std::regex_token_iterator


```
template<
    class BidirIt,
    class CharT = typename std::iterator_traits<BidirIt>::value_type,    (since C++11)
    class Traits = std::regex_traits<CharT>
> class regex_token_iterator
```

`std::regex_token_iterator` is a read-only `ForwardIterator` that accesses the individual sub-matches of every match of a regular expression within the underlying character sequence. It can also be used to access the parts of the sequence that were not matched by the given regular expression (e.g. as a tokenizer).

On construction, it constructs an `std::regex_iterator` and on every increment it steps through the requested sub-matches from the current match_results, incrementing the underlying regex_iterator when incrementing away from the last submatch.

The default-constructed `std::regex_token_iterator` is the end-of-sequence iterator. When a valid `std::regex_token_iterator` is incremented after reaching the last submatch of the last match, it becomes equal to the end-of-sequence iterator. Dereferencing or incrementing it further invokes undefined behavior.

Just before becoming the end-of-sequence iterator, a `std::regex_token_iterator` may become a *suffix iterator*, if the index `-1` (non-matched fragment) appears in the list of the requested submatch indexes. Such iterator, if dereferenced, returns a match_results corresponding to the sequence of characters between the last match and the end of sequence.

A typical implementation of `std::regex_token_iterator` holds the underlying `std::regex_iterator`, a container (e.g. `std::vector<int>`) of the requested submatch indexes, the internal counter equal to the index of the submatch, a pointer to `std::match_results`, pointing at the current submatch of the current match, and a `std::match_results` object containing the last non-matched character sequence (used in tokenizer mode).

Type requirements

- `BidirIt` must meet the requirements of `BidirectionalIterator`.

Specializations

Several specializations for common character sequence types are defined:

Defined in header `<regex>`

Type	Definition
`cregex_token_iterator`	`regex_token_iterator<const char*>`
`wcregex_token_iterator`	`regex_token_iterator<const wchar_t*>`
`sregex_token_iterator`	`regex_token_iterator<std::string::const_iterator>`
`wsregex_token_iterator`	`regex_token_iterator<std::wstring::const_iterator>`

Member types

Member type	Definition
`value_type`	`std::sub_match<BidirIt>`
`difference_type`	`std::ptrdiff_t`
`pointer`	`const value_type*`
`reference`	`const value_type&`
`iterator_category`	`std::forward_iterator_tag`
`regex_type`	`basic_regex<CharT, Traits>`

Member functions

(constructor)	constructs a new regex_token_iterator (public member function)
(destructor) (implicitly declared)	destructs a regex_token_iterator, including the cached value (public member function)
operator=	replaces a regex_token_iterator (public member function)
operator== **operator!=**	compares two regex_token_iterators (public member function)
operator* **operator->**	accsses current submatch (public member function)
operator++ **operator++**`(int)`	advances the regex_token_iterator to the next submatch (public member function)

Notes

It is the programmer's responsibility to ensure that the `std::basic_regex` object passed to the iterator's constructor outlives the iterator. Because the iterator stores a `std::regex_iterator` which stores a pointer to the regex, incrementing the iterator after the regex was destroyed results in undefined behavior.

Example

```
#include <fstream>
#include <iostream>
#include <algorithm>
#include <iterator>
#include <regex>
int main()
{
   std::string text = "Quick brown fox.";
   // tokenization (non-matched fragments)
   // Note that regex is matched only two times: when the third value is obtained
   // the iterator is a suffix iterator.
   std::regex ws_re("\\s+"); // whitespace
   std::copy( std::sregex_token_iterator(text.begin(), text.end(), ws_re, -1),
              std::sregex_token_iterator(),
              std::ostream_iterator<std::string>(std::cout, "\n"));

   // iterating the first submatches
   std::string html = "<p><a href=\"http://google.com\">google</a> "
                      "< a HREF =\"http://cppreference.com\">cppreference</a>\n</p>";
   std::regex url_re("<\\s*A\\s+[^>]*href\\s*=\\s*\"([^\"]*)\"", std::regex::icase);
   std::copy( std::sregex_token_iterator(html.begin(), html.end(), url_re, 1),
              std::sregex_token_iterator(),
              std::ostream_iterator<std::string>(std::cout, "\n"));
}
```

Output:

```
Quick
brown
fox.
http://google.com
http://cppreference.com
```

Retrieved from "http://en.cppreference.com/mwiki/index.php?title=cpp/regex/regex_token_iterator&oldid=41780"