Jeff DiPaola
CS 214
10/20/14

Assignment 3 – Overview

When index is called the two arguments are passed to the main function where it is checked that they are proper and they are put into character arrays. The input file is then passed to a function called “FileOrDir” which checks to see if the input is a file or a directory. If it is a directory, it is then passed by “FileOrDir” to “DirectorySearcher” where it is traversed for subdirectories and files. Each encountered file or directory will invoke a recursive call to the “FileOrDir” function, thus repeating the process.

If the input file, or an encountered file, is found to be an actual file and not a directory, it is passed to the function “FileReader”. This function opens the file for reading then uses a while loop to grab a single character at a time. Each character is checked to see if it is a letter or number; if it is, it’s added to a character array, if not it’s deemed a separator and the character array is passed to the function “WordAdder” and all the values are reset to traverse to the next word. Additionally, we account for the last word “escaping” by having a check outside the while loop.

Once the word is passed to the “WordAdder” function most of our heavy lifting is about to take place. The main struct is declared as a global variable to facilitate saving each word/file combination as well as to make it easy to access it throughout the program. The struct itself, called “List” consists of a char array “word”, a List next pointer, and a “FileList” structure. The “FileList” structure is basically set up the same but instead of holding the words, holds the file information for each word (word, count, and a next pointer). WordAdder determines if this word is present, if it has been seen with this filename before, and will take care of all the necessary additions and changes ensuring that the words are in alphabetical order and the file names are in chronological order.

Finally, once all of these files have finished being called, the process returns back to the main function. The primary struct “finalout” is declared as a global variable, so function “OutCreator” is called, passing it the name given to use for the output file. “OutCreator” first checks for the existence of the output file name, and if it does exist prompts for an overwrite. Once the file is established the program uses a set of nested while loops, one to traverse the list of words and one to traverse the sub-list of files to write the information to the output file in the specified format.

Since the final structure is the only actual memory being taken up, we can say that the program has about an O(n) memory space where n is the number of words multiplied by the number of files. The run time also probably is about O(n) considering on a worst case scenario all words are traversed to add a final word (something like zzzzz)