**Project-4**

**IMPLEMENTING A CONCORDANCE WITH A LINKED LIST**

**Name: Pawan Dhungana**

**Project no: 4**

**Due Date: October 4, 2018**

**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. This project uses a class to implement a concordance.

**Data Structures**

The program makes use of class named Concordance to implement a concordance. Word is used as the data type of the items in the concordance using typedef. One constructor is used : Concordance() which is an inline constructor.

**Functions**

Five functions are used in the program: insert(), length(), get\_count(), get\_node()and a friend function. Insert() inserts the given word into the concordance, length() returns the number of words in the concordance, friend function writes write the invoking concordance to an output stream; that is to write out the concordance. Besides these, there is a function called read\_file() that reads the contents of the file and store into an array of strings.

**Main Program**

In the main program, all these functions are being called. The user can know the count of the distinct words from the execution of the main program. In the main programs, an object of the class is created through which different functions in the class are called.

**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.

The program's name is Project4.cpp, to compile and run it, simply enter:

g++ Project4.cpp

a.out

If an input file is this:

This is a small

test file, containing a small

number of words.

then the corresponding output might look something like this:

Word Count

---------------

A 2

CONTAINI 1

FILE 1

IS 1

NUMBER 1

OF 1

SMALL 2

TEST 1

THIS 1

WORDS 1

---------------

The file contains 10 distinct words.

**Code Listing:**

#include <cstdlib>

#include <iostream>

#include <cstring>

#include<fstream>

using namespace std;

const int MAX = 8;

class Concordance

{

public:

// TYPEDEF

typedef char Word[MAX+1];

// CONSTRUCTOR

Concordance( ) { first = NULL; } // Inline function

// MODIFICATION MEMBER FUNCTION

void insert ( const Word word );

// CONSTANT MEMBER FUNCTIONS

std::size\_t length( );

std::size\_t get\_count(Word word);

// FRIEND FUNCTION for the List class:

friend std::ostream& operator << ( std::ostream& out\_s,const Concordance& c );

// DATA MEMBERS

struct Node

{

Word wd;

int count;

Node \*next;

};

Node \*first;

private:

// PRIVATE FUNCTION

Node\* get\_node ( const Word word, int count, Node\* link );

};

void read\_word(Concordance c);

Concordance::Node\* Concordance::get\_node(const Word word, int c, Node\* link)

{

Concordance::Node\* temp=new Node;

temp->count=c;

temp->next=link;

strcpy(temp->wd,word);

return temp;

}

//retuns how many times the repeats in the list

size\_t Concordance::get\_count( Word str)

{

Node \*temp;

if(first==NULL)

return 0;

else

{

temp=first;

while(temp!=NULL)

{

if(strcmp(temp->wd,str)==0)

return temp->count;

temp=temp->next;

}

}

return 0;

}

//prints the words and returns the length of the list

size\_t Concordance::length()

{

int l=0;

Node \*temp;

cout<<"Word"<<"\tCount"<<endl;

cout<<"-------------------------"<<endl;

if(first==NULL)

return 0;

else

{

temp=first;

while(temp!=NULL)

{

//printing the words in the list

cout<<temp->wd<<"\t"<<temp->count<<endl;

temp=temp->next;

l++;

}

cout<<"--------------------------"<<endl;

return 1;

}

}

//to insert the given words into concordance

//also checks if the word is already in the list

//if not increase the count of the word in the concordance

void Concordance::insert(const Word word)

{

Node \*newNode,\*temp,\*prev;

//if there are list is empty

if(first==NULL)

{

first=get\_node(word,1,NULL);

}

//if the list is not empty

else

{

prev=temp=first;

while(strcmp(temp->wd,word)<=0 && temp->next!=NULL)

{

prev=temp;

if(strcmp(temp->wd,word)==0)

{

temp->count=temp->count+1;

return;

}

temp=temp->next;

}

if(temp==first && strcmp(temp->wd,word)>0)

{

newNode=get\_node(word,1,first);

first=newNode;

}

else if(temp==first && strcmp(temp->wd,word)<0)

{

newNode=get\_node(word,1,NULL);

first->next=newNode;

}

else

{

if(temp->next==NULL)

{

if(strcmp(temp->wd,word)<0)

{

newNode=get\_node(word,1,NULL);

temp->next=newNode;

}

else

{

newNode=get\_node(word,1,temp);

prev->next=newNode;

}

}

else

{

newNode=get\_node(word,1,temp);

prev->next=newNode;

}

}

}

}

//main function

int main()

{

Concordance c;

read\_word(c);

cout<<"The file contains "<<c.length()<<" distinct words."<<endl;

return 0;

}

//read the characters in the file to make a word and insert the word into list

void read\_word(Concordance c)

{

ifstream in\_f;

in\_f.open("a.txt");

char ch;

char w[9];

int i=0;

while(in\_f.get()

{

w[0]=ch;

cout<<w[0];

ch=toupper(ch);//converts lowercase to uppercase

if(int(ch)>=65 && int(ch)<=90)

{

//saves only 8 characters

if(i<8)

w[i]=ch;

i++;

}

else

{

if(i>=8)

w[8]='\0';

else

w[i]='\0';

//insert the word

if(i!=0)

c.insert(w);

w[0]=w[1]=w[2]=w[3]=w[4]=w[5]=w[6]=w[7]=w[8]='';

i=0;

}

}

return;

}

**Test Document**

If an input file is this:

This is the first of the

test file, containing some of

the words to test.

then the corresponding output might look something like this:

Word Count

---------------

THIS 1

IS 1

THE 3

FIRST 1

OF 2

TEST 2

FILE 1

CONTAINI 1

SOME 1

TO 1

WORDS 1

---------------

The file contains 11 distinct words.

If an input file is this:

The second test

of the file, containing some

words, looks like the previous test.

then the corresponding output might look something like this:

Word Count

---------------

THE 3

SECOND 1

TEST 2

OF 1

FILE 1

CONTAINI 1

SOME 1

WORDS 1

LOOKS 1

LIKE 1

PREVIOUS 1

---------------

The file contains 11 distinct words.

If an input file is this:

This is a small

test file, containing a small

number of words.

then the corresponding output might look something like this:

Word Count

---------------

A 2

CONTAINI 1

FILE 1

IS 1

NUMBER 1

OF 1

SMALL 2

TEST 1

THIS 1

WORDS 1

---------------

The file contains 10 distinct words.

**Summary**

In this project, we implemented a program that implements a concordance of a text using a class. We a program to provide the user to input the file containing the words. The program calculates the count of the repetition of the words and calculates the number of distinct words from the execution of the main program.

From this project, I learned how to implement a concordance class with a linked list. This project helped me learn more clearly about how the string and array of characters work in a linked list program.