Textausgabe mit cout (C++-Standardausgabe)

Wenn Sie cout zum Ausgeben von Daten oder cin zum Einlesen verwenden möchten, müssen Sie die Headerdatei `<iostream>` und den Befehl zum Festlegen des Namensraums `using namespace std;` einbinden.

Beispiele:

```
#include <iostream>
using namespace std;

int main(int argc, char **argv)
{
int i;
float f;

i=1;
f=2.0;

cout << "Hello World" <<endl;
cout << "i hat den Wert" << i << endl << "f hat den Wert" << f << endl;
}

// endl sorgt für einen Zeilenumbruch
```

Leider können wir noch nicht erklären, welche Bedeutung die Schreibweise bei cin und cout hat.[1]

Ausgabeformatierung

Sie können die Ausgabe von Zahlen mit cout durch Manipulatoren (Steueranweisungen) formatieren.

Achtung! Die Headerdatei iomanip muss eingebunden werden!

```
#include <iomanip>
```

1. **Allgemeine Manipulatoren**
   - **setfill(<Zeichen in Hochkommas>)** setzt das Füllzeichen (dauerhaft)
   - **setw(int n)** setzt die Feldbreite **für die nächste Operation** auf n Spalten
   - **left / right** linksbündige / rechtsbündige Ausgabe
   - **internal** bei Zahlen: Vorzeichen links-, Wert rechtsbündig
2. **Manipulatoren für Ganzzahlen**
   - **dec** dezimale Darstellung (Standard)
   - **hex** hexadezimale Darstellung
   - **oct** oktale Darstellung
   - **showpos** / **noshowpos** + bei positiven Zahlen ausgeben / unterdrücken
   - **uppercase** / **nouppercase** Groß- /Kleinbuchstaben (Standard) bei Hex-Ausgabe

Beispiele:

1 Für Experten: cin und cout sind Stream-Objekte. Bei << und >> handelt es sich um überladene Operatoren, also im Prinzip um Methoden.

Variablendefinition	Ausgabebefehl	Ausgabe:
`int n = 4;`	`cout << n;`	4
	`cout << showpos << n;`	+4
	`cout << setw(3) << n;`	4
	`cout << left << setw(3) << n;`	4
	`cout << setfill('0') << setw(3) << n;`	004
`int n = -4;`	`cout << n;`	-4
	`cout << setw(3) << n;`	-4
	`cout << left << setw(3) << n;`	-4
`int n = 27`	`cout << dec << n;`	27
	`cout << oct << n;`	33
	`cout << hex << n;`	1b
	`cout << hex << showbase << n;`	0x1b

3. Manipulatoren für float- und double-Zahlen

- **fixed** Darstellung als Festpunktzahl
- **scientific** Darstellung mit 10er Exponent
- **showpoint** Ausgabe aller Ziffern entsprechend der eingestellten Genauigkeit
- **noshowpoint** Unterdrückung abschließender Nullen und (wenn möglich) des Punktes
- **setprecision(<n>)** Genauigkeit auf n Stellen setzen

Beispiele:

Variablendefinition	Ausgabebefehl	Ausgabe:
`float f = 4;`	`cout << f;`	4
	`cout << showpos << f;`	+4
	`cout << showpoint << f;`	4.00000
	`cout << setprecision(8) << f;`	4.0000000
	`cout << setprecision(8) << showpos << g;`	+4.0000000
`float g=-8.67;`	`cout << g ;`	-8.67
	`cout << showpos<< g;`	-8.67
	`cout << showpoint<< g;`	-8.67000
	`cout << setprecision(8) << showpos << g;`	-8.6700001
`float g= 234.34;`	`cout << setprecision(3)<< scientific << g;`	+2.343e+002
	`cout << setprecision(3)<< fixed << g;`	+234.340

In der Voreinstellung erfolgt die Ausgabe als Dezimalbruch, gerundet auf 6 Stellen insgesamt mit Unterdrückung nachfolgender Nullen und ohne Punkt bei ganzzahligen Werten.

Reichen die 6 Stellen nicht aus, wird auf exponentielle Darstellung umgeschaltet. Sobald das Format auf **fixed** oder **scientific** umgestellt wurde, bezieht sich die Genauigkeit auf die Nachkommastellen.

Einlesen mit cin

Mit Hilfe von cin können Sie Eingaben von der Tastatur in Variablen einlesen.

Syntax:

```
cin >> variable1 >> variable2;
```

Stößt ein Programm auf eine `cin`-Zeile, wartet es auf eine Tastatureingabe. Passt die Eingabe nicht zu den nach den >>-Zeichen angegebenen Variablen kommen sinnlose Variablenwerte heraus.

Werden mehrere Werte eingelesen, müssen diese durch Leerzeichen getrennt werden. Benutzerfreundlicher ist es, für jede Variable, die eingelesen wird, eine eigene `cin`-Anweisung zu verwenden.

Beispielcode:

```
      char antwort;
      int k, ergebnis;
      cout << "Geben Sie bitte einen Buchstaben und eine Zahl durch
Leerzeichen            getrennt ein: ";
      cin >> antwort >> k;
      cout << "Es wurde antwort="<< antwort<< " und k=" << k <<
"eingegeben\n";
```

Wir benutzen `cin` zunächst zum Einlesen von Zahlen. Später werden Sie dafür eine bessere, aber kompliziertere Lösung kennenlernen.

Für `cin` gibt es auch einen eigenen Manipulator:

- `setbase(<n>)` setzt die Zahlenbasis auf n=8, 10 oder 16.