Streams and Files

"streams hierarchy, streams errors, file handling"

Fundamentals of OOPs

Shakirullah Waseeb
shakir.waseeb@gmail.com

Nangarhar University

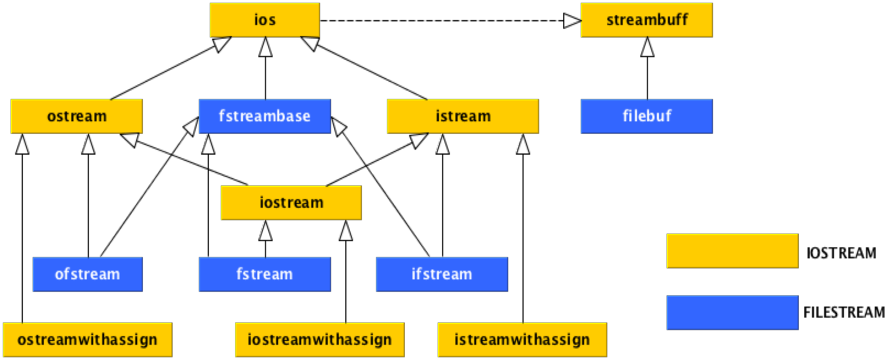
November 21, 2017

Agenda

Introduction

- **Stream:** a general name given to a flow of data
- In C++ streams are represented by an object of a particular class (cin and cout)
- Different streams are used to represent different kinds of data flow (e.g cin represents how data flow occurs between console and program)

Stream classes hierarchy



Classes Descriptions

- ios: base class for all stream classes
  - **Formatting Flags** (skipws, left, right, interal, dec, oct, hex, showbase, uppercase, showpos, scientific, unitbuf, stdio)
    Example: cout.setf(ios::left)
  - **Manipulators** formatting instructions directly into stream. without arguments (ws, dec, oct, hex, endl), with arguments (setw(), setfill())
  - **Functions** number of member functions can be used to set formatting flags and perform other tasks
    Example: ch = fill(); // returns the fill characters
    fill(ch); // set the fill character

Predefined Streams Objects

- cin: object of istream_withassign, used for keyboard input
- cout: object of ostream_withassign, used for screen display
  - cerr: an object of ostream_withassign for error messages
  - clog: an object of ostream_withassign for error messages
- Some error-status flags are (goodbit, eofbit, failbit, badbit, hardfail), functions for error flags are (int = eof(), int= fail(), int=bad(), int=good(), clear(ios::failbit))

Disk File I/O with streams

- Sometime it is required to save data to disk files and read it when demand
- Writing into files and reading from files requires another set of classes: ifstream for input, ofstream for output, and fstream for both input and output

Formatted I/O

- In formatted I/O, numbers are stored as characters
- Characters and strings are stored normally
- Writing data:
  - create an object of ofstream class
  - initialize the object to the file name (e.g somename.txt)
  - initialization sets aside various resources for the file, and open the file of given name on the disk
  - if file doesn't exists it will be created
  - use insertion operator to write variables of constant values of any basic type to file
  - when the program terminates, the object calls its destructor, which closes the file

Code Example

```
int x = 5;
float y=3.4;
string z = 'something';
ofstream ofile('formatted.txt')
ofile « x « ' ' « y « ' ' « z;
```

Formatted I/O -continue

- Reading data:
  - create an object of ifstream class
  - initialize to the file name (e.g somename.txt)
  - use extraction operator to read the written data

Code Example

```
int x;
float y;
string z;
ifstream ifile('formatted.txt')
ifile » x » y » z;
```

Strings with embedded blanks

- The previous approach will not work with char* strings containing embedded blanks
- This can be achieved by writing a specific delimiter character after each string
- And use getline() function, rather than the extraction operator while reading them

Code Snippet (writing)

```
ofstream ofile('embedded.txt') ;
ofile « "some string with embedded blanks \n";
ofile « "a quick brown fox jummped \n";
ofile « "over the lazy dog \n";
```

Code Snippet (reading)

```
char buffer[30];
ifstream ifile('embedded.txt');
while (ifile.good()){
  ifile.getline(buffer,30);
  cout « buffer;
}
```

Binary I/O

- We can also write data into file in binary format as it is in memory
- Two functions: write(); member function of ofstream, and read(); member function of ifstream
- The data is treated in terms of bytes (type char)
- The data is transferred as a buffer full of bytes from and to a disk file
- Parameters to write() and read() are the address of the data buffer and its length
- Address must be cast, using reinterpret_cast, to type char*, and the length is the length in bytes, not the number of data items in the buffer

Binary I/O -continue

Code Snippet (writing)

```
ofstream out('binary.dat', ios::binary);
int size=3;
int obuff[] = 1,2,3,4,5;
int ibuff[size];
out.write(reinterpret_cast<char*>(obuff), size* sizeof(int));
out.close();


ifstream in('binary.dat', ios::binary);
out.read(reinterpret_cast<char*>(ibuff), size* sizeof(int));
for (int x=0; x¡size; x++){
  cout « "ibuff[«x«] : "«ibuff[x]«endl;
}
```

Your Turn: Time to hear from you!

[1]

[1] https://fensafitters.files.wordpress.com/2013/07/3d095.jpg

References

Robert Lafore
Object-Oriented Programming in C++, 4th Edition .
2002.

Piyush Kumar
Object oriented Programming (Using C++)
http://www.compgeom.com/ piyush/teach/3330