Azariah Laulusa
Professor Biplav Srivastava
CSCE 240
02/02/2023

Project Assignment #1: Website Data Extractor

The goal of this project assignment was to extract data from the given CDC and WebMD websites about a specific disease of our choice. In this case, my program will extract data about the disease rabies from the CDC website "https://wwwnc.cdc.gov/travel/diseases/rabies" and the WebMD website "https://www.webmd.com/a-to-z-guides/what-is-rabies". The program takes in an html text format of both websites and outputs a text file containing the disease data and the content statistics. For the content statistics of the input and output files, the program will print out the number of words, the number of characters including spaces, the number characters not including spaces, and the number of lines. This program will only accept these specific html text files, which is why the program doesn't include user input.

Starting with the CDC website, the program has a function called **readWriteFileCDC** to read the input of the CDC html file, **rabies-cdc.txt**, and to write to an output text file, **rabies-cdc-output.txt**. The reading and writing function for CDC returns nothing and has no parameters. Before this function, the program also has two functions called **replacingSpecialChar** and **checkingCDCWebsites**. **ReplacingSpecialChar** is a function that returns a string and has a parameter of one string. This function is used to take in a string and check the string for any special characters. If there are special characters, then the function replaces the special characters with the character they represent. For example, a special character in the CDC file was *’* and the function replaces it with an apostrophe (') because that is what it represents in the html file. **CheckingCDCWebsites** is used to extract websites from each line if they exist. To do this, the function searches for *<a* and **. These characters are used in html files to enclose websites. If the function finds a website, it uses *subtr* to parse the line with a website and extracts the website data. The function encloses the websites with parentheses and quotation marks (").

In the **readWriteFileCDC** function, the function looks for the title, headers, sub-headers, paragraphs, and lists, while simultaneously counting the number of lines, words, and characters. Since the CDC html file splits its data into separate lines, we can simple search each line of the

file using a while loop and *getline*, which gets each line in the file. If a line has the characters *<title>*, then it is assumed to be the title of the file. Lines with * * and *h3* contain the data for the headers. While lines with *<p>* contain data with paragraphs, and lines with *<li>* contain data for lists. If any of these characters are found, then they will be parsed for data, print the data, and counted for the content statistics. The paragraphs are extracted as one line, so the program adds the line character */n* after a certain number of lines to cut the paragraphs into multiple lines.

To keep track of the content statistics, the program has three functions in the beginning of the program called **wordCounting**, **allCharCounting** and **charCounting**. All functions check for empty strings, returns an integer, and take in a string. The word counting function, **wordCounting**, takes in a string then, using a for loop, adds to the word count after it identifies a space or break in the line. Then, the function returns the amount it calculated. The character counting function, **charCounting**, works in a similar way except if the function identifies a space, break or indention, it will not add to the count. It will only add to the count if it's a character not including spaces. The all-character count function, **allCharCounting**, counts all characters in the string, including integers. To count for lines in the input and output files, the program just adds to the line count variable, **lineCount**, every time a line is taken in and adds to the output line count variable, **outLineCount**, every time a line is printed to the output file.

Moving on to the WebMD website, I created a function called **readWriteFileWebMD** to read the input of the WebMD html file, **rabies-webmd.txt**, and to write to an output text file, **rabies-webmd-output.txt**. The read and write function for the WebMD website does not return anything and has no parameters. After opening the input file and output file, the function uses a while loop to get each line of the WebMD html file. In the WebMD html file, all of data containing paragraphs, headers and lists are in sections, and all the sections are in extremely long lines. Before finding the sections, the program can find the title by looking for *<title>* and the main header by looking for *<h1>* and obtain their data. After getting the data for the title and the main header, the function will search through the long lines. First, the program counts the number of sections using the **sectionCount** variable. Then, the program uses a while loop to look for lines with sections using the characters *<section>*. Each time the program finds a section, it deletes it to avoid looking at the same section. If the program reaches the amount sections it counted, then the loop ends.

In the while loop, if a line contains a section, then the program looks for paragraphs, headers, and lists in the line. Like the **readWriteFileCDC** function, the **readWriteFileWebMD** function will look for headers using *<h2>*, look for paragraphs using * *, and look for lists using *<li>*. Before finding this data, since all most of the data is in one line, the program looks for and replaces website data with just the website URL. Some websites correspond with words, so the program also includes the word with the website. When looking for paragraphs, headers, and lists, the program uses while loops because there are multiple forms of that data in one line. Every time data is found, it is printed to the output file and it is also deleted from the line to avoid reading the same data over again. For example, you might have a line that has * words another word *. So, the program makes sure to look for all possible paragraphs, headers, and lists. At the end of the **readWriteFileCDC** function and the **readWriteFileWebMD** function, the program prints the content statistics to the output files.

In conclusion the program was able to extract data from two websites and print the data to output files. In the output files, the title is in the first line. Headers are indented with asterisks (*), and sub-headers are only indented. Paragraphs are indented once, and lists are indented with a dash (-). Websites have parentheses and quotation marks (“). Lastly, the content statistics are printed at the bottom of the files. I enjoyed this project assignment because of its freedom. In this assignment, I was able to exercise my problem-solving skills and learn more about C++. I am excited to learn even more about C++ and refine my programming skills more and more.

Works Cited

Editorial Contributors, WebMD. “Rabies: 9 Symptoms & What Do If You Are Bitten by a Rabid Animal.” *WebMD*, WebMD, 10 June 2022, https://www.webmd.com/a-to-z-guides/what-is-rabies.

“Rabies.” *Centers for Disease Control and Prevention*, Centers for Disease Control and Prevention, 5 May 2022, https://wwwnc.cdc.gov/travel/diseases/rabies.