Einführung in die C-Programmierung

Warum C?

- Sehr stark verbreitet (Praxisnähe)
- Höhere Programmiersprache
- Objektorientierte Erweiterung: C++
- Aber auch hardwarenahe Programmierung möglich (z.B. Mikrokontroller).
- Weitgehende Portabilität (d.h. sowohl Betriebsystem als auch Hardware unabhängig)
- C unterstützt die strukturierte Programmierung
- C Compiler erzeugen kompakten und schnellen Code
  - wichtig für (Ingenieur-)Simulationen

- Viele Bibliotheken erhältlich

Werkzeuge zum Schreiben eines C-Programms

Mindestens folgende Werkzeuge sind zum Erstellen eines C-Programms nötig:

- Editor (z.B. emacs): Zum Schreiben des Programmcodes. Die Quellcode-Dateien haben (meist) die Endung `.c` und `.h`
- Compiler (z.B. gcc)
- Linker

Compilers und Linker

Aufgaben des Compilers:

- Test auf syntaktische Korrektheit
- Übersetzen des Programms in Programm-Module (sog. object files) [Maschienencode]. Diese haben meist die Endung `.o`

Aufgaben des Linkers:
Zusammenfügen der Programm-Module und setzen der Sprungaddressen. Als Resultat erhält man ein ausführbares Programm.

Präprozessor

Der Präprozessor modifiziert von dem eigentlichen Compilieren den Quellcode. Mit seiner Hilfe werden weitere Quelltextdateien (Headerfiles) eingebunden, Teile des Codes ignoriert oder ersetzt.

Eigenschaften von C

- Formatfrei
- Ein Semikolon markiert das Ende einer Anweisung
- Variablen besitzen einen Typ (z.B. ganze Zahl, Charakter).

Kommentare

Kommentare werden in C mit `/*` eingeleitet und mit `*/` beendet.

Beispiel:

```
/* Das ist ein
   Kommentar */
```

Kommentare werden dazu eingesetzt, um ein Programm leichter verständlich zu machen.
Sie werden von Kompiler ignoriert.

Datentypen - Variablen

Grunddatentypen:

- char - ein einzelnes Zeichen
- short - ein kleiner ganzzahliger Wert
- int - ein ganzzahliger Wert
- long - ein großer ganzzahliger Wert
- float - ein Gleitkommawert
- double - ein Gleitkommawert mit doppelter Genauigkeit

Die Größen der Objekte sind maschinenabhängig.
Außerdem sind zusammengesetzte Datentypen, wie Vektoren und Strukturen möglich.

Vereinbarungen - Deklarationen

In C müssen alle Variablen vereinbart werden, bevor Sie benutzt werden. Dies wird als Deklaration bezeichnet. Beispiel:

```
int step;
float temparatur;
```

Funktionen

- Funktionen dienen zur Strukturierung von Programmen. Funktionalitäten können so gekapselt werden.
- Funktionen haben in C einen Typ (z.b. `void, int`). Der Rückgabewert muss dem Typ entsprechen.
- Funktionsaufrufe erkennt man an den runden Klammern nach dem Funktionsnamen: z.B. `getchar(), pow(x,y)`. Die Argumente stehen innerhalb der Klammern.
- Die Startfunktion wird in C mit `int main()` bezeichnet.

Kontrollstrukturen

- Schleifen
- Entscheidungen
- Zusammenfassung von Anweisungen: Blöcke. Die geschweiften Klammern { } dienen zur Definition eines Blockes.
- Unterprogramme (Funktionen)

Bibliotheken

Mehrere kompilierte Module mit Funktionen können zu Funktionsbibliotheken (sog. Libraries) zusammengefasst werden.

C selbst besitzt keine eingebauten Funktionalitäten. In C gibt es aber eine genormte Standard C Library. Sie enthält häufig benötigte Funktionen für Ein- und Ausgabe, mathematische Funktionen, Verarbeitung von Zeichenketten, Speicherverwaltung und andere Bereiche.

Benötigt man eine Library so muss man die Schnittstelle bekannt geben. Dies macht man mit der Präprozessordirektive `#include`

Beispiel:

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

Hilfe zu Library unter Linux

Unter linux erhalten sie Hilfe zu den meisten C-Standard Bibliotheksaufrufen mit dem Befehl

```
man 3 FUNKTIONSNAME
```

Beispiel:

```
man 3 printf
```

ergibt die Hilfe zum formatierten Schreiben mit der Funktion `printf()`.

Ein kleines Programm

```
/* Das Hello World Programm */

/* Information zur IO-Standardbibliothek einfuegen*/
#include <stdio.h>

/* Funktion namens main definieren. Diese ist vom Typ int und
empfaengt keine Argumente (leere Argumentenliste)*/
int main()
{ /* Blockanfang: Beginn der Funktion main()
/* Aufrufen der IO-Bibliotheksfunktion printf, um
die Zeichenfolge Hallo Welt zu drucken.
\n steht fuer den Zeilentrenner*/
  printf("Hallo Welt\n");

/* Der Rueckgabewert der Funktion ist 0 */
  return 0;
} /* Blockende: Ende der Funktion main() */
```

Übersetzen des Programms mit gcc

Übersetzen der Quelltextdatei halloWelt.c mit:
`gcc -Wall -c halloWelt.c`
`-Wall` : Alle möglichen Warnungen sollen ausgegeben werden.
`-c` : Es soll nur compiliert und nicht gelinkt werden. (*we want do it step by step*)

Jetzt existiert im laufenden Verzeichniss eine Objekt-Datei halloWelt.o .

Aus dieser lässt sich durch benutzen des Linkers eine ausführbare Datei erzeugen:
`gcc -o halloWelt halloWelt.o`

Die Option `-o halloWelt` bewirkt, dass das ausführbare Programm den Namen halloWelt bekommt. Stadardmäßig heisst die Datei a.out

Mit dem Aufruf: `./halloWelt` lässt sich das Programm starten.

Alternativ kann man mit einem Befehlsaufruf kompilieren und linken: `gcc -o halloWelt halloWelt.c`

Schlüsselwörter

Folgende Schlüsselwörter sind geschützt und sollten nicht für Variablen etc. verwendet werden:

```
auto break case char const continue default do double
else enum extern float for goto if int long register
return short signed sizeof static struct switch
typedef union unsigned void volatile while
```