4.6 Von C nach C++/Ausgabe mit cout

4.6.1 cout-Ausgaben verändern

Der Ausgabe-Datenstrom `cout` kann durch so genannte *Manipulatoren* beeinflusst werden. Einer dieser Manipulatoren ist `endl`, der einen sofortigen Zeilenumbruch ausgibt.

4.6.2 Sofort-Ausgabe: `flush`

Ein weiterer Manipulator ist `flush`. Er sorgt sofort dafür, dass alles, was bisher an `cout` geleitet wurde, ausgegeben wird. Insofern wirkt `flush` wie `fflush(stdout)` in C:

cout01.cpp

```
1  #include <iostream>
2  #include <unistd.h>
3  using namespace std;
4  int main(void)
5  {
6     for(int zahl=0; zahl<10; ++zahl)
7     {
8        cout << zahl << flush << ",␣";
9        sleep(1);
10    }
11    cout << endl;
12    return 0;
13 }
```

4.6.3 Bool: `boolalpha` und `noboolalpha`

In C++ gibt es den Datentyp `bool`, der die Wahrheitswerte `true` (gleich 1) und `false` (gleich 0) aufnehmen kann. Standardmäßig gibt `cout` 0 oder 1 aus; mit dem Manipulator `boolalpha` kann das geändert werden, so dass `true` oder `false` ausgegeben wird (rückgängig macht man das mit `noboolalpha`):

cout02.cpp

```
1  #include <iostream>
2  #include <unistd.h>
3  using namespace std;
4  int main(void)
5  {
6     bool x = true;
7     cout << x << ",␣"
8          << boolalpha << x << ",␣"
9          << noboolalpha << x
10         << endl;
11    return 0;
12 }
```

Terminal

```
schueler@debian964:~$ g++ cout02.cpp; a.out
1, true, 1
```

4.6.4 Vorzeichen: showpos und noshowpos

Nun zur Ausgabe von Zahlen. Mit dem Manipulator `showpos` kann man dafür sorgen, dass auch negative Zahlen und null mit Vorzeichen ausgegeben werden (rückgängig macht man das mit `noshowpos`).

cout03.cpp

```
1  #include <iostream>
2  using namespace std;
3  int main(void)
4  {
5     cout << -1 << ", " << 0 << ", " << 1 << endl;
6     cout << showpos;
7     cout << -1 << ", " << 0 << ", " << 1 << endl;
8     cout << noshowpos;
9     cout << -1 << ", " << 0 << ", " << 1 << endl;
10    return 0;
11 }
```

Terminal

```
schueler@debian964:~$ g++ cout03.cpp; a.out
-1, 0, 1
-1, +0, +1
-1, 0, 1
```

4.6.5 Zahlensysteme: dec, hex, oct, showbase und noshowbase

Mit `dec`, `oct` und `hex` kann man die Zahlenbasis angeben. Mit `showbase` wird 0 oder 0x vorangestellt, `noshowbase` macht das wieder rückgängig:

cout04.cpp

```
1  #include <iostream>
2  using namespace std;
3  int main(void)
4  {
5     int zahl;
6     cout << "Zahl: ";
7     cin >> zahl;
8     cout << dec << zahl << endl;
9     cout << hex << zahl << endl;
10    cout << showbase << zahl << endl;
11    cout << oct << zahl << endl;
12    cout << noshowbase << zahl << endl;
13    cout << dec << zahl << endl;
14    return 0;
15 }
```

Terminal

```
schueler@debian964:~$ g++ cout04.cpp; a.out
Zahl: 12345
12345
3039
0x3039
030071
```

```
30071
12345
```

4.6.6 Gleitkommazahlen: `scientific` und `fixed`

Gleitkommazahlen können in Festkomma- oder in Exponential-Darstellung angezeigt werden. Dafür gibt es die Manipulatoren `scientific` (entspricht %e), `fixed` (entspricht %f), `hexfloat` (entspricht %a) und defaultfloat (entspricht %g, Voreinstellung).

cout05.cpp

```
1  #include <iostream>
2  #include <iomanip>
3  using namespace std;
4  int main(void)
5  {
6     cout << "ohne........: " << 123.456 << endl;
7     cout << "fixed.......: " << fixed << 123.456 << endl;
8     cout << "scientific..: " << scientific << 123.456 << endl;
9     cout << "hexfloat....: " << hexfloat << 123.456 << endl;
10    cout << "defaultfloat: " << defaultfloat << 123.456 << endl;
11    return 0;
12 }
```

Terminal

```
schueler@debian964:~$ g++ cout05.cpp; a.out
ohne........: 123.456
fixed.......: 123.456000
scientific..: 1.234560e+02
hexfloat....: 0x1.edd2f1a9fbe77p+6
defaultfloat: 123.456
```

4.6.7 Breite der Ausgabe: `setw(n)`

Mit dem Manipulator `setw(n)` kann man dafür sorgen, dass das folgende Objekt mindestens mit einer Breite von n Zeichen ausgegeben wird:

cout11.cpp

```
1  #include <iostream>
2  #include <iomanip>
3  using namespace std;
4  int main(void)
5  {
6     int zahl;
7     cout << "Zahl: ";
8     cin >> zahl;
9     cout << setw(4) << zahl << endl;
10    return 0;
11 }
```

Terminal

```
schueler@debian964:~$ g++ cout11.cpp; a.out
Zahl: 20
  20
```

Falls die Ausgabe mehr Platz braucht, wird sie nicht gekappt. Als einzige der Manipulatoren wirkt `setw(n)` *nur* auf das *direkt folgende Objekt*, selbst dann , wenn es sich um Text handelt:

cout12.cpp

```
#include <iostream>
#include <iomanip>
using namespace std;
int main(void)
{
    int zahl;
    cout << "Zahl: ";
    cin >> zahl;
    cout << setw(4) << "A" << zahl << "B" << endl;
    return 0;
}
```

Terminal

```
schueler@debian964:~$ g++ cout12.cpp; a.out
Zahl: 20
  A20B
```

4.6.8 Füllzeichen festlegen: `setfill(c)`

Wenn man eine Uhrzeit anzeigen will, möchte man sie so anzeigen: `08:03:04`, nicht aber so: `8: 3: 4`. Dann muss die zweistellige Minute mit einer Null am Anfang aufgefüllt werden, nicht mit einem Leerzeichen. Dazu dient der Manipulator `setfill(c)`. Er bekommt als Parameter das Zeichen, mit dem aufgefüllt werden soll:

cout13.cpp

```
#include <iostream>
#include <iomanip>
using namespace std;
int main(void)
{
    int std=8, min=3, sek=4;
    cout << setfill('0');
    cout << "Es ist "
         << setw(2) << std << ":"
         << setw(2) << min << ":"
         << setw(2) << sek << " Uhr." << endl;
    return 0;
}
```

Terminal

```
schueler@debian964:~$ g++ cout13.cpp; a.out
Es ist 08:03:04 Uhr.
```

Wie man sieht, wirkt `setfill(c)`, anders als `setw(n)`, dauerhaft weiter.

4.6.9 Gleitkommazahlen: `setprecision(n)`

Bei Gleitkommazahlen kann man mit `setw(n)` nur die minimale Gesamtbreite festlegen:

cout16.cpp

```
#include <iostream>
#include <iomanip>
using namespace std;
int main(void)
{
   double messwert;
   cout << "Zahl: ";
   cin >> messwert;
   cout << setfill('0') << setw(12) << messwert << endl;
   return 0;
}
```

Terminal

```
schueler@debian964:~$ g++ cout16.cpp; a.out
Zahl: 1.23456789
000001.23457
```

Die Zahl der Nachkommastellen (und damit die absolute Genauigkeit) stellt man mit `setprecision(n)` ein, in diesem Beispiel mit dem Parameter 4:

cout17.cpp

```
#include <iostream>
#include <iomanip>
using namespace std;
int main(void)
{
   double messwert;
   cout << "Zahl: ";
   cin >> messwert;
   cout << setfill('0') << setw(12)
        << setprecision(4) << messwert << endl;
   return 0;
}
```

Terminal

```
schueler@debian964:~$ g++ cout17.cpp; a.out
Zahl: 1.23456789
00000001.235
```

Das gilt auch für die Exponentialdarstellung mit dem `scientific`-Manipulator:

cout18.cpp

```
#include <iostream>
#include <iomanip>
using namespace std;
int main(void)
{
   double messwert;
   cout << "Zahl: ";
   cin >> messwert;
   cout << scientific << setw(12)
        << setprecision(4) << messwert << endl;
```

```
11     return 0;
12 }
```

Terminal

```
schueler@debian964:~$ g++ cout18.cpp; a.out
Zahl: 1.23456789
  1.2346e+00
```

4.6.10 Ausrichtung: `left`, `right` und `internal`

Sobald man die minimale Ausgabebreite mit `setw(n)` eingestellt hat, erfolgt die Ausgabe rechtsbündig. Mit `left` kann man linksbündig ausrichten und mit `internal` wird das Vorzeichen nach links und der Betrag nach rechts ausgerichtet. Mit `right` kann man alles wieder rückgängig machen:

cout19.cpp

```
1  #include <iostream>
2  #include <iomanip>
3  using namespace std;
4  int main(void)
5  {
6      cout << setfill('.');
7      cout << setw(8) << -123 << endl;
8      cout << left  << setw(8) << -123 << endl;
9      cout << internal << setw(8) << -123 << endl;
10     cout << right << setw(8) << -123 << endl;
11     return 0;
12 }
```

Terminal

```
schueler@debian964:~$ g++ cout19.cpp; a.out
....-123
-123....
-....123
....-123
```