

Hydroinformatik I - WiSe 2018/2019

V6b: Input/Output (I/O) - Files

Prof. Dr.-Ing. habil. Olaf Kolditz

¹Helmholtz Centre for Environmental Research – UFZ, Leipzig

²Technische Universität Dresden – TUD, Dresden

³Center for Advanced Water Research – CAWR

Dresden, 16.11.2018

Semesterfahrplan

WiSe 2019/2020: Hydroinformatik I, Freitag (3. DS) 11:10-12:40, HÜL/S186/H					
No	KW	Datum	ID	Vorlesung	Dozent
1a	42	19.10.2019	BHYWI-08-01	Hydroinformatik - Einführung	Kolditz
1b	42	19.10.2019	BHYWI-08-02	Compiler (Installation)	Kolditz
2	43	25.10.2019	BHYWI-08-03	Datentypen	Kolditz
3	44	01.11.2019	BHYWI-08-05	Hausaufgaben	Kolditz
5	45	08.11.2019	BHYWI-08-04	Einführung in Python	Kolditz
4	46	15.11.2019	BHYWI-08-06	Programmieren in Nat-Ing	Kalbacher
6	47	22.11.2019	BHYWI-08-07	Klassen	Kolditz
7	48	29.11.2019	BHYWI-08-08	Input-Output (I/O)	Kolditz
8	49	06.12.2019	BHYWI-08-09	Strings - Textverarbeitung	NN
9	50	13.12.2019	BHYWI-08-10	Pointer	NN
10	51	20.12.2019	BHYWI-08-11	Container	Kolditz
11	1	03.01.2020	BHYWI-08-12	BigData & Water 4.0	Kolditz
12	2	10.01.2020	BHYWI-08-13	Hydrologische Modellierung	Kolditz
13	3	17.01.2020	BHYWI-08-14	Neuronale Netzwerke	Kolditz
14	4	24.01.2020	BHYWI-08-15	ANN / Bayes'sche Netzwerke	Kolditz
15	5	31.01.2020	BHYWI-08-16	BN / Maschinelles Lernen	Kolditz
16	6	07.02.2020	BHYWI-08-17	Klausurvorbereitung	Kolditz

fstream Klasse

Abb. 2 zeigt die Hierarchie der fstream Klassen.

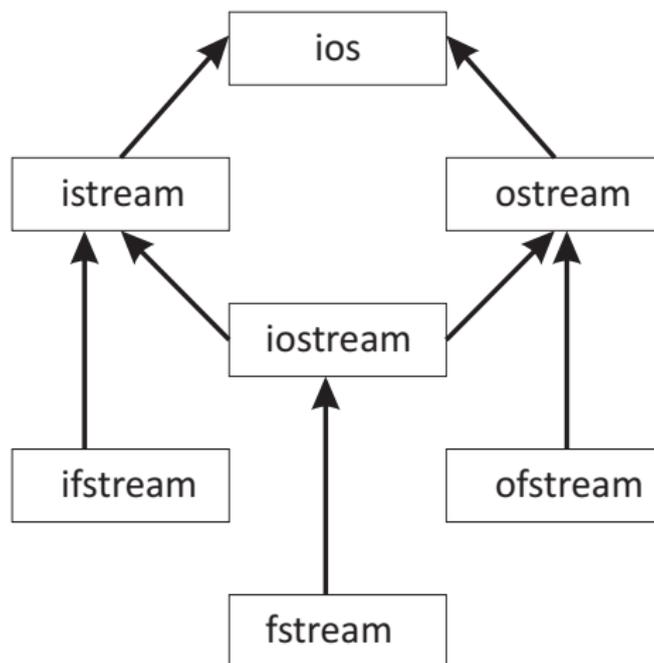


Abbildung: Die fstream Klassen

fstream Klasse

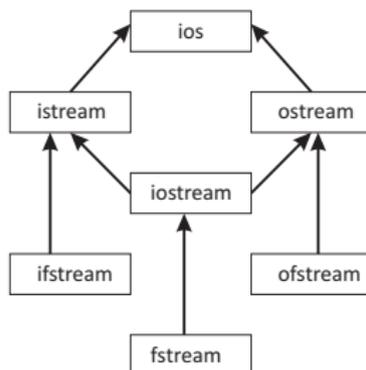


Abbildung: Die fstream Klassen

Diese sind von den bereits bekannten ios stream Klassen abgeleitet.

- ▶ ifstream: ist von istream abgeleitet für das Lesen von Dateien.
- ▶ ofstream: ist von ostream abgeleitet für das Schreiben von Dateien.
- ▶ fstream: ist von iostream abgeleitet für das Lesen und Schreiben von Dateien.

File streams anlegen

Eröffnungsmodus: Um eine Datei benutzen zu können, muss diese zunächst geöffnet werden.

Flag	Funktion
<code>ios::in</code>	Eine (existierende) Datei wird zum Lesen geöffnet.
<code>ios::out</code>	Eine Datei wird zum Schreiben geöffnet. Existierende Inhalte werden überschrieben.
<code>ios::app</code>	Die neuen Inhalte werden an die existierenden angehängt.
<code>ios::trunc</code>	Eine bestehende Datei wird beim öffnen auf die Länge 0 gekürzt.
<code>ios::ate</code>	Schreib- und Leseoperationen werden auf das Dateiende gesetzt.
<code>ios::binary</code>	Schreib- und Leseoperationen werden im Binärmodus ausgeführt.

Tabelle: Eröffnungsmodi für Dateien

Die default Werte sind:

- ▶ `ios::in` für `ifstream`
- ▶ `ios::out` | `ios::trunc` für `ofstream`

File-Streams schließen

Wir wissen schon, dass es bei objekt-orientierten Sprachen immer zwei passende Dinge gibt, z.B. Klassen-Konstruktoren und -Destruktoren. So ist zu erwarten, dass es zu einer Methode 'Datei öffnen' (`open()`) auch eine Methode 'Datei schließen' gibt (`close()`) (siehe Tabelle 6.4)

A simple copy function E623

```
#include <iostream> // for using cout
#include <fstream> // for using ifstream / ofstream
#include <string> // for using string
using namespace std; // namespace for std functions

int main()
{
    //-----
    ifstream input_file; // Instance of class ifstream
    input_file.open("input_file.txt"); // Open file "text_file.txt"
    string my_string; // Instance of class string
    input_file >> my_string; // Reading a string from file
    cout << my_string.data() << endl; // Output of string to screen
    //-----
    ofstream output_file; // Instance of class ifstream
    output_file.open("output_file.txt"); // Open file "text_file.txt"
    output_file << my_string; // Writing a string to a file
    //-----
    return 0;
}
```

Die Ein- >> und Ausgabeoperatoren << formatieren die Datentypen (z.B. int in der Übung E623 entsprechend den Einstellungen der fstream Klasse. Diese Einstellungen können durch Flags verändert werden (siehe nächsten Abschnitt).

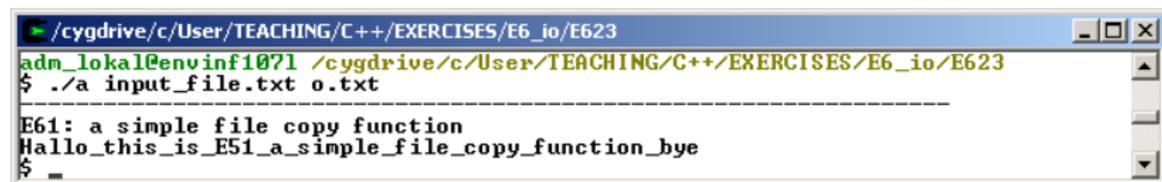
6.2.3 main() Function with parameters

Die `main()` Funktion kann auch mit einer Parameterliste (`int argc`, `char *argv[]`) versehen werden. Die Anzahl der Parameter (`argc`) wird automatisch erkannt. Mit jedem Leerzeichen in der Tastatureingabe entsteht ein neuer Eingabeparameter (Abb. 3).

```
int main(int argc, char *argv[])
{
    ifstream input_file;           // Instance of class ifstream
    input_file.open(argv[1]);      // Open file, name from cin
    ofstream output_file;        // Instance of class ifstream
    output_file.open(argv[2]);    // Open file, name from cin
    return 0;
}
```

6.2.3 main() Function with parameters

Die Benutzung der main Funktion mit Eingabeparametern ist in der folgenden Abbildung zu sehen.



```
/cygdrive/c/User/TEACHING/C++/EXERCISES/E6_io/E623
adm_lokal@envinf1071 /cygdrive/c/User/TEACHING/C++/EXERCISES/E6_io/E623
$ ./a input_file.txt o.txt
-----
E61: a simple file copy function
Hallo_this_is_E51_a_simple_file_copy_function_bye
$ _
```

Abbildung: Die main Funktion mit Parametern

6.2.4 Daten-Konverter

Ihre Frage nach dem Sinn der Übung E623 ist vollkommen berechtigt, wozu ein Programm schreiben, um eine Datei zu kopieren. Das kann ich doch auch mit dem Windows-Explorer oder mit `cp file1 file2` machen. Richtig, aber genauso funktionieren Kopierprogramme, Windows-Explorer ruft 'unser' Kopierprogramm auf. Wenn wir auch nur kleine Änderungen in unserer Datei vornehmen wollen (z.B. eine laufende Nummer in jede Zeile einfügen), kann uns der Windows-Explorer nicht mehr weiter helfen. Dies ist insbesondere dann ärgerlich, wenn die Datei ziemlich groß ist ... Auch hier sagen sie zu Recht, eine Nummerierung in eine größere Datei einfügen, das kann ich z.B. auch mit EXCEL machen. In der nächsten Übung schreiben wir einen kleinen Konverter, also genau was EXCEL im Hintergrund macht, wenn wir eine neue Spalte einfügen.

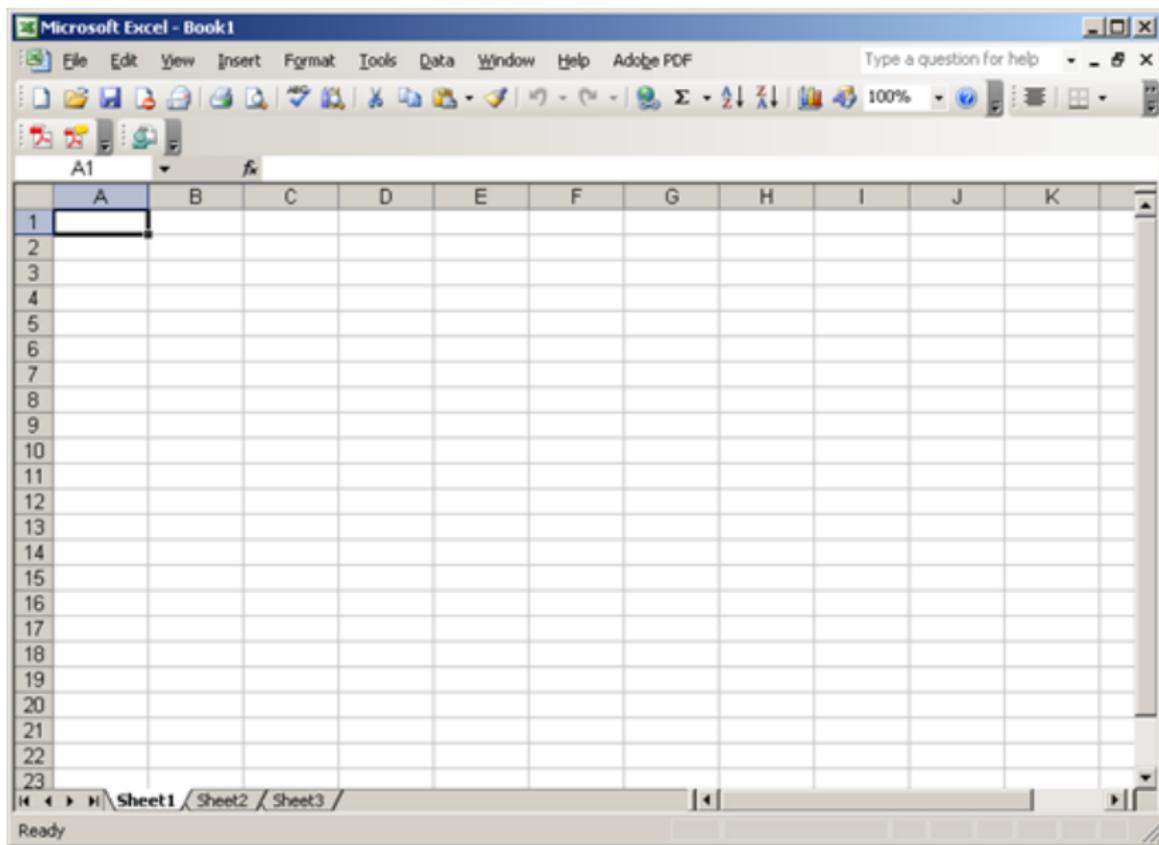
Windows Explorer window titled "DRESDEN" showing the file structure of a folder. The address bar indicates the path: C:\User\TEACHING\C++\LECTURES\DRESDEN.

The left pane shows the folder structure:

- GEOSYS
- PAPER
- PICTURES
- PRESENTATIONS
- PROJECTS
- PVT
- REVIEWS
- TEACHING
 - C++
 - Copy of SCRIPT
 - EXERCISES
 - http%3a%2f%2fftp.inf.tu-dresden.
 - LECTURES
 - DRESDEN
 - TUEBINGEN
 - SCRIPT
 - FIGURES
 - WEB
 - CMEFM
 - HABIL
 - HIGRADE
 - IPSWaT
 - OKAYAMA_2009
 - TUTORIALS

The right pane shows a list of files:

Name	Size	Type	Date Modified
L0.ppt	12.113 KB	Microsoft PowerPo...	17.04.2009 08:31
L1.ppt	1.734 KB	Microsoft PowerPo...	17.04.2009 08:26
L2.ppt	1.503 KB	Microsoft PowerPo...	24.04.2009 08:53
L3.ppt	2.066 KB	Microsoft PowerPo...	17.04.2009 08:10
L4.ppt	1.248 KB	Microsoft PowerPo...	08.05.2009 08:50
L5.pdf	332 KB	Adobe Acrobat Doc...	07.05.2009 17:03
L5.ppt	1.176 KB	Microsoft PowerPo...	08.05.2009 11:01
L6.ppt	732 KB	Microsoft PowerPo...	14.05.2009 17:19
L7.ppt	279 KB	Microsoft PowerPo...	24.04.2009 08:55



6.2.4 Daten-Konverter

```
int main()
{
    //-----
    ifstream input_file;           // Instance of class ifstream
    input_file.open("input.txt");  // Open file "text_file.txt"
    ofstream output_file;         // Instance of class ifstream
    output_file.open("output.txt"); // Open file "text_file.txt"
    //-----
    char line[80];
    int i=0;
    while(input_file.getline(line,80)) // Loop condition
    {
        output_file << i << " " << line << endl;
        i++;                          // Incrementor (+1)
    }
    //-----
    return 0;
}
```

C++ news tables

Was ist neu bei dieser Übung.

C++ Ding	Was tut's
<code>while()</code>	eine Kontrollstruktur für Schleifen (solange der Ausdruck in () wahr (true) ist wird die Schleife ausgeführt)
<code>i++</code>	der Inkremetor (zählt Eins hoch)

Table: C++ news

File-Streams und Klassen

Wir wollen eine Lesefunktion für die Klasse `CStudent` schreiben. Bevor wir damit beginnen, müssen wir uns Gedanken über eine geeignete Struktur für die Darstellung eines `CStudent` Datensatzes in einer Datei machen. Der Vorschlag für die Strukturierung von Datensätzen ist die Benutzung von Schlüsselwörtern zur Trennung von Datenblöcken, z.B.

```
#STUDENT
  $NAME_FIRST
    James
  $NAME_LAST
    Bond
  . . .
#STOP
```

File-Streams und Klassen

Wir benutzen zwei verschiedene Symbole für Schlüsselwörter:

- ▶ keyword # : zur Trennung von Datensätzen für eine Instanz von CStudent,
- ▶ subkeyword \$: zur Identifizierung der einzelnen Daten für die CStudent Instanz.
- ▶ #STOP zeigt das Ende der Datenbasis an. (Eigentlich wäre dies nicht nötig, da das Dateiende auch mit dem Parameter eof (end-of-file) abgefragt werden kann. Wir werden aber sehen, dass mit #STOP einiges einfacher zu programmieren ist.)

6.3 CStudent::Read() E63

```
ios::pos_type CStudent::Read(istream& input_file)
{
    //-----
    string input_line;
    char buffer[256]; // MAX_LINE
    ios::pos_type position;
    //-----
    while(true)
    {
        position = input_file.tellg();
        input_file.getline(buffer,256);
        input_line = buffer;
        if(input_line.size()<1) // empty line
            continue;
        // Dealing with keywords
        if(input_line.find('#')!=string::npos) // keyword found
            return position;
    }
    ...}
```

6.3 main() E63

```
#include <iostream> // for using cout
#include <fstream> // for using ifstream / ofstream
#include <string> // for using string
#include "student.h" // for using CStudents
using namespace std; // for std functions

int main()
{
    //-----
    // File handling
    ifstream input_file; // ifstream instance
    input_file.open("data_set.txt");
    if(!input_file.good() // Check is file existing
    {
        cout << "! Error in STD::Read: file could not be opened" << endl;
        return 0;
    }
    input_file.seekg(0L,ios::beg); // Rewind file
    //-----
    CStudent* m_std = new CStudent(); // CStudent instance
    m_std->Read(input_file);
    //-----
    input_file.close();
    return 0;
}
```

C++ news tables

Die main Funktion besteht aus zwei Teilen, dem File-Handling und dem Aufruf der Lesefunktion. Beim File-Handling wird der stream aus der Datei data_set.txt geöffnet, anschließend erfolgt der Test, ob das File erfolgreich geöffnet werden konnte; wenn nicht, wird die main Funktion sofort beendet.

Was ist neu bei dieser Übung.

C++ Ding	Was tut's
<code>ifstream& input_file</code>	eine Reference auf ein Objekt wird in Kapitel 7 ausführlich abgehandelt

Tabelle: C++ news

fstream Methoden

Methoden	Funktion
<code>open()</code>	öffnet die Datei
<code>good()</code>	testet erfolgreiche Öffnung der Datei
<code>seekg(pos, ios::beg)</code>	geht zur Position pos in der Datei
<code>seekg(0L, ios::beg)</code>	spoolt zum Dateianfang zurück
<code>tellg()</code>	merkt sich die aktuelle Position im stream
<code>getline(buffer, 256)</code>	holt eine Zeile der Länge 256 (Zeichen) aus dem stream und kopiert diese in buffer
<code>close()</code>	schließt Datei
<code>>></code>	Eingabeoperator für Dateien
<code><<</code>	Ausgabeoperator für Dateien

Tabelle: fstream Methoden

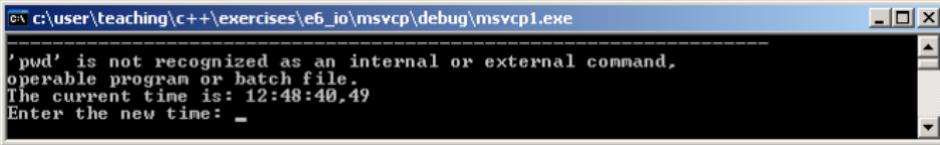
Die string Auswertung spielt bei der Lesefunktion eine wichtige Rolle, daher haben wir uns im vorangegangenen Kapitel mit der string Klasse beschäftigt.

Testfragen

1. Was ist die Basis-Klasse für alle Ein- und Ausgaben in C++ ?
2. Was sind die C++ Klassen für das Lesen und Schreiben von Dateien ?
3. Welchen Include benötigen wir für das Arbeiten mit I/O File-Klassen ?
4. Was sind die Standard-Flags für File-Streams (Lesen und Schreiben) ?
5. Mit welchem Flag können wir zu schreibende Daten an eine existierende Datei anhängen ?
6. Was ist der Unterschied zwischen ASCII- und Binär-Formaten ?
7. Mit welchem Flag können wir Daten in einem Binär-Format schreiben ?
8. Mit welcher Anweisung wird ein File geöffnet ?
9. Mit welcher Anweisung wird ein File geschlossen ?
10. Was bewirken die Stream-Operatoren << und >> ?
11. Wie können wir mit Dateinamen in unserem Hauptprogramm `main(...)` arbeiten ?
12. Welche Anweisung benötigen wir für die Erzeugung einer Instanz für einen Eingabe-Strom ?
13. Welche Anweisung benötigen wir für die Erzeugung einer Instanz für einen Ausgabe-Strom ?
14. Für die Erstellung einer Datenbank ist es wichtig einzelnen Datensätze zu trennen. Wie können wir soetwas in der Datenbank-Datei bewerkstelligen ?
15. Ist es wichtig das Ende einer Datenbank-Datei, z.B. mit einem Schlüsselwort `#STOP`, zu markieren ?
16. Mit welcher Abfrage können wir prüfen, ob die Öffnung einer Datei erfolgreich war ?
17. Mit welcher Anweisung können wir die aktuell gelesene Position in einer geöffneten Datei abfragen ?
18. Mit welcher Anweisung können wir zu einer bestimmten Position in einer geöffneten Datei springen ?
19. Mit welcher Anweisung können wir eine komplette Zeile aus geöffneten Datei auslesen ?

Quellcode und Compiler

Hier wartet schon die erste Überraschung auf uns: `pwd` wird nicht erkannt und wir werden aufgefordert eine Zeit einzugeben (Abb. 4).



```
CA c:\user\teaching\c++\exercises\e6_io\msvc\debug\msvcpl.exe
'pwd' is not recognized as an internal or external command,
operable program or batch file.
The current time is: 12:48:40,49
Enter the new time: _
```

Abbildung: Das Debug Verzeichnis

Was ist passiert ?

- ▶ `pwd` ist ein Linux-Befehl, den kennt der Windows-Compiler nicht.
- ▶ `time` gibt es auch als DOS-Befehl, hat aber eine ganz andere Bedeutung: nicht Ausgabe der Zeit sondern Zeit ändern.

Wir sehen also, dass unser Quellcode von verschiedenen Compilern unterschiedlich interpretiert wird.