Hydroinformatik I - WiSe 2018/2019

HyBHW-S1-01-V6b: Input/Output (I/O) - Files

Prof. Dr.-Ing. habil. Olaf Kolditz

[1]Helmholtz Centre for Environmental Research – UFZ, Leipzig

[2]Technische Universität Dresden – TUD, Dresden

[3]Center for Advanced Water Research – CAWR

[4]TUBAF-UFZ Center for Environmental Geosciences – C-EGS, Freiberg / Leipzig

Dresden, 27.11.2020

Semesterfahrplan

WiSe 2020/2021: Hydroinformatik I, Freitag (3. DS) 11:10-12:40, HÜL/S186/H

No	KW	Datum	ID	Vorlesung	Dozent
1	44	30.10.2020	HyBHW-1-01-01	Hydroinformatik - Einführung	Kolditz
2	44	30.10.2020	HyBHW-1-01-02	Compiler (Installation)	Kolditz
3	45	06.11.2020	HyBHW-1-01-03	Jupyter, Python	Kolditz
4	46	13.11.2020	HyBHW-1-01-04	Datentypen	Rink
5	47	20.11.2020	HyBHW-1-01-05	Klassen	Kolditz
6	48	27.11.2020	HyBHW-1-01-06	Input-Output (I/O)	Kolditz
7	49	04.12.2020	HyBHW-1-01-07	Strings - Textverarbeitung	Kolditz
8	50	11.12.2020	HyBHW-1-01-08	Pointer & Container	Kolditz
9	51	18.12.2020	HyBHW-1-01-09	Christmas Lecture	
10	1	08.01.2021	HyBHW-1-01-10	Hydrologische Modellierung	Kolditz
11	2	15.01.2021	HyBHW-1-01-11	BigData & Water 4.0	Kolditz
12	3	22.01.2021	HyBHW-1-01-12	Neuronale Netzwerke	Kolditz
13	4	29.01.2021	HyBHW-1-01-13	ANN / Bayes'sche Netzwerke	Kolditz
14	5	05.02.2021	HyBHW-1-01-14	BN / Maschinelles Lernen	Kolditz
15				Klausurvorbereitung	Kolditz

Informatik und Tools
Programmieren in C++
Hydrologische Modellierung

fstream Klasse

Abb. 2 zeigt die Hierarchie der fstream Klassen.

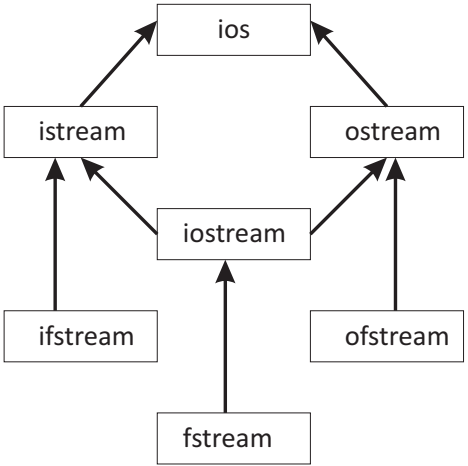


Figure: Die fstream Klassen

fstream Klasse

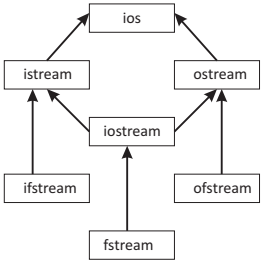


Figure: Die fstream Klassen

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

Table: 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 EX06c-io-files

```
#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).

```
1 int main(int argc, char *argv[])
2 {
3   ifstream input_file;                  // Instance of class ifstream
4   input_file.open(argv[1]);             // Open file, name from cin
5   ofstream output_file;                 // Instance of class ifstream
6   output_file.open(argv[2]);            // Open file, name from cin
7   return 0;
8 }
```

6.2.3 main() Function with parameters

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

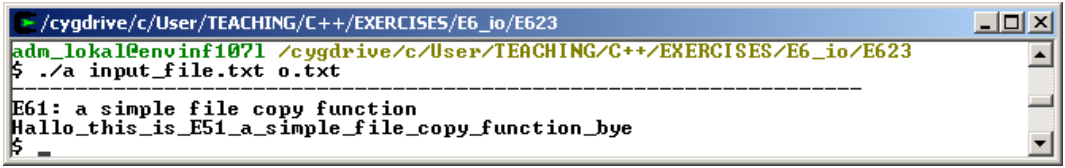
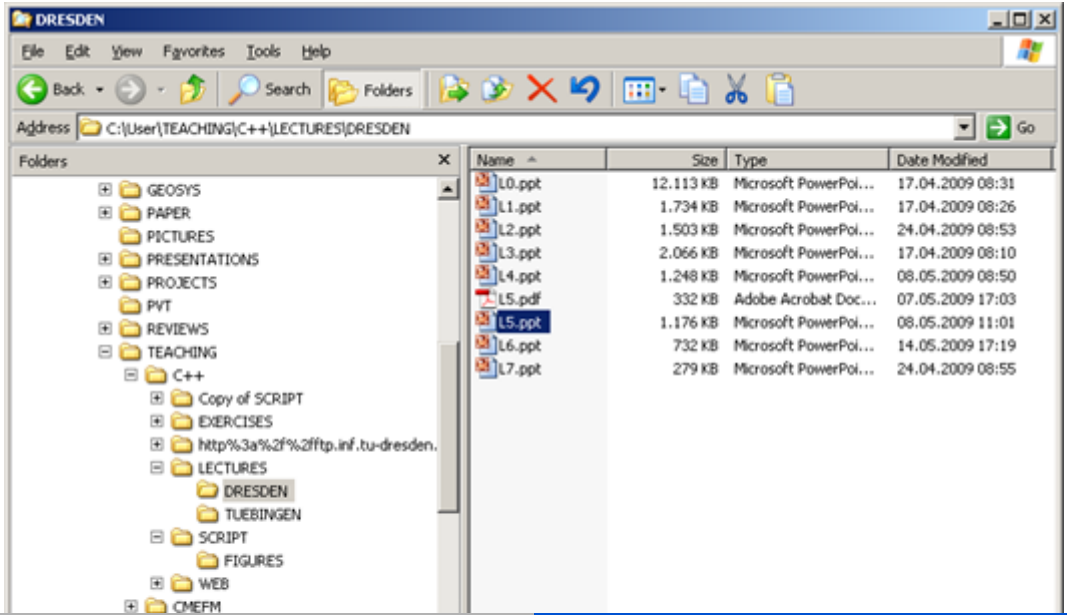
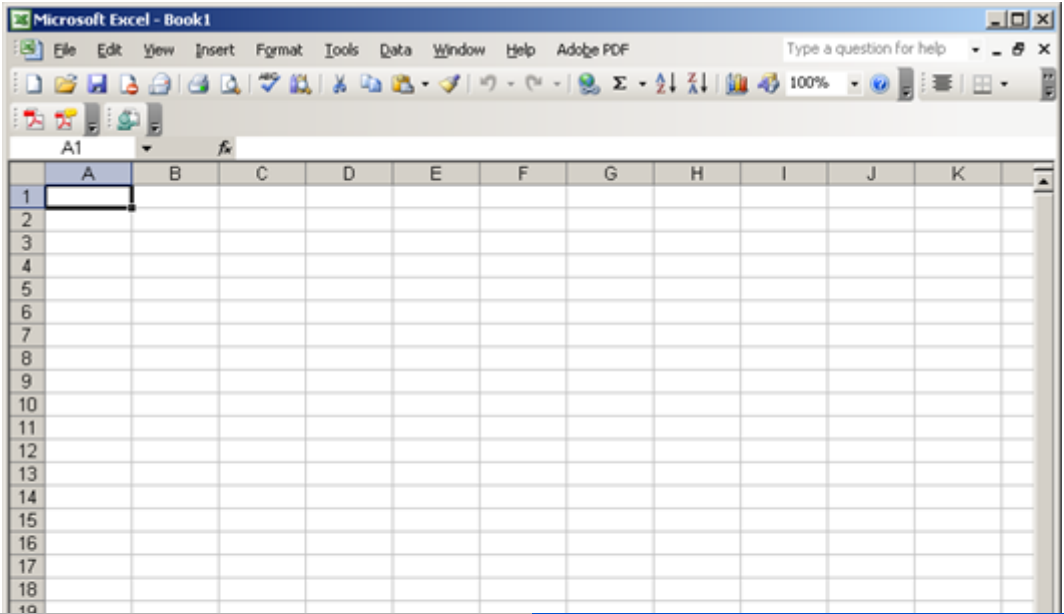


Figure: Die main Funktion mit Parametern

Daten-Konverter

Ihre Frage nach dem Sinn der Übung `EX06d-io-converter` 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 laufenden 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.


DRESDEN
File Edit View Favorites Tools Help
Back Search Folders
Address C:\User\TEACHING\C++\LECTURES\DRESDEN
Go
Folders
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
Name Size Type Date Modified
L0.ppt 12.113 KB Microsoft PowerPoi... 17.04.2009 08:31
L1.ppt 1.734 KB Microsoft PowerPoi... 17.04.2009 08:26
L2.ppt 1.503 KB Microsoft PowerPoi... 24.04.2009 08:53
L3.ppt 2.066 KB Microsoft PowerPoi... 17.04.2009 08:10
L4.ppt 1.248 KB Microsoft PowerPoi... 08.05.2009 08:50
L5.pdf 332 KB Adobe Acrobat Doc... 07.05.2009 17:03
L5.ppt 1.176 KB Microsoft PowerPoi... 08.05.2009 11:01
L6.ppt 732 KB Microsoft PowerPoi... 14.05.2009 17:19
L7.ppt 279 KB Microsoft PowerPoi... 24.04.2009 08:55


Microsoft Excel - Book1
File Edit View Insert Format Tools Data Window Help Adobe PDF
Type a question for help
100%
A1
A B C D E F G H I J K
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

6.2.4 Daten-Konverter

```
1  int main()
2  {
3    //---------------------------------------------------------------
4    ifstream input_file;                  // Instance of class ifstream
5    input_file.open("input.txt");         // Open file "text_file.txt"
6    ofstream output_file;                 // Instance of class ifstream
7    output_file.open("output.txt");       // Open file "text_file.txt"
8    //---------------------------------------------------------------
9    char line[80];
10   int i=0;
11   while(input_file.getline(line,80))    // Loop condition
12   {
13     output_file << i << " " << line << endl;
14     i++;                                // Incrementor (+1)
15   }
16   //---------------------------------------------------------------
17   return 0;
18 }
```

C++ news tables

Was ist neu bei dieser Übung.

C++ Ding	Was tut's
`while()`	eine Kontrollstruktur für Schleifen (solange der Ausdruck in () wahr (true) ist wird die Schleife ausgeführt)
`i++`	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(ifstream& 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;
   ...}
```

Listing 1: Title

6.3 main() E63

```
1  #include <iostream>  // for using cout
2  #include <fstream>   // for using ifstream / ofstream
3  #include <string>    // for using string
4  #include "student.h" // for using CStudents
5  using namespace std; // for std functions
6
7  int main()
8  {
9    //---------------------------------------------------------------
10   // File handling
11   ifstream input_file; // ifstream instance
12   input_file.open("data_set.txt");
13   if(!input_file.good())                // Check is file existing
14   {
15     cout << "! Error in STD::Read: file could not be opened" << endl;
16     return 0;
17   }
18   input_file.seekg(0L,ios::beg);        // Rewind file
19   //---------------------------------------------------------------
20   CStudent* m_std = new CStudent();     // CStudent instance
21   m_std->Read(input_file);
22   //---------------------------------------------------------------
23   input_file.close();
24   return 0;
25 }
```

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, wir die main Funktion sofort beendet.
Was ist neu bei dieser Übung.

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

Table: C++ news

fstream Methoden

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

Table: fstream Methoden

Die string Auswertung spielt bei der Lesefunktion eine wichtige Rolle, daher werden wir uns wir uns in der nächsten Vorlesung 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).

Figure: 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.