Berufskolleg Opladen
Stauffenbergstraße 23
51575 Leverkusen
Tel.: 0221 / 20234

C++ - Dateiein- und Ausgabe

Frank van den Boom

1. März 2015

Inhaltsverzeichnis

1 Einleitung

In unseren Programmen haben wir bereits viele Elemente der E / A-Bibliothek (Eingabe / Ausgabe-Bibliothek) verwendet:

- cin (ausgesprochen: see-in), das die Standardeingabe liest
- cout (ausgesprochen: see-out),das in die Standardausgabe schreibt
- Operator >>, der zum Lesen der Eingabe dient
- Operator <<, der zum Schreiben der Ausgabe verwendet wird

Dieses Skript erörtert die Unterstützung von Lese- und Schreibvorgängen in Dateien und Strings.

2 Die Standard E/A-Bibliothek

Die E/A-Typen werden in drei seperaten Headern definiert: iostream definiert die Typen, die zum Lesen und Schreiben in Konsolenfenstern verwendet werden. Diese haben wir schon oft benutzt. fstream legt die Typen fest, die zum Lesen und Schreiben benannter Dateien engesetzt werden, und sstream definiert die Typen, die zum Lesen und Schreiben von im Speicher befindlichen Strings genutzt wird (dazu kommen wir aber erst später).

Berufskolleg Opladen
Stauffenbergstraße 23
51575 Leverkusen
Tel.: 0221 / 20234

2.1 Einige Typen und Header der E/A-Bibliothek

Header	Typ
iostream	istream liest Streams ostream schreibt Streams iostream liest und schreibt Streams
fstream	ifstream liest Dateien ofstream schreibt Dateien fstream liest und schreibt Dateien
sstream	istringstream liest Strings ostringstream schreibt Strings stringstream liest und schreibt Strings

3 Dateiein- und -ausgabe

Der fstream Header definiert also drei Typen zur Unterstützung von Datei-E/A-Vorgängen.

1. ifstream, liest Dateien
2. ofstream, schreibt Dateien
3. fstream, liest und schreibt Dateien

3.1 Dateistreamobjekte verwenden

Bisher haben unsere Programme die von der Bibliothek definierten Objekte cin, cout und cerr verwendet. Wenn wir eine Datei lesen oder schreiben wollen, müssen wir unsere Objekte definieren und an die gewünschten Dateien binden. Wenn wir davon ausgehen, dass es sich bei iFileName und oFileName um Strings mit den Namen der Dateien handelt, die wir lesen und schreiben wollen, dann können wir zum Beispiel den folgenden Code schreiben, um ein fstream-Objektpaar zu definieren und zu öffnen: [caption=002.cpp] // ifstream ertstellen und an die Datei mit dem Namen iFileName binden ifstream infile(iFileName); // ofstream erstellen und an die Datei mit dem Namen oFileName binden ofstream outfile(oFileName); Bei infile handelt es sich um einen Stream, den wir lesen, und bei outfile um einen Stream, in den wir schreiben können. Die Bereitstellung eines Dateinamens als Initialisierer auf ein ifstream- oder ofstream-Objekt führt dazu, dass die genannte Datei geöffnet wird.

```
ifstream infile;  // ungebundener Eingabedateistream
ofstream outfile; // ungebundener Ausgabedateistream
```

Diese Definitionen legen infile als Streamobjekt fest, das die Datei liest, und outfile als Objekt, das wir zum Schreiben in eine Datei verwenden können. Bisher ist keiner der beiden Streams an eine Datei gebunden. Bevor die Streams verwenden können, müssen wir diese wie folgt an eine Datei binden:

```
infile.open("in.txt");     // Die Datei in.txt wird zum Lesen geoeffnet
outfile.open("out.txt");   // Die Datei out.txt wird zum Schreiben geoeffnet
```

3.2 Das erfolgreiche Öffnen einer Datei überprüfen

Nach dem Öffnen einer Datei ist es sinnvoll zu üperprüfen, ob das Öffnen erfogreich verlaufen ist:

```
// Pruefen, ob das Oeffnen erfolgreich war
if((!infile) {
    cerr << "error: unable to open input file: "
         << infile << endl;
}
```

Berufskolleg Opladen
Stauffenbergstraße 23
51575 Leverkusen
Tel.: 0221 / 20234

3.3 Einen Dateistream an eine neue Datei binden

Nachdem fstream geöffnet wurde, bleibt sie mit der genannten Datei verbunden. Um fstream mit einer anderen Datei zu verknüpfen, müsssen wir zunächst die vorhandene Datei schließen, den Status zurücksetzen und dann eine andere Datei öffnen:

```
ifstream infile("in.txt");   // Oeffnet die Datei in.txt zum Lesen

infile.close();              // Schliesst die Datei in.txt
infile.clear();              // Status auf O.K. setzen
infile.open("next.txt");     // Oeffnet die Datei next.txt zum Schreiben
```

4 Dateimodi

Es gibt sechs Dateimodi.

Name	Beschreibung
in	zur Eingabe öffnen
out	zur Ausgabe öffnen
app	vor dem Schreiben das Ende suchen
trunc	einen vorhandenen Stream beim Öffnen abschneiden
binary	E / A-Operationen im binären Modus ausführen

Jedesmal wenn wir eine Datei öffnen, wird implizit oder explizit ein Dateimodus festgelegt. Der Standardwert der implizit festgelegt wird, schwankt je nach Streamtyp. Standardmäßig werden im ifstream verknüpfte Dateien im in-Modus geöffnet. Die vom ofstream geöffneten Dateien werden im out-Modus geöffnet. Eine im out-Modus geöffnete Datei wird abgeschnitten: Alle in der Datei gespeicherten Daten werden vor dem Zugriff gelöscht. Alternativ können wir den Dateimodus angeben, in dem die Datei geöffnet wird.

```
// output-Modus als Standard; schneidet die Datei
// mit dem Namen fileA.txt ab
ofstream outFile("fileA.txt");
// gleichwertiger Effekt: fileB.txt wird explizit abgeschnitten
ofstream outFileB("fileB.txt", ofstream::out | ofstream::trunc);
// append-Modus; f gt am Ende einer vorhandenen Datei fileC.txt neue Daten ein
ofstream appfile("fileC.txt", ofstream::app);
```

4.1 Handschriftlicher Exkurs zu Zahlensystemen und bitweisen Operatoren

Siehe handschriftliche Mitschrift des Tafelbildes.