Hydroinformatik I - WiSe 2019/2020

V6a: Input/Output (I/O)

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

Dresden, 22.11.2019

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

Übung

Klassen (zur Erinnerung)

I/O Konzepte

Ein Programm (ob prozedural, modular, objekt-orientiert) ist eigentlich nichts weiter als eine Datenverarbeitung zwischen einer Eingabe (Input) und einer Ausgabe (Output). I und O können mehr oder weniger schick gemacht sein:

1. I/O Standardgeräte,
2. I/O Dateien,
3. Datenbanken (I) und Visualisierung (O).

UFZ – VISLab

Aus didaktischen Gründen müssen wir leider mit dem Langweiligsten - I/O Standardgeräte - anfangen. Spannend wird's, wenn Daten durch die Visualisierung 'lebendig' werden. Die Abb. **??** zeigt eine professionelle Datenaufbereitung einen Porenraummodells in unserem Labor für wissenschaftliche Visualisierung (TESSIN-VISLab) am UFZ in Leipzig.

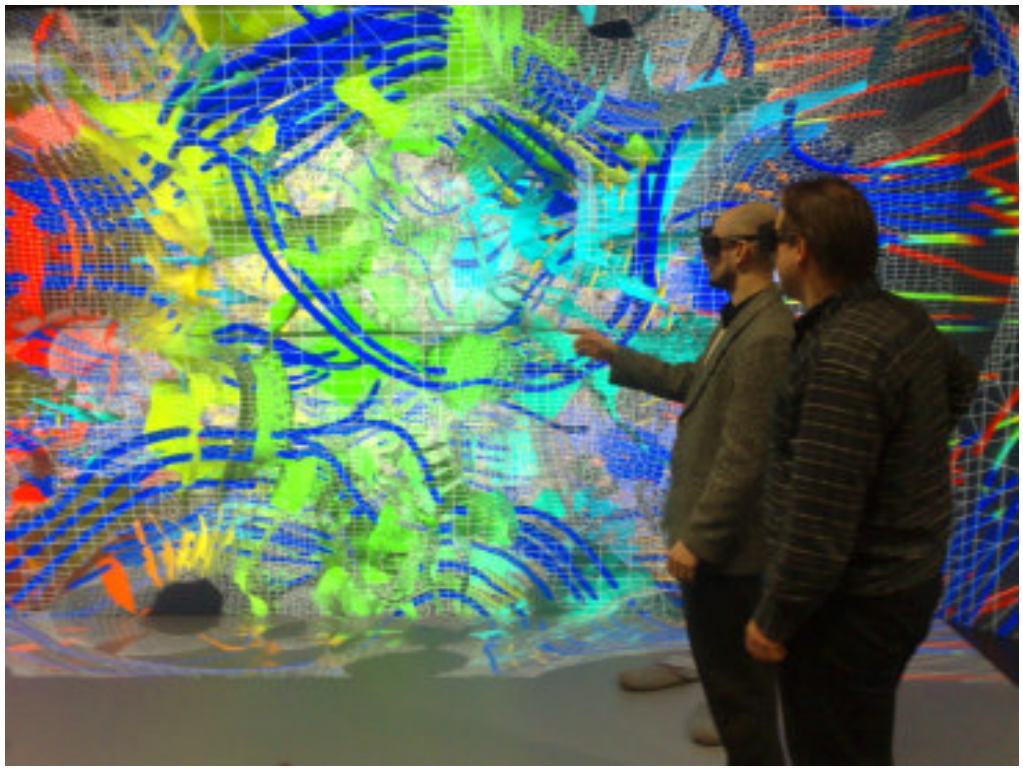

Abbildung: Wissenschaftliche Visualisierung

Die `iostream` Klasse

Die Klasse `iostream` geht durch Mehrfachvererbung aus den Klassen `istream` und `ostream` hervor. `iostream` stellt damit die Funktionalität beider I/O Klassen zu Verfügung.

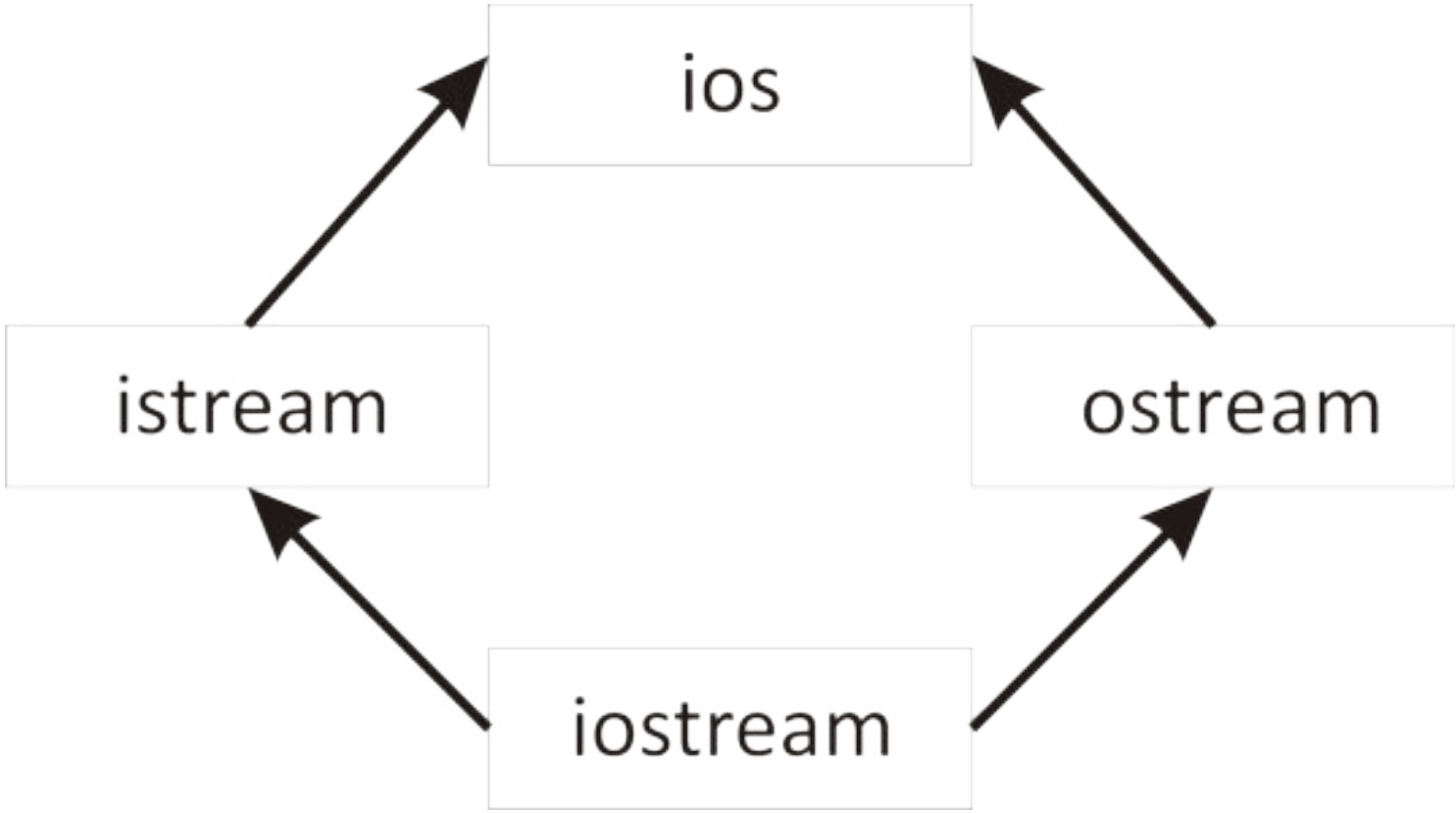

Abbildung: I/O stream Klassen

Die Standard-Streams

Es gibt vier Standard-Streams:

- `cin`: Standard-Eingabe über die Tastatur, Objekt der Klasse istream
- `cout`: Standard-Ausgabe auf dem Bildschirm, Objekt der Klasse ostream
- `cerr` und `clog`: zwei Objekte der Klasse ostream für die Fehlerausgabe.

Die Ein- $>>$ und Ausgabeoperatoren $<<$ transportieren die Ströme von und zu den Eingabe- bzw. Ausgabegeräten. Dabei formatieren sie die Datentypen (z.B. int in der Übung E3.1) entsprechend den Einstellungen der Klasse ios. Diese Einstellungen können durch Flags verändert werden (siehe nächsten Abschnitt).

Übung E31

```
#include <iostream>
using namespace std;
int main()
{
  int zahl;
  cout << "Bitte eine ganze Zahl eingeben: ";
  cin >> zahl;
  cout << zahl << endl;
  return 0;
}
```

Formatierte Ausgaben

Wir beschäftigen uns mit der Gestaltung, d.h. Formatierung, von Ausgaben, wir wollen die Bildschirmausgabe schick machen, z.B. in Tabellenform, dass alles schön untereinander steht. Der zweite Aspekt der Formatierung ist die Genauigkeit von ausgegebenen Zahlenwerten.

Formatierte Ausgabe von Ganzzahlen

In der nachfolgenden Übung E3.2.1 beschäftigen wir uns mit den verschiedenen Ausgabemöglichkeiten von ganzen Zahlen.

```
#include <iostream>
using namespace std;
int main()
{
  int zahl;
  cout << "Bitte eine ganze Zahl eingeben: ";
  cin >> zahl;
  cout << uppercase // für Hex-Ziffern
       << " oktal \t\t dezimal \t hexadezimal \n "
       << oct << zahl << " \t\t "
       << dec << zahl << " \t\t "
       << hex << zahl << endl;
  return 0;
}
```

Formatierte Ausgabe von Gleitpunktzahlen

In der nachfolgenden Übung beschäftigen wir uns mit den verschiedenen Ausgabemöglichkeiten von realen Zahlen.

Methoden	Wirkung
`int precision(int n)`	Genauigkeit wird auf n gesetzt

Formatierte Ausgabe von Gleitpunktzahlen

Übung E3.2.2

```
#include <iostream>
using namespace std;
int main()
{
  double zahl;
  cout << "Bitte eine Gleitkommazahl eingeben: ";
  cin >> zahl;
  cout.precision(7); // auf sieben Stellen genau
  cout << "Standard: \t"                  << zahl << endl;
  cout << "showpoint: \t"  << showpoint   << zahl << endl;
  cout << "fixed: \t\t"    << fixed       << zahl << endl;
  cout << "scientific: \t" << scientific  << zahl << endl;
  return 0;
}
```

Ausgabe von Speicherbedarf

In dieser Übung E3.2.3 benutzen wir den `sizeof` Operator, um den Speicherbedarf von Standard Daten-Typen zu bestimmen.

```
#include <iostream>
using namespace std;
int main()
{
  cout << "Type\tNumber of bytes\n";
  cout << "----------------------\n";
  cout << "bool\t\t" << sizeof(bool) << endl;
  cout << "char\t\t" << sizeof(char) << endl;
  cout << "short\t\t" << sizeof(short) << endl;
  cout << "int\t\t" << sizeof(int) << endl;
  cout << "long\t\t" << sizeof(long) << endl;
  cout << "float\t\t" << sizeof(float) << endl;
  cout << "double\t\t" << sizeof(double) << endl;
  cout << "long double\t" << sizeof(long double) << endl;
  return 0;
}
```