iostream actions at Stockholm
Jerry Schwarz

The iostream working group met in Stockholm and recommends the following actions with regards to open issues

27-607: void* inserter and extractor

There should be members of num_get and num_put to control formatting of pointers, and the extractor and inserter for void* should indicate that these are used.

Add public member to class num_get in \[lib.locale.num.get\]

\[
\text{iter\_type get(iter\_type in, iter\_type end, ios\_base\&,}
\]

\[
\text{ios\_base::iostate\& err, void* p) const;}
\]

Add protected member to class num_get in \[lib.locale.num.get\]

\[
\text{virtual iter\_type do\_get(iter\_type in, iter\_type end, ios\_base\&,}
\]

\[
\text{ios\_base::iostate\& err, void* p) const;}
\]

Add above variant of get to \[lib.facet.num.get.members\]

\[
\text{iter\_type get(iter\_type in, iter\_type end, ios\_base\&,}
\]

\[
\text{ios\_base::iostate\& err, void* p) const;}
\]

Add above variant of do\_get to \[lib.facet.get.virtuals\]

\[
\text{iter\_type get(iter\_type in, iter\_type end, ios\_base\&,}
\]

\[
\text{ios\_base::iostate\& err, void* p) const;}
\]

and include in description of effects

For conversions to void* the specifier is %p.

Add public member to class num_put in \[lib.locale.num.put\]

\[
\text{iter\_type put(iter\_type s, ios\_base\&f, char\_type fill, void* p) const;}
\]

Add protected member to class num_put in \[lib.locale.num.put\]

\[
\text{virtual iter\_type do\_put(iter\_type s, ios\_base\&f, char\_type fill,}
\]

\[
\text{void* p) const;}
\]

Add above variant of put to \[lib.facet.num.put.members\]

\[
\text{iter\_type put(iter\_type s, ios\_base\&f, char\_type fill, void* p) const;}
\]

Add above variant of do\_put to \[lib.facet.put.virtuals\]
iter_type get(iter_type in, iter_type end, ios_base&, ios_base::iostate& err, void* p) const;

and include in description of effects

For conversions from void* the specifier is %p.

27-312 streambuf::sync

There was a discussion of various alternatives of what streambuf::sync ought to do with an non-empty get area. It was finally agreed that it should do nothing, and to emphasize that we recommend adding a sentence to the description of the effects of streambuf::sync in [lib.streambuf.virt.buffer]

If gptr() is non-null and gptr() != egptr() then do nothing.

27-414 check good()

In the definition of readsome, putack and unget in [lib.istream.unformatted] add as the first sentence

If !good() calls setstate(failbit) which may throw an exception and returns.

27-501 padding for char inserter

The working group has discussed a change to the char extractor many times in the past. It has always been understood that the current description in the WP describes a misfeature of iostreams classic. At this meeting the working group took a straw vote and recommended by 4 to 2 that the time is ripe for changing this misfeature.

Change the description in [lib.ostream.insertors] of

basic_ostream<charT,traits>& operator<<(char_type c);

to

Effects: Convert the char_type c with the conversion specifier c.

Returns: *this

If N0918/96-0100 is accepted then make the corresponding change in both the template functions for char_type and char.

27-651 behavior of setfill

The definition of setfill needs to be different than that for other manipulators because there may not be a conversion from int to char_type. The working group considered several possibilities and finally choose the following variation.

Change the description of setfill in [lib.std.manip] to

template<class charT> smanip setfill(charT c);

Returns: An object s of implementation specified type such that if out is (or is derived from) basic_ostream<charT,traitsT> and c has type charT then out << setfill(c) behaves as if f(s) were called where f could be defined as ...

27-203 testing state of stream

iostream classic operator void* to test the state of a stream. We changed that to operator bool when bool was introduced into the language. However, because the presence of operator bool turns the common beginner’s mistake of cout >> 1 into a well formed program we recommend reverting to operator void*.

Replace the declaration from 27.4.4[lib.ios]

operator bool() const;
operator void*() const;
Replace the definition in 27.4.4.3[lib.iostate.flags] with
operator void*() const;
Returns: If fail() a null pointer, otherwise some non-null value.