Assignment 4 Report

May 9, 2017

Program Description & Purpose This program contains the implementation of the Regex Library (Regular Expression) in order to parse a string and HTML files in order to retrieve specified data. There are essentially two files, a main.cpp and a stroustrup.txt file. The main file provides solutions to problems 3,4, & 5, and can be built and executed in order to show the desired output (the instructions for building and running the program are down below). Problem 3 in the assignment rubric asks to find and print the keywords "98" and "thanks" from a string, Problem 4 asks to find and print a line located inside an HTML's header tags, and Problem 5 asks to print all of the .ppt file names located in an HTML file (stroustup.txt).

Data Structures & Algorithm Description The primary data structure utilized in this program was the array data structure (utilized by the STL's string class & the Regex Library) in order to store our strings for Regex parsing. The HTML file (stroustrup.txt) was passed in through the I/O stream and was parsed line-by-line in order to retrieve the desired data from the HTML file. The Regex library also utilizes the array data structure in order to store the groups (or in the program's case) "matches", which are essentially just substrings acquired from the original parsed string. Internally, it is assumed that the Regex library essentially receives a regex syntax pattern, searches the input (string) for the matching content based on the syntax pattern, pulls the content or "groups" from the string as a substring, and stores the content in an array of strings for storage and later access.

I/O description

- **Problem #3 -** The input for problem #3 was a single string that read "**I would like the number 98 to be found and printed, thanks.**" The goal was to find "98" and "thanks" in the string using Regex. The output for problem #3 is shown below.
- **Problem #4 -** The input for problem #4 was a single string that was from an HTML file that read "**<html><head>Wow such a header <title>This is a title</title>" "So top</head>Much body</html>**". The goal was to retrieve only the items inside of the header tag but not inside of the title tag in the string using Regex. The output for problem #4 is shown down below.
- **Problem #5 -** The input for problem #5 was an HTML file that was converted to a .txt file in order to pass it into a stream in order to parse the file line by line to find and print out the file name of every hyperlinked powerpoint file. The output for problem #4 is shown down below.

Regex Patterns and Functions

- **Problem #3 -** The Regex pattern for problem #3 was two separate patterns "**\d\d**" and "**(thanks)**". Unfortunately I could not find a way to incorporate both into a single pattern so I had to use two separate patterns to achieve the desired result. "**\d\d**" command retrieves digits [0-9] "98" from the input string, and "**(thanks)**" command retrieves "thanks" from the input string.
- **Problem #4 -** The Regex pattern for problem #4 was **"(<head>(.*)<title>.*<\/title>(.*)<\/head>)".** This pattern essentially retrieves two substrings, the contents in between the <head> tag and the <title> tag, and the contents inbetween the </title> tag and the </head> tag, returning **"Wow such a header So top".**

- **Problem #5 -** The Regex pattern for problem #5 was "**((?:<a href=")(.+)(\.ppt))**". This pattern retrieves two substrings or "groups", (with the "**<a href="**" HTML syntax excluded), the file name, and the file extension. With the program reading in the stroustrup.txt file line-by-line from the stream, each .ppt file is printed on a new line, giving a list of all .ptt files in the HTML file.
- **Regex_search() -** The purpose of Regex_search is to search an input string for matches based off of an input syntax pattern.
- **Regex_match() -** Determines if the regular expression matches the entire string.

C++/STL Features The primary C++ native features used in the program were the STL string class, the Regex library, and the I/O stream in order to store, parse, and access various data regaurding input string, input files, and parsing said input data.

How to Compile and Run

1. In a Unix terminal, navigate to the respective source directory **PA4/** using the **cd** command.
2. Once in the **PA4/** directory type **make all** to compile the respective program.
3. Run the executable for the Regex program by typing **./main** for the Regex program.
4. Compiled on build.tamu.edu host

Conclusion Regex (Regular Expression) is a helpful tool that allows the user to parse data quickly if the user if acquainted with Regex syntax instead of parsing an input string/file/data manually. This essentially universalizes data parsing and improves program readability across the globe from user to user and is a time saver if the users are acquainted with Regex and the syntax.