Operatoren

>> Operatoren sind **Symbole**, die den Compiler anweisen, bestimmte mathematische oder logische Manipulationen durchzuführen. Operatoren arbeiten mit **Operanden**, die Variablen, Konstanten oder Ausdrücke sein können. Die Kombination von Operatoren und Operanden bildet **Ausdrücke**.

>> In C gibt es zahlreiche Operatoren. Wir betrachten hier nur die wichtigsten. Andere stellen wir später an den passenden Stellen vor.

Wir beginnen mit arithmetischen Operatoren. Dann schauen wir uns Vergleichsoperatoren und logische Operatoren an. Unterwegs behandeln wir einige Zuweisungsoperatoren.

Arithmetische Operatoren und Zuweisung

Den Zuweisungsoperator (=) haben wir bereits kennengelernt. Damit können wir einen Wert in eine Variable einfügen.

Arithmetische Operatoren erlauben grundlegende mathematische Operationen: *Addition* (+), *Subtraktion* (-), *Multiplikation* (*) und *Division* (/).

Beispiele:

```
int y = 10;
int x = y + 1;
x = x * 5;
x = x + 3 * y;
```

Bei einer Zuweisung wird zuerst der Wert rechts vom = berechnet und dann der Variablen links davon zugewiesen.

Modulo-Operator

Der Modulo-Operator (`%`) gibt den Rest an, wenn zwei Zahlen geteilt werden.

Beispiel:

```
int m = 13 % 4; // m ist jetzt 1
```

Es gibt einige praktische Anwendungen beim Programmieren. Beispiele:

1. Schattierung jeder zweiten Zeile in einer Tabelle, d.h. alle Zeilen für die gilt, dass `rowNumber % 2` gleich 0 ist.

2. Generierung von Zufallszahlen innerhalb eines gewünschten Bereichs.

Kurzformen

Für einige Operatoren gibt es eine Kurzform. Beispiel:

`x *= 5;` ist das gleiche wie `x = x * 5;`

Solche Operatoren gibt es für alle grundlegenden arithmetischen Operatoren: +, −, *, /, %

Es gibt auch eine Kurzform für das Erhöhen oder Verringern um 1:

`x++` inkrementiert x

`x--` dekrementiert x

Der Code sieht dadurch eleganter aus. Die Verwendung ist optional.

Boolesche Ausdrücke

… sind entweder *wahr* oder *falsch*. Grundlage für Entscheidungen in Programmen; steuern Ablauf der Ausführung („control flow“):

- Welche Verzweigung soll in einer Bedingung genommen werden?
- Soll eine Schleife weiterlaufen?

Nicht notwendig, *bool* zu verwenden, ein *int* geht bspw. auch.

In C wird jeder Wert, der nicht 0 ist, als *true* angesehen – also nicht nur die 1.

Nur die 0 ist *false*.

Logische Operatoren

Logisches **UND (&&)**: wahr, wenn beide Operanden wahr sind, sonst falsch;
Ergebnis ist 0 (falsch) oder 1 (wahr)

Logisches **ODER (||)**: wahr, wenn mind. einer der Operanden wahr ist, sonst falsch;
Ergebnis ist 0 oder 1

Logisches **NICHT (!)**:
invertiert den Wert eines Operanden;
Ergebnis ist 0 oder 1

Beispiel:

```
int x = 1 && 0;
int y = !x;
int z = x || y; // z ist nun 1
```

Vergleichs-operatoren

Die Vergleichsoperatoren von C verhalten sich ähnlich wie elementare arithmetische Vergleiche.

Sie geben 0 (für falsch) oder 1 (für wahr) zurück. Folgende Vergleiche existieren:

`x < y` kleiner als

`x <= y` kleiner als oder gleich

`x > y` größer als

`x >= y` größer als oder gleich

`x == y` Gleichheit

`x != y` Ungleichheit

Überblick

Operator: Symbol, das den Rechner anweist, mathematische oder logische Manipulationen durchzuführen.

Operand: Wert, auf den ein Operator wirkt, z.B. Variable, Literal oder Ausdruck.

Ausdruck (expression): Kombination aus Operatoren und Operanden (oder einzelnes Literal oder Variable); Ausdrücke werden zu einem einzigen Wert ausgewertet.

Zuweisung (assignment): Spezieller Ausdruck mit Seiteneffekt: weist einer Variablen einen Wert zu.

Boolescher Ausdruck: Ausdruck, der als wahr oder falsch ausgewertet wird.

EXTRAS IN 3 MINUTEN

FRAGEN – ANTWORTEN – RÄTSEL
UND KURZE ZUSAMMENFASSUNG

F1: `int x = 10; x *= 3; x %= 7;`
Welchen Wert hat x jetzt?

F2: Warum bekomme ich bei `5 / 2`
als Ergebnis die Zahl 0 statt 2.5?

F3: Wie prüft man mit Operatoren am
einfachsten, ob eine Zahl gerade ist?

F4: Wird der Ausdruck als wahr (1) ausgewertet?
`(5 + 3>7) && (4 * 2 == 8)`

Operator-Rangfolge (operator precedence)
beachten!

test.c

Users > dh > Desktop > C test.c > main()

```c
#include <stdio.h>

int main() {
    int a = 5, b = 10, c;

    c = (a = b);
    printf("%d\n", (c = a = b = 20) == 20);
    return 0;
}
```

Was passiert in diesem Programm?

```c
#include <stdio.h>


int main() {
    int a = 5, b = 5;

    // Using postfix increment
    int r1 = (a++ == 5);
    printf("a = %d, r1 = %d\n", a, r1);

    // Using prefix increment
    int r2 = (++b == 5);
    printf("b = %d, r2 = %d\n", b, r2);

    return 0;
}
```

Rätsel-frage

```
Ist a 0? => 0
Ist b 0? => 0
Dann muss a AND b 1 sein.
a AND b: 0
Nanu?
```

```c
#include <stdio.h>

int main() {
    int a = 10, b = 5;

    printf("Ist a 0? => %d\n", a == 0);
    printf("Ist b 0? => %d\n", b == 0);
    printf("Dann muss a AND b 1 sein.\n");
    printf("a AND b: %d\n", a & b);
    printf("Nanu?\n");

    return 0;
}
```

Operanden sind Variablen, Konstanten oder Ausdrücke.

Ausdrücke haben einen Wert.

Arithmethische Operatoren

+, -, *, /

% (Modulo)

Zuweisung (=):
Ausdruck mit Seiteneffekt

Kurzformen (+= usw.)

Inkrement (++)

Dekrement (– –)

Vergleichsoperatoren
<, <=, >, >= sowie ==, !=

Ergebnis ist 0 oder 1

Grundlage für Entscheidungen im Programmablauf

Boolesche Ausdrücke: Jeder Wert ≠ 0 ist wahr, nur 0 falsch

Logische Operatoren &&, || und !

Steuern den Programmfluss (z.B. in Bedingungen, Schleifen)

Auswertungsreihenfolge (rechts nach links) und Operatorrangfolge.

Achtung, Unterschied:
&&, || und &, | sowie = und ==