CSE 230
Intermediate Programming in C and C++
Stream Input/Output in C++

Fall 2017

Stony Brook University

Instructor: Shebuti Rayana

http://www3.cs.stonybrook.edu/~cse230/

Ref. Book: C How to Program, 8th edition by Deitel and Deitel

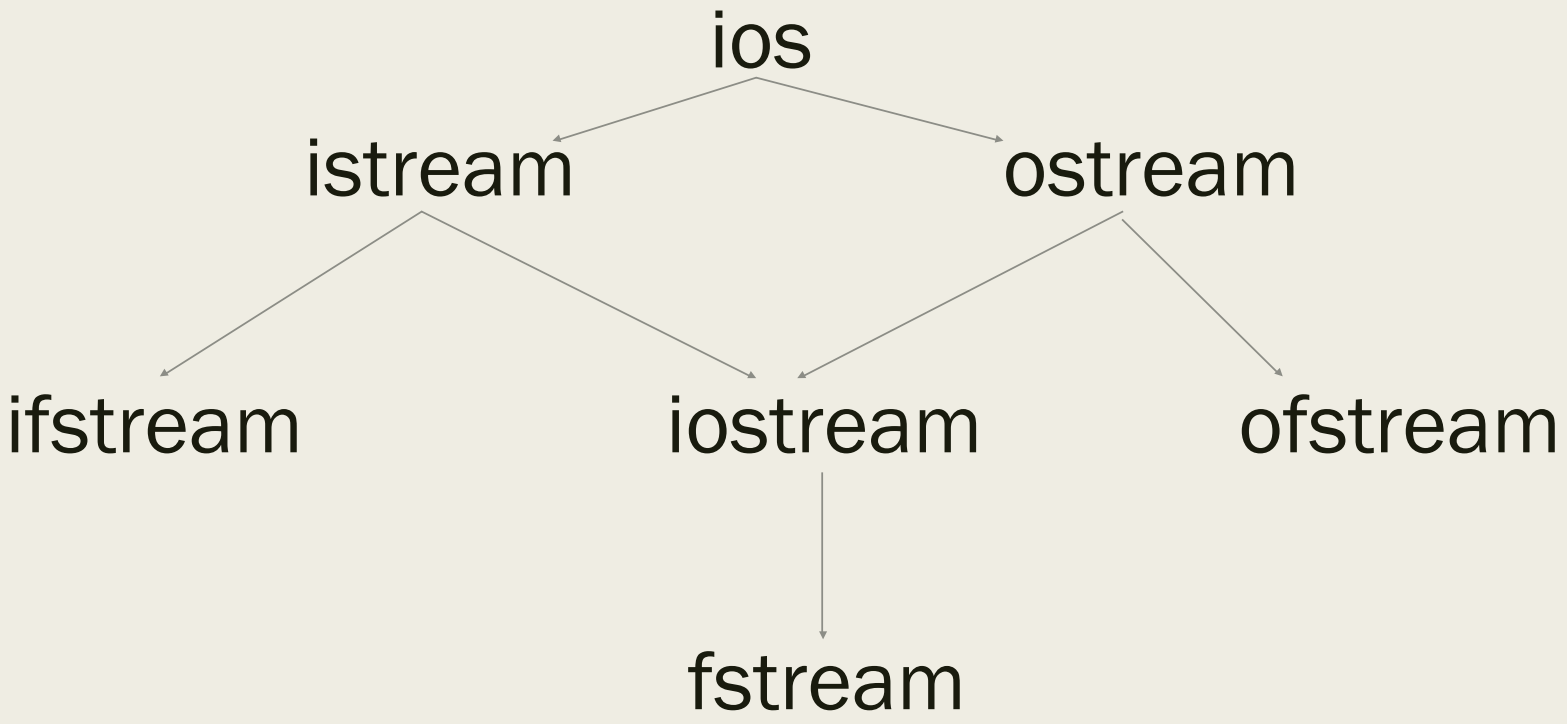
Stream Input/Output

- C++ standard libraries provide an extensive set of input/output capabilities.
- I/O occurs in *streams* of bytes.
  - A stream is simply a sequence of bytes. In input operations , the bytes flow from a device to main memory. In output operations, the bytes flow from main memory to a device.
- C++ provides both *low-level* (i.e. unformatted) and *high-level* (i.e. formatted) I/O capabilities.

Stream Input/Output

- Use the C++-style I/O exclusively in C++ programs, even though C-style I/O is available to C++ programmers.
- C++ I/O is type safe.
- C++ enables a common treatment of I/O for predefined types and user-defined types. This commonality facilitates software development and reuse.

Stream Input/Output Classes



Functions peek, putback and ignore

- *ignore* skips over a designated number of characters (default is one) or terminates upon encountering a designated delimiter (default is EOF), whichever comes first .
  Examples:

  cin.ignore(); cin.ignore(3); cin.ignore(25,' ');
- *peek* returns the next character from an input stream, but does not remove the character from the stream.
- *putback* places the previous character obtained by a *get* from input back onto the stream.

Unformatted I/O

- Performed with read and write member functions of istream and ostream, respectively.
- read inputs bytes to an built-in array of chars in memory.
- write outputs byes from a built-in array of chars.
- These bytes are not formatted. They are input output raw bytes.

Stream Manipulators

- C++ provides various *manipulators* that perform formatting tasks.
- *manipulators* provide capabilities such as:
  - setting field widths and precisions.
  - setting and unsetting format flags.
  - setting the fill character in fields.
  - flushing streams, inserting a new line in output
  - inserting a null character in the output and
  - skipping white space in the input stream.

Stream Format State Flags

- Various *format flags* specify the kinds of formatting to be performed during stream I/O operations.
- The *setf*, *unsetf* and *flags* member functions control the flag settings.
- The parameterized stream manipulators *setiosflags* and *resetiosflags* may be used instead of *setf* and *unsetf*.
- The bitwise-or operation, | , combines various options into a single long value.
- Calling the *flags* member function sets the options and returns a *long* value containing the prior options. The result value may be used to restore the previous stream options.
- *flags* function must specify settings of all the flags.

Example: Format State Flags

- ios::left left justify output. Padding appear to the right.
- ios::right right justify output. Padding appear to the left.
- ios::internal sign is left justified, magnitude is right-justified.
- ios::dec integers be treated as decimal values.
- ios::oct integers be treated as octal values.
- ios::hex integers be treated as hexadecimal values.
- ios::skipws skip whitespace character on an input stream.
- ios::fixed output floating number with given precision.
- ios::scientific output floating number in scientific notation.
- ios::showbase output the base, 0 for octal and 0x for hex.
- ios::showpoint output with a decimal point & trailing zeros.

Stream Error States

- The state of a stream may be tested through bits in class *ios*, the base class for input/output streams.
- The *eofbit* is set for an input stream after end-of-file is encountered. The call cin.eof() returns true if end-of-file has been encountered on *cin* and false otherwise.
- The *failbit* is set when a format error occurs on the stream. The call cin.fail() tests the status of this bit.
- The *badbit* is set when an error occurs that results in the loss of data. The call cin.bad() reports the status.
- The *goodbit* is set if none of the above bits are set. The call cin.goodbit() tests the status of this bit. I/O operations must only be performed on “good” streams.

Stream Error States (cont.)

- The *rdstate* function returns the error state of the stream, which could then be tested by a switch statement that examines *ios::eofbit*, *ios::badbit*, *ios::failbit*, *ios::goodbit*.
- The *clear* function is normally used to restore a stream's status to "good" so I/O may proceed on that stream.

```
cin.clear();      // default is to set the goodbit
cin.clear(ios::failbit);     // set failbit, not clear it
```