8
Strukturen (struct) und Aufzählungen (enum) in C

Jörn Loviscach

Versionsstand: 25. September 2013, 18:09

Die nummerierten Felder sind absichtlich leer, zum Ausfüllen beim Ansehen der Videos:
http://www.j3L7h.de/videos.html

Bitte hier notieren, was beim Bearbeiten unklar geblieben ist

1 Strukturen

Ein Array speichert eine Sammlung von Daten von *einem* Typ. Eine Struktur `struct` kann dagegen Daten *verschiedenen* Typs oder verschiedener Rollen zusammenfassen, zum Beispiel bei Datensätzen der folgenden Arten:

[1]

In C und C++ kann man einen solchen `struct`-Typ nach diesem Muster deklarieren:

```
2
```

Hiermit ist noch keine neue Variable eingeführt, nur ein neuer Datentyp. Das Semikolon am Ende ist in neueren Sprachen verschwunden. (In C und C++ kann man vor dem Semikolon sofort Variablen dieses neuen Typs deklarieren.) Der Einfachheit halber sollte man in C (nicht in C++) noch dazu schreiben:

```
3
```

Dann wird `Sample` ein vollwertiger Typname, wie es in C++ auch ohne diese Zeile passiert.

Eine Variable dieses Typs legt man so an:

```
4
```

Dann steht natürlich wie üblich Unsinn in den Bits (Demo im Debugger). Man sollte deshalb zum Initialisieren Werte vorgeben, zum Beispiel so:

```
5
```

Wenn man hinten Werte weglässt, wird wie bei einem Array mit Nullen aufgefüllt. Also ist dies die einfachste Art, alle Einträge auf null zu setzen:

```
6
```

In C99 sind diverse andere Arten der Initialisierung von Strukturen hinzugekommen, aber die finden sich noch nicht in C++ wieder.

Auf die einzelnen Elemente einer Struktur greift man nach diesem Muster zu:

7

Wie einfache Variablen und anders als Arrays kann man Strukturen mit = zuweisen (also kopieren), sie an Funktionen übergeben und sogar als Rückgabewerte von Funktionen benutzen. Allerdings werden dabei arg viele Daten kopiert. Beispiel:

8

Es gibt allerdings anders als bei einfachen Variablen keinen Vergleich von Strukturen mit ==. Das muss man zu Fuß machen:

9

Die spannendste Anwendung von Strukturen ist, aus ihnen Arrays zu bauen:

10

2 Aufzählungen

Oft möchte man für eine Variable nur eine Handvoll verschiedener Möglichkeiten haben, insbesondere für Zustände, zum Beispiel:

11

Das könnte man im Prinzip mit Konstanten und ganzen Zahlen machen:

12

Im Prinzip könnte man das sogar mit Zeichenketten machen – aber aus welchen Gründen lieber nicht?

Die elegantere Lösung statt der zu Fuß definierten Konstanten ist ein Aufzählungstyp (Enumeration):

13

Das macht `MachineState` zu einem Ganzzahltyp mit benannten Werten, beginnend bei null.

Wieder der Einfachheit halber sollte man in C (nicht in C++) noch dazu schreiben:

14

Dann wird `MachineState` ein vollwertiger Typname, wie es in C++ auch ohne diese Zeile passiert.

Danach kann man schreiben:

15

Ideal für die Arbeit mit einer Enumeration ist die Verzweigung mit `switch`:

[16]

Denselben Effekt könnte man mit einer unübersichtlich tiefen Schachtelung von `if` erreichen:

[17]