Einfache Ein- und Ausgabe

- Standard-Ein- und Ausgabe
  - Eingabe
  - Eingabe von Zeichenketten
  - Ausgabe
- Ein- und Ausgabe mit Dateien

Einfache Ein- und Ausgabe

Standard-Ein- und Ausgabe

Ein Programm empfängt einen Strom von Eingabedaten, verarbeitet diese Daten und gibt einen Strom von Ausgabedaten aus. Ein „Strom“ (englisch *stream*) ist eine Folge von Bytes, die nacheinander vom Programm interpretiert bzw. erzeugt werden.

In C++ sind die folgenden Ein- und Ausgabeströme vordefiniert:

`cin`	Standard-Eingabe (Tastatur)
`cout`	Standard-Ausgabe (Bildschirm)
`cerr`	Standard-Fehlerausgabe (Bildschirm)

Eingabe

Der Operator >> sorgt bei der Eingabe dafür, dass automatisch die nötigen Umformungen vorgenommen werden.

```
int zahl; cin >> zahl;
```

bewirkt, dass eine Folge von Ziffern bis zu einem Nicht-Ziffernzeichen eingelesen und in die interne Darstellung einer `int`-Zahl umgewandelt wird.

Einfache Ein- und Ausgabe

Eingabe

Die Auswertung des Eingabestroms durch den >> -Operator hat die folgenden Eigenschaften:

- Führende Zwischenraumzeichen (englisch *whitespace*) werden ignoriert.

  Zwischenraumzeichen sind
  - Leerzeichen `' '`,
  - Tabulatorzeichen `'\t'`,
  - Zeilenrücklauf `'\r'`,
  - Zeilensprung `'\v'`,
  - Seitenvorschub `'\f'` und die
  - Zeilenendekennung `'\n'`.
- Zwischenraumzeichen werden als Endekennung genutzt.
- Andere Zeichen werden entsprechend dem verlangten Datentyp interpretiert.

Sollen Zwischenraumzeichen nicht ignoriert werden, so ist die Funktion `get()` zu verwenden, die zum Einlesen einzelner Zeichen benutzt werden kann:

```
// ein einzelnes Zeichen einlesen
char c;
cin.get(c);
```

Einfache Ein- und Ausgabe

Eingabe

Der nach außen hin nicht sichtbare Ablauf einer Tastaturabfrage mit „`cin >>`“ besteht aus den folgenden Schritten, wobei angenommen wird, dass noch nichts auf der Tastatur eingegeben worden ist:

1. Aufforderung an das Betriebssystem zur Zeichenübergabe.
2. Eingabe der Zeichen auf der Tastatur (mit Korrekturmöglichkeit durch die Backspace-Taste). Die Zeichen werden vom Betriebs-system der Reihe nach in einem besonderen Speicherbereich abgelegt, dem Tastaturpuffer.
3. Abschluss der Eingabe mit der RETURN-Taste. Damit wird das Zeichen `'\n'` als Zeilenende-kennung im Tastaturpuffer abgelegt, und der Puffer wird vom Betriebssystem an das C++-Programm übergeben.
4. Auswertung des Tastaturpuffer-Inhalts durch den Operator >> je nach Datentyp der gefragten Variablen.
5. Daten, die nach der Auswertung übrigbleiben, weil sie nicht zum Datentyp passen, verbleiben im Tastaturpuffer und können mit dem nächsten `cin >> ...` gelesen werden.

Einfache Ein- und Ausgabe

Eingabe

Das folgende Programm verlangt der Reihe nach eine `int`- und eine `double`-Zahl. Falls das Format mit dem aktuell erwarteten Datentyp nicht übereinstimmt, wird die Schleife abgebrochen, z.B. bei Eingabe eines Buchstabens.

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

int main()
{
   int i;
   double d;

   while(cin >> i >> d)
       cout << i << endl << d << endl;

   return 0;
}
```

Eingabe: `100 12.4`
Ausgabe:

Eingabe: `2.7`
Ausgabe:

Einfache Ein- und Ausgabe

Eingabe von Zeichenketten

Die Eingabe von Zeichenketten unterscheidet sich nicht von der Eingabe von Zahlen. Häufig möchte man jedoch nicht nur durch Zwischenraumzeichen getrennte Zeichenketten einlesen, sondern ganze Zeilen.

Beispiel: Einlesen von Vor- und Nachname

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
   cout << "Bitte Vor- und Nachnamen eingeben: ";
   string derName;
   cin >> derName;
   cout << derName;

   return 0;
}
```

Eingabe: `Daisy Duck`

Ausgabe:

Einfache Ein- und Ausgabe

Eingabe von Zeichenketten

Beispiel: Einlesen von Vor- und Nachname - Version 2

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

int main()
{
   cout << "Bitte Vor- und Nachnamen eingeben: ";
   string derName;
   getline(cin, derName);
   cout << derName;

   return 0;
}
```

Eingabe: `Paulchen Panther`

Ausgabe: **`Paulchen Panther`**

Einfache Ein- und Ausgabe

Ausgabe

Der Operator << formt automatisch aus der internen Darstellung in eine Textdarstellung um. Es wird stets die mindestens notwendige Weite genommen:

```
cout << 7 << 11;
```

erscheint als

```
711
```

Formatierungen sind möglich, zum Beispiel:

```
cout << 7;
cout.width(6);
cout << 11;
```

erscheint als

```
7     11
```

Einfache Ein- und Ausgabe

Ein- und -Ausgabe mit Dateien

Die Ein- und Ausgabeoperatoren >> und << sowie die Funktionen `get()` zur Eingabe eines Zeichens und `put()` zur Ausgabe eines Zeichens sind auch bei Dateien verwendbar. Der Header `<fstream>` enthält die vom Compiler benötigten Beschreibungen für Dateiobjekte.

Ein Programm, das auf Dateien zugreift, enthält die folgenden wesentlichen Elemente:

- Es wird ein Dateiobjekt definiert. Der Typ des Dateiobjekts ist `ifstream` für Ein- und `ofstream` für Ausgabedateien.
- Die Verbindung des Dateiobjekts zu einer existierenden oder anzulegenden Datei wird mit der Funktion `open()` hergestellt, d.h. die Datei wird „geöffnet“. Der externe Dateiname wird der Funktion `open()` als Zeichenkette übergeben.
- Die Verbindung wird mit der Funktion `close()` wieder gelöst, d.h. die Datei wird „geschlossen“.

Bei Programmende wird automatisch ein `close()` durchgeführt. Es empfiehlt sich jedoch, eine Datei zu schließen, sobald nichts mehr geschrieben werden soll und noch weitere Programmteile folgen.

Einfache Ein- und Ausgabe

Ein- und -Ausgabe mit Dateien

Beispiel: Kopieren einer beliebigen Datei

```
#include <iostream>
#include <string>
#include <cstdlib>          // fuer exit()
#include <fstream>
using namespace std;

int main()
{
    // Definieren und Oeffnen der Eingabedatei
    ifstream quelle;
    string quelldateiname;
    cout << "Quelldatei? ";
    cin >> quelldateiname;

    // Datei oeffnen; nicht portabel:
    // quelle.open(quelldateiname.c_str());
    quelle.open(quelldateiname.c_str(),
                ios::binary|ios::in);
    // Ohne ios::binary sind unter DOS/Windows
    // und OS/2 nur Textdateien kopierbar
    // bei ifstreams muss ios::binary
    // mit ios::in verknuepft werden

    if (!quelle)    // Fehlerabfrage
    {
        cerr << quelldateiname
             << " kann nicht geoeffnet werden!\n";
        exit(-1);
    }
```

Einfache Ein- und Ausgabe

Ein- und -Ausgabe mit Dateien

Beispiel: Kopieren einer beliebigen Datei (Fortsetzung)

```
    string zieldateiname;
    cout << "Zieldatei? ";
    cin >> zieldateiname;
    // Definieren und Oeffnen der Ausgabedatei
    // hier in einem Schritt; nicht portabel:
    // ofstream ziel(zieldateiname.c_str());
    ofstream ziel(zieldateiname.c_str(),
                  ios::binary|ios::out);
    // bei ofstreams muss ios::binary
    // mit ios::out verknuepft werden

    if (!ziel)      // Fehlerabfrage
    {
        cerr << zieldateiname
             << " kann nicht geoeffnet werden!\n";
        exit(-1);
    }

    // zeichenweise kopieren
    char ch;
    while (quelle.get(ch))
        ziel.put(ch);

    // Dateien schliessen
    quelle.close();
    ziel.close();

    return 0;
}
```