
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 `basic_string<>` 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)`
`bool regex_match(str,regex,flags)`
`bool regex_match(beg,end,regex)`
`bool regex_match(beg,end,regex,flags)`
	Check full match of *regex*

`bool regex_match(str,matchRet,regex)`
`bool regex_match(str,matchRet,regex,flags)`
`bool regex_match(beg,end,matchRet,regex)`
`bool regex_match(beg,end,matchRet,regex,flags)`
	Check and return full match of *regex*

`bool regex_search(str,regex)`
`bool regex_search(str,regex,flags)`
`bool regex_search(beg,end,regex)`
`bool regex_search(beg,end,regex,flags)`
	Search match of *regex*

`bool regex_search(str,matchRet,regex)`
`bool regex_search(str,matchRet,regex,flags)`
`bool regex_search(beg,end,matchRet,regex)`
`bool regex_search(beg,end,matchRet,regex,flags)`
	Search and return match of *regex*

strRes `regex_replace(str,regex,repl)` *strRes* `regex_replace(str,regex,repl,flags)` *outPos* `regex_replace(outPos,beg,end,regex,repl)` *outPos* `regex_replace(outPos,beg,end,regex,repl,flags)`	Replace match(es) according to *regex*

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 `regex_replace()`, you have to pass the replacement specification *repl* as a string (again, it is overloaded for both objects of class `basic_string<>` 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 *outPos*, 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<>`).