

String Output with `cout.width()` and `setw()` (2003.01.18)

Important note: *The examples given here were written before the current upgrade to the G++ compiler. In order for them to compile on the current compiler, function main must be return type int and must have as its last statement “return 0;”.*

In standard C++, string output using `cout` presents issues that are not readily apparent. For example, `cout.width` and `setw` are both effective with string literals. However, they are not effective with named string constants or variable identifiers. For example:

```
d47.ucs.louisiana.edu% more test.cc
#include <iostream>
#include <string>
#include <iomanip>

using namespace std;

void main()
{
    cout.width(40);                // effective
    cout << "literal output" << endl;
    cout << setw(60) << "literal output" << endl;    // effective

    const string CONSTANT = "constant output";
    cout.width(20);                // no effect
    cout << CONSTANT << endl;
    cout << setw(10) << CONSTANT << endl;          // no effect

    string variable = "variable output";
    cout.width(20);                // no effect
    cout << variable << endl;
    cout << setw(10) << variable << endl;          // no effect
}
d47.ucs.louisiana.edu% !g
g++ test.cc
d47.ucs.louisiana.edu% a.out
                literal output                literal output

constant output
constant output
variable output
variable output
d47.ucs.louisiana.edu% █
```

In the run of the example program “test.cc”, the output demonstrates that `cout.width()` and `setw()` have no effect on string class constants or variables.

To solve this problem, the string method `length()` can be used to find out the number of characters stored in a string variable (or constant). This number can be subtracted from the space to be allocated to the output of the string and the remaining amount of spaces used to define the number of spaces given to a single space given as a literal string.

For example, if we wished to output the value of a string variable called “variable” within 20 spaces and have the string variables value be justified to the right, we could write the following code:

```
cout.width( 20 - variable.length() );  
cout << " " << variable;
```

The space would be output as taking up 20 spaces, less the number of characters in the value of variable.

The same technique can be used with setw. For example:

```
cout << setw( 20 - variable.length() )  
    << " " << variable;
```

Here is an example of using these techniques with string literals and string variables:

```
d47.ucs.louisiana.edu% more test.cc  
#include <iostream>  
#include <string>  
#include <iomanip>  
  
using namespace std;  
  
void main()  
{  
    cout.width(40);  
    cout << "literal output" << endl;  
    cout << setw(60) << "literal output" << endl;  
  
    string variable = "variable output";  
    cout.width(20 - variable.length());  
    cout << " " << variable << endl;  
    cout << setw(40 - variable.length()) << " " << variable << endl;  
}  
d47.ucs.louisiana.edu% !g  
g++ test.cc  
d47.ucs.louisiana.edu% a.out  
                                literal output  
  
    variable output                literal output  
  
                                variable output  
d47.ucs.louisiana.edu% █
```