Doniyorkhon Sobirov

Professor: Shweta Jain

CSCI 373

September 25, 2016

Assignment #2

**Instructions: The main objective of this assignment is to practice the linked list concept.**

(A) You are going to write a class called Prof that stores a professor's name, office location (room number) and phone number. This part is very easy.

(B) You will write a linked list class that can be used to maintain a sorted list of professors. We talked about this in class in details. Now you have to read and implement. This class should

1) sort the names in alphabetical order of the last names. If there is a tie, look at the first name to break the tie.

2) should have the capability to both insert and delete.

3) search for a professor in the list by the full name. (Other keys for the search is possible but not necessary to implement because the essential functionality is same.).

4) should be able to display the list. Overload the insertion operator (<<) so that you can both display the list on the screen or save it in a file. The display should look "nice". Choose a format that is pleasing to the eye.

 5) Overload the extraction operator (>>) so that you can receive inputs from the console or read from a file.

 (C) Write a driver program that uses the list class to enter data and to display it.  The driver program should be able to receive data from a file or user input. (It is essentially calling the overloaded operator provided by the list class).

(D) Test using the list of professors in the Math and CS department at JJ.

**Answers:**

**Header files:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*

\* File: LinkedList.h

\* Author: Doniyorkhon Sobirov

\*

\* Created on September 25, 2016, 10:51 AM

\*/

#ifndef LinkedList\_h

#define LinkedList\_h

#include <iostream>

#include <string>

#include "Prof.h"

using namespace std;

struct Node {

Prof prof;

Node \*next;

};

class LinkedList{

private:

Node \* head;

int length;

// public members

public:

// Default Constructor

LinkedList();

bool insertProfessor( Node \* newNode, int position );

bool removeProfessor( int position );

void printProfessors();

void sortProfessors();

void searchProfessors(string name);

bool insert( Node \* newNode);

// Destructor

~LinkedList();

};

#endif /\* LINKEDLIST\_H \*/

**Header files:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*

\* File: Prof.h

\* Author: Doniyorkhon Sobirov

\*

\* Created on September 25, 2016, 12:41 PM

\*/

#ifndef PROF\_H

#define PROF\_H

#include <iostream>

#include <string>

using namespace std;

class Prof

{

public:

string first\_name;

string last\_name;

int room\_number;

string phone\_number;

};

#endif /\* PROF\_H \*/

**Main files:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*

\* File: main.cpp

\* Author: Doniyorkhon Sobirov

\*

\* Created on September 25, 2016, 12:51 PM

\*/

#include <cstdlib>

#include <string>

#include <iostream>

#include "LinkedList.h"

#include "Prof.h"

using namespace std;

LinkedList professorsList;

int main()

{

LinkedList professorsList;

int choice=0;

cout<<"Please select your choice.\n 1. Insert.\n 2.Search.\n 3.Delete.\n 4.Exit."<<endl;

cin >> choice;

string keyword="";

int position;

switch(choice){

case 1:

cout<<"Please select you source of input.\n 1.Console.\n 2.File.\n";

cin >> choice;

break;

case 2:

cout<<"Please the name you would like to search.\n";

cin >> keyword;

professorsList.searchProfessors(keyword);

break;

case 3:

cout<<"Please the position of the professor you would like to delete.\n";

cin >> position;

professorsList.removeProfessor(position);

return 0;

break;

case 4:

exit(0);

return 0;

break;

}

return 0;

}

void getInput(int choice){

string first\_name;

string last\_name;

string phonenumber;

int room\_number;

string filename;

Node \* professor = new Node;

switch(choice){

case 1:

cout<<"Please enter the professor's first name.\n";

cin >> first\_name;

cout<<"Please enter the professor's last name.\n";

cin >> last\_name;

cout<<"Please enter the professor's phone number.\n";

cin >> phonenumber;

cout<<"Please enter the professor's room number.\n";

cin >> room\_number;

professor->prof.first\_name = first\_name;

professor->prof.last\_name = last\_name;

professor->prof.phone\_number = phonenumber;

professor->prof.room\_number = room\_number;

professorsList.insert(professor);

break;

case 2:

cout<< "Enter the file name.\n"<<endl;

cin>>filename;

break;

}

}

**Source files:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

#include "LinkedList.h"

#include "Prof.h"

// Default Constructor creates the head node.

LinkedList::LinkedList()

{

head -> prof;

head -> next;

length = 0;

}

bool LinkedList::insertProfessor( Node \* newNode, int position )

{

if ((position <= 0) || (position > length + 1))

{

cout << "Error: No such position\n";

return false;

}

if (head -> next == NULL)

{

head -> next = newNode;

length++;

return true;

}

int count = 0;

Node \* p = head;

Node \* q = head;

while (q)

{

if (count == position)

{

p -> next = newNode;

newNode -> next = q;

length++;

return true;

}

p = q;

q = p -> next;

count++;

}

if (count == position)

{

p -> next = newNode;

newNode -> next = q;

length++;

return true;

}

cout << "Error: The professor was not added to the list.\n";

return false;

}

void LinkedList::sortProfessors()

{

Node \* p = head;

Node \* q = head;

Node \* k = head;

Node \* m = head;

int count = 0;

while(count<length){

while (q)

{

p = q;

k = p -> next;

m = k -> next;

if(p -> prof.first\_name > k->prof.first\_name){

p->next = m;

k->next = p;

} else if(p -> prof.first\_name == k->prof.first\_name){

if(p -> prof.last\_name > k->prof.last\_name){

p->next = m;

k->next = p;

}

}

q = p -> next;

}

count++;

}

}

void LinkedList::searchProfessors(string name){

bool found;

Node \* p = head;

Node \* q = head;

cout << "Professors";

cout << "\n---------------------------\n";

while (!found)

{

p = q;

if((p->prof.first\_name==name) || (p->prof.last\_name==name)){

cout << "Found!!\n"<< endl;

cout << "\t" << p -> prof.first\_name<<" "<<p -> prof.last\_name << endl;

cout << "\tPhone number: " << p -> prof.phone\_number << endl;

cout << "\tRoom Number: "<< p -> prof.room\_number << endl;

cout << "\n";

found = true;

}

q = p -> next;

}

}

bool LinkedList::removeProfessor( int position )

{

if ((position <= 0) || (position > length + 1))

{

cout << "Error: No such position.\n";

return false;

}

if (head -> next == NULL)

{

cout << "Error: The list is empty.\n";

return false;

}

int count = 0;

Node \* p = head;

Node \* q = head;

while (q)

{

if (count == position)

{

p -> next = q -> next;

delete q;

length--;

return true;

}

p = q;

q = p -> next;

count++;

}

cout << "nError: No one was removed.\n";

return false;

}

void LinkedList::printProfessors()

{

Node \* p = head;

Node \* q = head;

cout << "Professors";

cout << "\n---------------------------\n";

while (q)

{

p = q;

cout << "\t" << p -> prof.first\_name<<" "<<p -> prof.last\_name << endl;

cout << "\tPhone number: " << p -> prof.phone\_number << endl;

cout << "\tRoom Number: "<< p -> prof.room\_number << endl;

cout << "\n"<<endl;

q = p -> next;

}

}

bool LinkedList::insert( Node \* newNode){

while(false){

if (head -> next == NULL)

{

head -> next = newNode;

length++;

return true;

} else {

Node \* next = head -> next;

head=next;

false;

}

}

}

LinkedList::~LinkedList()

{

Node \* p = head;

Node \* q = head;

while (q)

{

p = q;

q = p -> next;

if (q) delete p;

}

}

**Screen shots:**

