Lab 6 – C++ Advanced Input
CPSC 1021- Fall 2017

CLEMSON
School of COMPUTING

Introduction

In today's lab, we will read in a file using C++ I/O Manipulators and functions, including an introduction to the std::string object.

Lab Objectives

- C++ File I/O + Manipulators
- Reading a file with no length marker and complex input
- Use nested loops in a single-pass algorithm

Prior to Lab

Make sure you can read in and write out to files in C++. This involves using C++ style streams. Read more here: http://www.cplusplus.com/doc/tutorial/files/

We will be using std::string's in this lab, but they will not be covered extensively. Still, these pages will be useful:

http://www.cplusplus.com/reference/string/string/
http://en.cppreference.com/w/cpp/string/basic_string/getline
http://www.cplusplus.com/reference/iomanip/

Lab Instructions

Create one file, **lab6.cpp.** This program should accept **two command line arguments:** an input text file and an output text file for writing results.

You will be given an input file of variable length that contains records from a book auction service. Each line will contain a record of the form:

1 $43.50 $74.50 $33.60 The Dark Tower
2 $23.15 $43.32 $19.53 Lord of the Rings Return of the King
3 $32.34 $50.65 Ender's Game
4 $21.50 $32.92 Clash of Kings
5 $34.32 $31.21 $37.43 $19.99 Feast for Crows
6 $28.50 $22.92 Hitchhiker's Guide to the Galaxy

Each record begins with an ID, a variable list of auction prices, and a book title.

You may only use C++ style file I/O to manipulate data. **You may not use printf or scanf in your program**.

Your Objectives

- Read through the input file breaking each line into its components
- Find the average of all the list prices for each book
- Print out a formatted record for each book, using setw() and other manipulators
- If a book title is too long, replace the end of the title with "…"

Sample Output

Before we go any further, here is the expected output for the example file above:

ID	Title	Price
1	The Dark Tower	50.53
2	Lord of the Rings Retu...	28.67
3	Ender's Game	41.50
4	Clash of Kings	27.21
5	Feast for Crows	30.74
6	Hitchhiker's Guide to ...	25.71

The ID column should be set to a width of **4** characters
The Title column should be set to a width of **25** characters
The Price column should be set to a width of **10** characters

This means that **if a book title is greater than 25 characters** you must erase everything past the 25th character and replace the last 3 characters with "…". To do that, let's talk briefly about strings.

Strings

In C++, we have access to a new variable type known as a **string**. Strings are C++ objects which handle their own memory requirements, and define many functions that are useful when manipulating text. For the purposes of this lab, it is fine to think of strings as char*'s which do not require mallocs/frees.

To create a string, #include the <string> library and simply declare it like any other variable:

std::string some_string;

Using C++ style file I/O, we can read in to a string just like any other type of variable:

```
ifstream input(argv[1]);
input >> some_string;
```

I/O Manip: Currency

We will use new I/O manipulators in this lab, C++11 standard functions get_money() and put_money().

http://www.cplusplus.com/reference/iomanip/get_money/
http://www.cplusplus.com/reference/iomanip/put_money/

These manipulators make it much easier to read in complex currency strings as values we can perform math on. I'll refer you to the reference pages for examples of using these functions, however note that we must **set the locale** of our program for these functions to work correctly.

Simply use these lines before attempting to input/output a currency value, given that **in** is an ifstream and **out** is an ofstream.

```
in.imbue(std::locale("en_US.UTF-8"));
out.imbue(std::locale("en_US.UTF-8"));
```

We have to do this because the functions support multiple formats of currency and we want them to interpret our values as dollar amounts.

I/O Manip: getline()

Reading in integers is easy, and as we saw currency can be read with the money manipulators. The last field we need to deal with is the book title. Since the titles contain whitespace, we need to use the getline() function to collect all characters up to the end of the record.

The arguments to this function are the stream to read from and the string to read in to (this is one reason we are using strings). Again, use of this function is fairly straightforward.

String: Length, Erase, Replace

Although we haven't covered classes and objects in C++ yet, basic string operations are useful and easy to use. You might have noticed in most of your C code that you often must keep track of how long your char* strings are yourself with some integer variables.

The C++ string variable type provides many functions which make our life easier. These code and usage of these functions is provided here without explanation: it will be covered in depth in

the coming weeks of lecture. For the purposes of this lab, we only need to be able to plug these functions into our solution:

std::string some_title = "Braveheart";

some_title.length(); // returns value 10
some_title.erase(5, std::string::npos); // string value is now "Brave"
some_title.replace(2, std::string::npos, 3, 'r'); // string value is now "Brrrr"

0	1	2	3	4	5	6	7	8	9
B	r	a	v	e	h	e	a	r	t

length() performs exactly as you would expect, simply returning a number.
erase() takes a start location and a number of characters to erase.
replace() takes a start location, end location, number of characters to insert, and the character to insert

Note the use of **std::string::npos** in the above examples: this constant represents the end of the string. So the erase call above will erase all characters from location 5 to the end of the string.

The Read Loop

Your program should parse the file as follows:

print to file column titles (see sample output above)
while (read in ID succeeds)
while (read in price value succeeds)
keep sum of prices for the current record
keep count of prices to find average later
done

use in.clear() to get ready to read the title
use getline to read the title into a std::string

if the title is great than 25 in length
erase all characters after the 25th
replace characters 23 – 25 with '.'

print to file as three columns of size 4, 25, and 10 (remember to use put_money!)
done

What to turn in

- **lab6.cpp** which reads in a file of book records, finds the average price, and writes the record back out in a well formatted way.

SOME HINTS!

String locations are indexed from 0 just like arrays.

Test your code piece by piece. Make sure you can read in all fields and print them out before trying to manipulate or format.

If a read fails (like when reading in the price list for a record) an error flag is set which stops all future reads. To reset these flags and continue reading, use .clear():

in_file.clear()

Compile and Execute

Use G++ to compile your code as follows **and include the C++11 standard!**:

g++ -std=c++11 -Wall -g lab6.cpp -o books

Execute the program

./books <input> <output>

FORMATTING:

1. Your program should be well documented
2. Each file should include a header:
3. Your program should consist of proper and consistent indention
4. No lines of code should be more than 80 characters

```
// Sample Header
/******************
  your name
  username
  Lab 1
  Lab Section:
  Name of TA
******************/
```

5 – 10 points will be deducted for each of the above formatting infractions.

Submission Instructions

- Test your program on the School of Computing server prior to submitting.
- Use the tar utility to tar.gz all source files. **Do not tar an entire directory! When I untar your archive, I should see all the files you included, not a top-level directory! Failure to correctly tar may result in up to a 25-point penalty!**

EX. tar –czvf nglyder-lab6.tar.gz lab6.cpp

- Name your tarred file **<username>-lab<#>.tar.gz** (ex. nglyder-lab6.tar.gz)
- Use handin (http://handin.cs.clemson.edu) to submit your archive