

Um die Ein- und Ausgabe nach eigenen Wünschen gestalten zu können, haben Sie in C++ die Möglichkeit, den Datenstrom durch Manipulatoren zu formatieren.

In der Vorlesung wurde die Ein- und Ausgabe verschiedener Daten mit Strömen vorgestellt. Dabei waren Sie auf die Standardformatierungen der iostream Library angewiesen. Problematisch kann dies dann werden, wenn Sie bestimmte Wünsche hinsichtlich der Gestaltung der Ausgabe haben. Sie möchten zum Beispiel eine Tabelle mit dem grossen Einmaleins ausgeben. Dann ist es wünschenswert, dass die Zahlen in den Spalten rechtsbündig untereinander stehen und sich Leerzeichen zwischen den Zahlen befinden. Diese Ausgabeformatierungen können Sie mittels besonderer **Strom-Manipulatoren** erreichen. Die Manipulatoren werden wie normale Ausgabefelder in den Ein- oder Ausgabestrom eingefügt. Das folgende Beispiel wird Ihnen Klarheit verschaffen.

```
#include <iostream>
#include <iomanip>

using namespace std;

for (int i = 1; i < 20; ++i) {
    for (int j = 1; j < 20; ++j) {
        cout << right << setw(4) << i*j;
    }
    cout << endl;
}
```

Hierbei sorgt der Manipulator **right** für eine rechtsbündige Ausrichtung der Ausgabe, die wegen **setw(4)** immer vier Zeichen breit ist. Schliesslich bewirkt **endl** die Ausgabe einer Zeilenumschaltung.

setbase(int B)	Basis 8, 10 oder 16 definieren
setfill(char c)	Füllzeichen festlegen
setprecision(int n)	Flisskommapprezession
setw(int w)	Weite setzen

Tabelle 1: Manipulatoren mit Parametern

Ein anderes Problem bei der Ausgabe numerischer Werte ist die Darstellung mit einer bestimmten Genauigkeit. Das nächste Beispiel zeigt, wie mittels Verwendung des Manipulators **setprecision** die Kreiszahl pi mit unterschiedlicher Genauigkeit ausgegeben werden kann. Dabei wird die Ausgabe korrekt gerundet:

```
#include <iostream>
#include <iomanip>
#include <cmath>

using namespace std;

double pi = 4.0 * atan(1.0);
for (int p = 12; p > 1; p -=3) {
    cout << setprecision(p) << fixed << pi << endl;
}

// Ausgabe:   3.141592653590
//           3.141592654
//           3.141593
//           3.142
```

boolalpha	alphanumerische Ausgabe von bool-Werten (true/false statt 1/0)
noboolalpha	Gegenteil von boolalpha
showbase	Ausgabe des Präfixes bei ganzzahligen Werten (0 bei oktaler Ausgabe, 0x bei hexadezimaler, nichts bei dezimaler Ausgabe)
noshowbase	Gegenteil von showbase
showpoint	zwingende Ausgabe nachfolgender Nullen bei Fließkommawerten (nur in Zusammenhang mit fixed)
noshowpoint	Gegenteil von showpoint
showpos	zwingende Ausgabe von + bei positiven Werten
noshowpos	Gegenteil von showpos
skipws	überlese Whitespaces beim Input
noskipws	Gegenteil von skipws
unitbuf	Puffer nach jeder Ausgabe leeren
nounitbuf	Ausgabe puffern
uppercase	Ausgabe von Grossbuchstaben bei Formatzeichen wie z.B. e und x
nouppercase	Gegenteil von uppercase
left	rechtsbündige Ausgabe
right	linksbündige Ausgabe
internal	Auffüllen der Ausgabe zwischen Vorzeichen und Wert
dec	dezimale Ausgabe von ganzzahligen Werten
oct	oktale Ausgabe von ganzzahligen Werten
hex	hexadezimale Ausgabe von ganzzahligen Werten
fixed	Ausgabe von Fließkommawerten in Festpunktschreibweise <i>ddd.dd</i> (vgl. "%f")
scientific	Ausgabe von Fließkommawerten in Exponentialschreibweise <i>d.ddddd±dd</i> (vgl. "%e")

Tabelle 2: Manipulatoren ohne Parameter,

Mit `cout` wird eine gepufferte Ausgabe durchgeführt, d.h. die auszugebenden Zeichen werden nicht sofort, sondern erst etwas später ausgegeben. Gerade bei der Fehlersuche, aber auch bei Benutzerhinweisen und -aufforderungen, führt dies zu ungewollten Effekten. Mit `flush` kann man das Programm zwingen alles im Ausgabepuffer stehende komplett auszugeben. Dies kann ebenfalls mit der Ausgabe von `endl` erreicht werden. Es bewirkt neben dem Zeilenvorschub auch ein `flush`. Z.B.

```
cout << 1234.56789 << "\n" << flush;
```

```
cout << 1234.56789 << endl;
```

Dieses Verhalten einer nicht gepufferten Ausgabe kann auch mit dem Flag oder Manipulator `unitbuf` erreicht werden.

<code>flush</code>	Leeren des Ausgabestoms
<code>ws</code>	Überlesen von Whitespaces im Eingabestrom
<code>endl</code>	Ausgabe von '\n' und anschliessendes flush
<code>ends</code>	Ausgabe von '\0' und anschliessendes flush

Tabelle 3: Spezielle Manipulatoren