1 \leq m < 10$.
l) Stellen Sie die Zahl 24 in der Form $m \cdot 2^n$ dar mit einer ganzen Zahl n und einer Zahl m mit $1 \leq m < 2$.
m) Stellen Sie die Zahl 0.15 in der Form $m \cdot 2^n$ dar mit einer ganzen Zahl n und einer Zahl m mit $1 \leq m < 2$.
n) 10 Kibi-Byte entsprechen 10240 Byte. Wie viele Byte hat dann ein Kibi-Byte?