Movies

Collin Worth

November 2023

1 Introduction

```
/*
    Computer Science 2
    Main File

    Collin Worth 11-7-23
*/

#include <iostream>
#include <fstream>
#include "Tree.h"
#include "List.h"
using namespace std;

int main(){

    Tree tr;
    int exit = 0;

    ifstream fData("Movies.txt"); // open file for use
    int test = 0;
    while(exit == 0){
        exit = tr.GetData(fData);
    }
    fData.close(); // close file

    //Display all movies in the tree (only the titles!).
    tr.DisplayTitles();

    //Display all actors of a given movie in the tree: Bullitt, Man of the Year,
    cout << endl;
    cout << "Actors in Bullitt:" << endl;
    tr.DisplayAllActors("Bullitt ");
```

```
cout << endl;
cout << "Actors in Man of the Year:" << endl;
tr.DisplayAllActors("Man of the Year");

cout << endl;
cout << "Actors in The April Fools:" << endl;
tr.DisplayAllActors("The April Fools");

cout << endl;
cout << "Actors in Good Will Hunting:" << endl;
tr.DisplayAllActors("Good Will Hunting");

cout << endl;
cout << "Actors in Forrest Gump:" << endl;
tr.DisplayAllActors("Forrest Gump");

//Display all movies of a given actor: Tom Cruise, Carrie Fisher, Roger Moor
cout << endl;
cout << "Tom Cruise's Movies:" << endl;
tr.DisplayCertinActors("Tom Cruise");

cout << endl;
cout << "Carrie Fisher's Movies:" << endl;
tr.DisplayCertinActors("Carrie Fisher");

cout << endl;
cout << "Roger Moore's Movies:" << endl;
tr.DisplayCertinActors("Roger Moore");

cout << endl;
cout << "Clint Eastwood's Movies:" << endl;
tr.DisplayCertinActors("Clint Eastwood");

cout << endl;
cout << "Matt Damon's Movies:" << endl;
tr.DisplayCertinActors("Matt Damon");

//Display all movies released in 1970 and one other year of your choice.
cout << endl;
cout << "Movies Released in 1970:" << endl;
tr.DisplayMoviesReleased(1970);

cout << endl;
cout << "Movies Released in 2000:" << endl;
tr.DisplayMoviesReleased(2000);
```

```
    tr.deleteTree();


    return 0;
}

/*
    Computer Science 2
    Tree Header File

    Collin Worth 11-7-23
*/

#ifndef Tree_H
#define Tree_H
#include <string>
#include "List.h"
using namespace std;

class Tree {

    public:

        List l;

        struct Node {
            string title;
            List names;
            int year;
            Node* leftPtr;
            Node* rightPtr;
        };

        Tree();
        ~Tree();

        bool IsLeaf(Node*);
        void AddNodeR(string, int, List);
        void removeNode();
        void printTree();
        void deleteTree();
        void printFullNode(Node*);
        int GetData(ifstream&);
        void DisplayTitles();
        void DisplayCertinActors(string);
```

```
        void DisplayAllActors(string);
        void DisplayMoviesReleased(int);



    private:

        Node* rootPtr;

        void AddNodeR(Node* &, string, int, List);
        void deleteTree(Node* &);
        void printTree(Node*);
        void DisplayTitles(Node*);
        void DisplayCertinActors(Node*, string);
        void DisplayAllActors(Node*, string);
        void DisplayMoviesReleased(Node*, int);

};



#endif

/*
    Computer Science 2
    Tree Implementation File

    Collin Worth 11-7-23
*/
#include <iostream>
#include <fstream>
#include "Tree.h"
#include "List.h"
using namespace std;



Tree::Tree(){
    rootPtr = NULL;
};

Tree::~Tree(){

};
```

```
int Tree::GetData(ifstream& fData){
    if (!fData.is_open()) {
    cerr << "Error: File not found!" << endl;
    return 1;
    }
    string curString;
    string storTitle;
    int storYear = 0;
    string storName;
    int strLoc = 0;


    getline(fData, curString);

    while(curString.empty()){ //insures that we are using a string with data in
        getline(fData, curString);
    }

    while(curString[strLoc] != '('){
        storTitle += curString[strLoc];
        strLoc ++;
    }

    strLoc ++;

    while (curString[strLoc] != ')'){
        storYear = (storYear * 10) + (curString[strLoc] - '0');
// Convert char to int
        strLoc++;
    }

    // NAMES
    ////////////////////////

    List newList;
    while(!curString.empty()){

        if(fData.eof()){return 1;}

        newList.AddNode(curString);

        getline(fData, curString);
    }

    AddNodeR(storTitle, storYear, newList);
    if(fData.eof()){
```

```
        return 1;
    }else{
        return 0;
    }
}

void Tree::AddNodeR( string title, int year, List names){
    AddNodeR( rootPtr, title, year, names);
}

// Add Node based on movie title
void Tree::AddNodeR(Node* &t, string title, int year, List Nlist) {
    if (t == nullptr) {
        t = new Node;
        t->title = title;
        t->year = year;
        t->names = Nlist;
        t->leftPtr = nullptr;
        t->rightPtr = nullptr;
    } else if (title <= t->title) {
        AddNodeR(t->leftPtr, title, year, Nlist);
    } else {
        AddNodeR(t->rightPtr, title, year, Nlist);
    }
}

void Tree::printTree(){
    printTree(rootPtr);
}

void Tree::printTree(Node* t){

    if(t != NULL){
        printFullNode(t);
        printTree(t->leftPtr);
        printTree(t->rightPtr);
    }
}

void Tree::printFullNode(Node* t){
    cout << endl;
    cout << "Title:" << t->title << endl;
    cout << "Year:" << t->year << endl;
    cout << "Actors:";
    t->names.PrintList();
    cout << endl;
```

```
}

void Tree::deleteTree(Node* &t) {
    if (t != nullptr) {
        deleteTree(t->leftPtr);  // Delete left subtree
        deleteTree(t->rightPtr); // Delete right subtree
        delete t;                // Delete current node
        t = nullptr;             // Set node to null to avoid dangling pointers
    }
}

void Tree::deleteTree() {
    deleteTree(rootPtr); // Call the recursive function starting from the root
    rootPtr = nullptr;   // Set the root to null after the tree is deleted
}

// Return true if node is a leaf
bool Tree::IsLeaf( Node* treePtr) {
    return ((treePtr->leftPtr == NULL) && (treePtr->rightPtr == NULL) );
}

//public method
void Tree::DisplayTitles(){
    DisplayTitles(rootPtr);
}

void Tree::DisplayTitles(Node* t) {
    if (t != nullptr) {
        if (t->leftPtr != nullptr) {
            DisplayTitles(t->leftPtr);
        }
        cout << t->title << endl;
        if (t->rightPtr != nullptr) {
            DisplayTitles(t->rightPtr);
        }
    }
}

void Tree::DisplayCertinActors(string s){
    DisplayCertinActors(rootPtr, s);
}

void Tree::DisplayCertinActors(Node* t, string s){

    //while walking through tree
    if(t != NULL){
```

```
        //if one of the actors == string
        if(t->names.LookFor(s)){
            //print movie title
            cout << t->title << endl;
        }
        DisplayCertinActors(t->leftPtr, s);
        DisplayCertinActors(t->rightPtr, s);
    }
}

//public
void Tree::DisplayAllActors(string s){
    DisplayAllActors(rootPtr, s);
}

//private
void Tree::DisplayAllActors(Node* t, string s){
    if(t != NULL){
        if(t->title == s){
            t->names.PrintList();
        }
        DisplayAllActors(t->leftPtr, s);
        DisplayAllActors(t->rightPtr, s);
    }
}

//public
void Tree::DisplayMoviesReleased(int i){
    DisplayMoviesReleased(rootPtr, i);
}

void Tree::DisplayMoviesReleased(Node* t, int i){
    if(t != NULL){
        if(t->year == i){
            cout << t->title << endl;
        }
        DisplayMoviesReleased(t->leftPtr, i);
        DisplayMoviesReleased(t->rightPtr, i);
    }
}

/*
        CS 121
        Header File

        Collin Worth     9-25-2023
```

```
*/

#ifndef LIST_H
#define LIST_H

#include <string>
using namespace std;

class List {
public:
    List(); // Constructor
    ~List(); // Destructor

    void PrintList();
    void Delete();
    void AddNode(string);
    bool LookFor(string);

private:
    struct Node {
        string data;
        Node* next;
    };
    Node* head;
};

#endif

/*
        CS 121
        Class File

        Collin Worth     9-25-2023
*/

#include "List.h"
#include <iostream>

using namespace std;

List::List() : head(nullptr) {}

List::~List() {
    // Deconstructor
}
```

```
//takes a string and adds it to the list
void List::AddNode(string str) {

    Node* ptr = new Node;
    ptr -> data = str;

    if(head == NULL){
        ptr -> next = NULL;
    }else{
        ptr -> next = head;
    }
    head = ptr;
}

// Print list in order to test code
void List::PrintList() {
    Node* current = head;

    while (current != nullptr) {
        cout << current->data;
        if (current->next != nullptr) {
            cout << ",-";
        }
        current = current->next;
    }
    cout << endl;
}

// Implement the Delete function to clean up the list
void List::Delete() {
        while (head != nullptr) {
                Node* temp = head;
                head = head->next;
                delete temp;
        }
}

//returns true if string "s" was found in the list
bool List::LookFor(string s){
    Node* temp = head;

    while(temp != NULL){
        if(temp->data == s){
            return true;
        }
        temp = temp->next;
```

```
    }
    return false;
}
```