

Lab : Iterators

Information

In-class labs are meant to introduce you to a new topic and provide some practice with that new topic.

Topics: Using iterators with STL List

Solo work: Labs should be worked on by each individual student, though asking others for help is permitted. Do not copy code from other sources, and do not give your code to other students. **Students who commit or aid in plagiarism will receive a 0% on the assignment and be reported.**

Building and running: If you are using Visual Studio, make sure to run with debugging. **Do not run without debugging!**

Using the debugger will help you find errors.

To prevent a program exit, use this before `return 0;`

<code>cin.ignore();</code> <code>cin.get();</code>
--

Turn in: Once you're ready to turn in your code, prepare the files by doing the following: **(1)** Make a copy of your project folder and name it `LASTNAME-FIRSTNAME-LABNAME`. (Example: `HOPPER-GRACE-LAB-UNIT-TESTS`) **(2)** Make sure that all source files (`.cpp`, `.hpp`, and/or `.h` files) and the `Makefile` files are all present. **(3)** Remove all Visual Studio files - I only want the source files and `Makefiles`. **(4)** Zip your project folder as `LASTNAME-FIRSTNAME-LABNAME.zip`

Never turn in Visual Studio files!

Starter files: Download from GitHub.

Grading: Grading is based on completion, if the program functions as intended, and absence of errors. **Programs that don't build will receive 0%.** Besides build errors, runtime errors, logic errors, memory leaks, and ugly code will reduce your score.

Contents

1.1	About	3
1.1.1	Setting up the project	3
1.2	Lab specifications	3
1.2.1	ReadBook	3

1.1 About

Iterators are a way we can more efficiently traverse through certain types of data structures, such as linked lists. For this lab, we will have a list of strings to use iterators with.

1.1.1 Setting up the project

Download the starter zip file, `LAB-ITERATORS.zip`, off GitHub. This zip contains the following:

- `main.cpp`
- `Menu.hpp`
- `aesop.txt`
- `fairytale.txt`

Make sure your text files are in your **project directory** so that they can be read.

The `LoadBook` function is already implemented for you, you just have to implement the `ReadBook` function according to the instructions to make a book reading program.

1.2 Lab specifications

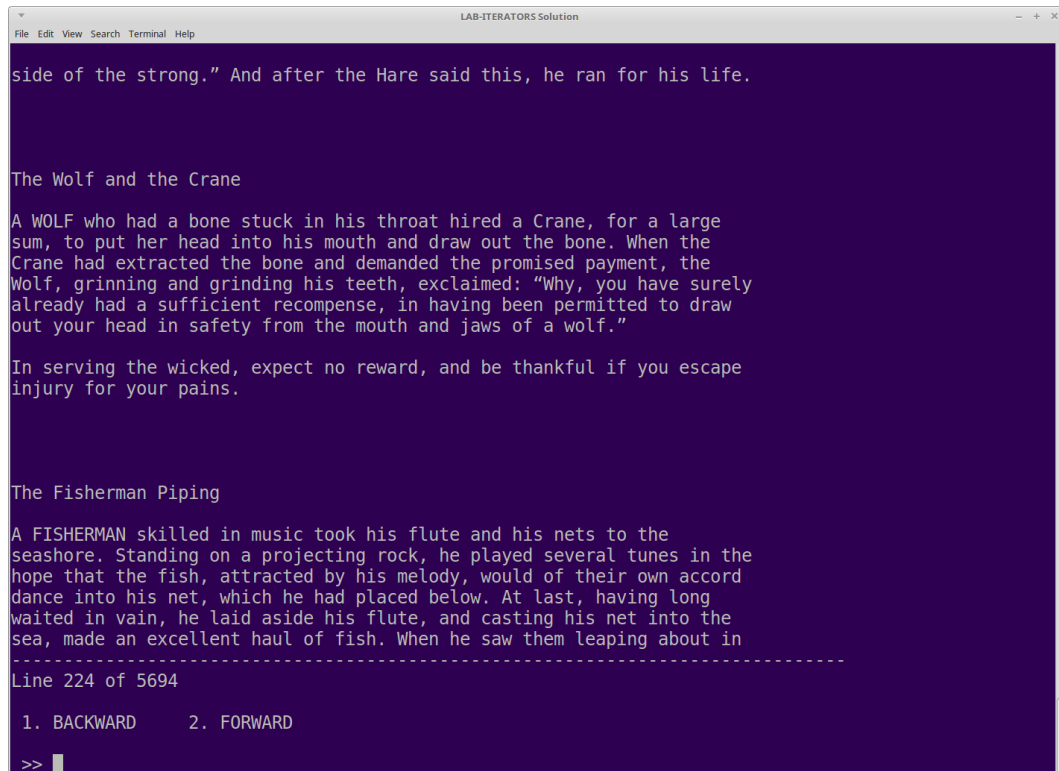
1.2.1 ReadBook

The book reader will output one-screen-height of lines of text at a time. A standard console window is `80x30`. There will be a menu bar at the bottom of the screen, taking up several lines as well. On the bottom of the page, there will be a count of what line you're currently on, and the total amount of lines in the book.

You'll need two int counters: `counter` (to decide when to end the page), and `lines` (the line count for the current progress in the book). You will also want to make an integer named `pageLength` to store the amount of lines to draw before ending the page (since we have to make space for the bottom

menu.)

Initialize `counter` and `lines` to 0, and `pageLength` to 28. Depending on the size of your console window, you might have to change it around.



```
LAB-ITERATORS Solution
File Edit View Search Terminal Help

side of the strong." And after the Hare said this, he ran for his life.

The Wolf and the Crane
A WOLF who had a bone stuck in his throat hired a Crane, for a large
sum, to put her head into his mouth and draw out the bone. When the
Crane had extracted the bone and demanded the promised payment, the
Wolf, grinning and grinding his teeth, exclaimed: "Why, you have surely
already had a sufficient recompense, in having been permitted to draw
out your head in safety from the mouth and jaws of a wolf."

In serving the wicked, expect no reward, and be thankful if you escape
injury for your pains.

The Fisherman Piping
A FISHERMAN skilled in music took his flute and his nets to the
seashore. Standing on a projecting rock, he played several tunes in the
hope that the fish, attracted by his melody, would of their own accord
dance into his net, which he had placed below. At last, having long
waited in vain, he laid aside his flute, and casting his net into the
sea, made an excellent haul of fish. When he saw them leaping about in
-----
Line 224 of 5694

1. BACKWARD      2. FORWARD

>> |
```

After you've created the initial variables, you will begin a loop with your iterator:

```
1 list<string>::iterator it;
2 for ( it = bookText.begin(); it != bookText.end(); )
3 {
4 }
```

This sets the start and the end of the loop, and we will handle whether to move forward or backward within the loop. Display the current line of the book, update the counters, and move the iterator forward:

```
1 cout << *it << endl;      // Display line of text
2 counter++;                // Line counter for this page
3 lines++;                  // Line counter for book
4 it++;                     // Move iterator forward
```

You'll also need an if statement to see if your `counter` has hit the `pageLength`, then display the menu to the user...

```
1  if ( counter == pageLength )
2  {
3      cout << "Line " << lines << " of "
4          << bookText.size() << endl << endl;
5
6      int choice = Menu::ShowIntMenuWithPrompt( {
7          "BACKWARD",
8          "FORWARD"
9      }, false );
10
11     if ( choice == 1 ) // backwards
12     {
13         // Move the iterator back pageLength*2 spaces.
14     }
15
16     counter = 0; // reset line/page counter
17 }
```

You cannot use `--` with the iterator, so you'll need another loop to make the iterator back up `pageLength*2` times. Also, make sure to decrement `lines` by the `pageLength` amount as well.

Afterward, your book reader program should work, letting you scroll between pages with the 1 and 2 commands.