The basic idea of the program focuses on two cases: direct file input and directory entry input.
If file entry input, we process the file for words, write the words and file counts to xml and then end program. If directory input, we open the directory and for each file entry inside the do the same as the previous case. If there is another directory entry, we recursively call the directory process.

The main unit of analysis lies in processFile, which opens a file and finds an instance of an alpha character. We then count the number of alphanumeric characters following to find the length of the word, and then copy the word into an array. If the word spans the range of the buffer, the word is buffered into a carry array and then the next segment of the file is processed, considering the carried elements. Once a word is found, we update the data structure containing words.

Words are left in a linked list of word objects which have the text of the word, and another linked list of file structures which have the individual counts for that file for that word.

Once all documents and directories have been processed, words are sorted via selection sort, and then for each word, files are sorted via selection sort.

Once everything is sorted, it is simple as writing out to the .xml file.

As far as time analysis goes, files are processed in O(n) time where n is the number of characters in the file. Words are added in linked list manner, O(n) insertion time in list for a word.

Sorting each list (both word and file for word) is done using selection sort and runs in $O(n^2)$.

Space complexity, we need a large amount of space for document buffering as well as carry over array. Large amount of data to hold the linked list of words with linked lists of files.