Einseitige Auswahl (Entscheidung)

Die einseitige Auswahl kennen Sie schon von Karol: `wenn ... dann ... *wenn`

Syntax in C++:

```
if  (<logischer Ausdruck>)
{
          /* Anweisung(en) */
}
```

> In Syntax-Angaben kennzeichnen spitze Klammen wie bei `<logischer Ausdruck>` einen Klartext, der ersetzt werden muss. Die spitzen Klammern selbst gehören nicht zur Syntax.

Die geschweiften Klammen kann man weglassen, wenn sie nur eine Anweisung einschließen.

Zweiseitige Auswahl (Entscheidung, Verzweigung)

Auch das ist für uns nichts neues:

```
if (<logischer Ausdruck>)
{
          /* Anweisung(en) */
}
else
{
          /* andere Answeisung(en) */
}
```

Vergleichsoperatoren

Für einen logischen Ausdruck benötigt man in der Regel einen der folgenden Vergleichsoperatoren:

Vergleichsoperator	Bedeutung	Beispiel für einen logischen Ausdruck
== !=	ist gleich ist ungleich	(x == 3) (x != 3)
< <=	ist kleiner als ist kleiner oder gleich	(x < 3) (x <= 3)
> >=	ist größer als ist größer oder gleich	(x > 3) (x >= 3)

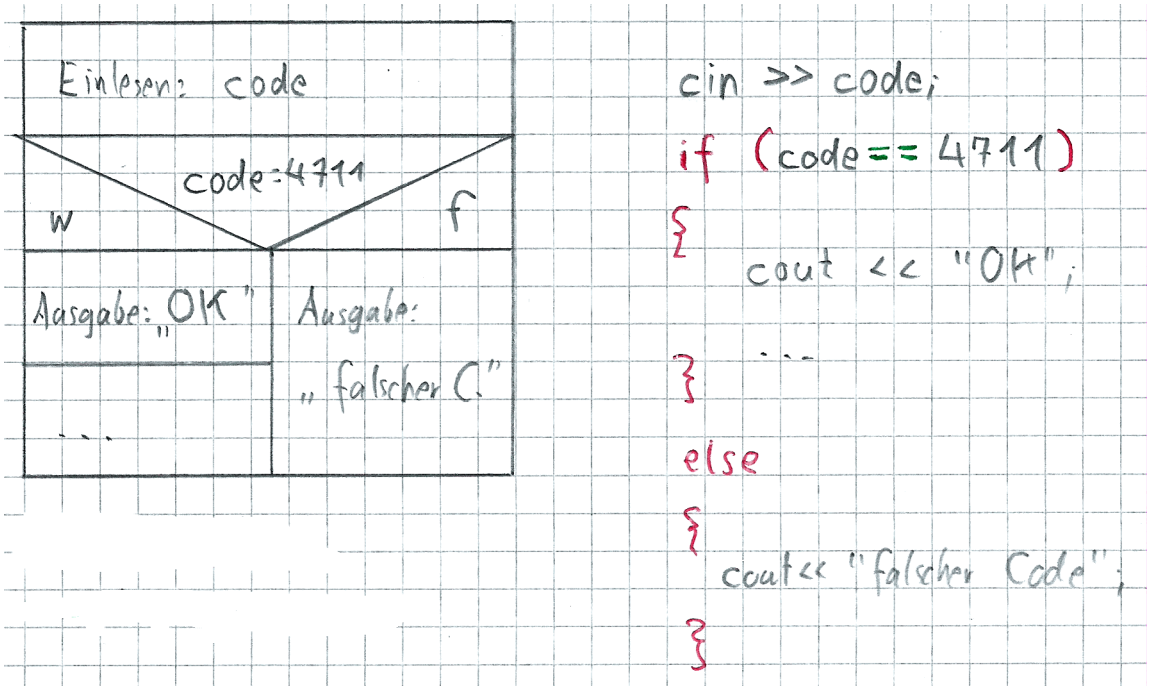
Das Ergebnis eines logischen Ausdrucks ist entweder `true` oder `false`. Man nennt es deshalb auch Wahrheitswert.

Wahrheitswerte in altem Quellcode

Früher gab es keinen Datentyp für Wahrheitswerte. Es wurde der Typ `int` benutzt, um sie nachzubilden: 0 entspricht `false` und alle anderen Werte entsprechen `true`.

Das funktioniert heute noch, aber Sie sollten es möglichst nicht mehr verwenden.

Beispiel 1: Einfaches Codeschloss



Beispiel 2: Vorzeichen einer Zahl

Der Benutzer gibt eine Zahl ein. Es wird geprüft, ob die Zahl größer oder gleich 0 ist. Wenn ja, wird der Text „Die Zahl ist nicht negativ" ausgegeben, sonst „Die Zahl ist negativ".

```
int main()
{
    int zahl;

    cout << "Bitte eine ganze Zahl eingeben: ";
    cin  >> zahl;

    if ( zahl>=0 )
       cout << "Die Zahl ist nicht negativ." << endl;
    else
       cout << "Die Zahl ist negativ." << endl;

return 0;
}
```

Verzweigungen können auch verschachtelt werden. Das Beispiel wird so geändert, dass die Eingabe einer 0 extra behandelt wird:

```
if ( zahl > 0 )
    cout << "Die Zahl ist positiv." << endl;
else
{
   if (zahl < 0)
       cout << "Die Zahl ist negativ." << endl;
   else
       cout << "Die Zahl hat den Wert 0." << endl;
}
```