# Operatoren. Programmiersprachen. Programmablaufplan und Struktogramm.

Jörn Loviscach

Versionsstand: 29. September 2012, 20:51

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



## 1 Zahlensysteme in C

C selbst versteht ab Werk Dezimalzahlen wie `123`, Hexadezimalzahlen wie `0xE8` und Oktalzahlen wie `057`. Die Oktalzahlen kommen eher selten vor, sind aber eine fiese Tücke: Man darf in C-Code vor Dezimalzahlen keine führende Null schreiben, sonst werden sie als Oktalzahlen verstanden!

Alle Ganzzahlen werden im Rechner im Zweiersystem dargestellt. Mit den meisten Debuggern kann man sich die Inhalte von Variablen dezimal, hexadezimal und binär anzeigen lassen (Demo). Auch den Taschenrechner von Microsoft Windows kann man umschalten – und für bitweise Operationen benutzen (Demo).

## 2 Operatoren für Ganzzahlen in C

Die Sprache C macht keinen nennenswerten Unterschied zwischen Rechenoperationen (Addition, Multiplikation usw.), logischen Operationen (Oder, Und usw.) und Vergleichen (gleich, größer usw.). Hier sind einige wichtige davon:

1

Die logischen Operatoren `&&` und `||` sind nicht nur schlicht „Und“ und „Oder“, sondern haben eine besondere Raffinesse namens „Short Circuit“ (Kurzschluss): Wenn bei `a && b` das `a` bereits falsch ist, liefert der Ausdruck sofort falsch, ohne dass `b` angesehen wird. Analog bei `a || b`. (Und zwar wie?) Dieses Verhalten ist zum jetzigen Zeitpunkt aber noch nicht spannend.

Diese Liste der Operatoren ist in der Reihenfolge der Präzedenz geordnet: Erst werden die Operationen oben in der Liste ausgeführt, dann die unten. Das ist „Punktrechnung vor Strichrechnung“, aber bis ins Kleinste für alle Operationen definiert. Mehrere Operationen auf gleicher Stufe der Liste werden mit wenigen Ausnahmen von links nach rechts ausgewertet. Das ist vor allem bei Ausdrücken wie `a*b/c*d` spannend.

Der Ausdruck

[2]

ist also zu verstehen als:

[3]

Schreiben Sie am Anfang lieber mehr Klammern als nötig. Das ist wie Fahrradfahren mit Stützrädern: Es sieht nicht professionell aus, aber man fällt seltener auf die Nase.

Diese Operationen können in C kreuz und quer gemixt werden. Neuere Sprachen wie Java und C# sind strenger bei der Unterscheidung von Zahl (z.B. `int`) und Wahrheitswert (`bool`).

# 3 Inkrement, Dekrement, Verbindung mit Zuweisung

Oft will man ein Ergebnis derselben Variable zuweisen, aus der man gerade einen Wert geholt hat:

[4]

Dafür gibt es in C Schreibweisen, die kürzer und klarer sind:

[5]

Das Erhöhen (meist um den Wert 1) heißt Inkrement, das Verringern (meist um den Wert 1) Dekrement. Dazu gibt es auch als Tätigkeit: „`i` wird dekrementiert.“

In professionellen Programmen sehen Sie `j++` und eher noch `++j` als Teil von komplexeren Ausdrücken wie `i = 2*++j`. Tun Sie das nicht. Sie müssen nicht mit so etwas zeigen, für wie clever Sie sich halten. Das lohnt sich nicht – schon allein wegen der erschwerten Fehlersuche.

# 4 Zoo der Programmiersprachen

Um einen Computer zu programmieren – ihm mitzuteilen, was er tun soll –, muss man sich verständlich machen. Das geht bisher nicht durch gutes Zureden, sondern nur mit Hilfe streng formaler Programmiersprachen.

Im Seminar haben wir bereits die ersten Schritte auf dem MSP430... in der Programmiersprache C gemacht. Das ist die klassische Programmiersprache für Mikrocontroller. Hier geht es um deren Version C99. Auch Programmiersprachen werden heute mehr oder minder häufig upgedatet.

Aus C haben sich unzählige andere Programmiersprachen entwickelt, so dass man die Kenntnisse in C an vielen Stellen nutzen kann:[6]

Daneben gibt es andere Familien von Programmiersprachen, zum Beispiel Pascal, aus dem Modula und Delphi hervorgegangen sind, oder BASIC (Beginner's All-purpose Symbolic Instruction Code), der Stammvater von Visual Basic, Visual Basic .NET und VBScript.

Unter Physikern ist Fortran (FORmula TRANslation) berüchtigt für jahrzehntelang gepflegte Numerikbibliotheken. In der Finanzwelt stößt man noch auf antikes COBOL (COmmon Business-Oriented Language).

Je nach Anwendung ist eine andere Programmiersprache sinnvoll oder sogar nötig. Man lernt nicht eine einzige Programmiersprache und ist dann fertig! Der Job von Informatikerinnen und Informatikern ist im Zweifelsfall sogar, eine neue, für das aktuelle Problem passende Programmiersprache zu entwickeln.

Ein paar Live-Beispiele, wo welche Programmiersprachen eingesetzt werden:

- Webseiten mit Formularen oder komplexen Bedienelementen benutzen typischerweise JavaScript.

- Auf dem Web-Server laufen dabei bei kleinen Angeboten gerne Skripte in der Sprache PHP (PHP: Hypertext Processor).
- Die üblichen Textverarbeitungen und Tabellenkalkulationen lassen sich mit Makros in BASIC automatisieren.
- Die Muttersprache praktisch jeden Prozessors ist nicht C oder PASCAL oder ..., sondern eine je nach Prozessortyp verschiedene, binäre Maschinensprache. Sie wird meist in einer halbwegs lesbaren Assemblersprache dargestellt.
- Die Sprache von MATLAB® ist auf den Umgang mit Vektoren und Matrizen optimiert.
- Es gibt auch grafische Programmiersprachen, zum Beispiel MATLAB® Simulink. Was man dort mit Kästchen und Pfeilen zusammenklickt, ist ein Programm.
- In wissenschaftlichen Veröffentlichungen verwendet man meist keine echte Programmiersprache, sondern „Pseudocode“, ein nach persönlichem Geschmack vereinfachtes Pascal oder C.

# 5 Arten von Programmiersprachen

Die meisten heute aktuellen Programmiersprachen sind, was man als *imperativ* („befehlend“) bezeichnet, manchmal auch als *prozedural*, was aber ein mehrdeutiger Begriff ist. Mit imperativen Sprachen schreibt man sozusagen Rezepte auf: Tue dieses, prüfe jenes usw. Dies gilt auch für die *objektorientierten* Programmiersprachen im nächsten Semester. Andere Arten an Programmiersprachen basieren auf dem mathematischen Funktionsbegriff (*funktionale* Sprachen) oder beschreiben logische Bedingungen für eine gesuchte Lösung.

Eine weitere Unterscheidung kann man zwischen Skriptsprachen und „vollwertigen“ Programmiersprachen machen. Skriptsprachen sind dazu gedacht, auf die Schnelle ohne viel Brimborium kleine Programme zu schreiben, zum Beispiel Makros für die Textverarbeitung. Vertreter dieser Gattung sind:

[7]

Die meisten (aber nicht alle) Sprachen der C-Familie gelten dagegen als „vollwertige“ Programmiersprachen:

[8]

Diese Unterscheidung ist nicht ohne Diskussion: So ist Facebook zu großen Teilen in der Skriptsprache PHP geschrieben.

Skriptsprachen sind gerne „dynamisch typisiert“: Wo in C durch `int a;` festgelegt ist, dass die Variable `a` nur ganze Zahlen speichert, kann man etwa in MATLAB® schreiben:

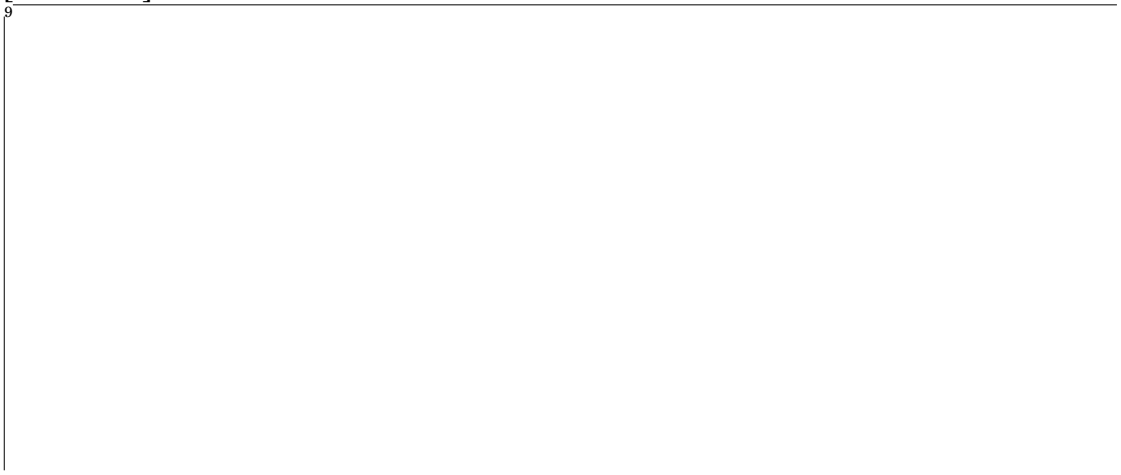
```
a = 42;
a = [1 2 3];
a = 'xyz';
```

Man könnte auch versuchen, Skriptsprachen und „vollwertige“ Programmiersprachen zu unterscheiden, indem man interpretierte von kompilierten Sprachen trennt. Diese Unterscheidung klappt heute aber nicht mehr gut. Demnächst mehr dazu.

# 6 Flussdiagramm

Ein imperatives Programm kann man als Programmablaufplan = Flussdiagramm [flowchart] darstellen:

[9]

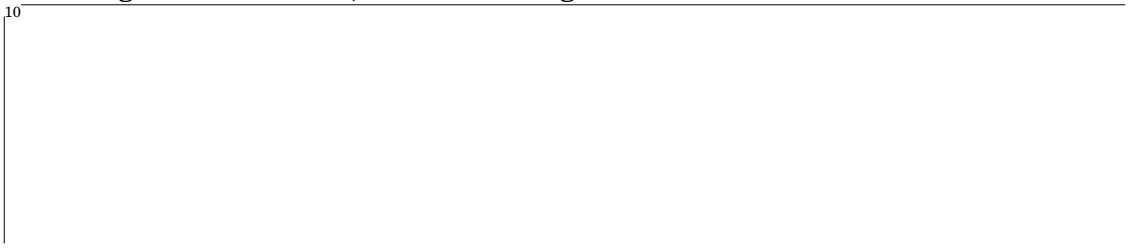


Stadionförmige Rechtecke stehen für Start und Ende, Rechtecke für Operationen, Parallelogramme für Ein- und Ausgabe, Rauten für Verzweigungen.

# 7 Strukturierte Programmierung, Struktogramm

Imperative Programmierung kann im Prinzip zu einem Flussdiagramm führen, das wie Spaghetti aussieht – sogenannter Spaghetticode, bei dem man nur schwierig erkennen kann, was wann ausgeführt wird:

[10]



In BASIC und anderen Sprachen wurde/wird dies durch den Befehl `GOTO` gefördert.

Seit Jahrzehnten arbeitet man zu Vermeidung von Spaghetti fast ausschließlich mit sauberen Blöcken wie `if` und `while`. Das heißt *strukturierte* Programmierung.

Mit einem Struktogramm = Nassi-Shneiderman-Diagramm lässt sich das aufzeichnen: