the trusted technology learning source

The Standard Template Library: Generic Programming

By Stanley B. Lippman

Jan 25, 2002

This chapter is from the book

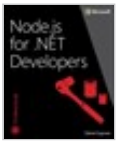
C# Primer: A Practical Approach

Learn More Buy

Using the iostream Iterators

Related Resources

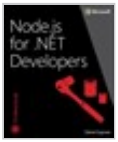
Store | Articles | Blogs



Node.js for .NET Developers

By David Gaynes

Book $15.99



Node.js for .NET Developers

By David Gaynes

eBook (Watermarked) $12.79

Microsoft .NET - Architecting Applications for the Enterprise, 2nd Edition

By Dino Esposito, Andrea Saltarello

Book $35.99

See All Related Store Items

Imagine that we have been given the task of reading a sequence of string elements from standard input, storing them into a vector, sorting them, and then writing the words back to standard output. A typical solution looks like this:

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
using namespace std;

int main()
{
  string word;
  vector<string> text;

  // ok: let's read each word in turn until done
  while ( cin >> word )
      text.push_back( word );

  // ok: sort it
  sort( text.begin(), text.end() );

  // ok: let's write them back
  for ( int ix = 0; ix < text.size(); ++ix )
     cout << text[ ix ] << ' ';
}
```

The standard library defines both input and output iostream iterator classes, called `istream_iterator` and `ostream_iterator`, that provide a shorthand method of reading and writing elements of a single type. To use either iterator class, we must include the `iterator` header file:

```
#include <iterator>
```

For example, let's see how we can use the `istream_iterator` class to read our string sequence from standard input. As with all iterators, we need a `first,last` pair of `istream_iterators` that mark the range of elements. The definition

```
istream_iterator<string> is( cin );
```

provides us with a `first` iterator. It defines `is` as an `istream_iterator` bound to standard input that reads elements of type string. We also need a `last` iterator that represents 1 past the last element to be read. For standard input, end-of-file represents 1 past the last. How do we indicate that? The definition of an `istream_iterator` with `eof`,

```
istream_iterator<string> eof;
```

This chapter is from the book

C# Primer: A Practical Approach

Learn More Buy

represents end-of-file. How do we actually use this pair? In the following example, we pass them to the generic algorithm `copy()`, together with the vector to store the string elements. Because we don't know what size to make the vector, we adapt it with a `back_inserter`:

```
  copy( is, eof, back_inserter( text ));
```

Now we need an `ostream_iterator` to mark where to write each string element. We stop when there are no more elements to write out. The following defines `os` to be an `ostream_iterator` tied to standard output that holds elements of type string:

```
ostream_iterator<string> os( cout, " " );
```

The second argument is either a C-style character string or a string literal that indicates the delimiter to output between the elements. By default, the elements are written without any delimiter between them. In this example, I've chosen to output each element separated by a space. Here is how we might use it:

```
copy( text.begin(), text.end(), os );
```

`copy()` writes each element stored within `text` to the ostream indicated by `os`. Each element is separated by a space. Here is the complete program:

```
#include <iostream>
#include <iterator>
#include <algorithm>
#include <vector>
#include <string>
using namespace std;

int main()
{
  istream_iterator< string > is( cin );
  istream_iterator< string > eof;

  vector< string > text;
  copy( is, eof, back_inserter( text ));

  sort( text.begin(), text.end() );

  ostream_iterator<string> os( cout, " " );
```

```
  copy( text.begin(), text.end(), os );
}
```

Often, rather than read from standard input or write to standard output, we read from and write to a file. How can we do that? We simply bind the `istream_iterator` to an `ifstream` class object and the `ostream_iterator` to an `ofstream` class object:

```
#include <iostream>
#include <fstream>
#include <iterator>
#include <algorithm>
#include <vector>
#include <string>
using namespace std;

int main()
{
  ifstream in_file( "as_you_like_it.txt" );
  ofstream out_file( "as_you_like_it_sorted.txt" );

  if ( ! in_file || ! out_file )
  {
    cerr << "!!unable to open the necessary files.\n";
    return -1;
  }

  istream_iterator< string > is( in_file );
  istream_iterator< string > eof;

  vector< string > text;
  copy( is, eof, back_inserter( text ));

  sort( text.begin(), text.end() );
```

```
  ostream_iterator<string> os( out_file, " " );
  copy( text.begin(), text.end(), os );I
}
```

Exercise 1

Write a program to read a text file. Store each word in a map. The key value of the map is the count of the number of times the word appears in the text. Define a word exclusion set containing words such as *a*, *an*, *or*, *the*, *and*, and *but*. Before entering a word in the map, make sure that it is not present in the word exclusion set. Display the list of words and their associated count when the reading of the text is complete. As an extension, before displaying the text, allow the user to query the text for the presence of a word.

Exercise 2

Read in a text file—it can be the same one as in Exercise 1— storing it in a vector. Sort the vector by the length of the string. Define a function object to pass to `sort()`; it should accept two strings and return `true` if the first string is shorter than the second. Print the sorted vector.

Exercise 3

Define a map for which the index is the family surname and the key is a vector of the children's names. Populate the map with at least six entries. Test it by supporting user queries based on a surname and printing all the map entries.

Exercise 4

Write a program to read a sequence of integer numbers from standard input using an `istream_iterator`. Write the odd numbers into one file using an `ostream_iterator`. Each value should be separated by a space. Write the even numbers into a second file, also using an `ostream_iterator`. Each of these values should be placed on a separate line.

+ Share This Save To Your Account < Back **Page 10** of 10

Discussions

Comments for this thread are now closed.

0 Comments InformIT 1 Login

Recommend Share Sort by Oldest

This discussion has been closed.

ALSO ON INFORMIT WHAT'S THIS?

The Inefficiency of Multitasking: Why Your Smartphone Is a …

2 comments • 5 months ago

Avat Tony Howard — Nothing new here. There used to be time-management courses where …

Auto Property Enhancements in C# 6

1 comment • a month ago

Avat imran —  YES™ < PLUS PINT .I have profited 104 thousand

OS X for iOS Developers: Why You Should Be Building …

1 comment • 3 months ago

Avat eskadah — Great article/tutorial.. I always thought that regular expressions in cocoa were ascii

Migrating C/C++ from 32-Bit to 64-Bit | Just What Exactly Is '64-

1 comment • 3 months ago

Avat Kevin Nealon — An excellent article and very relevant. It is very helpful to see all the issues

Subscribe Add Disqus to your site Privacy