Universität Augsburg
Institut für Informatik

Übungen zum Vorkurs Informatik WS 2020/2021
Fakultät für Angewandte Informatik
Lehrprofessur für Informatik
Prof. Dr. Lorenz, Johannes Metzger 12.10.2020

Übungsblatt 1

Allgemeine Hinweise:

- Achten Sie bei allen Programmieraufgaben auf *Kompilierbarkeit* und *Einhaltung der Coding Conventions* (werden zu jedem Thema in den Vorlesungsfolien erklärt); 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 *(Compiler)*

a) Starten Sie ein Kommandozeilen-Programm

b) Testen Sie mit Hilfe des Kommandozeilen-Programms, ob der gcc-Compiler auf Ihrem Rechner installiert ist

c) Installieren Sie ggf. den gcc-Compiler (Anleitungen stehen für Windows und Mac online zur Verfügung)

Haben Sie trotz Befolgung der Anleitungen Probleme, erhalten Sie ***schon ab dem 5.10.*** *Unterstützung unter* `https://discord.gg/SRYedBz`*:*

Discord *ist ein Chatsystem, das wir zur durchgehenden Betreuung im Vorkurs verwenden. Der angegebene Server wird von Studierenden der Fachschaft für Studierende der Informatik und Mathematik betrieben und wird temporär von uns zur Vorkurs-Betreuung mitgenutzt. In der Kategorie "Vorkurs Informatik" gibt es verschiedene Räume, in denen Fragen zur GCC-Einrichtung (ggf. mittels Screensharing) und (später) den einzelnen Aufgaben gestellt werden können. Alle Räume außerhalb dieser Kategorie werden nicht von uns betreut und dienen der reinen Kommunikation zwischen Studierenden. Um die Einhaltung der Regeln im Disclaimer-Raum wird gebeten.*

Aufgabe 2 *(Kommandozeile)*

a) Starten Sie ein Kommandozeilen-Programm

b) Legen Sie mit Hilfe des Kommandozeilen-Programms an geeigenter Stelle ein Verzeichnis für diesen Vorkurs an und erstellen Sie in diesem Verzeichnis jeweils ein Unterverzeichnis für jedes Kapitel des Vorkurses (in diesen Unterverzeichnissen legen Sie später Ihre Programme ab).

(Anleitungen stehen für Windows und Mac online zur Verfügung)

Aufgabe 3 *(Texteditor)*

a) Installieren Sie einen Texteditor zur Erstellung der Programme oder wählen Sie dafür einen bereits auf Ihrem Rechner installierten Texteditor.

Im Kurs unterstützen wir den Texteditor `Atom` (verfügbar für alle Betriebssysteme - siehe auch Vorlesungsfolien).

Für Windows kann man auch `Notepad++` empfehlen (siehe auch Vorlesungsfolien).

b) Setzen Sie in Ihrem Texteditor (nach Möglichkeit) die Tabulatorlänge auf `8` und die Zeichencodierung auf `UTF-8`.

Aufgabe 4 *(Test)*

a) Erstellen Sie ein C-Datei `Aufgabe4a.c` mit unten stehendem Quellcode, kompilieren Sie diese und führen Sie das erstellte Programm aus.

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

int main(void)
{
        int x, y;
        x = rand();
        y = 2 * x;
        printf("Das␣Ergebnis␣ist:␣%i", y);
        return 0;
}
```

Hinweise:

- *Tippen Sie den Quellcode ab (kein Copy / Paste!)*
- *Benutzen Sie ihren Texteditor zur Erstellung der C-Datei*
- *Benutzen Sie ein Kommandozeilen-Programm, um die C-Datei zu kompilieren und das erstellte Programm auszuführen*

b) Erstellen Sie ein C-Datei `Aufgabe4b.c` mit unten stehendem Quellcode, ohne diesen anders zu formatieren, kompilieren Sie diese und führen Sie das erstellte Programm aus.

```c
#include <stdio.h>
#include <stdlib.h>
int main(void){int x,y;x=rand();y=
2*x;printf("Das␣Ergebnis␣ist:␣%i\n"
,y);return 0;}
```

Die Programme aus den Teilaufgaben a) und b) bestehen aus denselben Anweisungen und haben dieselbe Ausgabe. Der Quellcode unterscheidet sich nur durch seine Formatierung. Programmierkonventionen sorgen u.a. für eine übersichtliche Formatierung wie in Teilaufgabe a). Das macht den Code besser verständlich und erleichtert die Fehlersuche. Teilaufgabe b) ist ein Beispiel für unstrukturierten und damit schlechten Quellcode.

c) Variieren Sie den Quellcode aus der C-Datei `Aufgabe4a.c` wie in den folgenden Teilaufgaben beschrieben, kompilieren Sie jeweils den modifizierten Quellcode und beobachten Sie die Ausgaben des Compilers (können Sie sich diese schon erklären?):

- Fügen Sie an verschiedenen Stellen eine Leerzeile ein.
- Fügen Sie an verschiedenen Stellen ein Leerzeichen ein.
- Lassen Sie an verschiedenen Stellen ein Leerzeichen weg.
- Lassen Sie einen der Strichpunkte weg.
- Tauschen Sie die Zeilen 2 und 3.
- Lassen Sie Zeile 1 weg.
- Beginnen Sie `main` mit einem Großbuchstaben.
- Beginnen Sie `printf` mit einem Großbuchstaben.
- Beginnen Sie `include` mit einem Großbuchstaben.
- Beginnen Sie `stdio` mit einem Großbuchstaben.
- Beginnen Sie `int` mit einem Großbuchstaben.
- Beginnen Sie `return` mit einem Großbuchstaben.
- Lassen Sie eines der `#`-Zeichen weg.
- Lassen Sie eines der `<`-Zeichen weg.
- Lassen Sie eines der `(`-Zeichen weg.
- Lassen Sie eines der }-Zeichen weg.
- Lassen Sie nach `rand` das Klammerpaar `()` weg.
- Schreiben Sie `x` an einer Stelle groß.
- Schreiben Sie `x` an allen Stellen groß.
- Ersetzen Sie `x` an allen Stellen durch `zahl`.

*Manche der Modifizierungen erzeugen Warnungen oder Fehler, manche nicht. Kommt es weder zu einer Warnung, noch zu einem Fehler, hat die Änderung lediglich Auswirkung auf das Format und ggf. die Übersichtlichkeit des Quellcodes. Eine **Warnung** verhindert nicht die Erzeugung des Maschinenprogramms, sondern deutet nur auf schlechten Programmierstil hin. Ein **Fehler** führt dazu, dass das Maschinenprogramm nicht erstellt werden kann, da der Compiler den Quellcode nicht interpretieren kann - d.h. die sog. **Syntax** des Quellcodes ist nicht korrekt. In den folgenden Tagen lernen Sie, nach welchen Regeln Sie syntaktisch korrekten Quellcode erstellen können.*

Aufgabe 5 *(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 $m \cdot 10^x$ dar mit einer ganzen Zahl x und einer Zahl m mit $1 \leq m < 10$ (m heißt *Mantisse* und x *Exponent* der sog. *normierten Gleitpunktdarstellung zur Basis* 10 $m \cdot 10^x$).

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 $1 \leq m < 10$ (m heißt *Mantisse* und x *Exponent* der sog. *normierten Gleitpunktdarstellung zur Basis* 10 $m \cdot 10^x$).

l) Stellen Sie die Zahl 24 in der Form $m \cdot 2^x$ dar mit einer ganzen Zahl x und einer Zahl m mit $1 \leq m < 2$ (m heißt *Mantisse* und x *Exponent* der sog. *normierten Gleitpunktdarstellung zur Basis* 2 $m \cdot 2^x$).

m) Stellen Sie die Zahl 0.15 in der Form $m \cdot 2^x$ dar mit einer ganzen Zahl x und einer Zahl m mit $1 \leq m < 2$ (m heißt *Mantisse* und x *Exponent* der sog. *normierten Gleitpunktdarstellung zur Basis* 2 $m \cdot 2^x$).