Doc: X3J16/96-0100=WG21/N0918
Date: May 25, 1996
By: Philippe Le Mouël

Insertion and Extraction of char, signed char, unsigned char

Description

The current WP does not specify the following:

- Insertion and Extraction of characters of type *char* into or from streams instantiated on other character types.
- Insertion and Extraction of characters of type *signed char* or *unsigned char* into or from streams instantiated on other character types.

Discussion

The problem of inserting and extracting *char* from streams instantiated on other character types can be divided into two parts, technical and semantic. The technical part can be resolved by using partial specialization, as pointed out by Jerry Schwarz. His solution is to remove the insertors on the character type from *basic_ostream* and to replace them by the following global insertors:

```
template<class charT, class traits>
basic_ostream<charT, traits>& operator<< (basic_ostream<charT, traits>&, charT);
```

Inserts a character of type *charT* different from *char* into a stream instantiated on *charT*.

```
template <class charT, class traits>
basic_ostream<charT, traits>& operator<< (basic_ostream<charT, traits>&, char);
```

Inserts a tiny character into a stream instantiated on other character types.

```
template <class traits>
basic_ostream<char, traits>& operator<< (basic_ostream<char, traits>&, char);
```

Inserts a tiny character into a stream instantiated on tiny character.

```
template<class charT, class traits>
basic_ostream<charT, traits>& operator<< (basic_ostream<charT, traits>&, const  charT*);

template <class charT, class traits>
basic_ostream<charT, traits>& operator<< (basic_ostream<charT, traits>&, const char*);

template <class traits>
basic_ostream<char, traits>& operator<< (basic_ostream<char, traits>&, const char*);
```

The same scheme can be applied to extract tiny characters.

The remaining problem is to find the meaning of inserting or extracting tiny characters from or into an arbitrary stream. A conversion needs to occur between tiny characters and characters of type *charT*. There are several schemes that can be used to perform such a conversion:

- We can require the character type *charT* to have an explicit constructor taking a *char* argument; we already have it for *‘/n’* and *0*.

- We can use the locale *ctype* member functions *widen* and *narrow* to convert between tiny characters and characters of type *charT*.

- We can use the *codecvt* facet member functions to convert between tiny characters and characters of type *charT*.

The two last solutions are the most reasonable. If we use the *ctype* member functions *widen* and *narrow*, we assume that the user wants to convert only between the code set used by its machine to encode tiny characters, and the code set used to encode characters of type *charT*. Therefore, we do not consider the case where inserting tiny characters has the meaning of inserting a multibyte sequence of characters, as in reading shift JIS encoding from a file and converting it as Unicode internally. If we insert a null terminated sequence of *char* we may want to treat it as a shift JIS multibyte sequence, and therefore use the *codecvt* facet to perform the conversion.

If we look closely at the solution using the *codecvt* facet, we see that we have two locale objects used in iostreams: The first one is located in the *streambuf*, and the second one in the stream. For design consistency, we should use the one located in the stream, therefore the extra flexibility we gain translates, to a new burden for the user, who has to take care of imbuing the right *codecvt* facet in the stream object. We also need to be careful in resolving the iostreams issue 27-205. Imbuing in the stream object must have no effect on the locale imbued in the *streambuf*.

The ideal solution is to use a combination of both schemes. With this approach, insertion and extraction of tiny characters from or into streams instantiated on other character types, use the *codecvt* facet imbued in the stream object to perform the conversion between tiny characters and characters of type *charT*. The default behavior of the *codecvt* facet is-as using the *ctype<charT>* facet member functions *narrow* and *widen* to perform the conversion between tiny characters and characters of type *charT*. This solution allows users to attach particular meaning to the insertion and extraction of tiny characters, and still get a reasonable default behavior.

Proposed resolutions

There are three proposed resolutions for this first problem:

1. Do not allow insertion or extraction of tiny characters from or into streams instantiated on some other character types.

2. Allow insertion and extraction of tiny characters from or into streams instantiated on some other character types. Resolve the technical aspect as described above by removing insertors and extractors on the character type from respectively *basic_ostream* and *basic_istream* and adding global member template functions to insert and extract characters. Resolve the semantic aspect by using the *ctype* facet member functions *widen* and *narrow* to perform the conversion between characters of type *char* and *charT*.

3. Same as 2 except for the semantic aspect, which is resolved by using the *codecvt* facet from the locale object imbued in the stream to perform conversion between characters of type *char* and *charT*. By

default the conversion performed by the *codecvt* facet member functions is-as using the *ctype* facet member functions *narrow* and *widen*.

Discussion

The second problem we have to deal with is the insertion and extraction of characters of type *signed char* or *unsigned char* into or from streams instantiated on other character types. This problem is closely related to the one presented above, and therefore the same comments and solutions apply. However, we have to consider another possibility, which is to allow insertion and extraction only on streams instantiated on tiny characters. This will maintain compatibility with the “old iostreams” library. This option which maintains compatibility with the “old iostreams” library, would add the following global template functions:

template <class traits>
basic_ostream<char, traits>& operator<< (basic_ostream<char, traits>&, unsigned char);

template <class traits>
basic_ostream<char, traits>& operator<< (basic_ostream<char, traits>&, signed char);

template <class traits>
basic_ostream<char, traits>& operator<< (basic_ostream<char, traits>&, const unsigned char*);

template <class traits>
basic_ostream<char, traits>& operator<< (basic_ostream<char, traits>&, const signed char*);

The same scheme can be applied to extract *unsigned char* and *signed char*.

Note: The semantic behavior of these functions is the same as in the “old iostreams” library.

Proposed resolutions

They are the same as the ones proposed in the case of insertion and extraction of tiny characters in arbitrary streams, plus the following:

4. Allow insertion and extraction of *signed char* and *unsigned char* only from or into streams instantiated on tiny characters (as described above).