Lab 9: Parsing Strings

CPSC 1021 – Spring 2020

CLEMSON School of COMPUTING

Due: Friday. 3/13 before 5pm EST.

Objectives

1. Understand how to use string variables in C++.
2. Perform string processing using the methods of the string class.
3. Read input from a file.
4. Output formatted strings to the standard output.
5. Create and submit a tarball.

Part I – Introduction

Twitter and similar micro-blogging platforms allow users to broadcast short messages to a potentially large audience. Twitter messages are called “tweets” and consist of 140 characters or less. Tweets are often sent from mobile phones, and tweeters can choose whether their tweets are public or not.

The platforms allow people to communicate in ways that were not possible even a few years ago, and they have already had a profound effect on society. They even affect how society reacts to various disasters, such as fires, floods, and disease outbreaks—ordinary citizens can now broadcast timely and location specific information that is of vital importance to both emergency management agencies and other members of the public. Many users have used tweets during the recent hurricane Dorian to alert rescue workers, law enforcement, or to warn others about certain dangers/events.

Sifting through the tweets for useful information is difficult, however. As a result, there have been proposals to manually add structure to tweets sent during disasters. One proposal is called Tweak the Tweet (TtT) (you can find additional info at http://epic.cs.colorado.edu/?page_id=11 if you are curious). In TtT, information is marked using a “hashtag” such as *#loc* (for location), shown in the example below:

MT @carlseelye: #lovelandfire #Wind just switched and now the smoke is thick around our house #loc 40.352,-105.2045

Adding the structure makes it far easier for a computer to classify the tweets.

In this lab, you will use methods of the string class to process messages similar to TtT tweets. The data is made from real TtT data but it has been altered to suit the lab and make it a little bit easier to work with. You will use several string methods to pull out information from the text, manipulate it, and print it out to the screen.

Part II – The string class and methods

What is a string?

C++ provides *two types* of string representations −

- The C-style character string
  - You are already familiar with it. It uses a one-dimensional array of characters which is terminated by a *null* character '\0'. A number of functions are used to manipulate the string, such as *strcopy, strlen, strcat, strcmp*, and some others. To use the C-style you need to include the <cstring> file.

- The string class introduced with Standard C++ library.
  - This class supports all the operations that C-style string use, plus provides much more functionality. To use this class you need to include a <string> library.

  - Please visit the following link and read about the string functionality. The page provides several code examples that show how to instantiate a string, how to determine its size, etc. https://www.geeksforgeeks.org/stdstring-class-in-c/

- Please examine the string interface that shows you all the class members and methods: http://www.cplusplus.com/reference/string/string. Element access, capacity, and string operation methods are probably the most useful for this lab. Please note that some of the string methods have the same names as the class vector, and you can also create an iterator and access the front and the end of the string.

What is a stringstream?

A part of C++ Standard library, *stringstream* lives in the header file <sstream>. It associates a string object with a stream, and allows you to read from the string, as if it were a stream (like cin) using stream extraction operator >>. It also allows you to write to a string, as if it is an output stream (cout) using stream insertion operator <<.

Here is an example of how to create a stringstream object and read from it into a variable

```
string str = "12345"; // declare a string

// create an object of class stringstream using its constructor
// and pass the above string to initialize it.
stringstream ss(str);

// The object has the value 12345. Now declare an int and
// read the value from the stringstream into that int
int x;
ss >> x;

// Now the variable x holds the value 12345
cout << "Value of x : " << x;
```

And here is an example of how you can read a line from an input file and use it to initialize an stringstream object. Then you can read from the stringstream converting string into two integers.

```
ifstream ifs;
stringstream ss;
string line;

ifs.open(some_infile);

getline(ifs, line); // width and height
ss << line;
ss >> width >> height;
```

Stringstream is very useful for converting strings to numbers. You can use a *stringstream* object in your program when you are parsing the longitude and latitude. Take a look at some of the examples on this page: https://www.geeksforgeeks.org/converting-strings-numbers-cc/

Part III – Parsing the tweet

- Design a class TweetParser that will have all necessary member variables and methods. You will have *tweetParser.h* and *tweetParser.cpp* files. It is up to you to decide what members and methods you need and how you should name them. There is more than one way to write this program, and you can be as creative as you wish.

- Your main driver (*driver.cpp)* will take input file as a command line argument and pass it to the class TweetParser for parsing. Please include all sanity checks in the main, such as checking that user passed the required command line argument, and that the specified file exists. Pass the file to the TweetParser constructor. Your class will read the file, process the data, and print output to the screen.

- You will read tweets from the *tweets.txt* input file that is provided to you.
  - The tweets encode the following information using so-called “hashtags”:
    - report type (#typ);
    - some further detail (#det);
    - location (#loc) such as a street address or location description;
    - latitude (#lat);
    - longitude (#lng).

  - The tweet structure remains the same across all tweets, and the order and number of the # elements will also be the same. (Take a look at the input file)

  - Report type indicates the meaning of the tweet (i.e., whether it is a request for help, or it reports some factual information), and the report detail some provides additional information. Each of the hashtags is followed by some string value (such as an actual

latitude or longitude value). When writing your code, you can assume that all of the tweets have the following format:

#typ value; #det value; #loc value; #lat value; #lng value;

They consist of a series of hashtags, each followed by a value, and each value followed by a semicolon (;).

- Print your output to the screen in the following format.

```
Type:       OFFER
Detail:     free essential supplies 4 evacs pets.
Location:   2323 55th st. boulder
Latitude:   40.022
Longitude:  -105.226

Type:       STRUCTURE
Detail:     damaged
Location:   224 left fork road (shed) (house okay)
Latitude:   40.029854
Longitude:  -105.391055

Type:       WIND
Detail:     just switched- and now the smoke is thick around our house
Location:   40.352--105.2045
Latitude:   40.3523
Longitude:  -105.2045
```

etc.

What/How to submit

That's it, you are done! You can now submit your zipped tarball to canvas to be graded.

NOTE: If you submit the archive that cannot be open, or is corrupt, you will not be given the opportunity to redo the work and resubmit. You have one shot to get it right.