
14.5. Replacing Regular Expressions

Finally, let's look at the interface that allows you to replace character sequences that match a regular expression. Consider the following example:

Click here to view code image

```
// regex/regexreplace1.cpp

#include <string>
#include <regex>
#include <iostream>
#include <iterator>
using namespace std;

int main()
{
    string data = "<person>\n"
                  " <first>Nico</first>\n"
                  " <last>Josuttis</last>\n"
                  "</person>\n";

    regex reg("<(.*)>(.*)</(\\1)>");

    // print data with replacement for matched patterns
    cout << regex_replace (data,                          // data
                           reg,                           // regular
expression
                           "<$1 value=\"$2\"/>")          // replacement
         << endl;

    // same using sed syntax
    cout << regex_replace (data,                          // data
                           reg,                           // regular
expression
                           "<\\1 value=\"\\2\"/>",        // replacement
                           regex_constants::format_sed)   // format flag
         << endl;

    // use iterator interface, and
    // - format_no_copy:     don't copy characters that don't match
    // - format_first_only: replace only the first match found
    string res2;
    regex_replace (back_inserter(res2),                   // destination
                   data.begin(), data.end(),              // source range
                   reg,                                   // regular
expression
                   "<$1 value=\"$2\"/>",                  // replacement
                   regex_constants::format_no_copy        // format flags
                    | regex_constants::format_first_only);
    cout << res2 << endl;
}
```

Here again, we use a regular expression to match XML/HTML-tagged values. But this time, we transform the input into the following output:

Click here to view code image

```
<person>
 <first value="Nico"/>
 <last value="Josuttis"/>
</person>

<person>
 <first value="Nico"/>
 <last value="Josuttis"/>
</person>

<first value="Nico"/>
```

To do this, we specify a replacement where we can use matched subexpressions with the character `$` (see Table 14.1). Here, we use `$1` and `$2` to use the tag and the value found in the replacement:

Click here to view code image

```
"<$1 value=\"$2\"/>"          // replacement using default syntax
```

Table 14.1. Regex Replacement Symbols

Default Pattern	sed Pattern Meaning	
$&	&	The matched pattern
$*n*	*n*	The *n*th matched capture group
$‘		The prefix of the matched pattern
$’		The suffix of the matched pattern
$$		The character $

Again, we can avoid having to escape the quotes by using a raw string:

Click here to view code image

```
R"(<$1 value="$2"/>)"         // replacement using default syntax
```

By passing a regex constant `regex_constants::format_sed`, you can instead use the replacement syntax of the UNIX command `sed` (see the second column in Table 14.1):

Click here to view code image

```
"<\\1 value=\"\\2\"/>"        // replacement using sed syntax
```

Again, by using a raw string, we can avoid escaping backslashes:

Click here to view code image

```
R"(<\1 value="\2"/>)"         // replacement using sed syntax specified as raw string
```