Username: Pralay Patoria **Book:** The C++ Standard Library: A Tutorial and Reference, Second Edition. No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

14.10. Basic Regex Signatures in Detail

Table 14.6 lists the signatures of the basic regex operations regex_match() (see Section 14.1, page 717), regex_search() (see Section 14.1, page 717), and regex_replace() (see Section 14.5, page 730). As you can see,

there are always overloads to operate on strings, which can be both objects of class <code>basic_string<></code> and ordinary C-strings, such as string literals, and iterators, which specify the begin and end of the character sequence to process. In addition, you can always pass format flags as an optional last argument.

Table 14.6. Regex Operation Signatures

Signature	Effect
bool regex_match(str, regex)	Check full match of regex
<pre>bool regex_match(str,regex,flags)</pre>	
bool regex_match(beg,end,regex)	
<pre>bool regex_match(beg,end,regex,flags)</pre>	
<pre>bool regex_match(str, matchRet, regex)</pre>	Check and return full match
<pre>bool regex_match(str, matchRet, regex, flags)</pre>	of regex
<pre>bool regex_match(beg,end,matchRet,regex)</pre>	
<pre>bool regex_match(beg,end,matchRet,regex,flags)</pre>	
bool regex_search(str, regex)	Search match of regex
bool regex_search(str, regex, flags)	
<pre>bool regex_search(beg,end,regex)</pre>	
bool regex_search(beg,end,regex,flags)	
bool regex_search(str, matchRet, regex)	Search and return match
<pre>bool regex_search(str, matchRet, regex, flags)</pre>	of regex
<pre>bool regex_search(beg,end,matchRet,regex)</pre>	
<pre>bool regex_search(beg,end,matchRet,regex,flags)</pre>	
strRes regex_replace(str, regex, repl)	Replace match(es) according
<pre>strRes regex_replace(str, regex, repl, flags)</pre>	to regex
<pre>outPos regex_replace(outPos, beg, end, regex, repl)</pre>	
<pre>outPos regex_replace(outPos, beg, end, regex, repl, flags)</pre>	

Both regex_match() and regex_search() return true if a match was found. They also allow you to optionally pass an argument matchRet that returns details of the (sub)matches found. These arguments of type std::match_results<> (introduced in Section 14.2, page 721) must be instantiated for an iterator type that corresponds with the character type:

• For C++ strings, it is the corresponding **CONST** iterator. For types **String** and **WSTring**, the corresponding types **Smatch** and **WSMatch** are defined:

Click here to view code image

```
typedef match_results<string::const_iterator> smatch;
typedef match_results<wstring::const_iterator> wsmatch;
```

• For C-strings including string literals, it is the corresponding pointer type. For c-strings of **char** and **wchar_t** characters, the corresponding types **cmatch** and **wcmatch** are defined:

Click here to view code image

```
typedef match_results<const char*> cmatch;
typedef match_results<const wchar_t*> wcmatch;
```

For <code>regex_replace()</code> , you have to pass the replacement specification <code>repl</code> as a string (again, it is overloaded for both objects of class <code>basic_string<></code> and ordinary C-strings, such as string literals). The string version returns a new string with the corresponding replacements. The iterator version returns the first argument <code>outPos</code>, which has to be an output iterator specifying where the replacements are written to.

Finally, note that to avoid ambiguities no implicit type conversion from strings or string literals to type regex is provided. Thus, you always explicitly have to convert any string holding a regular expression to type std::regex (or $std::basic_regex<>$).