Object Oriented Methodology

Week – *6*, Lecture – *2*

Input-Output in C++

SAURABH SRIVASTAVA

VISITING FACULTY

IIIT LUCKNOW

Go back and revise some C :P

The basic concepts in C and C++ have major overlaps between the two languages

We discussed these topics during ITP in the previous semester

- Difference between Textual Data and Binary Data
- The idea of “opening” a file for reading or writing, and then “closing” it when we are done
- Opening files for text based access as well as opening file for byte based access
- The different modes, in which a file could be opened
- Accessing files sequentially, as well as making jumps (using the `ftell()` and `fseek()` functions)
- The use of `fprintf()` as an equivalent to `printf()` for writing textual data to files
- The use of `fscanf()` as an equivalent to `scanf()` for reading textual data from files
- The use of `fread()` to read binary data from a file
- The use of `fwrite()` to write binary data to a file

Please revise all these topics, if they are not fresh in you memory

I/O in C++

If you liked the way I/O is done in C, you can continue to use it

- Any C++ compiler will also support the C mode of I/O
- This means, you can always use `printf()` instead of `cout` !!

Although the I/O philosophy in C++ is fairly similar, there are some differences

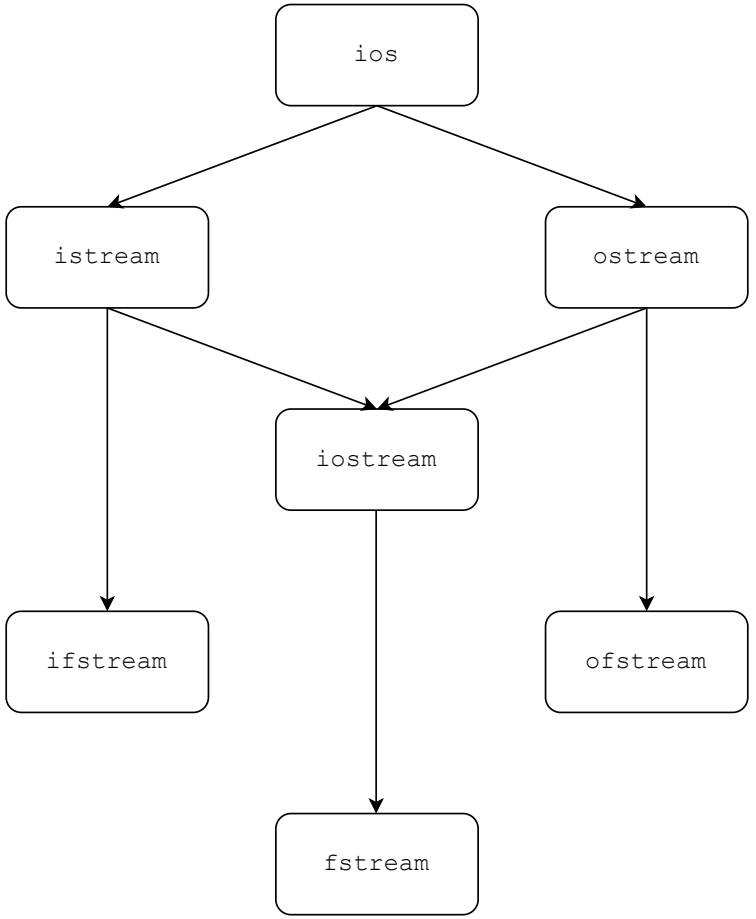
First, in C++, instead of a file pointer (like `FILE* ptr` in C), you create *streams*

- Streams can be considered as pipes – through which, data flows
- A stream can either be unidirectional, or bidirectional …
- … i.e., it may send data from a program to a destination, provide data from a source to a program, or both

Second, C++ uses inheritance to provide functionalities related to I/O

- This allows many common features to be available across a wide variety of use cases, without duplication

Major I/O Classes in C++



This is also a great example of how Inheritance can be used practically

Creating and using Streams

Based on your use case, you should first choose the correct stream

You have been using two streams very often anyhow – `cout` and `cin`

- They are of type `ostream` and `istream` respectively

If you want to perform I/O over files, you can choose a different set of streams

- The `ifstream` class is a derived class of `istream` for taking inputs from files
- The `ofstream` class is a derived class of `ostream` for outputting values to files
- The `fstream` (and it base class, `iostream`) can be used for both input as well as output w.r.t. to files

To open a stream, you can either pass the file name to the stream constructor, e.g.

- `ifstream in("file.txt");`

Or, you can use the `open()` method in the class, e.g.

- `ifstream in; in.open("file.txt);`

Stream modes

The `open()` method as well as the constructor, can be provided specific *mode* arguments

- These are very similar to the arguments provided to `fopen()`
- For example, the mode value for opening a file, and keeping the stream binary is:
  `ifstream in("file.dat", ios::in | ios::binary);`

There are modes to open files in read, write and append formats

- Check the Additional Reading section at the end for reference

After the operations, you should call the `close()` method on the stream to free the resources

The read/write methods

The inheritance hierarchy allows the file streams to be used similar to `cout` and `cin`

- So, this is perfectly fine:
  `ostream out("file.txt"); out<<"Hello World";`

There is almost a one-to-one mapping between functions in C and C++ for I/O

There are methods `read()` and `write()`, which are equivalent to `fread()` and `fwrite()`

For random access, you have `seek_p()` and `seek_g()` – equivalent to `fseek()`

- The two functions are for *write* and *read* positions respectively
- To know the current position within the stream, you can use `tell_p()` and `tell_g()`

Let us now see some code !!

Additional Reading

This short and simple reference probably contains everything for practical purposes

- http://www.cplusplus.com/doc/tutorial/files/

If you can catch hold of this book – *C++: The Complete Reference, by Herbert Schildt, 4th Edition*

- Read Chapters 20 and 21
- Chapter 32 can be used as a reference text