Small Changes to IOStreams

The following changes to Clause 27, Input/Output library, are minor changes˝affecting correctness of descriptions and specifications, and should all be non-controversial. The changes go beyond˝simple editorial corrections, however, since they affect semantics and function signatures, for example. All˝the changes listed here come from public comments, editorial boxes in the CD, or already-open issues of the˝Library working group.

Section	Action	Existing Text	Replacement Text
27	Change	Replace all occurrences of `win, wout, werr,` and `wlog` with `wcin, wcout, wcerr,` and `wclog` respectively.	
27.1.1	Add	To the definition of “character”...	The character class has a trivial constructor and destructor. It also has a bitwise copy that preserves its value and semantics.
27.1.2.5	Add		27.1.2.5 Type SZ_T A type that represents one of the signed basic intregal types. It is used to represent the number of characters transferred in an I/O operation, or the size of I/O buffers.
27.1.2.6	Add		27.1.2.6 Type STATE_T `STATE_T` is an implementation-defined value-oriented type. It holds the conversion state, and is compatible with the function `locale::codecvt().`
27.3	Delete	Remove the “include <fstream>“ from 27.3 synopsis	
27.3.1	Change	Paragraph 2 says, Ater the object cin is initialized, cin.tie() returns cout.	After the object cin is initialized, cin.tie() returns &cout
27.3.2	Change	Paragraph 2 says, “After the object wcin is initialized, wcin.tie() returns wcout”.	After the object wcin is initialized, wcin.tie() returns &wcout.
27.4, 27.4.1	Change	INT_T streamsize	SZ_T streamsize
27.4, 27.4.1	Change	Move `streamoff` and `streampos` to Appendix D.2 (note that `streampos` is missing from the 27.4 synopsis)	
27.4, 27.4.1	Delete	Remove `wstreamoff`	
27.4.2	Change	`typedef` ***To be specified*** `state_type`	typedef `STATE_T state_type`
27.4.2.1	Delete	Section to 27.1.2.6, `ios_traits` types	
27.4.2	Change		Paragraph 2 currently says “an implementation shall provide” instantiations of `ios_traits<char>` and `ios_traits<wchar_t>;` this is correct, however this change was made without approval so we need to approve it now.
27.4.2.2	Add		In `not_eof():` “Effects: For specializations on type char to_int_type widens the char to int without sign-extending. For other types, it performs an implementation defined conversion. “
27.4.2.2	Change	In `length():` “Effects: Determines ...”	“Returns: ...”

27.4.2.4	Change	In `to_char_type()`: "Effects: Converts ..."	"Returns: ..."
27.4.2.4	Change	In `to_int_type()`: "Effects: Converts ..."	"Returns: ..."
27.4.2.4	Add		In copy(): "Returns: `dst`"

27.4.3	Change	`int precision() const;`
		`int precision(int` *`prec`*`);`
		`int width() const;`
		`int width(int` *`wide`*`);`
			`streamsize precision() const;`
			`streamsize precision(streamsize` *`prec`*`);`
			`streamsize width() const;`
			`streamsize width(streamsize` *`wide`*`);`)

27.4.3.1.6	Add		In the description of the Init destructor, it also calls wcout.`flush(), wcerr.flush(),` and `wclog.flush().`

27.4.3.2	Change	`int precision() const;`
		`int precision(int` *`prec`*`);`
		`int width() const;`
		`int width(int` *`wide`*`);`
			`streamsize precision() const;`
			`streamsize precision(streamsize` *`prec`*`);`
			`streamsize width() const;`
			`streamsize width(streamsize` *`wide`*

27.4.3	Change	In synopsis: `int* iarray`	`long* iarray`
27.4.3	Change	Synopsis, move `fill` to `basic_ios`, 27.4.4	
27.4.3.2	Change	Move `fill` to `basic_ios`, 27.4.4.4	
27.4.3.2 Footnote 212)	Change	`Integer array`	`long array`
27.4.3.4	Change	In `iword`(): allocates an array of `int`	allocates an array of `long`
27.4.3.4	Delete	In `iword`(): remove the Notes	
27.4.3.4	Add		In `iword()` Effects add: "The reference returned may become invalid after another call to iword with a different index after a copyfmt or when the object is destroyed."
27.4.3.4	Delete	In `pword()`: remove the Notes	
27.4.3.4	Add	In `pword`() effects section	The reference returned may become invalid after another call to `pword` with a different index after a copyfmt or when the object is destroyed.
27.4.3.5	Change	Effects: Constructs an object of class `ios_base`, assigning initial values to its member objects. The postconditions of this function s are indicated in Table 72.	Effects: The initialization of ios_base members are initializated in an unspecified way by basic_ios::init. The ios_base constructor does no initialization.
27.4.3.5	Move	Table 72 moved to `basic_ios`	Move Table 72 to 27.4.4.1
27.4.4.1	Add		Make Table 72-ios_base() effects index unspecified.
27.4.4.1	Add	"**Effects**: `basic_ios`, assigning initial values..."	**Effects**: Constructs an object of class `basic_ios`, leaving its member objects uninitialized. The object shall be initialized by calling `init ( basic_streambuf *sb)` before it is destroyed.

27.4.4.1	Add	**Effects:** `basic_ios::init(basic_streambuf *sb)` has no semantics	**Effects**: The `f`unction ensures the initializations described in Table 72 happen in an implementation-defined manner. The object shall be initialized by calling `init(basic_streambuf *sb)` before the object is destroyed. **Postconditions**: `rdbuf() == sb, tie() == 0, ios_base` initialized according to Table 72.
27.4.4.2	Change	`Basic_ios::tie` **Returns**: An output sequence that is *tied* to (synchronized with) an input sequence controlled by the stream buffer.	`Basic_ios::tie` **Returns**: An output sequence that is *tied* to (synchronized with) the sequence controlled by the stream buffer
27.4.4.2	Change	`basic_ios::imbue` **Effects:** Calls `ios_base::(loc)` (27.4.3.3) and `rdbuf()-> pubimbue (loc)`	`basic_ios::imbue` **Effects:** Calls `ios_base::imbue(loc)` (27.4.3.3) and if `rdbuf() != NULL` then `rdbuf()-> pubimbue(loc)`. Returns: `ios_base::imbue(loc)`.
27.4.4.2	Change	`basic_ios::copyfmt` **Effects:** ... — `rdstate()` is left unchanged ...	basic_ios::copyfmt Effects: ... — rdbuf() and rdstate() are left unchanged ...
27.5.2	Change	`int in_avail();`	`streamsize in_avail();`
27.5.2	Change	`int sgetn(char_type* s, streamsize n);`	`streamsize sgetn(char_type* s, streamsize n);`
27.5.2.2.3	Change	`int in_avail();` **Returns:** If a read position if available, returns `gend() - gnext()`. Otherwise, returns `showmanyc`(). (27.5.2.4.3).	`streamsize in_avail();` **Returns:** If a read position if available, returns `egptr() - gptr()`. Otherwise, returns `showmanyc`(). (27.5.2.4.3).
27.5.2.2.3	Change	`int_type sbumpc();` **Returns**: If the input sequence read positions if not available, returns `uflow()`. Otherwise, returns `char_type (*gptr())` and increments the next point for the input sequence.	`int_type sbumpc();` **Returns**: If the input sequence read positions if not available, returns `uflow()`. Otherwise, returns `*gptr()` and increments the next point for the input sequence.
27.5.2.2.3	Change	`int_type gsetc();` **Returns**: If the input sequence read positions if not available, returns `underflow()`. Otherwise, returns `char_type (*gptr()).`	`int_type gsetc();` **Returns**: If the input sequence read positions if not available, returns `underflow()`. Otherwise, returns `*gptr().`
27.5.2.2.3	Change	`int sgetn(char_type* s, streamsize n);`	`streamsize sgetn(char_type* s, streamsize n);`
27.5.2.2.5	Change	`int sputc(char_type c);` **Returns**: If the input sequence read positions if not available, returns `overflow(c)`. Otherwise, stores `c` at the next pointer for the output sequence, increments the ponter, and returns `*pptr().`	`int_type sputc(char_type c);` **Returns**: If the input sequence read positions if not available, returns `overflow(c)`. Otherwise, stores `c` at the next pointer for the output sequence, increments the ponter, and returns `traits::to_int_type(c).`
27.5.2.2.5	Change	`int_type sputn(const char_type* s, streamsize n);`	`streamsize sputn(const char_type* s, streamsize n);`

27.5.2.4.2	Change	`int sync();` **Effects:** Synchronizes the controlled sequences with the arrays. That is, if `pbase()` is non-null, the characters between `pbase()` and `pptr()` are written to the controlled sequence, and if `gptr()` is non-null, the characters between `gptr()` and`egptr()` and restored to the input sequence. The pointers may then be reset as appropriate.	`int sync();` **Effects:** Synchronizes the controlled sequences with the arrays. That is, if `pbase()` is non-null, the characters between `pbase()` and `pptr()` are written to the controlled sequence. The pointers may then be reset as appropriate.
27.5.2.4.3	Change	Footnote 217) The morphemes of `showmany` are ...	Footnote 217) The morphemes of `showmanyc` are ...
27.5.2.4.3	Change	`showmanyc` **Returns:** a guaranteed lower bounds on the number of characters that can be read from the input sequence before a call to `uflow()` or `underflow()` returns `traits::eof()`. A positive return value of indicated that the next such call will not return traits::eof().	[This came from Editorial Box 146] `showmanyc` **Returns:** Returns an estimate of the number of characters available in the sequence, or -1. If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at leasts that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.
27.5.2.4.3	Change	`underflow` **`Returns:`** `traits::eof()` to indicate failure.	`underflow` **Returns:** consolidate with first **Returns** clause.
27.5.2.4.4	Change	`int pbackfail(int` *`c`* `= traits::eof());`	`int_type pbackfail(int_type` *`c`* `= traits::eof());`
27.6	Change	`<iomanip>` synopsis has includes for `<istream>` and `<ostream>`	replace with include for `<ios>`
27.6	Change	`<iomanip>` does *not* define a single type `smanip`.	Change `smanip` to type `T1, T2, ...` Each manipulator has an optionally unique return type. It is unspecified what the return type is
27.4.2 synopsis	Change	static int_type not_eof(char_type c);	static int_type not_eof(int_type c);
27.4.2 synopsis	Change	static bool is_whitespace(const ctype<char_type> ctype&, char_type c);	static bool is_whitespace(int_type c, const ctype<char_type> ct&);
27.4.2.2	Change	Returns: a value other than the end-of-file, even if c==eof(). Returns: int_type(c) if c!=eof().	Returns: c, if c != eof(), otherwise an unpsecificed value!= eof().
27.4.2.3	Change	static bool is_whitespace(char_type c, const ctype<char_type> ctype&); Returns: true if c represents a whitespace character. The default definition is as if it returns ctype.isspace(c). An implementation of the iostream class templates ... provided from the base struct string_char_traits<CHAR_T>.	static bool is_whitespace(int_type c, const ctype<char_type> ct&); Returns: true if c represents a whitespace character. The default definition is as if it returns ct.is(ct.space, c). [last sentence deleted, since there is no base struct]

27.5.2 Synopsis	Change	int sungetc();	int_type sungetc();
27.5.2 Synopsis also 27.5.2.2.4	Change	int sputc(char_type c);	int_type sputc(char_type c);
27.5.2 Synopsis	Change	int_type sputn(const char_type* s, streamsize n);	streamsize sputn(const char_type* s, streamsize n);
27.6.1.1 Synopsis	Change	... seekg(pos_type&); ... seekg(off_type&, iosbase::seekdir);	... seekg(const pos_type&); ... seekg(const off_type&, iosbase::seekdir);
27.6.1.1 pp 2	Delete	They may use other public members of istream except that they do not invoke any virtual members of rdbuf() except uflow().	They may use other public members of istream.
27.6.1.1.2	Change	Notes: ... uses the function bool traits::is_whitepace(charT, const locale*) ...	Notes: ... uses the function bool traits::is_whitepace(charT, const ctype<charT>&) ...

27.6.1.2.1 pp 3	Change sample code	if ((TYPE)tmp != tmp) { // set fail bit
		} else val = (TYPE) tmp;
			if(<tmp can be safely converted to TYPE>)
			  val = (TYPE)tmp;
			else // set fail bit

27.6.1.2.2	Change	[description of basic_istream::operator>>(char_type*)] If width() is greater than zero, the maximum number of characters stored n is width(); otherwise it is numeric_limits<int>:max().	If width() is greater than zero, the maximum number of characters stored, n, is width(); otherwise n is the size of the largest array of char_type that can also store the terminating null.
27.6.1.2.2	Change	[description of basic_istream::operator>>(basic_streambuf<charT, traits>*)] If sb is null, calls setstate(badbit) which may throw ... - an exception occurs (in which case the exception is caught). setstate(badbit) is not called.	If sb is null, calls setstate(failbit) which may throw ... - an exception occurs (in which case the exception is caught).
27.6.1.3	Change	[description of basic_istream::get(basic_streambuf& sb, char_type)] Effects: Extracts characters and inserts them in the output sequence controlled by rdbuf(). ...	Effects: Extracts characters and inserts them in the output sequence controlled by sb. ...

27.6.1.3	Change	[description of basic_istream::readsome(char_type* s, streamsize n)] Effects: Extracts characters and stores them into successive locations of an array whose first element is designated by s. Returns: A value based on in_avail(): - If in_avail() < 0, calls setstate(eofbit) (which may throw ios_base::failure (27.4.4.3)), and returns zero; - If in_avail() == 0, returns 0; - If in_vail() > 0, returns read(s, min(in_avail(), n)).	Effects: Extracts characters and stores them into successive locations of an array whose first element is designated by s. - If rdbuf()->in_avail() == -1, calls setstate(eofbit) (which may throw ios_base::failure (27.4.4.3)), and extracts no characters; - If rdbuf()->in_avail() == 0, extracts no characters; - If rdbuf()->in_avail() > 0, extracts rmin(rdbuf()->in_avail(), n) characters. Returns: The number of characters extracted.
27.6.1.3	Change	[description of basic_isream::putback] Effects: Calls rdbuf->sungetc(). If that function returns traits::eof(), calls setstate(badbit) (which may throw ios_base::failure (27.4.4.3)).	Effects: If rdbuf() is not null, calls rdbuf->sungetc(). If rdbuf() is null, or if sungetc() returns traits::eof(), calls setstate(badbit) (which may throw ios_base::failure (27.4.4.3)).
27.6.1.3	Change	[description of basic_istream::unget] Effects: Calls rdbuf->sungetc(). If that function returns traits::eof(), calls setstate(badbit) (which may throw ios_base::failure (27.4.4.3)).	Effects: If rdbuf() is not null, calls rdbuf->sungetc(). If rdbuf() is null, or if sungetc() returns traits::eof(), calls setstate(badbit) (which may throw ios_base::failure (27.4.4.3)).
27.6.1.3	Change	[description of basic_istream::sync] Effects: If rdbuf() is a null pointer, returns traits::eof(). Otherwise, calls rdbuf()->pubsync() and, if that function returns traits::eof(), calls setstate(badbit), (which my throw ios_base::failure (27.4.4.3), and returns traits::eof(). Otherwise, returns zero. Notes: Uses traits::eof().	Effects: If rdbuf() is a null pointer, returns -1. Otherwise, calls rdbuf()->pubsync() and, if that function returns -1, calls setstate(badbit), (which my throw ios_base::failure (27.4.4.3)), and returns -1. Otherwise, returns zero. [Notes deleted]
27.6.2.1 Synopsis	Change	... seekp(pos_type&); ... seekp(off_type&, iosbase::seekdir);	... seekp(const pos_type&); ... seekp(const off_type&, iosbase::seekdir);
27.6.2.1 pp 2	Change	If the called function throws an exception, the output function calls setstate(badbit), which may throw ios_base::failure (27.4.4.3), and if badbit is on in exceptions() rethrows the exception.	If the called function throws an exception, the output function calls setstate(badbit), which may throw ios_base::failure (27.4.4.3), in which case it rethrows the exception.
27.6.2.4.1	Change	Tables are not notes, they are normative	
27.6.2.4.1	Change	Table 76 needs a more descriptivetitle, e.g. "Numeric conversions"	

27.6.2.4.2 Paragraph 1	Change	`istreambuf_iterator`	`ostreambuf_iterator`
27.6.2.4.1 Table 77	Change		This was a change made at the Austin meeting without approval. It is correct in the current draft.
27.6.2.4.2	Add		In function `basic_ostream::op<< (basic_streambuf *sb)` Add to Effects: "If *sb* is a null pointer, calls `setstate(badbit)` (which may throw `ios_base::failure` (**27.4.4.3**))."
27.6.2.5 Paragraph 3	Change	"If that function returns `traits::eof()`"	"If that function returns -1"
27.7	Delete	Table 77 <cstdlib> remove entire table	
27.7 Synopsis	Change	`template<class charT, class traits = int_charT_traits...`	`template<class charT, class traits = ios_traits...`
27.7.1.1	Delete		In function `basic`_streambuf constructor **Effects:** Delete second instance of "initializing the base class with `basic_streambuf()`"
27.7.1.3	Change	in `underflow` "Returns: `char_type(*gptr())`"	"Returns: `int_type(*gptr())`"
27.7.1.3	Change	In `pbackfail` **Effects**: "Returns: `char_type(c)`"	**Effects:** "Returns: `c`"
27.7.1.3	Change	In `pbackfail` **Effects:** "If `c == traits::eof()`... Returns: `c`"	**Effects:** "If `c == traits::eof()`... Returns: `traits::not_eof(c)`"
27.8.1.1 Paragraph 3	Delete	"If the file is not open for reading or for update"	Delete "for update": "If the file is not open for reading"
27.8.1.1 Paragraph 3	Delete	"If the file is not open for writing or for update"	Delete "for update": "If the file is not open for writing"
27.8.1.3 Table 83	Delete		Remove the second entry for "`in | out`"
27.8.1.3 Table 83	Change		Table 83 is currently a "Note". Keep the table and associated text, but it should not be a Note.
27.8.1.3	Change	"Effects: If `is_open == false`, returns a null pointer. Otherwise, calls basic_streambuf<charT, traits>::basic_streambuf(). It then opens... as if by calling `::fopen`	`basic_filebuf::open` "Effects: If `is_open == true`, returs a null pointer. Otherwise, initializes the filebuf as required. It then opens...as if by calling `::fopen`
27.8.1.4	Delete	Remove "**Requires:** `is_open() == true`" from `showmanyc`	
27.8.1.4	Delete	Remove "**Requires:** `is_open() == true`" from `underflow`	
27.8.1.4	Delete	Remove "**Requires:** `is_open() == true`" from `pbackfail`	
27.8.1.4	Delete	Remove "**Requires:** `is_open() == true`" from `overflow`	
27.8.1.4	Delete	Remove "**Requires:** `is_open() == true`" from `seekoff`	
27.8.1.4	Delete	Remove "**Requires:** `is_open() == true`" from `seekpos`	

27.8.1.4	Change	"**Returns:** `traits::eof()` to indicate failure, otherwise `c`." from `pbackfail`	"**Returns:**`traits::eof()` to indicate failure, otherwise `not_eof(c)`."
27.8.1.4	Delete	Remove **Default Behavior:** clause from `pbackfail`	
27.8.1.4	Change	"**Returns:** `traits::eof()` to indicate failure." from function `pbackfail`	"**Returns:** `traits::eof()` to indicate failure, otherwise returns some value other than `traits::eof()` to indicate success."
27.8.1.4	Delete	Remove **Effects:** clause from `imbue`. The paragraph doesn't apply to anything	
27.8.1.7	Delete	Remove the `explicit` qualifier for `rdbuf()`	
27.8.1.8	Change	"`explicit basic_ofstream(const char* s, openmode mode = out)`"	"`explicit basic_ofstream(const char* s, openmode mode = out | trunc)`"
27.8.1.9	Change	"`explicit basic_ofstream(const char* s, openmode mode = out);`"	"`explicit basic_ofstream(const char* s, openmode mode = out | trunc);`"
27.8.1.10	Change	"`void open(const char* s, openmode mode = out);`"	"`void open(const char* s, openmode mode = out | trunc);`"
27.8.2 Table 84 <cstdio>	Change	Change second `vprintf` to `vfprintf`	
27.8.2 Table 84 <cstdio>	Delete	Delete duplicate entry for `tmpfile`	
27.8.2 Table 84 <cstdio>	Add		Add entry for `putchar`
27.8.2 Table 84 <cwchar>	Delete	Delete this table. It is already duplicatd in clause 21 table 44	
27.8.2	Delete	Remove reference to `<cwchar>`	
27 Table 65	Delete	Remove reference to `<cwchar>`	
27.8.2 Paragraph 2	Delete	Delete this paragraph	
D.3.1.3	Change	`eof()`	Change all references of `eof()` to `EOF`
D.3.1.3	Change	`overflow` "Returns `(char)`*c*."	"Returns `(unsigned char)`*c*."
D.3.1.3	Change	`pbackfail` "Returns `(char)`*c*."	"Returns `(unsigned char)`*c*."
D.3.1.3	Delete	`setbuf` Remove **Default behavior:** clause	

D3.1.3	Change	`underflow`
		--If the input... returning
		`(char* )gnext`
		-- otherwise... returns
		`(char)*gnext`
			--If the input...returning
			`(unsigned char)*gnext`
			--otherwise... returns
			`(unsigned char)*gnext`