**Sandesh Ghimire**

**CSCI 301 section 3**

**Computer Science 2**

**Project 3**

**20th September**

**Introduction**

An ordered list is a sequence of elements, all of the same type, that is ordered by the elements ' values. One way to represent an ordered list is in a linked list. This project contains three files, " *List.h* “, " *List.cpp* " and " *main.cpp* “. The program initializes the linked list and displays menu to the user. The user can manipulate the linked list by performing one of the functions mentioned in the menu. The user is asked to enter what action does he want to be performed next and the following action is performed.

**Data Structures:**

The data structures used in this project is pointers. The pointers are used in order to initialize the linked list and access the contents of the linked list.

**Functions:**

The program contains a class named List in the header file " *List.h* “. The class list contains many public functions that are used to manipulate the linked list according to the user's decision the various function to manipulate the linked list are as follows:

*List() {first = NULL;}*: This is the constructor for the class List.

*~List():* This is the destructor for the class List.

*void make\_empty():* This function re - initializes an existing List to be empty.

*void insert (Item empty):* This function inserts an into a List.

*void remove (Item target):* This function removes a value from the List.

*bool empty():* This function checks if the list is empty.

*length ():* This function returns the length of a list.

*bool present(const Item& target):* This function checks if an item exists on the list or not.

*Item kth (Item position):* This function returns the value of item in kth position.

*void Display:* This function prints the existing list.

*friend std::ostream& operator << (std::ostream& out\_s,const list& l):* This is a friend function to overload the “<<” operator.

**The Main Program:**

The main program utilizes the class and print outs the menu as well as selects the cases and performs actions accordingly.

At first, the main program calls the *display()* function to display the menu. Then the program asks the user to enter the desired operation. The program then selects the case that the user inputs and performs the actions accordingly. The programs continue to run until the user enters the letter " q ".

**Code**

**Main.cpp**

#include <iostream>

#include <cassert>

#include "list.h"

#include "list.cpp"

void display();//declaring the function

using namespace std;

void display(){ //function to display menu

cout<<"This program responds to commands the user enters to manipulate an ordered "<<endl<<

"list of integers, which is initially empty. In the following commands, k is a position in the list, and v is an integer."<<endl;

cout<<"i v -- Insert the value v into the list."<<endl;

cout<<"r v -- Remove the value v from the list."<<endl;

cout<<"m -- Is the list empty?"<<endl;

cout<<"l -- Report the length of the list."<<endl;

cout<<"p v -- Is the value v present in the list?"<<endl;

cout<<"k k1 -- Report the k1th value in the list."<<endl;

cout<<"w -- Write out the list."<<endl;

cout<<"h -- See this menu."<<endl;

cout<<"q -- Quit."<<endl;

}

int main(){ //The main function

list l;

display();

char choice;

int num;

bool quit = false;

while(choice != 'q') //using loop to stay on the program

{

cout<<" --> ";

cin>>choice;

switch (choice)

{

case 'i':

cin >> num;

l.insert(num);

break;

case 'r': //if choice is r call remove

{

cin>>num;

l.remove(num);

break;

}

case 'm': //check if empty

if (l.empty())

cout << " The list is empty."<<endl;

else

cout << " The list is NOT empty."<<endl;

break;

case 'l': //if choice is l call length

{

int len=l.length();

cout<<"The list contains "<<len <<" elements"<<endl;

break;

}

case 'p': //if choice is p call present

{

cin>>num;

if (l.present(num)){

cout<<"The value "<<num<<" is present in the list"<<endl;

}

else{

cout<<"The value "<<num<<" is NOT present in the list"<<endl;

}

break;

}

case 'w': //if choice is w call display

{

cout<<"The list: (";

l.Display();

cout<<")"<<endl;

break;

}

case 'k': //give the value of kth element

cin >> num;

if ((l.empty()) || (num > l.length()))

cout << " Not the valid selection."<<endl;

else

cout<<" The "<<num<<"th element of the list is: "<<l.kth(num)<<endl;

break;

case 'h': //if choice is h call display menu

{

display();

break;

}

case 'q':{ //if choice is q call quit the program

quit=true;

break;

}

default:{ //if entered extra then menu

cout<<"Wrong choice"<<endl;

break;

}

if (quit) //if not quit

{

break;

}

cin>>choice;

}

}

}

**List.cpp**

#include <iostream>

#include <cassert>

#include "list.h"

using namespace std;

list::~list() //distructor

{

Node\*temp; //initalizing the temp pointer

while(first != NULL){ //using precondition loop

temp=first;

first=first->next;

delete temp;

}

}

void list::Display(){

Node \* temp; //declare the variable

if (first==NULL){cout<<"The list is empty.";} //pre condition

else{

int i=0;

temp=first;

while (temp != NULL){//post condition

i++;

cout<<temp->data<<", ";

temp=temp->next;

}

}

}

void list::make\_empty(){ //function to make list empty

Node\*temp;//declaring the variable

while(first != NULL){//looping

temp=first;

first=first->next;

delete temp;

}

}

void list::insert(Item entry){//function to insert

Node\*prev;

if(first==NULL||entry<first->data){first=get\_node(entry,first);}//pre condition

else{

prev=first;

while(prev->next != NULL && prev->next->data<entry){

prev=prev->next;}//post condition

prev -> next = get\_node(entry, prev->next);

}

}

void list::remove(Item target)//to remove

{

Node \* prev; //declare pointers

Node \* temp;

prev=first;

if(first->data==target){//pre condition

first=first->next;

delete prev;

}

else{

while(prev->next->data != target){//post condition

prev=prev->next;}

temp= prev->next;

prev->next=temp->next;

delete temp;

}

}

bool list::present(const Item& target){

Node\* p;

p=first;

while(p != NULL && p->data != target){//loop to get to that value

p=p->next;}

return (p!=NULL);}

int list::length(){ //check the length

Item count = 1;

Node \* temp;

temp = first;

while (temp->next != NULL){ //starting the loop by increasing counter

temp=temp->next;

count++;//increasing the counter according to condition

}

return count;

}

ostream& operator<<(ostream& out\_s, const list& b){ //overloading the << operator

list::Node\*n;

n= b.first;

cout<<n->data<<"";//displaying if null

while (n->next != NULL)

{

n=n->next;

out\_s<<n->data<<"";//displaying if data are present

}

return out\_s;

}

bool list::empty(){ //check if list is empty or not

if(first==NULL){//pre condition

return true;

}

else{//post condition

return false;

}

}

list::Node\* list::get\_node(const Item value, Node\* link){

Node\*temp;

temp= new Node;

temp->data=value;

temp->next=link; //getting the value of particular node

return temp;

}

int list::kth(int target)//finding the kth item from list

{

Node\*p;

int Count=1;

p=first;

while(p != NULL && p->data != target&& Count<target){//pre condition

Count++;

p=p->next;

}

return p->data; //returning this value to print on the screen

}

**List.h**

#include <iostream>

#ifndef LIST\_H

#define LIST\_H

using namespace std;

class list{

public://public functions of the class

typedef int Item; //defining the Item using int

//Item first;

list(){first=NULL;}; //constructor

~list();//Destructor

void make\_empty();//to make list empty

void insert (Item Empty);//to insert the data in the list

void remove(Item target);//to remove the data from the list

//Member functions

bool Empty(){ //checks if the list is empty or not

return first=NULL;}

int length();//finds the length of list

bool present(const Item& target);// finds if the item is present in list or not

bool empty();

Item kth(Item position);//finds the value of kth position

void Display();//prints out the contents of list

friend std::ostream& operator << (std::ostream& out\_s,const list& l); //this is a friend function

private://private functions of class

struct Node //Making a Node for pointing the data of list

{

Item data;//data

Node\* next;//pointer

};

Node\* first;//It points first data of list

Node\* get\_node(Item Empty, Node \*link);//to get the data in list

};

#endif // list\_H

**User Document**

The program can be used by the user to initialize a linked list. The user will be able to reinitialize the list to be empty, insert values inside the list, remove values from the list, display the kth term in the list, check if a value is present in the list, display the length of the list. To compile the program user can open the executable file. As soon as the .exe file opens, the user will be asked to enter the desired action to be taken. The user will be able to manipulate through the list by using special codes which are available in the menu, are available in the menu.

**Testing**

-Test 1

This program responds to commands the user enters

manipulate an ordered list of integers, which is

initially empty. In the following commands, kl is a

position in the, and v is an integer.

e – Re-initialize the list to be empty.

i v -- Insert the value into the list.

r v -- Remove the value from the list

m -- Is the list empty?

l – Report the length of the list.

p v -- Is the value v present in the list?

k kl -- Report the kth value in the list.

w -- write out the list.

h -- See this menu.

q – Quit.

-->i 27

-->i 42

-->i 15

-->i 33

-->i 14

--> m

This is NOT empty.

--> w

List: (14,15,27,42 )

--> p 22

The value is NOT present in the list.

--> k 3

The 3th element of the list is 27

--> q

-Test 2

This program responds to commands the user enters

manipulate an ordered list of integers, which is

initially empty. In the following commands, kl is a

position in the, and v is an integer.

e – Re-initialize the list to be empty.

i v -- Insert the value into the list.

r v -- Remove the value from the list

m -- Is the list empty?

l – Report the length of the list.

p v -- Is the value v present in the list?

k kl -- Report the kth value in the list.

w -- write out the list.

h -- See this menu.

q – Quit.

-->i 11

-->i 9

-->i 1

-->i 56

-->i 35

--> m

This is NOT empty.

--> w

List: (1,9,11,35,56 )

--> p 31

The value 31 is NOT present in the list.

--> p 9

The value 9 is present in the list.

--> k 3

The 3rd element of the list is 11

--> q

**Summary**

In this project, we have implemented a program that initializes a list and allows the user to manipulate the list. The program reads the special code from the user. The user can know about the special codes from the menu which is displayed to the user as soon as the program starts. The user can then insert remove and perform many other included actions to the list along with printing the list. I used three files to keep my codes organized and easy to access. The problems and their solutions involving linked list that we learnt in class helped a to solve the given question.