C++11 Regular-Expression Library

By Brian Overland

Jun 25, 2013

This chapter is from the book

C++ for the Impatient

Learn More Buy

20.5. "Find All," or Iterative, Searches

When searching for substrings, you typically want to find all the matching substrings rather than just the first one. The easiest way to do this is to use an iterative search.

This technique involves creating two iterators: one that iterates through a target string and is associated with a **regex** object, and another that is simply an "end" condition.

```
sregex_iterator  iter_name(str_obj.begin(), str_obj.end(), regex_obj
sregex_iterator  end_iter_name;
```

This second declaration—the one that creates *end_iter_name*—should strike you as something new: an uninitialized iterator that has a use. This is different from other STL containers, which require a call to their **end** function to get an end-position iterator. Not so with regex iterators: An uninitialized regex iterator automatically represents the ending position—one position past the end of the string.

The **sregex_iterator** type is an adapter that uses a template, **regex_iterator**, with the **string** class. You can build iterators upon other string types.

With these two declarations in place—providing an iterator and an end position—it's then an easy matter to iterate through all the substrings found. For example:

```
#include <regex>
```

```
#include <string>
using std::regex;
using std::sregex_iterator;
using std::string;

regex reg1("([A-Za-z]+) \\1", std::regex_constants::icase);
string target = "The the cow jumped over the the moon";
sregex_iterator it(target.begin(), target.end(), reg1);
sregex_iterator reg_end;
for (; it != reg_end; ++it) {
     std::cout << "Substring found: ";
     std::cout << it->str() << ", Position: ";
     std::cout << it->position() << std::endl;
}
```

These statements print:

```
Substring found: The the, Position: 0
Substring found: the the, Position: 24
```

This next example finds all words beginning with an uppercase or lowercase "B". Case sensitivity must be turned off in order to find "Barney" as well as "bat" and "big". In addition, the word-boundary character (\b) is used to ensure that the only patterns found are those in which the letter "B" comes at the beginning of the word.

```
regex reg2("\\bB[a-z]*", regex_constants::icase);
string bstr = "Barney goes up to bat with a big stick.";
sregex_iterator it(bstr.begin(), bstr.end(), reg2);
sregex_iterator reg_end;
for (; it != reg_end; ++it) {
     std::cout << "Substring found: ";
     std::cout << it->str() << ", Position: ";
     std::cout << it->position() << std::endl;
}
```

These statements print:

```
Substring found: Barney, Position: 0
Substring found: bat, Position: 18
Substring found: big, Position: 29
```

Share This Save To Your Account

< Back **Page 5** of 10 Next >