```
/*
 *
 * Copyright (c) 2003
 * John Maddock
 *
 * Use, modification and distribution are subject to the
 * Boost Software License, Version 1.0. (See accompanying file
 * LICENSE_1_0.txt or copy at http://www.boost.org/LICENSE_1_0.txt)
 *
 */

 /*
  *   LOCATION:    see http://www.boost.org for most recent version
  *   FILE         regex_token_iterator_example_2.cpp
  *   VERSION      see <boost/version.hpp>
  *   DESCRIPTION: regex_token_iterator example: spit out linked URLs
  */


#include <fstream>
#include <iostream>
#include <iterator>
#include <boost/regex.hpp>

boost::regex e("<\\s*A\\s+[^>]*href\\s*=\\s*\"([^\"]*)\"",
               boost::regex::normal | boost::regbase::icase);

void load_file(std::string& s, std::istream& is)
{
   s.erase();
   if(is.bad()) return;
   //
   // attempt to grow string buffer to match file size,
   // this doesn't always work...
   s.reserve(static_cast<std::string::size_type>(is.rdbuf()->in_avail()));
   char c;
   while(is.get(c))
   {
      // use logarithmic growth stategy, in case
      // in_avail (above) returned zero:
      if(s.capacity() == s.size())
         s.reserve(s.capacity() * 3);
      s.append(1, c);
   }
}


```

```
int main(int argc, char** argv)
{
   std::string s;
   int i;
   for(i = 1; i < argc; ++i)
   {
      std::cout << "Findings URLs in " << argv[i] << ":\n" << std::endl;
      s.erase();
      std::ifstream is(argv[i]);
      load_file(s, is);
      is.close();
      boost::sregex_token_iterator i(s.begin(), s.end(), e, 1);
      boost::sregex_token_iterator j;
      while(i != j)
      {
         std::cout << *i++ << std::endl;
      }
   }
   //
   // alternative method:
   // test the array-literal constructor, and split out the whole
   // match as well as $1....
   //
   for(i = 1; i < argc; ++i)
   {
      std::cout << "\nFindings URLs in " << argv[i] <<
           ", alternative method:\n" << std::endl;
      s.erase();
      std::ifstream is(argv[i]);
      load_file(s, is);
      is.close();
      const int subs[] = {1, 0,};
      boost::sregex_token_iterator i(s.begin(), s.end(), e, subs);
      boost::sregex_token_iterator j;
      while(i != j)
      {
         std::cout << *i++ << std::endl;
      }
   }

   return 0;
}
```