Project 4

Concordance of a string using Linked List

Full Name: Prashul Shrestha

Section #: 2

Project #: 4

Due Date: February 21, 2017.

**Design** **Document**

**Introduction**

A **concordance** of a text is an alphabetical table of the words that appear in the text and the number of times each word appears. Concordances summarize the frequencies of words in text and are used in statistical analyses of authors' works and to determine authorship of disputed works. For example; In this sentence, “A dog is a loyal animal”. Here, the concordance of the word ‘a’ is two and the number of words are six which helps us get accurate content of a sentence.

**Data** **Structures**

The program uses a class called Class **List**(), an char **array**[], string **array**[] and a linked **list** to store the each word of the sentence taken from the file. First, each word is taken and stored in a char array where it is filtered for characters other than alphabets and stored into another string and the string is further taken into the original string **array**[]. The elements from the array is taken and checked for its repetition in the list using **is**\_**Present**() function and then stored in the list using node and **get\_node** function.

**Functions**

The program uses **Six** functions to implement the concordance using the linked List. The functions are called from main() and some are member function to return the result within the function which called it. The list of the functions are given below:

* List() : A constructor to initialize the list to be empty.
* ~List(): A destructor to delete nodes in the list and is called by default.
* **Void insert**(**word**) – This will insert word in the invoking concordance in the correct position. If the word is already in the concordance, increment its count.
* **Int** **get\_count**(**word**) – This will return the count associated with **word** in the invoking concordance. This function returns zero if **word** is not in the concordance.
* **Int** **length**() – This function will returns the length of the invoking concordance; that is, the number of distinct words that is present in the lists.
* Output - Overloads the "<<" operator to write the invoking concordance to an output stream.
* **get**\_**node**(**word**,**count**,**link**) - Returns a pointer to a new node that contains **word**, **count**, and the pointer **link**. This function will be **private**

The program uses **string**.**size**() from **string**.h library to get the length of the string. It also uses **isalpha(ch)** function from the **c++** library to check the string is not space and remove the non-alphabetical character from the string sentence.

**Menu**() is a the main function from where we first create an object or an instance of class list l and which helps invoke the functions such as l.insert(word); and l.get\_count() to insert the sentence and get the count of the word.

**The Main Program**

User Document

A **concordance** of a text is an alphabetical table of the words that appear in the text and the number of times each word appears. Concordances summarize the frequencies of words in text and are used in statistical analyses of authors' works and to determine authorship of disputed works. For example; In this sentence, “A dog is a loyal animal”. Here, the concordance of the word ‘a’ is two and the number of words are six which helps us get accurate content of a sentence.

The main program named Recursive.cpp can be compiled and run, using the code:

g++ Linked.cpp

a.out

**g++** function will compile the function and make it ready to be run using **a.out**. The function will prompt the following output:

**Ouptut:**

:- The user will prompt the computer as per their desired command and the program will respond accordingly.

**Summary**

Completing this project, I learnt the use of linked list in the real-world experience. Using pointers to create a linked list of a sentence and check its concordance made me help better understand pointers and linked list.

In this program, I used char **array** [] of length 8 to store each word from the text up to only eight elements of the word and used **isalpha**(ch) to check for characters and store it in the array and later store into the string and store it into another string array[]. From the string array, each element is first checked using is\_Present() function and if present it will implement the count with the word and if it is not present than the word is concorded into the linked list.