

D:/H-Drive/public\_html/91.204/91.204-2012-13f/204-lecs/code/BoostRegexTests/regex\_token\_iterator\_eg\_2.cpp

```
1  /*
2  *
3  * Copyright (c) 2003
4  * John Maddock
5  *
6  * Use, modification and distribution are subject to the
7  * Boost Software License, Version 1.0. (See accompanying file
8  * LICENSE_1_0.txt or copy at http://www.boost.org/LICENSE_1_0.txt)
9  *
10 */
11
12 /*
13 *   LOCATION:    see http://www.boost.org for most recent version
14 *   FILE        regex_token_iterator_example_2.cpp
15 *   VERSION      see <boost/version.hpp>
16 *   DESCRIPTION: regex_token_iterator example: spit out linked URLs
17 */
18
19
20 #include <fstream>
21 #include <iostream>
22 #include <iterator>
23 #include <boost/regex.hpp>
24
25 boost::regex e("<\\s*A\\s+[^>]*href\\s*=\\s*\"([^\"]*)\">",
26               boost::regex::normal | boost::regbase::icase);
27
28 void load_file(std::string& s, std::istream& is)
29 {
30     s.erase();
31     if(is.bad()) return;
32     //
33     // attempt to grow string buffer to match file size,
34     // this doesn't always work...
35     s.reserve(static_cast<std::string::size_type>(is.rdbuf()->in_avail()));
36     char c;
37     while(is.get(c))
38     {
39         // use logarithmic growth strategy, in case
40         // in_avail (above) returned zero:
41         if(s.capacity() == s.size())
42             s.reserve(s.capacity() * 3);
43         s.append(1, c);
44     }
45 }
46
47
```

D:/H-Drive/public\_html/91.204/91.204-2012-13f/204-lecs/code/BoostRegexTests/regex\_token\_iterator\_eg\_2.cpp

```
48 int main(int argc, char** argv)
49 {
50     std::string s;
51     int i;
52     for(i = 1; i < argc; ++i)
53     {
54         std::cout << "Findings URLs in " << argv[i] << ":\n" << std::endl;
55         s.erase();
56         std::ifstream is(argv[i]);
57         load_file(s, is);
58         is.close();
59         boost::sregex_token_iterator i(s.begin(), s.end(), e, 1);
60         boost::sregex_token_iterator j;
61         while(i != j)
62         {
63             std::cout << *i++ << std::endl;
64         }
65     }
66     //
67     // alternative method:
68     // test the array-literal constructor, and split out the whole
69     // match as well as $1....
70     //
71     for(i = 1; i < argc; ++i)
72     {
73         std::cout << "\nFindings URLs in " << argv[i] <<
74             ", alternative method:\n" << std::endl;
75         s.erase();
76         std::ifstream is(argv[i]);
77         load_file(s, is);
78         is.close();
79         const int subs[] = {1, 0,};
80         boost::sregex_token_iterator i(s.begin(), s.end(), e, subs);
81         boost::sregex_token_iterator j;
82         while(i != j)
83         {
84             std::cout << *i++ << std::endl;
85         }
86     }
87
88     return 0;
89 }
```