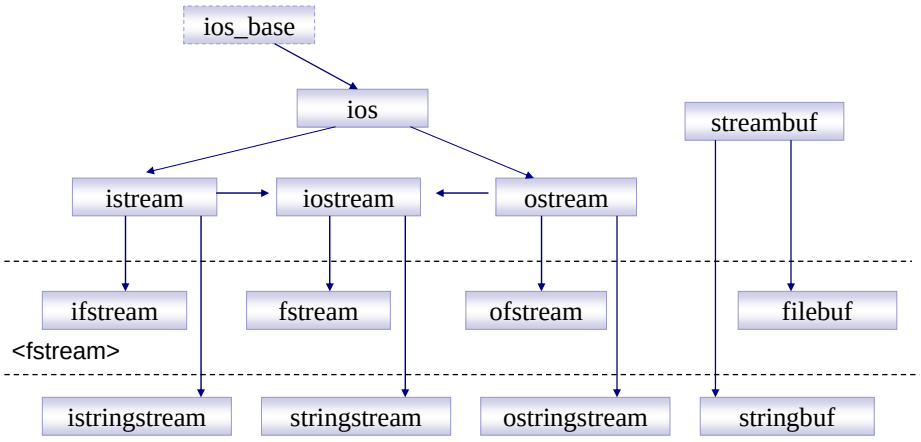
BBT

CPP

Iostream class

BBT

iostream hierarchy



ios

- Constructor:
  - **ios::ios *([streambuf* sb [, ostream* tie])***
- Destructor:
  - **ios::~ios *()***

ios methods

- **ios::operator void* () *const***
  - it is true if no failures have occurred (ios::fail is not true).
- **ios::operator !() *const***
  - the operator ! returns true if ios::fail is true (an operation has failed).
- **iostate ios::rdstate *() const***
  - Return the state flags for this stream

iostate

- **goodbit**
  - There are no indications of exceptional states on this stream.
- **eofbit**
  - End of file.
- **failbit**
  - An operation has failed on this stream; this usually indicates bad format of input.
- **badbit**
  - The stream is unusable

Test iostate methods

- **int ios::good *() const***
  - Test the state flags associated with this stream; true if no error indicators are set.
- **int ios::bad *() const***
  - Test whether a stream is marked as unusable. (Whether ios::badbit is set.)
- **int ios::eof *() const***
  - True if end of file was reached on this stream. (If ios::eofbit is set.)
- **int ios::fail *() const***
  - Test for any kind of failure on this stream: *either* some operation failed, *or* the stream is marked as bad. (If either ios::failbit or ios::badbit is set.)

Set iostate methods

- **void ios::setstate *(iostate state)***
  - Set the state flag for this stream to state in addition to any state flags already set.
- **void ios::clear *(iostate state)***
  - Set the state indication for this stream to the argument *state*. You may call ios::clear with no argument, in which case the state is set to good (no errors pending).

Choices in formatting

- **char ios::fill () const**
  - Report on the padding character in use.
- **char ios::fill *(char padding)***
  - Set the padding character
- **int ios::precision *() const***
  - Report the number of significant digits currently in use for output of floating point numbers. Default is 6.
- **int ios::precision *(int signif)***
  - Set the number of significant digits to signif
- **int ios::width *() const***
  - Report the current output field width setting , Default: 0, which means to use as many characters as necessary.
- **int ios::width *(int num)***
  - Set the input field width setting to *num*. Return the *previous* value for this stream. This value resets to zero (the default) every time you use `<<'

fmtflags

- **fmtflags ios::flags *() const***
  - Return the current value of the complete collection of flags controlling the format state.
- **ios::dec, ios::oct, ios::hex** (mask: **ios::basefield**)
  - What numeric base to use in converting integers from internal to display representation, or vice versa: decimal, octal, or hexadecimal, respectively.
- **ios::fixed ,ios::scientific** (mask: **ios:floatfield**)
  - Use fixed number of digits or scientific notation

Fmtflags (2)

- **ios::left,ios::right, ios::internal** (mask: **ios::adjustfield**)
  - Where output is to appear in a fixed-width field; left-justified, right-justified, or with padding in the middle respectively.
- **ios::showbase**
  - Display the conventional prefix as a visual indicator of the conversion base: no prefix for decimal, `0' for octal, `0x' for hexadecimal.
- **ios::showpoint**
  - Display a decimal point and trailing zeros after it to fill out numeric fields
- **ios::showpos**
  - Display a positive sign on display of positive numbers.

Fmtflags (3)

- **ios::skipws**
  - Skip white space. (On by default).
- **ios::stdio**
  - Flush the C stdio streams stdout and stderr after each output operation
- **ios::unitbuf**
  - Flush after each output operation
- **ios::uppercase**
  - Use upper-case characters for the non-numeral elements in numeric displays; for instance, `0X7A' rather than `0x7a', or `3.14E+09' rather than `3.14e+09'

Set fmtflags

- **fmtflags ios::flags *(fmtflags value)***
  - Set *value* as the complete collection of flags controlling the format state.
- **fmtflags ios::setf *(fmtflags flag)***
  - Set one particular flag ,return the complete collection of flags *previously* in effect.
- **fmtflags ios::setf *(fmtflags flag, fmtflags mask)***
  - Clear the flag values indicated by *mask*, then set any of them that are also in *flag*.
- **fmtflags ios::unsetf *(fmtflags flag)***
  - Make certain *flag* is not set for this stream;

manipulators

- **ws** Skip whitespace.
- **flush** Flush an output stream.
- **endl** Write an end of line character `\n', then flushes the output stream.
- **ends** Write `\0' (the string terminator character).

Need include <iomanip>

- **setprecision** *(int signif)*
- **setw** *(int n)*
- **setbase** *(int base)*
- **dec** equivalent to `setbase(10)'.
- **hex** equivalent to `setbase(16)'.
- **oct** equivalent to `setbase(8)'.
- **setfill** *(char padding)*

Synchronizing related streams

- ostream* **ios::tie** *() const*
  - Report on what output stream, if any, is to be flushed before accessing this one.
- ostream* **ios::tie** *(ostream* assoc)*
  - Declare that output stream assoc must be flushed before accessing this stream.

Reaching the underlying streambuf

- **streambuf* ios::rdbuf *() const***
  - Return a pointer to the streambuf object that underlies this ios.

class ostream

- **ostream::ostream *()***
- **ostream::ostream *(streambuf* sb [, ostream tie])***
- **ostream& ostream::put *(char c)***
  - Write the single character *c*.
- **ostream& ostream::write *(string, int length)***
  - Write *length* characters of a string to this ostream, beginning at the pointer *string*.
- ***ostream& ostream::form (const char *format, ...) (gnu ext.)***
  - *A GNU extension, similar to fprintf(file, format, ...).*
- ***ostream& ostream::vform (const char *format, va_list args)***

Repositioning an ostream

- **streampos ostream::tellp *()***
  - Return the current write position in the stream.
- **ostream& ostream::seekp *(streampos loc)***
  - Reset the output position to loc
- **ostream& ostream::seekp *(streamoff loc, rel)***
  - Reset the output position to *loc*, relative to the beginning, end, or current output position in the stream. *rel* can be:
  - beg
  - cur
  - end

Other utility

- **ostream& flush ()**
  - Deliver any pending buffered output for this ostream.

class istream

- **istream::istream *()***
- **istream::istream *(streambuf *sb [, ostream tie])***

Reading one character

- **int istream::get *()***
  - Read a single character (or EOF) from the input stream, returning it (coerced to an unsigned char) as the result.
- **istream& istream::get *(char& c)***
  - Read a single character from the input stream, into &c.
- **int istream::peek *()***
  - Return the next available input character, but *without* changing the current input position.

Reading strings

- **istream& istream::get *(char* c, int len [, char delim])***
  - Read a string from the input stream, into the array at c.
  - The remaining arguments limit how much to read: up to `len-1' characters, or up to (but not including) the first occurrence in the input of a particular delimiter character *delim*---newline (\n) by default.
  - get writes `\0' at the end of the string
- **istream& istream::getline *(charptr, int len [, char delim])***
  - Read a line from the input stream, into the array at *charptr*. *charptr* may be any of three kinds of pointer: char*, unsigned char*, or signed char*.
  - If getline succeeds in reading a "full line", it also discards the trailing delimiter character from the input stream.

Reading strings –cont.

- **istream& istream::read *(pointer, int len)***
  - Read len bytes into the location at pointer, unless the input ends first.
  - If the istream ends before reading *len* bytes, read sets the ios::fail flag.
- ***istream& istream::gets (char **s [, char delim]) (gnu ext.)***
  - A GNU extension, to read an arbitrarily long string from the current input position to the next instance of the delim character (newline \n by default).
- ***istream& istream::scan (const char *format ...) (gnu ext.)***
  - similar to fscanf(file, format, ...). The format is a scanf-style format control string, which is used to read the variables in the remainder of the argument list from the istream.

Repositioning an istream

- **streampos istream::tellg *()***
  - Return the current read position
- **istream& istream::seekg *(streampos p)***
  - Reset the input pointer (if the input device permits it) to *p*,
- **istream& istream::seekg *(streamoff offset, ios::seek_dir ref)***
  - Reset the input pointer (if the input device permits it) to *offset* characters from the beginning of the input, the current position, or the end of input.

istream utilities

- **int istream::gcount *()***
  - Report how many characters were read from this istream in the last unformatted input operation.
- **istream& istream::ignore *([int n] [, int delim])***
  - Discard some number of characters pending input. The first optional argument *n* specifies how many characters to skip. The second optional argument *delim* specifies a "boundary" character. By default, *delim* is EOF
  - If you do not specify how many characters to ignore, ignore returns after discarding only one character.

istream utilities (2)

- **istream& istream::putback *(char ch)***
  - Attempts to back up one character, replacing the character backed-up over by *ch*. Returns EOF if this is not allowed.
- **istream& istream::unget *()***
  - Attempt to back up one character.

class iostream

- If you need to use the same stream for input and output, you can use an object of the class iostream, which is derived from *both* istream and ostream.

Classes for Files and Strings

- ifstream
  - Methods for reading files.
- ofstream
  - Methods for writing files.
- istringstream
  - Methods for reading strings from memory.
- ostringstream
  - Methods for writing strings in memory.

Reading files -<fstream>

- **ifstream::ifstream ()**
  - Make an ifstream associated with a new file for input.
  - you need to call ifstream::open before actually reading anything
- **ifstream::ifstream (int fd) (not ANSI)**
  - Make an ifstream for reading from a file that was already open, using file descriptor fd.
- **ifstream::ifstream (const char* fname [, int mode [, int prot]])**
  - Open a file *fname for this ifstream object.
  - By default, the file is opened for input (with ios::in as *mode*)

modes

- ios::in
  - Open for input. (Included in ANSI draft.)
- ios::out
  - Open for output. (Included in ANSI draft.)
- ios::ate
  - Set the initial input (or output) position to the end of the file.
- ios::app
  - Seek to end of file before each write. (Included in ANSI draft.)
- ios::trunc
  - Guarantee a fresh file; discard any contents that were previously associated with it.

modes –cont.

- ios::nocreate
  - Guarantee an existing file; fail if the specified file did not already exist.
- ios::noreplace
  - Guarantee a new file; fail if the specified file already existed.
- ios::binary
  - Open as a binary file (on systems where binary and text files have different properties, typically how `\n' is mapped; included in ANSI draft).

Open file

- **void ifstream::open *(const char* fname [, int mode [, int prot]])***
  - Open a file explicitly after the associated ifstream object already exists (for instance, after using the default constructor). The arguments, options and defaults all have the same meanings as in the fully specified ifstream constructor.

Writing files

- **ofstream::ofstream ()**
- **ofstream::ofstream (int fd)**
- **ofstream::ofstream (const char* fname [, int mode [, int prot]])**
- **ofstream::~ofstream ()**
- **void ofstream::open (const char* fname [, int mode [, int prot]])**
- **void fstreambase::close ()**
  - Close the file associated with this object, and set ios::fail in this object to mark the event.
  - both ifstream and ofstream inherit this additional method

Reading and write in memory

- **istringstream (openmode mode = in);**
- **istringstream (const string & str, openmode mode = in);**
- **ostringstream(openmode mode = out);**
- **ostringstream (const string & str, openmode mode = out);**
- **stringstream(openmode mode = in|out);**
- **stringstream (const string & str, openmode mode = in|out);**

stringstream

- **string str ( ) const;**
  - The first syntax returns a copy of the string object currently associated with the internal buffer
- **void str (const string & s);**
  - sets a new value for the string object associated with the buffer.
- **stringbuf* strstreambase::rdbuf ()**

Using the streambuf Layer

- The istream and ostream classes are meant to handle conversion between objects in your program and their textual representation.
- By contrast, the underlying streambuf class is for transferring raw bytes between your program, and input sources or output sinks. Different streambuf subclasses connect to different kinds of sources and sinks.

- Areas
  - The **put area** contains characters waiting for output.
  - The **get area** contains characters available for reading.

PutArea operations (virtual protected)

- **char* streambuf::pbase *() const***
  - Returns a pointer to the start of the put area
- **char* streambuf::epptr *() const***
  - Returns a pointer to the end of the put area.
- **char* streambuf::pptr *() const***
  - If pptr() < epptr (), the pptr() returns a pointer to the current put position.
- **void streambuf::pbump *(int N)***
  - Add *N* to the current put pointer. No error checking is done.
- **void streambuf::setp *(char* P, char* E)***
  - Sets the start of the put area to *P*, the end of the put area to *E*, and the current put pointer to *P* (also).

GetArea operations (virtual protected)

- **char* streambuf::eback *() const***
  - Returns a pointer to the start of the get area
- **char* streambuf::egptr *() const***
  - Returns a pointer to the end of the get area
- **char* streambuf::gptr *() const***
  - If gptr() < egptr (), then gptr() returns a pointer to the current get position.
- **void streambuf:gbump *(int N)***
  - Add *N* to the current get pointer. No error checking is done.
- **void streambuf::setg *(char* B, char* P, char* E)***
  - Sets the start of the get area to *B*, the end of the get area to *E*, and the current put pointer to *P*.

Simple output re-direction by redefining overflow

- **int overflow (int *c* = EOF);**
  - Put character at current put position. The character whose value is parameter *c* is stored at current put position and the put pointer is increased by one.
  - This virtual protected function is called by sputc and sputn in case there is **no room** in the internal output sequence to successfully perform the output operation.
- **int sync ( );**
  - Synchronize stream buffer. return 0 OK, -1 error.

input re-direction by redefining underflow

- **int underflow ( );**
  - Returns the character at current get position, or EOF (traits::eof) if the current get pointer is at the end of the input sequence.
  - This virtual protected function is called by sgetc and other input member functions when the characters remaining to be read in the internal input buffer **have exhausted** (i.e., gptr>=egptr)
  - Return: The character available at the get position or EOF if the get pointer is at the end of the input sequence.