Lab 3. File I/O and Data Processing

Ashokan Reservoir

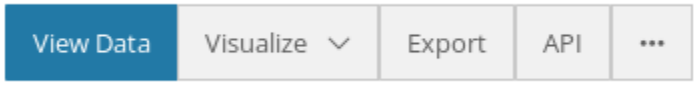
Located in Ulster County, about 13 miles west of Kingston and 73 miles north of New York City. Formed by the damming of the Esopus Creek, which eventually flows northeast and drains into the Hudson River. Consisting of **two basins** separated by a concrete dividing weir and roadway, it **holds 122.9 billion gallons** at full capacity and was placed into service in 1915.

The Ashokan is one of two reservoirs in the City's Catskill Water Supply System. The other is the Schoharie, located 27 miles to the north, whose water flows into the Ashokan via the Shandaken Tunnel and the Esopus Creek. Including the water it receives from the Schoharie Reservoir, the Ashokan supplies about **40% of New York City's daily drinking water** needs in non-drought periods.

Data on Ashokan water levels

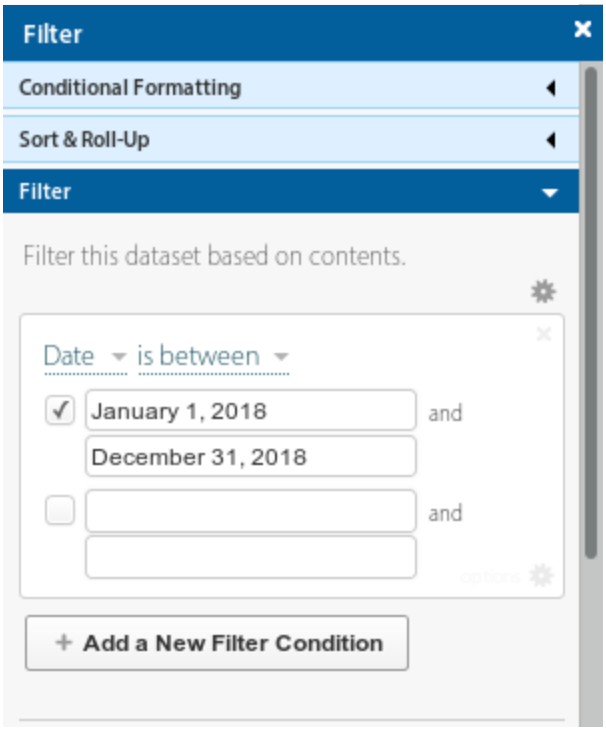
In this lab, we will be studying the Ashokan water levels for the year 2018. It is available from *NYC Open Data*. Please follow these instructions to download the dataset:

1. Follow the link NYC Open Data Current Reservoir Levels.
2. Choose the **View Data** menu option (it will reload the page).



3. Filter data including only the year 2018. To do that:
    - Click the blue button **Filter**

- In its submenu **Filter**, click **Add a New Filter Condition**
- Choose **Date is between** `January 1, 2018` and `December 31, 2018`



4. Sort entries *by Date* in the *Ascending* order.
5. Export the data:
   - Click the light blue button **Export**
   - Choose **TSV for Excel** (it will produce a plain text data file in *tab-separated values* format).
   - Save obtained file `Current_Reservoir_Levels.tsv` on your hard drive.

Data format

Don't open the datafile in Excel (that can mess up its formatting). Instead, you can open it with your text editor (gedit).

The datafile is a plain text file whose first line is a header followed by rows of data. The entries in each row are separated by the *tab* symbol, hence the name of the file format: *TSV* (tab-separated-values). It is the most convenient format for reading by our C++ program.

Each row has five fields: *Date*, *Storage* (in billions of gallons) and *Elevation* (in feet) for the East basin and for the West basin of the reservoir:

```
Date            EastStorage    EastElevation    WestStorage    WestElevation
01/01/2018          59.94            574            32.67          574.33
01/02/2018          59.89          573.99           32.57          574.29
01/03/2018          59.89          573.97           32.44          574.24
01/04/2018          59.9           573.97           32.22          574.07
...
```

To read the datafile, we have to open an **input file stream** (represented by an object of type `ifstream`, here we called it `fin`):

```
ifstream fin("Current_Reservoir_Levels.tsv");
if (fin.fail()) {
```

```
    cerr << "File cannot be opened for reading." << endl;
    exit(1); // exit if failed to open the file
}
```

Remember that the first line in the file is a header line. We have to skip it before we get to process the actual data. We can do that by reading that line into a temporary variable that we can call `junk`:

```
string junk;        // new string variable
getline(fin, junk); // read one line from the file
```

After that, the file can be read line by line. The most idiomatic C++ way to read such well-formatted file until the end would be the following:

```
while(fin >> date >> eastSt >> eastEl >> westSt >> westEl) {
    // this loop reads the file line-by-line
    // extracting 5 values on each iteration

    fin.ignore(INT_MAX, '\n'); //skips to the end of line,
                          //ignorring the remaining columns

    // for example, to print the date and East basin storage:
    cout << date << " " << eastSt << endl;
}
```

Here, variable `date` can be of type `string`, and the others are numeric variables of type `double` extracting the storage and elevation in East and West basins.

After you are done reading the file, close the stream:

```
fin.close();
```

Need to include additional header files

The above code is using a new function `exit` and a stream class `ifstream`. To make them work, we have to include two new **headers** at the beginning of the program:

```
#include <fstream>
#include <cstdlib>
#include <climits>
```

Task A

Write a program `east-storage.cpp` that asks the user to input a **string** representing the date (in MM/DD/YYYY format), and prints out the **East basin storage** on that day.

Example

```
$ ./east-storage
Enter date: 05/20/2018
East basin storage: 80.96 billion gallons
```

Task B. Minimum and maximum storage in 2018

Write a program `minmax.cpp` that finds the minimum and maximum storage in East basin in 2018.

Example (using made up numbers):

```
$ ./minmax
minimum storage in East basin: 59.88 billion gallons
MAXimum storage in East basin: 81.07 billion gallons
```

Hint:

The program should read the file line by line, while keeping track of what is the highest and the lowest storage level in the basin so far. In the end, after reading the entire file, the found values will be the minimum and the maximum storage levels for the entire year.

Task C. Comparing elevations

Write a program `compare.cpp` that asks the user to input two dates (the beginning and the end of the interval). The program should check each day in the interval and report **which basin had higher elevation** on that day by printing "**East**" or "**West**", or print "**Equal**" if both basins are at the same level.

Example:

```
$ ./compare
Enter starting date: 09/13/2018
Enter ending date: 09/17/2018

09/13/2018 West
09/14/2018 West
09/15/2018 West
09/16/2018 West
09/17/2018 West
```

Explanation:

Date	East (ft)	West (ft)	
09/13/2018	581.94	**582.66**	West is higher
09/14/2018	581.8	**582.32**	West is higher
09/15/2018	581.62	**581.94**	West is higher
09/16/2018	581.42	**581.55**	West is higher
09/17/2018	581.16	**581.2**	West is higher

Task D. Reverse chronological order

Write a program `reverse-order.cpp` which asks the user to input two dates (earlier date then later date). The program should report the **West basin elevation** for all days in the interval in the reverse chronological order (from the later date to the earlier).

Example:

```
$ ./reverse-order
Enter earlier date: 05/29/2018
Enter later date: 06/02/2018

06/02/2018  590.32 ft
06/01/2018  590.26 ft
05/31/2018  590.24 ft
05/30/2018  590.23 ft
05/29/2018  590.22 ft
```

Hint: If for the previous tasks you did not use arrays, here you really have to read the data into arrays first, and only then report them in the required order.