**Project-5**

**PROCESSING A LINKED LIST RECURSIVELY**

**Name: Pawan Dhungana**

**Project no: 5**

**Due Date: October 11, 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. The destructor is ~Concordance().

**Functions**

The program uses Six functions to implement the Concordance through Singly-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:

• Concordance() : A constructor to initialize the list to be empty.

• ~Concordance(): A destructor to delete nodes in the list and is called by default.

• Void insert(ifstream& in\_File) – This will help private insert function to send one word as char array in private insert() function.

• Bool is\_empty(): checks if the linked list is empty or not.

• Output - Overloads the "<<" operator to write the invoking concordance to an output stream.

• get\_node(entry, node p, int count) – Returns or initializes new node to store the char array in to singly linked list.

The program uses string.size() from string.h library to get the length of the string. It also uses strcmp() and strcpy() to compare the char array and the string, and copy one char array into another.

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

**Main Program**

Menu() is a the main function from where we first create an object or an instance of class Cord and which helps invoke the functions such as Cord.insert(in\_File); to input characters into the node. The program is quite simple first, the strings are inserted into the node while checking through the list using strcmp() and inserting if it’s not present or inserting and incrementing the count while removing the inserted same string.

**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++ Project5.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>

#include<iomanip>

using namespace std;

const int MAX = 8;

void readword(ifstream& in\_f, char (&array)[MAX+1]);

void buildlist(ifstream& in\_f, char(&array)[MAX+1]);

class Concordance

{

public:

// TYPEDEF

typedef char Word[MAX+1];

// CONSTRUCTOR

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

//Destructor

~Concordance();

// MODIFICATION MEMBER FUNCTION

void insert (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 );

private:

// DATA MEMBERS

struct Node

{

Word wd;

int count;

Node \*next;

};

Node \*first;

// PRIVATE FUNCTION

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

void insert(Node\* p, Word word);

size\_t length(Node \*p);

size\_t get\_count(Node \*p, Word word);

void write(ofstream &out\_s, Concordance c);

};

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

{

Node\* temp;

temp=new Node;

strcpy(temp->wd,word);

temp->next=link;

temp->count=count+1;

return temp;

}

Concordance::~Concordance()

{

Node \*temp;

while(first != NULL)

{

temp = first;

first = first->next;

delete temp;

}

}

//to insert the given words into concordance

void Concordance::insert(Word& word)

{

insert(first,word);

}

void Concordance::insert(Node \*p, Word word)

{

if(p==NULL)

p=get\_node(p,word);

else

insert(p->next,word);

}

//retuns how many times the repeats in the list

size\_t Concordance::get\_count( Node \*p, Word& word)

{

if(p==NULL)

return 0;

else

{

strcmp(p->wd,word);

return get\_count(p->count,word);

}

}

size\_t Concordance::get\_count(Word& word)

{

return get\_count(first,word);

}

//recusive function

size\_t Concordance::length(Node \*p)

{

if(p==NULL)

return 0;

else

return length(p->next);

}

//returns the length of the list

size\_t Concordance::length()

{

return length(first);

}

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

{

c.write(out\_s,first);

return out\_s;

}

void Concordance::write(ofstream &out\_s, Node\*p)

{

if(p!=NULL)

{

out\_s<<p->data;

write(out\_s,p->next);

}

}

//read the characters in the file to make a word

void readword(ifstream& in\_f, char (&array)[MAX+1])

{

char ch;

int i=0;

while(isalpha(ch) && !isspace(ch))

{

if(i > MAX-1)

{

while(!isspace(ch))

in\_f.get(ch);

break;

}

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

array[i] = ch;

i++;

in\_f.get(ch);

}

for(int j=0; j<2; j++)

cout<<array[j];

cout<<endl;

}

void buildlist(ifstream &in\_f, char (&array)[MAX+1])

{

Concordance c;

int x=0;

while(!in\_f.eof())

{

readword(in\_f,array);

c.insert(array,x);

}

}

//main function

int main(int argc, char \*argv[])

{

Concordance c;

char file[100];

ifstream in\_f;

char array[MAX+1];

cout<<"File name: ";

cin>>file;

in\_f.open(file);

buildlist(in\_f,array);

cout<<"Word"<<setw(10)<<" "<<"Count"<<endl;

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

cout<<c; //write out the words in the concordance

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

in\_f.close();

return EXIT\_SUCCESS;

}

**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 did the iterative version in the last project, but we are using recursive calls to do the same task as last project. The program provides 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. The last project helped me learn more clearly about how the string and array of characters work in a linked list program. And this project enhanced my knowledge from last project. I learned how to implement linked list recursively.

Other thing, that I learnt while doing this project is that, while we pass a char array[], a pointer cannot point to the array as whole, so we need to use strcpy() to copy the elements of the array into another.

Working with pointers and their semantics is helping to broaden my knowledge on the use of pointer in making complex program into simpler versions.