Ausgabe *(vorläufig)*

Syntax: `cout` << Arg_1 << Arg_2 << ... << Arg_n `;`

- Anschauliche Vorstellung: Argumente (Daten) fließen nacheinander in die Standardausgabe `cout`.
- Ausdrucksanweisung, bei der Operanden des Ausdrucks durch den Linksshiftoperator `<<` verknüpft sind.
- *Manipulatoren* als Operanden zulässig, beeinflussen Formatierung der Ausgabe (Änderung des Stromzustands)
- Wirkung bleibt in Kraft bis zu einer expliziten Änderung (Ausnahme: Feldbreite wirkt nur auf die nächste Ausgabe)
- Header `<iostream>` für vordefinierten Namen wie etwa `std::cout` und Manipulatoren ohne Argument, z.B. `std::fixed`.
- Header `<iomanip>` für Manipulatoren mit Argument, z.B `std::setw(n)` und `std::setprecision(n)`
- Durch Verwendung von `using namespace std;` Weglassen von `std::` möglich.

Ausgabe - Fortsetzung

- Wichtigste Formatierungen: Feldbreite, Genauigkeit bei Gleitpunktzahlen, Bündigkeit (Voreinst.: rechtsbündig)

```
double x=4.5;
cout << fixed << setw(8) << setprecision(4)
     << x << endl;
```

Ausgabe: ␣␣4.5000 (Zeilenvorschub)

- `endl` bewirkt Zeilenvorschub

```
int i=12,j=4;
cout << setw(4) << i << setw(4) << j << endl;
```

Ausgabe: ␣␣12␣␣␣4 (Zeilenvorschub)

- Ausgabe erfolgt mit minimal erforderlicher Breite, wenn nicht angegeben oder zu klein

```
int i=12,j=4;
cout << setw(1) << i << j << endl;
```

Ausgabe: 124 (Zeilenvorschub)

Ausgabe - Fortsetzung II

- Für das Gleitpunktformat gibt es 3 Darstellungen:

Exponentialformat	`scientific`
Festpunktformat	`fixed`
Allgemeinformat	C++11: `defaultfloat` (Voreinst.)

  Entsprechungen für printf (C/C++/Java): `%e`, `%f`, `%g`
- In C++98 Zurücksetzen auf Allgemeinformat mit `cout.unsetf(ios::floatfield)`
- Für das Ganzzahlformat gibt es 3 Darstellungen:

Dezimaldarstellung	`dec` (Voreinst.)
Oktaldarstellung	`oct`
Hexadezimaldarstellung	`hex`
- Weitere Flags ($\rightarrow$ Inf.bl.3, S.1)
- Statt Manipulatoren können auch die älteren Komponentenfunktionen verwendet werden ($\rightarrow$ Inf.bl.3, S.1)

Eingabe *(vorläufig)*

Syntax: `cin >>` Arg_1 `>>` Arg_2 `>> ... >>` Arg_n `;`

- Anschauliche Vorstellung: Aus der Standardeingabe `cin` fließen die Daten nacheinander in die angegebenen Variablen.
- Rechtsshiftoperator `>>` an Stelle des Linksshiftoperators
- Header analog zur Ausgabe
- Führender Zwischenraum (inkl. Zeilenenden) wird überlesen (Voreinstellung).
  Änderung durch Manipulator `noskipws`
- Gelesen wird bis zum ersten Zeichen, das *nicht* in der Darstellung vorkommen darf.
- Lesen unzulässiger Zeichens führt zu fehlerhaftem Stromzustand. *Ohne* Löschen der Fehlerbits und evtl. Überlesen von Folgezeichen im Eingabepuffer scheitern dann weitere Eingaben (später!)

Eingabe - Fortsetzung

- ```
  int i,j,k;
  cin >> i >> j >> k;
  ```
  Eingabe: `20␣100␣1000`
  $i = 20$, $j = 100$, $k = 1000$ (okay)

  Eingabe: `20␣100`
  `1000`
  $i = 20$, $j = 100$, $k = 1000$ (okay)

  Eingabe: `+20*100␣1000`
  $i = 20$, j undef., k undef. (nicht okay)
  Stromzustand fehlerhaft!

- Gleitpunktzahleingabe auch im Exponentialformat möglich:

  ```
  double x,y;
  cin >> x >> y;
  ```
  Eingabe: `3.14␣␣␣12.1e3`
  $x = 3.14$, $y = 12100$ (okay)
  ```

Eingabe - Fortsetzung II

- Überlesen von führendem Zwischenraum:

überlesen	`skipws` (Voreinst.)
nicht überlesen	`noskipws`

- Darstellung ganzer Zahlen:

Dezimaldarstellung	`dec` (Voreinst.)
Oktaldarstellung	`oct`
Hexadezimaldarstellung	`hex`

- Statt Manipulatoren können auch die älteren Komponentenfunktionen verwendet werden (→ Inf.bl.3, S.2)