Programmieren I

Programmieren in C

WS 2024

1 Einführung

C ist eine imperative Programmiersprache, die bereits seit 1972 existiert und somit eine der ältesten Programmiersprachen ist. Aufgrund der Eigenschaft, dass C eine Low-Level Sprache ist, wird sie oft für Betriebssysteme, Treiber oder embedded Systeme wie Arduinos, RasPis oder Autos verwendet.

Anders als bei vielen moderneren Programmiersprachen, gibt es in C relativ wenige Beschränkungen. Prinzipiell macht C genau das, was man ihm sagt. Das bedeutet aber auch, dass man sich selbst um viele Dinge kümmern muss und es in vielen Fällen zu ungewollten Nebeneffekten kommen kann, wenn man nicht aufpasst.

Beispiel 1.1. Es existiert ein Array (der Datentyp ist dabei egal) mit 10 Elementen. Um auf ein Element dieses Arrays zuzugreifen, wird die Syntax `array[i]` verwendet, wobei `i` der Index (also die Stelle) des Elements ist. Das heißt, der Aufruf `array[0]` gibt das erste Element des Arrays zurück, `array[1]` das zweite, usw.

Ruft man allerdings das Element `array[10]` auf, würde man einen Fehler erwarten, da das Array nur 10 Elemente beinhaltet. In vielen Programmiersprachen wäre das auch tatsächlich der Fall; Java würde eine `ArrayIndexOutOfBoundsException` werfen, Python einen `IndexError`, usw. C hingegen verfolgt das Mindset, dass der Entwickler schon weiß, was er tut und gibt somit einfach den Wert zurück, der sich an der 11. Stelle des Arrays befinden würde. Das klingt vielleicht etwas verwirrend, da es gar keinen 11. Wert gibt, aber ein Array ist im Endeffekt nichts anderes als eine durchgängige Reihe an Speicherzellen im Arbeitsspeicher. Über den Index berechnet man die Speicheradresse der jeweiligen Zelle und wenn der Index größer als die Länge des Arrays ist, wird auf eine Speicherzelle zugegriffen, die nicht zum Array gehört. Daher kann man also auch nicht vorhersehen, welcher Wert in dieser Zelle steht. Das ist zumden natürlich auch ein Sicherheitsrisiko, da unbefugt auf fremden Speicher zugegriffen wird und dieser im Wort-Case sensible Daten wie Passwörter beinhalten könnte.

2 Hello, World!

Oft ist das erste Programm, das man in einer (neuen) Programmiersprache schreibt, ein "Hello, World"-Programm. Das ist ein simples Programm, das den Text "Hello, World!" auf der Konsole ausgibt. Der Sinn dahinter ist, zu überprüfen, ob bei der Installation der Entwicklungsumgebung, Runtime, usw. alles korrekt verlief oder ob es zu Fehlern kam.

Möchte man in C sieht ein solches Programm schreiben, könnte das wie folgt aussehen:

```c
#include <stdio.h>

int main() {
  printf("Hello, World!\n");
  return 0;
}
```

Fig. 1: Hello, World! in C

3 Datentypen, Variablen und Operatoren

Der Code *Fig. 1* zeigt bereits die grundlegende Struktur eines C-Programms. Der Startpunkt ist die `main`-Funktion, welche somit beim Start des Programms aufgerufen wird. Der genaue Aufbau einer Funktion wird in Kapitel 4 genauer erläutert.

3.1 Datentypen

Ein grundlegender Bestandteil von Programmiersprachen sind Datentypen. Mithilfe von Datentypen unterscheidet man, welchen Wert eine Variable speichern kann. Eine Art von Datentypen sind die **primitiven Datentypen**. Diese bilden die Basis für alle anderen Datentypen. Folgende primitive Datentypen gibt es in C:

Datentyp	Keyword	Speichergröße	Wert
Integer	int	16 bit / 32 bit	Ganze Zahlen
Short	short	16 bit	Ganze Zahlen
Long	long	32 bit	Ganze Zahlen
Long Long	long long	64 bit	Ganze Zahlen
Float	float	32 bit	Gleitkommazahlen
Double	double	64 bit	Gleitkommazahlen
Character	char	8 bit	Ganze Zahlen

Table 1: Primitive Datentypen in C

Für alle ganzzahligen Datentypen kann außerdem ein `unsigned` vorangestellt werden, um nur positive Werte zuzulassen und somit den Wertebereich "nach oben" zu verdoppeln.

Die einzelnen Speichergrößen sind zudem abhängig von der verbauten CPU sowie dem verwendeten Compiler. So können insbesondere die Speichergrößen von Integern und Longs variieren.

3.2 Variablen

Wenn man in C von einer Variable spricht, meint man damit eigentlich nur einen fest reservierten Platz im Arbeitsspeicher. In C haben Variablen außerdem einen festen Datentypen, den man bestimmt, wenn man die Variable **deklariert** (also dem Compiler mitteilt, dass diese Variable existiert). Um eine Variable zu deklarieren, schreibt man in C:

```
<Datentyp> <Variablenname>;
```

Beispiel 3.1. Um eine Variable `a` vom Typ `int` zu deklarieren, verwendet man in C den Aufruf:

```
int a;
```

Durch eine reine Deklaration wird der Variable allerdings noch kein Wert zugewiesen. Würde man die Variable nun so aufrufen, würde sie den Wert zurückgeben, der noch im Speicher an der Stelle vorhanden ist. Daher sollte eine Variable immer bei der Deklaration einen Wert zugewiesen bekommen:

```
int a = 0;
```

Diese Zuweisung nennt man **Initialisierung**.

3.3 Operatoren

Operatoren bieten unter anderem die Möglichkeit, Variablen miteinander zu verknüpfen. Die grundlegendsten Operatoren, die man bereits aus der Mathematik kennt, sind:

Operator	Bedeutung
+	Addition
-	Subtraktion
*	Multiplikation
/	Division
%	Modulo (Rest einer Division)

Table 2: Grundlegende Operatoren in C

Beispiel 3.2. Gegeben sind zwei Variablen `a` und `b` und man möchte den Rest der ganzzahligen Division von `a` durch `b` berechnen. Der Code dazu könnte wie folgt aussehen:

```
int a = 10;
int b = 3;
int rest = a % b; // = 1, da 10/3 = 3 Rest 1
```

4 Funktionen