File I/O and Data Processing

Topics: I/O streams, fstream, data storage
Lab 3: https://maryash.github.io/135/labs/lab_03.html

- Sadab Hafiz

Streams in C++

In cpp, there are objects known as streams that allow input and output. Chances are, you've already used streams for user input and standard output (I/O).

The <iostream> library provides cin and cout. As you already know, cin is an input stream and cout is an output stream. These are standard which means the input data comes from the terminal and the output data is outputted to the terminal.

```
#include <iostream>                            // iostream library required for standard I/O

using namespace std;

int main(){

     int input;                                // variable that will be used to store input

     cin >> input;                             // standard user input

     cout << input << endl;                    // standard output

}
```

Stream I/O operators

The input and output streams have different operators. The input operator shows that the data is going from the input stream to a variable (stream >> variable):

```
int input;                         // variable that will be used to store input

cin >> input;                      // standard user input
```

The output operator shows that the data is going to the output stream. The standard output stream will output to the terminal (stream << data):

```
cout << “Hello, World!” << endl;      // standard user output
```

These operators can be used with a single stream to do multiple I/O operations:

```
cin >> variable1 >> variable2 >> variable3 >> variable4;

cout << “Output1” << “Output2” << endl;
```

File stream

What if we wanted to read and write using a file? In such cases, <fstream> library is useful. Similar to cin and cout, <fstream> provides `ifstream` and `ofstream` for file input and output.

We know that cin and cout by default is ‘connected’ to the terminal for I/O. In contrast, `ifstream` and `ofstream` objects need to be ‘connected’ to the file through initialization. The stream objects must be initialized before they can be used:

```
ifstream fin("input_filename.txt");          // initialize ifstream object named ‘fin’
ofstream fout(“output_filename.txt”);        // initialize ofstream object named ‘fout’
```

Similar to variables, the ifstream and ofstream objects can be named whatever you like. Needless to say, the file that you are trying to use must exist in the same directory as your cpp file.

File input example

The operators for ifstream and ofstream are the same same cin and cout. However the streams must be closed after you’re done using them. Usage:

```
#include <fstream>                            // fstream library required for file I/O

using namespace std;

int main(){

     ifstream fin(“input.txt”);               // initialize ifstream to read from “input.txt” file

     int input;                               // variable that will be used to store input

     fin >> input;                            // read an integer from “input.txt” file

     fin.close();                             // we don’t need the file anymore so close the stream

}
```

Learn more: https://www.tutorialspoint.com/cplusplus/cpp_files_streams.htm

Handling error while working with fstreams

There may be many reasons why input from a file can fail. In order to catch such failures, use the following example provided in lab 3:

```
ifstream fin("Current_Reservoir_Levels.tsv");                       // initialize ifstream

if (fin.fail()) {                                                   // check for failure

     cerr << "File cannot be opened for reading." << endl;          // output failure message

     exit(1);                                                       // exit program

}
```

The above code requires the following libraries:

```
#include <cstdlib>

#include <climits>
```

The getline() function

While reading from a file, the input operator (>>) only reads until it encounters a whitespace character. Whitespace characters include spaces, tabs etc.

However, there may be times when you may find it better to read the whole line from the file. In such cases use the getline() function. Getline takes an input stream(cin or ifstream) and a string variable where the single line read is stored. Example from the lab:

```
ifstream fin(“input.txt”);                        // initialize fstream

string junk;                                      // new string variable

getline(fin, junk);                               // read one line from the file and store in `junk`
```

In this case, getline is used to eat up a single line from the ifstream that we don’t have any use for. Combined with a while loop, you can read an entire file line by line using getline().

Using arrays to store input data

Last week we talked about arrays. While this week's program can be done without arrays, I highly recommend using arrays to make your lives easier.

Create five arrays to store the Date, EastStorage, EastElevation, WestStorage, WestElevation respectively:

```
string dateArr[365];

double eastStorageArr[365];

double eastElevationArr[365];

double westStorageArr[365];

double westElevationArr[365];
```

You can populate these arrays with the data from the input file.

Visualizing Data Table (with arrays)

By storing the data into arrays, we can access all the values of a particular date based on array indexes. You can visualize it in your mind something like this:

Array Index	dateArr	eastStorageArr	eastElevationArr	westStorageArr	westElevationArr
0	01/01/2018	59.94	574	32.67	574.33
1	01/02/2018	59.89	573.99	32.57	574.29
2	01/03/2018	59.89	573.97	32.44	574.24
...	...	...	...	...	...
364	12/31/2018	74.66	584.75	43.09	587.96

Reading the file using while-loop

We only care about the first five values in each line. Using a while-loop, ifstream and variables, you can read five values from each line for the entire file:

```
string date;                                           // string variable to store date

double eastSt, eastEl, westSt, westEl;                 // double variables to store numeric values

// this loop reads the file line-by-line extracting 5 values on each iteration

while (fin >> date >> eastSt >> eastEl >> westSt >> westEl) {     // while reading 5 values from fin

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

    … store the input variables into arrays here

}

fin.close()

// Do whatever you want you the data stored in arrays
```