13.4 — Stream classes for strings

BY ALEX, ON MARCH 18TH, 2008

So far, all of the I/O examples you have seen have been writing to cout or reading from cin. However, there is another set of classes called the stream classes for strings that allow you to use the familiar insertions (<<) and extraction (>>) operators to work with strings. Like istream and ostream, the string streams provide a buffer to hold data. However, unlike cin and cout, these streams are not connected to an I/O channel (such as a keyboard, monitor, etc…). One of the primary uses of string streams is to buffer output for display at a later time, or to process input line-by-line.

There are six string classes for streams: istringstream (derived from istream), ostringstream (derived from ostream), and stringstream (derived from iostream) are used for reading and writing normal characters width strings. wistringstream, wostringstream, and wstringstream are used for reading and writing wide character strings. To use the stringstreams, you need to #include the sstream header.

1) Use the insertion (<<) operator:

```
stringstream os;
os << "en garde!" << endl; // insert "en garde!" into the stringstream
```

2) Use the str(string) function to set the value of the buffer:

```
stringstream os;
os.str("en garde!"); // set the stringstream buffer to "en garde!"
```

There are similarly two ways to get data out of a stringstream:

1) Use the str() function to retrieve the results of the buffer:

```
stringstream os;
os << "12345 67.89" << endl;
cout << os.str();
```

This prints:

```
12345 67.89
```

2) Use the extraction (>>) operator:

```
stringstream os;
os << "12345 67.89"; // insert a string of numbers into the stream

string strValue;
os >> strValue;

string strValue2;
os >> strValue2;

// print the numbers separated by a dash
cout << strValue << " - " << strValue2 << endl;
```

This program prints:

```
12345 - 67.89
```

Note that the >> operator iterates through the string -- each successive use of >> returns the next extractable value in the stream. On the other hand, str() returns the whole value of the stream, even if the >> has already been used on the stream.

Conversion between strings and numbers

Because the insertion and extraction operators know how to work with all of the basic data types, we can use them in order to convert strings to numbers or vice versa.

First, let's take a look at converting numbers into a string:

```
stringstream os;

int nValue = 12345;
double dValue = 67.89;
os << nValue << " " << dValue;

string strValue1, strValue2;
os >> strValue1 >> strValue2;

cout << strValue1 << " " << strValue2 << endl;
```

This snippet prints:

```
12345 67.89
```

Now let's convert a numerical string to a number:

```
stringstream os;
os << "12345 67.89"; // insert a string of numbers into the stream
int nValue;
double dValue;

os >> nValue >> dValue;

cout << nValue << " " << dValue << endl;
```

This program prints:

```
12345 67.89
```

Clearing a stringstream for reuse

There are several ways to empty a stringstream's buffer.

1) Set it to the empty string using str():

```
stringstream os;
os << "Hello ";

os.str(""); // erase the buffer

os << "World!";
```

```
cout << os.str();
```

2) Call erase() on the result of str():

```
stringstream os;
os << "Hello ";

os.str(std::string()); // erase the buffer

os << "World!";
cout << os.str();
```

Both of these programs produce the following result:

```
World!
```

When clearing out a stringstream, it is also generally a good idea to call the clear() function:

```
stringstream os;
os << "Hello ";

os.str(""); // erase the buffer
os.clear(); // reset error flags

os << "World!";
cout << os.str();
```

clear() resets any error flags that may have been set and returns the stream back to the ok state. We will talk more about the stream state and error flags in the next lesson.

 **13.5 -- Stream states and input validation**

 **Index**

 **13.3 -- Output with ostream and ios**

Share this:

10 comments to 13.4 — Stream classes for strings

Grant
September 4, 2008 at 2:39 am · Reply

Hi Alex,
First a typo in the >> example, the comment says “print the numbers separated by a dash” but the code just prints the first value.

Second you need to include < sstream > to get at stringstream.

Third, the behaviour of >> and str() is subtly different as shown by this code:

```
stringstream os;
os << "12345 67.89"; // insert a string of numbers into the stream
string strValue;
os >> strValue; // extract 1st value
cout << "value1 " << strValue << endl;
os >> strValue; // extract 2nd value
cout << "value2 " << strValue << endl;
cout << "whole string " << os.str() << endl; // print whole string!
```

which gives the following output:

```
value1 12345
value2 67.89
whole string 12345 67.89
```

In other words the >> operator takes values from the string, moving along every time it is used, but the str() function always returns the whole string no matter how many times it is called (even after using >>)

Alex
September 10, 2008 at 10:56 pm · Reply

Thanks, I fixed the typo and made the distinction between >> and str() more clear.

Grant
September 4, 2008 at 5:54 am · Reply

Hi Alex,

There is an error in the section *Clearing a stringstream for reuse*. The second suggestion of using erase does not work. I think this is because os.str() returns a *copy* of the buffer and erase then just erases the copy leaving the original in tact. So the code in this case will still print “Hello World!”

Another point to note is that setting a buffer using str(“xx”) and then using the << operator on the stream will not append (as I at first thought) but will overwrite. So the following code:

```
stringstream os;
os.str("xxxxxxxxxx");
os << "World!";
cout << os.str() << endl;
```

will output:

World!xxxx

Alex
September 10, 2008 at 11:00 pm · Reply

You are correct, using erase() doesn't work because it clears a copy of the buffer.

That is interesting about the mixing of str() and the << operator. Doing so is kind of like mixing metaphors though -- better to stick with one or the other.

Gammerz
September 10, 2010 at 9:10 am · Reply

The text for this example still refers to erase().

serenity
December 4, 2009 at 6:15 am · Reply

Templates (covered later in this tutorial) can help you create a pretty nice, generic Convert function, that converts between arbitrary compatible values, such as strings and different number types (and even custom classes, if they overload operator<< and operator>> properly).

```
#include <iostream>
#include <string>
#include <sstream>

template <typename In_type, typename Out_type>
Out_type Convert(const In_type &value) {
    Out_type out;
    std::stringstream s;
    s << value;
    s >> out;
    return out;
}

int main() {
    std::string strNum = "1234";
    int dNum = Convert<std::string, int>(strNum);
    std::cout << strNum << " == " << dNum << std::endl;

    std::cout << "Enter a floating point value: ";
    float fNum;
    std::cin >> fNum;
    std::string strFloat = Convert<float, std::string>(fNum);
    std::cout << fNum << " == " << strFloat << std::endl;

    return 0;
}
<!--formatted-->
```

Output:

```
1234 == 1234
Enter a floating point value: 481.3
481.3 == 481.3
```

Auasp
July 13, 2011 at 8:14 pm · Reply

I Don't see any erase() function called.

mustsafa
March 12, 2013 at 12:12 pm · Reply

Hello, or should I say "Hello world!". I am still learning C++, and I would appreciate anyone how would help me.

I am trying to make a C++ program that would arrange three words in alphabetical order. Note that I did one for arranging three integers from least to greatest. Any help?

Knight
March 31, 2013 at 12:52 am · Reply

In the code given below, it is printing value of s2 but not printing value of S1. Can someone help me why it is the case?

```
using namespace std;
#include<iostream>
#include<string>
#include<sstream>

int main()
{
stringstream os;

int nD=234234;
os << nD;
string s2;
os >> s2;
cout << s2 << endl;

os << "12345 234234" << endl;

string S1;
os >> S1;
cout << S1 << endl;
os >> S1;
cout << S1 << endl;

return 0;
}
```

ict_ifycent2
February 18, 2014 at 1:52 am · Reply

great !!!