Hydroinformatik I - WiSe 2019/2020
HyBHW-S1-01-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
[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

Übung

Klassen (zur Erinnerung)

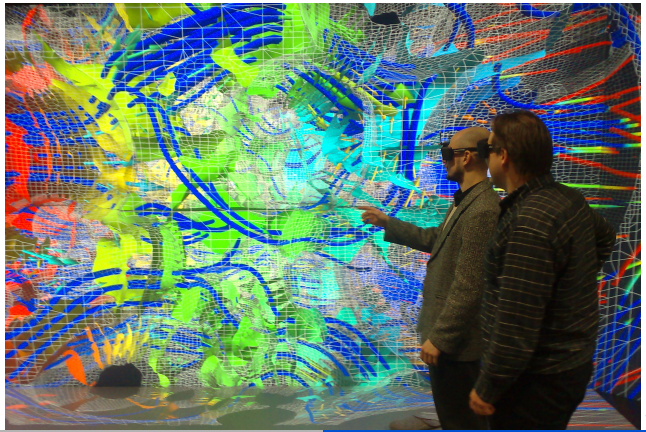
I/O Konzepte

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. 1 zeigt eine professionelle Datenaufbereitung einen Porenraummodells in unserem Labor für wissenschaftliche Visualisierung (TESSIN-VISLab) am UFZ in Leipzig.



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.

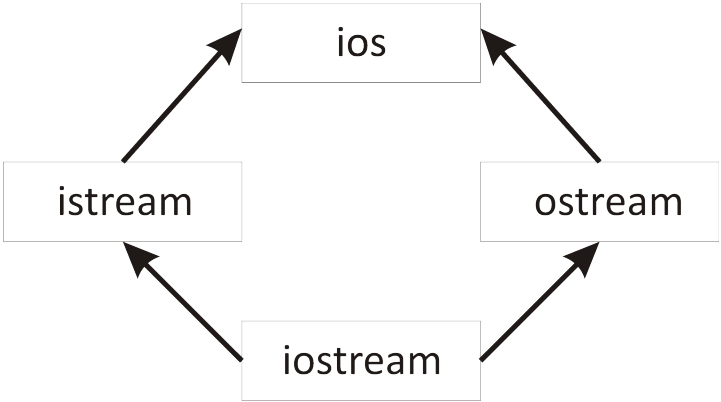


Figure: I/O stream Klassen

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 EX06a-io

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5    int zahl;
6    cout << "Bitte eine ganze Zahl eingeben: ";
7    cin >> zahl;
8    cout << zahl << endl;
9    return 0;
10 }
```

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.

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5    int zahl;
6    cout << "Bitte eine ganze Zahl eingeben: ";
7    cin >> zahl;
8    cout << uppercase // fuer Hex-Ziffern
9         << " oktal \t\t dezimal \t hexadezimal \n "
10        << oct << zahl << " \t\t "
11        << dec << zahl << " \t\t "
12        << hex << zahl << endl;
13   return 0;
14 }
```

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 EX06b-io-double

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5    double zahl;
6    cout << "Bitte eine Gleitkommazahl eingeben: ";
7    cin >> zahl;
8    cout.precision(7); // auf sieben Stellen genau
9    cout << "Standard: \t"                 << zahl << endl;
10   cout << "showpoint: \t"  << showpoint  << zahl << endl;
11   cout << "fixed: \t\t"    << fixed      << zahl << endl;
12   cout << "scientific: \t" << scientific << zahl << endl;
13   return 0;
14 }
```

Ausgabe von Speicherbedarf

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

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5    cout << "Type\tNumber of bytes\n";
6    cout << "-----------------------\n";
7    cout << "bool\t\t" << sizeof(bool) << endl;
8    cout << "char\t\t" << sizeof(char) << endl;
9    cout << "short\t\t" << sizeof(short) << endl;
10   cout << "int\t\t" << sizeof(int) << endl;
11   cout << "long\t\t" << sizeof(long) << endl;
12   cout << "float\t\t" << sizeof(float) << endl;
13   cout << "double\t\t" << sizeof(double) << endl;
14   cout << "long double\t" << sizeof(long double) << endl;
15   return 0;
16 }
```