Praktikum 1 – Erste Programme in C

Dauer: 2 Lektionen

Allgemeine Hinweise

Ich überlasse es Ihnen, ob Sie alleine oder in Zweiergruppen arbeiten wollen. Natürlich können (und sollen!) Sie auch mit anderen Gruppen diskutieren und Fragen stellen; dennoch sollten Sie das in einer Aufgabe zu lösende Problem immer zuerst selbst in Ihrer Gruppe analysieren und sich einen möglichen Lösungsvorschlag ausdenken, bevor Sie diesen mit anderen Kollegen besprechen. Der daraus resultierende Lerneffekt ist viel grösser, als wenn Sie gleich von Beginn weg jemand anderes nach einer Lösung fragen oder gar den Code irgendwo abschreiben.

Es wird jeweils ein Lösungsvorschlag zu den Praktika auf der Webseite zur Veranstaltung zur Verfügung gestellt. Betrügen Sie sich aber nicht selbst indem Sie die Praktika nicht selbst absolvieren, einfach nachträglich die Lösung anschauen und denken „das hätte ich wahrscheinlich schon gekonnt“ oder ähnlich. Beim Programmieren führt kein Weg am selbstständigen, praktischen Üben vorbei.

Entwicklungsumgebung

Das Dokument CDT_Anleitung_Raspberry_Pi.pdf, auf OLAT abgelegt, zeigt auf, wie Sie sich von Ihrem Notebook aus mit dem Raspberry Pi verbinden können und wie Sie mit der zu verwendenden Entwicklungsumgebung ein erstes Programm erstellen können.

Falls Sie nicht mit Raspberry Pi arbeiten, beschreibt das Dokument cdt_environment.pdf, ebenfalls auf OLAT abgelegt, die zu verwendende Entwicklungsumgebung bei Benützung der bereitgestellten Ubuntu Virtual Machine.

Grundsätzliches zu Dateinamen, Kompilieren und Ausführen

In C sollten die Files, die den Programmcode enthalten, die Endung .c haben; in Aufgabe 2 also zB temperatur.c. Um temperatur.c unter Linux/Unix zu kompilieren, geben Sie auf der Kommandozeile folgendes ein:

```
$> gcc -ansi temperatur.c
```

Die Option `-ansi` ist nicht zwingend nötig, empfehle ich Ihnen aber denn damit wird vom Compiler nur korrekter ANSI-C Code akzeptiert. Falls keine Fehler auftreten, wird die ausführbare Datei a.out generiert (a.exe bei Verwendung von cygwin). Das Programm kann dann mit

```
$> a.out        (oder ./a.out)
```

ausgeführt werden. Sie können mit der Option `-o` anstelle von a.out einen anderen Namen spezifizieren, zB

```
$> gcc -ansi -o temperatur temperatur.c
```

Eine weitere nützliche Option ist `-Wall`, wodurch der Compiler alle Warnungen ausgibt und damit auf potentielle Programmierfehler hinweisen kann:

```
$> gcc -ansi -Wall -o temperatur temperatur.c
```

Mit den grafischen Entwicklungsumgebungen verwenden Sie einfach die entsprechenden Menubefehle um Programme zu kompilieren und auszuführen.

Aufgabe 1: Hello World

Bringen Sie als erste Aufgabe das Hello World Beispiel aus den Vorlesungsunterlagen in ihrer C-Umgebung zu laufen. Tippen Sie es in einem Editor ein, kompilieren Sie es, und lassen Sie es anschliessend laufen.

Aufgabe 2: Konvertierung Celsius ↔ Fahrenheit

Schreiben Sie ein Programm in C, welches für die Temperaturwerte von -100 bis +200 Grad Fahrenheit (in 20er Schritten) die entsprechenden Werte in Celsius ausgibt. Die Ausgabe soll in einer sauber formatierten Tabelle analog zum folgenden Beispiel erscheinen (Celsius auf zwei Stellen gerundet, vier Leerzeichen zwischen F'heit und Celsius):

```
F'heit    Celsius
-----------------
  -100     -73.33
   -80     -62.22
         .
         .
         .
   160      71.11
   180      82.22
   200      93.33
```

Hinweise:

- Die Formel für die Umrechnung von Fahrenheit (f) in Celsius (c) Werte lautet:
  $c = 5/9 * (f - 32)$
- Für die Aufgabe brauchen Sie eine Schleife und die printf-Funktion. Wenn wir diese in der Vorlesung noch nicht behandelt haben, so arbeiten Sie die entsprechenden Seiten in den Unterlagen selbstständig durch: Seiten 23 & 24 für Schleifen, Seiten 27 & 28 für die printf-Funktion.

Aufgabe 3: Zeichen und Wörter zählen

Schreiben Sie ein Programm in C, welches die Zeichen und Wörter einer mit der Tastatur eingegebenen Zeile zählt. Wortzwischenräume sind entweder Leerzeichen (' ') oder Tabulatorzeichen ('\t'). Eine Zeile wird bei der Eingabe mit einem newline-character ('\n') abgeschlossen, worauf das Programm die Anzahl Zeichen (ohne den abschliessenden newline-character) und Wörter ausgegeben und terminieren soll.

Hinweis:

Lesen Sie die einzelnen Zeichen mit der Funktion `char getchar(void)` aus der stdio.h Library ein. Die Funktion getchar kehrt nicht gleich bei Eingabe des ersten Zeichens zurück, sondern puffert die Daten, bis die Eingabe einer kompletten Zeile mit Return abgeschlossen wird. Dann wird das erste Zeichen aus dem Puffer zurückgegeben und mit weiteren Aufrufen von getchar können die nachfolgenden Zeichen aus dem Puffer gelesen werden. Gibt getchar \n zurück, ist die Zeile komplett zurückgegeben und der Puffer ist wieder leer.

Testen mittels Shell Script (optional):

Um das Programm automatisch mit verschiedenen Inputs zu testen, kann ein einfaches Shell – Script verwendet werden. Dieses arbeitet ein Input-File mit vorbereiteten Sätzen ab und überprüft, ob der Output (Anzahl Zeichen/Wörter) korrekt berechnet wurde.
Eine Anleitung und die vorbereiteten Dateien finden Sie im OLAT unter /Testing.