

程序设计报告

（2021 / 22学年第一学期）

题目： 热播电视剧的评价及推荐系统

|  |  |
| --- | --- |
| **专业** | **信息安全** |
| **组长学号姓名** | **B20060527张宸冉** |
| **组员学号姓名** | **B20031007徐雨慧** |
|  | **B20031008韩旻倩** |
| **指导教师** | **王少辉** |
| **指导单位** | **计算机学院信息安全系** |
| **日期** | **2021.11.21** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **成员分工** | | | | |
| B20060527张宸冉 | 用户类 文件流输入输出 | | | |
| B20031007徐雨慧 | 管理员类 | | | |
| B20031008韩旻倩 | 主函数 | | | |
| **教师评价** | | | | |
| 课程目标 | 评价准则（每项10分） | 团队成员 | | |
|  |  |  |
| **目标1**通过课题的设置与课程的实施，引导学生主动思考、积极实践，鼓励学生不断提升作品的质量与水平，强化学生的品质意识与责任担当，以更好的为社会服务。 | 出勤与上机表现规范，积极承担、完成团队分配的任务。 |  |  |  |
| 具备较好的责任意识，完成了课题要求的各项开发任务，功能实现全面。 |  |  |  |
| 具备初步的质量意识，作品经过充分测试，界面美观，接口合理，交互友好。 |  |  |  |
| **目标2**通过课题的设置与课程的实施，锻炼学生的分析与设计能力，加深学生对编程语言的理解与掌握，加强学生对编程工具的熟悉和使用，全面锻炼学生的综合编程能力。 | 掌握程序设计开发的相关基础知识，具备初步的分析能力，能对求解的工程问题进行合理分析。 |  |  |  |
| 具备初步的程序设计能力，能结合计算机软硬件资源，合理选用算法、数据结构、数据存储方式等技术手段。 |  |  |  |
| 熟练掌握编程语言，代码编写逻辑清晰、格式规范、注释完整。 |  |  |  |
| 熟悉编程环境的使用，熟练掌握调试方法与工具，能对开发中出现的问题进行分析、跟踪及调试。 |  |  |  |
| **目标3**通过课题的设置与课程的实施，让学生经历完整的问题分析、算法设计、程序实现、代码测试、报告撰写与答辩过程，培养学生查找资料、探索分析问题的能力，锻炼学生的文档编写能力、团队合作意识与沟通表达能力。 | 具备一定自学能力与探索创新意识，能够充分利用各种网络与图书馆资源自学新知识与新技能。 |  |  |  |
| 具备一定口头表达能力与文字处理能力。回答问题准确。报告内容详实，格式规范。 |  |  |  |
| 具备较好的团队协作精神，能够努力沟通解决遇到的各种开发、工程问题。 |  |  |  |
| 评价总分 | |  |  |  |
| **能力达成评价** | |  |  |  |
| 备注：  能力达成评价：优秀、良好、中等、及格、不及格 | | | | |
| **指导教师： 年 月 日** | | | | |

**热播电视剧的评价及推荐系统**

**一、课题内容和要求**

1、随着人们对精神文明需求的增加，优秀的电视剧作品总是拥有众多粉丝。本课题旨在设计并实现一个热播电视剧的评价及推荐系统，帮助观众在有限的时间内选择到优秀作品，同时也分享自己对作品的评价，便于为其他人做参考。

该系统有两种角色的人员——管理员和普通用户。管理员需要凭密码登录，可以对热播电视剧的基本信息进行创立、增加、删除、修改；可以查看所有用户对所有电视剧的评价信息。普通用户可以查看排名（排名根据所有用户的评分进行计算和排序）前10的热播电视剧具体信息、也可以按指定剧名查询，可以选择观看及评价某电视剧，从多个方面进行评价等。

2、本程序可提供以下几个基本功能：

（1）一些基础信息，比如：电视剧的基本信息可以由程序设计人员预先从键盘上录入（通过记事本或写字板建立文本文件），存为文本文件；此后管理员可以通过程序对这些信息进行访问、增加、删除、修改。

（2）为提高人机交互性及清晰性，设计并实现相关界面（管理员和普通用户所能看到的界面不一样），如果有必要，需要使用分级菜单；

（3）管理员登录系统必须输入正确的用户名和密码，密码字符以\*形式显示，不可以显示明文，输入错误时提示用户，连续输入错误3次则无法进入系统；

（4）普通用户的功能：可以查询自己感兴趣的电视剧的基本信息、可以选择自己需要评价的电视剧并且给出各方面的评价分、可以查看所有人对某电视剧的具体评价、也可以查询最近热播的前10名电视剧，为自己选看提供参考；

（5）按照用户的浏览习惯，给出推荐。

（6）为普通用户提供分类查热播功能，比如，如果最关注特效的，就按这一指标查询排名前10的电视剧，而不是按统一的整体评分这一指标查询。

（7）为管理员提供分段统计及查看功能，比如，对用户评价的各个指标，对所有人的打分，按分数段进行统计，得出分析统计的一个整体报表。

1. **需求分析**

热播电视剧评价及推荐系统的功能框架图如图1所示。

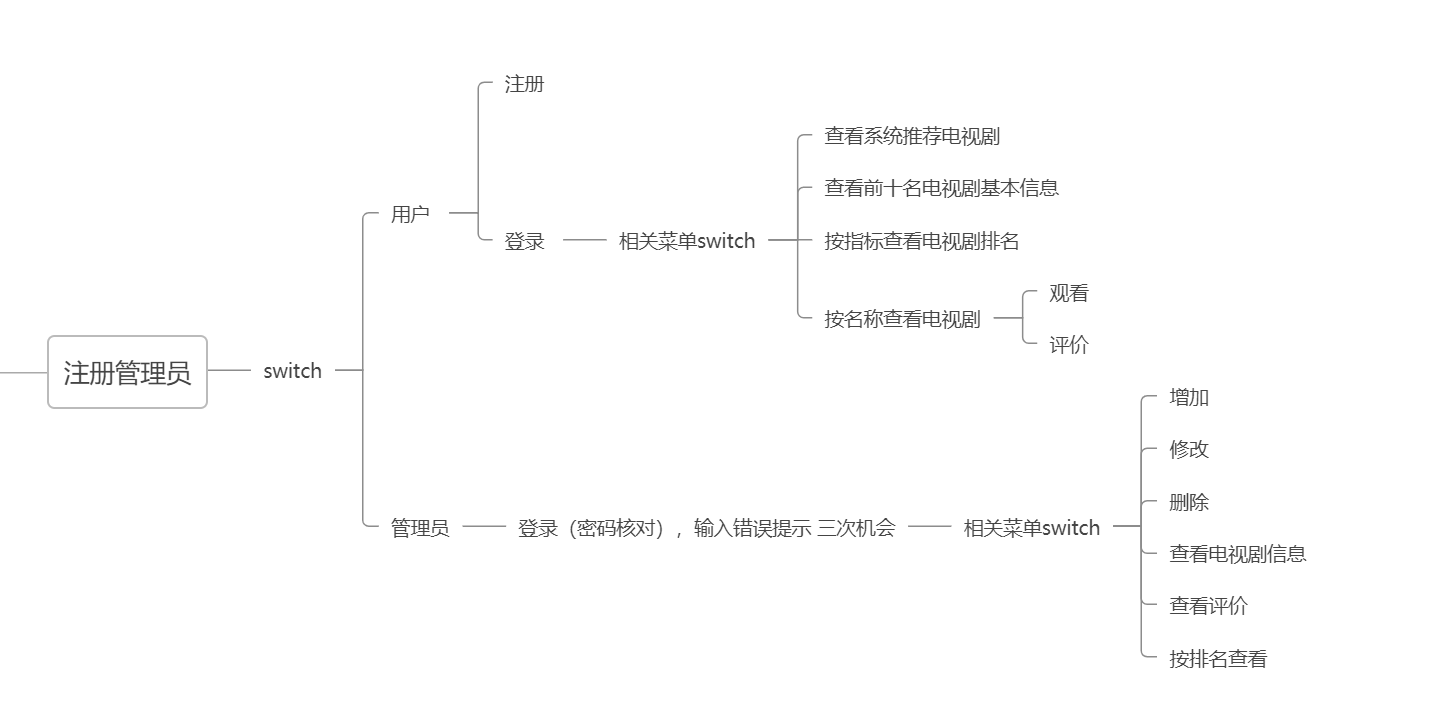


图1. 功能框架图

1. 注册/登录系统：通过用户输入账号、密码进行注册，将用户输入的数据录入文档中，用于以后用户登录时的验证。
2. 角色选择菜单：输出角色选择菜单，用户输入不同的数值分别进入不同角色功能菜单。
3. 角色功能菜单：输出角色功能菜单，用户通过输入不同选项来运行不同的功能，进而运行不同的函数。
4. **概要设计**

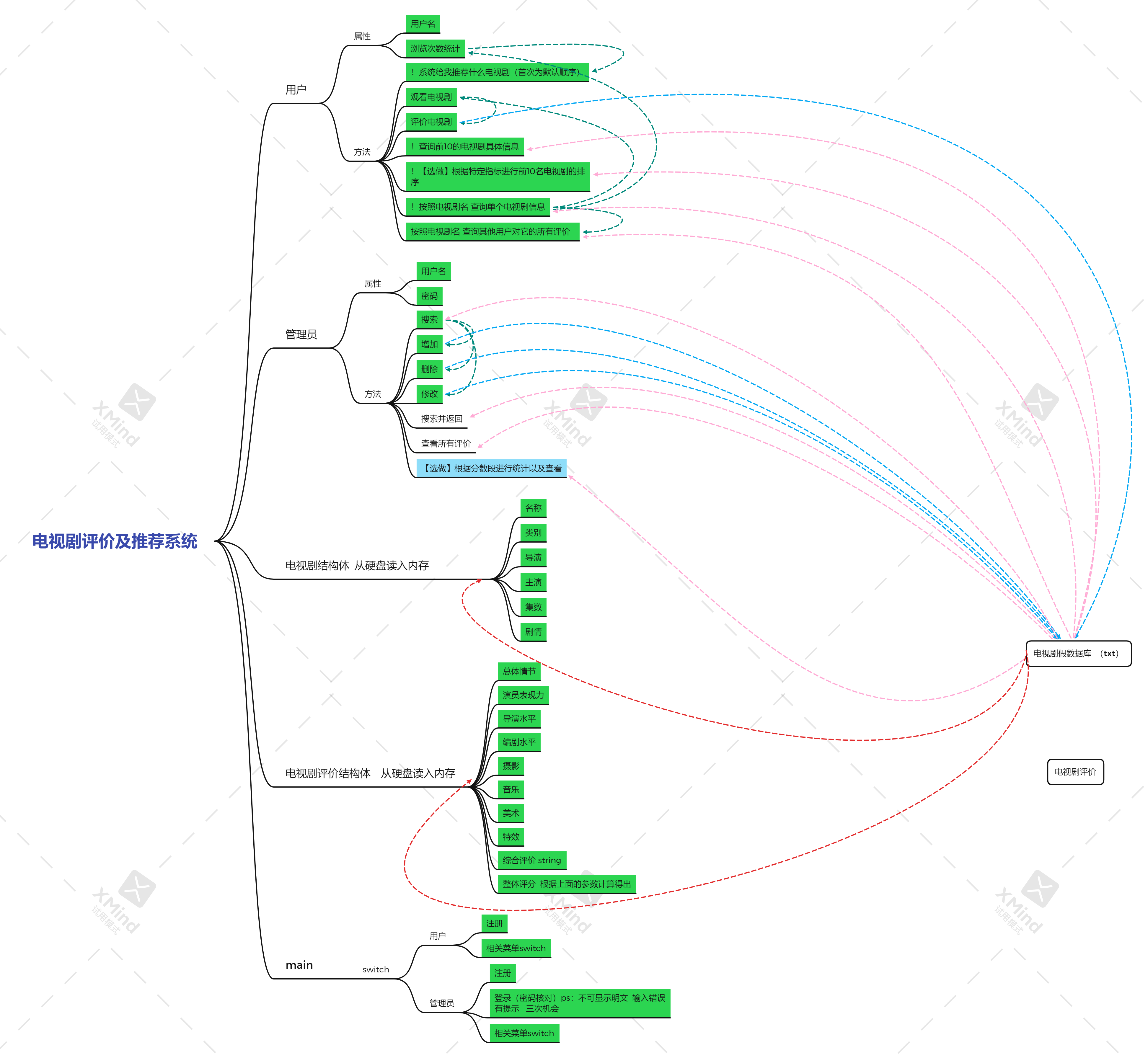


图2. 概要设计思维导图

1. **源程序代码**

#pragma once

#include<string>

#include<iostream>

using namespace std;

const int N = 40; //电视剧最大个数

const int MAXUSER = 10; //用户最大个数

typedef struct estimate {

string userName; //这条评论的用户名称

int plot; //情节评分

int behavior; //演员表现力

int levelOfDirector; //导演水平

int levelOfWriter; //编剧水平

int photographer; //摄影

int music; //音乐

int art; //美术

int effect; //特效

string overallEstimate; //综合评价

double average ; //整体评分

}Esitimate;

typedef struct infoOfTv

{

string name; //电视剧名字

string type; //电视剧类型

string director; //导演

string actor; //主演

int number; //集数

string intro; //剧情

}InfoOfTv;

class user

{

private:

string name; //用户名称

int userAttempt[N] = { 0 }; // 用于存放用户对于每一个电视剧的查询次数

public:

user();

user(string name);

string getName();

int recommendTv(int\*); //查看系统推荐的电视剧

void TopTenTv(Esitimate [N][MAXUSER], infoOfTv\* tv); //查看前十电视剧基本信息

void TopTenTvByIndex(int, Esitimate[N][MAXUSER], infoOfTv\*); //按照一定指标对前10电视剧进行排名

int searchTv(InfoOfTv\*,int\*); //按照电视剧名搜索电视剧基本信息

void esitimateTv(int,int,Esitimate[N][MAXUSER],infoOfTv\*); //评价改电视剧

void esitimateFromOtherUser(int,Esitimate [N][MAXUSER], infoOfTv\* ); //查看其它用户对这个电视剧的评价

void rankIndex(int pos[N], double averageOfTv[N]); //选择法排序

};

#include"用户.h"

using namespace std;

//选择法排序

void user::rankIndex(int pos[N],double averageOfTv[N]) {

//参数 pos是电视剧地址数组 averageOfTv是当前指标数据数组

//无返回值 直接更改相关数组

int index; //暂存最值

int i, k, temp1;

double temp;

for (k = 0; k < N - 1; k++) {

index = k;

for (i = k + 1; i < N; i++) {

if (averageOfTv[i] > averageOfTv[index]) {

index = i;

}

}

if (index != k) {

//交换平均值和地址

temp = averageOfTv[index];

averageOfTv[index] = averageOfTv[k];

averageOfTv[k] = temp;

temp1 = pos[index];

pos[index] = pos[k];

pos[k] = temp1;

}

}

}

user::user() {

name = "未定义";

}

user::user(string na) :name(na){

cout << "用户已经注册成功！！" << endl;

}

string user::getName() {

return name;

}

int user::recommendTv(int\* posOfDelete) {

//参数 管理员可能删除电视剧 参数是删除的电视剧的位置 用于调整当前用户偏好列表

//返回值是地址值

int max=0; //用户偏好中数值最大的电视剧的地址

//调整当前用户偏好数组

if (\*posOfDelete != N) {

for (int k = \*posOfDelete; k<N-1; k++) {

userAttempt[k] = userAttempt[k + 1];

}

userAttempt[N-1] = 0;

}

//寻找偏好数组中的最大值

for (int i = 0; i < N; i++) {

if (userAttempt[i] > userAttempt[max]) {

max = i;

}

}

return max;

}

void user::TopTenTv(Esitimate esitimateOfTv[N][MAXUSER], infoOfTv\*tv ) {

//参数 esitimateOfTv是电视剧评价二维数组 tv是电视剧信息数组

//返回值 无返回值

//构建电视剧平均分数组

double averageOfTv[N];

int pos[N];

double sum = 0.00;

//计算每个电视剧的平均分

for (int i = 0; i < N; i++) {

for (int j = 0; j < MAXUSER; j++) {

sum += esitimateOfTv[i][j].average;

}

averageOfTv[i] = sum / MAXUSER;

pos[i] = i;

sum = 0.00;

}

rankIndex(pos, averageOfTv); //排序

for (int i = 0; i < 10; i++) {

cout << "第" << i +1<< "名:" << endl;

cout << "名称:" << tv[pos[i]].name << endl;

cout << "类型:" << tv[pos[i]].type << endl;

cout << "导演:" << tv[pos[i]].director << endl;

cout << "主演:" << tv[pos[i]].actor << endl;

cout << "集数:" << tv[pos[i]].number << endl;

cout << "简介:" << tv[pos[i]].intro << endl<<endl;

}

}

void user::TopTenTvByIndex(int choice, Esitimate esitimateOfTv[N][MAXUSER], infoOfTv\* tv) {

//参数 choice是指标 esitimateOfTv评价二维数组 tv电视剧数组

//无返回值

double averageOfTv[N];

int pos[N];

double sum = 0.00;

if (choice == 1) {

for (int i = 0; i < N; i++) {

for (int j = 0; j < MAXUSER; j++) {

sum += esitimateOfTv[i][j].plot;

}

averageOfTv[i] = sum / MAXUSER;

pos[i] = i;

sum = 0.00;

}

rankIndex(pos, averageOfTv);

for (int i = 0; i < 10; i++) {

cout << "第" << i + 1 << "名:" << endl;

cout << "名称:" << tv[pos[i]].name << endl;

}

}

else if (choice == 2) {

for (int i = 0; i < N; i++) {

for (int j = 0; j < MAXUSER; j++) {

sum += esitimateOfTv[i][j].behavior;

}

averageOfTv[i] = sum / MAXUSER;

pos[i] = i;

sum = 0.00;

}

rankIndex(pos, averageOfTv);

for (int i = 0; i < 10; i++) {

cout << "第" << i + 1 << "名:" << endl;

cout << "名称:" << tv[pos[i]].name << endl;

}

}

else if (choice == 3) {

for (int i = 0; i < N; i++) {

for (int j = 0; j < MAXUSER; j++) {

sum += esitimateOfTv[i][j].levelOfDirector;

}

averageOfTv[i] = sum / MAXUSER;

pos[i] = i;

sum = 0.00;

}

rankIndex(pos, averageOfTv);

for (int i = 0; i < 10; i++) {

cout << "第" << i + 1 << "名:" << endl;

cout << "名称:" << tv[pos[i]].name << endl;

}

}

else if (choice == 4) {

for (int i = 0; i < N; i++) {

for (int j = 0; j < MAXUSER; j++) {

sum += esitimateOfTv[i][j].levelOfWriter;

}

averageOfTv[i] = sum / MAXUSER;

pos[i] = i;

sum = 0.00;

}

rankIndex(pos, averageOfTv);

for (int i = 0; i < 10; i++) {

cout << "第" << i + 1 << "名:" << endl;

cout << "名称:" << tv[pos[i]].name << endl;

}

}

else if (choice == 5) {

for (int i = 0; i < N; i++) {

for (int j = 0; j < MAXUSER; j++) {

sum += esitimateOfTv[i][j].photographer;

}

averageOfTv[i] = sum / MAXUSER;

pos[i] = i;

sum = 0.00;

}

rankIndex(pos, averageOfTv);

for (int i = 0; i < 10; i++) {

cout << "第" << i + 1 << "名:" << endl;

cout << "名称:" << tv[pos[i]].name << endl;

}

}

else if (choice == 6) {

for (int i = 0; i < N; i++) {

for (int j = 0; j < MAXUSER; j++) {

sum += esitimateOfTv[i][j].music;

}

averageOfTv[i] = sum / MAXUSER;

pos[i] = i;

sum = 0.00;

}

rankIndex(pos, averageOfTv);

for (int i = 0; i < 10; i++) {

cout << "第" << i + 1 << "名:" << endl;

cout << "名称:" << tv[pos[i]].name << endl;

}

}

else if (choice == 7) {

for (int i = 0; i < N; i++) {

for (int j = 0; j < MAXUSER; j++) {

sum += esitimateOfTv[i][j].art;

}

averageOfTv[i] = sum / MAXUSER;

pos[i] = i;

sum = 0.00;

}

rankIndex(pos, averageOfTv);

for (int i = 0; i < 10; i++) {

cout << "第" << i + 1 << "名:" << endl;

cout << "名称:" << tv[pos[i]].name << endl;

}

}

else {

for (int i = 0; i < N; i++) {

for (int j = 0; j < MAXUSER; j++) {

sum += esitimateOfTv[i][j].effect;

}

averageOfTv[i] = sum / MAXUSER;

pos[i] = i;

sum = 0.00;

}

rankIndex(pos, averageOfTv);

for (int i = 0; i < 10; i++) {

cout << "第" << i + 1 << "名:" << endl;

cout << "名称:" << tv[pos[i]].name << endl;

}

}

}

int user::searchTv(InfoOfTv\* tv,int\* posOfThis) {

//参数 电视剧数组指针

//返回值 找到返回1 没找到返回0 用于做条件判断

cout << "请输入想要寻找的电视剧名称：" << endl;

string nameOfTv;

cin >> nameOfTv;

int flag = 0; //用于显示电视剧名称是否在数组之中

int pos; //用于存储当前电视剧在电视剧基本信息数组中的位置

for (int i = 0; i < N; i++) { //查找

if (nameOfTv == tv[i].name) {

pos = i;

flag = -1;

}

if (flag == -1) {

break;

}

}

if (flag != 0) {

//找到了就返回基本信息

cout << endl;

cout <<"《"<< tv[pos].name << "》的基本信息是：" << endl;

cout << "名称:" << tv[pos].name << endl;

cout << "类型:" << tv[pos].type << endl;

cout << "导演:" << tv[pos].director << endl;

cout << "主演:" << tv[pos].actor << endl;

cout << "集数:" << tv[pos].number << endl;

cout << "简介:" << tv[pos].intro << endl;

//并且在偏好数组中给这个电视剧加1

userAttempt[pos]++;

\*posOfThis = pos;

return 1;

}

else

{

return 0;

}

}

void user::esitimateTv(int posOfThis,int posOfUser,Esitimate esitimateOfTv[N][MAXUSER],infoOfTv\* tv) {

//参数 posOfThis是当前评价的电视剧文件数组中的位置 posOfUser是当前用户的位置 esitimateList是文件数组 tv是电视剧基本信息数组

//无返回值

esitimateOfTv[posOfThis][posOfUser].userName = this->getName();

cout << "请根据一下不同层面给《"+tv[posOfThis].name+"》打分（0-10分~）" << endl;

cout << "情节：" ;

cin >> esitimateOfTv[posOfThis][posOfUser].plot;

while ((esitimateOfTv[posOfThis][posOfUser].plot != 0 && esitimateOfTv[posOfThis][posOfUser].plot != 1 && esitimateOfTv[posOfThis][posOfUser].plot != 2 && esitimateOfTv[posOfThis][posOfUser].plot != 3 && esitimateOfTv[posOfThis][posOfUser].plot != 4 && esitimateOfTv[posOfThis][posOfUser].plot != 5 && esitimateOfTv[posOfThis][posOfUser].plot != 6 && esitimateOfTv[posOfThis][posOfUser].plot != 7 && esitimateOfTv[posOfThis][posOfUser].plot != 8 && esitimateOfTv[posOfThis][posOfUser].plot != 9 && esitimateOfTv[posOfThis][posOfUser].plot != 10)||cin.fail()) {

cout << "输入错误，请重新输入！" << endl;

cin.clear();

cin.ignore(1000, '\n');

cin >> esitimateOfTv[posOfThis][posOfUser].plot;

}

cout << endl;

cout << "演员表现力：";

cin >> esitimateOfTv[posOfThis][posOfUser].behavior;

while ((esitimateOfTv[posOfThis][posOfUser].behavior != 0 && esitimateOfTv[posOfThis][posOfUser].behavior != 1 && esitimateOfTv[posOfThis][posOfUser].behavior != 2 && esitimateOfTv[posOfThis][posOfUser].behavior != 3 && esitimateOfTv[posOfThis][posOfUser].behavior != 4 && esitimateOfTv[posOfThis][posOfUser].behavior != 5 && esitimateOfTv[posOfThis][posOfUser].behavior != 6 && esitimateOfTv[posOfThis][posOfUser].behavior != 7 && esitimateOfTv[posOfThis][posOfUser].behavior != 8 && esitimateOfTv[posOfThis][posOfUser].behavior != 9 && esitimateOfTv[posOfThis][posOfUser].behavior != 10) || cin.fail()) {

cout << "输入错误，请重新输入！" << endl;

cin.clear();

cin.ignore(1000, '\n');

cin >> esitimateOfTv[posOfThis][posOfUser].behavior;

}

cout << endl;

cout << "导演水平：";

cin >> esitimateOfTv[posOfThis][posOfUser].levelOfDirector;

while ((esitimateOfTv[posOfThis][posOfUser].levelOfDirector != 0 && esitimateOfTv[posOfThis][posOfUser].levelOfDirector != 1 && esitimateOfTv[posOfThis][posOfUser].levelOfDirector != 2 && esitimateOfTv[posOfThis][posOfUser].levelOfDirector != 3 && esitimateOfTv[posOfThis][posOfUser].levelOfDirector != 4 && esitimateOfTv[posOfThis][posOfUser].levelOfDirector != 5 && esitimateOfTv[posOfThis][posOfUser].levelOfDirector != 6 && esitimateOfTv[posOfThis][posOfUser].levelOfDirector != 7 && esitimateOfTv[posOfThis][posOfUser].levelOfDirector != 8 && esitimateOfTv[posOfThis][posOfUser].levelOfDirector != 9 && esitimateOfTv[posOfThis][posOfUser].levelOfDirector != 10) || cin.fail()) {

cout << "输入错误，请重新输入！" << endl;

cin.clear();

cin.ignore(1000, '\n');

cin >> esitimateOfTv[posOfThis][posOfUser].levelOfDirector;

}

cout << endl;

cout << "编剧水平：";

cin >> esitimateOfTv[posOfThis][posOfUser].levelOfWriter;

while ((esitimateOfTv[posOfThis][posOfUser].levelOfWriter != 0 && esitimateOfTv[posOfThis][posOfUser].levelOfWriter != 1 && esitimateOfTv[posOfThis][posOfUser].levelOfWriter != 2 && esitimateOfTv[posOfThis][posOfUser].levelOfWriter != 3 && esitimateOfTv[posOfThis][posOfUser].levelOfWriter != 4 && esitimateOfTv[posOfThis][posOfUser].levelOfWriter != 5 && esitimateOfTv[posOfThis][posOfUser].levelOfWriter != 6 && esitimateOfTv[posOfThis][posOfUser].levelOfWriter != 7 && esitimateOfTv[posOfThis][posOfUser].levelOfWriter != 8 && esitimateOfTv[posOfThis][posOfUser].levelOfWriter != 9 && esitimateOfTv[posOfThis][posOfUser].levelOfWriter != 10) || cin.fail()) {

cout << "输入错误，请重新输入！" << endl;

cin.clear();

cin.ignore(1000, '\n');

cin >> esitimateOfTv[posOfThis][posOfUser].levelOfWriter;

}

cout << endl;

cout << "摄影：" ;

cin >> esitimateOfTv[posOfThis][posOfUser].photographer;

while ((esitimateOfTv[posOfThis][posOfUser].photographer != 0 && esitimateOfTv[posOfThis][posOfUser].photographer != 1 && esitimateOfTv[posOfThis][posOfUser].photographer != 2 && esitimateOfTv[posOfThis][posOfUser].photographer != 3 && esitimateOfTv[posOfThis][posOfUser].photographer != 4 && esitimateOfTv[posOfThis][posOfUser].photographer != 5 && esitimateOfTv[posOfThis][posOfUser].photographer != 6 && esitimateOfTv[posOfThis][posOfUser].photographer != 7 && esitimateOfTv[posOfThis][posOfUser].photographer != 8 && esitimateOfTv[posOfThis][posOfUser].photographer != 9 && esitimateOfTv[posOfThis][posOfUser].photographer != 10) || cin.fail()) {

cout << "输入错误，请重新输入！" << endl;

cin.clear();

cin.ignore(1000, '\n');

cin >> esitimateOfTv[posOfThis][posOfUser].photographer;

}

cout << endl;

cout << "音乐：";

cin >> esitimateOfTv[posOfThis][posOfUser].music;

while ((esitimateOfTv[posOfThis][posOfUser].music != 0 && esitimateOfTv[posOfThis][posOfUser].music != 1 && esitimateOfTv[posOfThis][posOfUser].music != 2 && esitimateOfTv[posOfThis][posOfUser].music != 3 && esitimateOfTv[posOfThis][posOfUser].music != 4 && esitimateOfTv[posOfThis][posOfUser].music != 5 && esitimateOfTv[posOfThis][posOfUser].music != 6 && esitimateOfTv[posOfThis][posOfUser].music != 7 && esitimateOfTv[posOfThis][posOfUser].music != 8 && esitimateOfTv[posOfThis][posOfUser].music != 9 && esitimateOfTv[posOfThis][posOfUser].music != 10) || cin.fail()) {

cout << "输入错误，请重新输入！" << endl;

cin.clear();

cin.ignore(1000, '\n');

cin >> esitimateOfTv[posOfThis][posOfUser].music;

}

cout << endl;

cout << "美术：";

cin >> esitimateOfTv[posOfThis][posOfUser].art;

while ((esitimateOfTv[posOfThis][posOfUser].art != 0 && esitimateOfTv[posOfThis][posOfUser].art != 1 && esitimateOfTv[posOfThis][posOfUser].art != 2 && esitimateOfTv[posOfThis][posOfUser].art != 3 && esitimateOfTv[posOfThis][posOfUser].art != 4 && esitimateOfTv[posOfThis][posOfUser].art != 5 && esitimateOfTv[posOfThis][posOfUser].art != 6 && esitimateOfTv[posOfThis][posOfUser].art != 7 && esitimateOfTv[posOfThis][posOfUser].art != 8 && esitimateOfTv[posOfThis][posOfUser].art != 9 && esitimateOfTv[posOfThis][posOfUser].art != 10) || cin.fail()) {

cout << "输入错误，请重新输入！" << endl;

cin.clear();

cin.ignore(1000, '\n');

cin >> esitimateOfTv[posOfThis][posOfUser].art;

}

cout << endl;

cout << "特效：";

cin >> esitimateOfTv[posOfThis][posOfUser].effect;

while ((esitimateOfTv[posOfThis][posOfUser].effect != 0 && esitimateOfTv[posOfThis][posOfUser].effect != 1 && esitimateOfTv[posOfThis][posOfUser].effect != 2 && esitimateOfTv[posOfThis][posOfUser].effect != 3 && esitimateOfTv[posOfThis][posOfUser].effect != 4 && esitimateOfTv[posOfThis][posOfUser].effect != 5 && esitimateOfTv[posOfThis][posOfUser].effect != 6 && esitimateOfTv[posOfThis][posOfUser].effect != 7 && esitimateOfTv[posOfThis][posOfUser].effect != 8 && esitimateOfTv[posOfThis][posOfUser].effect != 9 && esitimateOfTv[posOfThis][posOfUser].effect != 10) || cin.fail()) {

cout << "输入错误，请重新输入！" << endl;

cin.clear();

cin.ignore(1000, '\n');

cin >> esitimateOfTv[posOfThis][posOfUser].effect;

}

cout << endl;

cout << "你还可以写写评论哦~" << endl;

cin >> esitimateOfTv[posOfThis][posOfUser].overallEstimate;

cout << endl;

esitimateOfTv[posOfThis][posOfUser].average = (esitimateOfTv[posOfThis][posOfUser].plot + esitimateOfTv[posOfThis][posOfUser].behavior + esitimateOfTv[posOfThis][posOfUser].levelOfDirector + esitimateOfTv[posOfThis][posOfUser].levelOfWriter + esitimateOfTv[posOfThis][posOfUser].photographer + esitimateOfTv[posOfThis][posOfUser].art + esitimateOfTv[posOfThis][posOfUser].music + esitimateOfTv[posOfThis][posOfUser].effect) / 8;

}

void user::esitimateFromOtherUser(int posOfThis,Esitimate esitimateOfTv[N][MAXUSER], infoOfTv\* tv) {

//参数 posOfThis是当前评价的电视剧文件数组中的位置 esitimateList是文件数组 tv是电视剧基本信息数组

//无返回值

cout << "其他用户对《" + tv[posOfThis].name + "》的评价是：" << endl;

for (int i = 0; i < MAXUSER; i++) {

if (esitimateOfTv[posOfThis][i].userName == "未输入") {

continue;

}

cout << esitimateOfTv[posOfThis][i].userName << ": " << esitimateOfTv[posOfThis][i].overallEstimate << " ta的总体评价是：" << esitimateOfTv[posOfThis][i].average << endl;

}

}

#pragma once

#include<iostream>

#include<fstream>

#include"用户.h"

using namespace std;

class manager

{

private:

string name;

string password;

public:

manager();

manager(string name, string password);

bool checkPassword(char\* password);

void Add(InfoOfTv\*, Esitimate esitimateOfTv[N][MAXUSER],int\*);//增加电视剧信息

void modify(InfoOfTv\*);//修改电视剧信息

void deletetv(InfoOfTv\*, Esitimate esitimateOfTv[N][MAXUSER], int\*,int\*);//删除电视剧信息

void searchofmanager(InfoOfTv\*);//按照名称返回电视剧信息

void esitimateFromuser(Esitimate esitimateOfTv[N][MAXUSER], InfoOfTv\*);

void rank(int,Esitimate esitimateOfTv[N][MAXUSER], InfoOfTv\*);//按照排名分数段查看

};

#include"管理员.h"

#include<fstream>

using namespace std;

int count1 = 30;

//选择法排序

void rankIndex(int pos[N], double averageOfTv[N]) {

//参数 pos是电视剧地址数组 averageOfTv是当前指标数据数组

//无返回值

int index; //暂存最值

int i, k, temp1;

double temp;

for (k = 0; k < N - 1; k++) {

index = k;

for (i = k + 1; i < N; i++) {

if (averageOfTv[i] > averageOfTv[index]) {

index = i;

}

}

if (index != k) {

//交换平均值和地址

temp = averageOfTv[index];

averageOfTv[index] = averageOfTv[k];

averageOfTv[k] = temp;

temp1 = pos[index];

pos[index] = pos[k];

pos[k] = temp1;

}

}

}

manager::manager() {

name = "未定义";

password = "未定义";

}

manager::manager(string name, string password) :name(name), password(password) {

cout << "管理员注册成功！" << endl;

cout << endl;

}

bool manager::checkPassword(char\* password) {

//参数 管理员输入的密码

// 返回值 密码是否正确的bool值

//判断输入密码是否一致

if (password == this->password) {

return 1;

}

else

{

return 0;

}

}

//增加电视剧信息

void manager::Add(InfoOfTv\* tv, Esitimate esitimateOfTv[N][MAXUSER],int\* countOfTv){

//参数 tv电视剧数组 esitimateOfTv评价二维数组 countOfTv主函数电视剧个数

//无返回值

if (count1 != N)//判断电视剧总数有没有超过40

{

//tv数组对应位置增加电视剧信息

cout << "请输入电视剧的名称、类型、导演、主演、集数、剧情" << endl;

cout << "名称：";

cin >> tv[count1].name;

cout << "类型：";

cin >> tv[count1].type;

cout << "导演：";

cin >> tv[count1].director;

cout << "主演：";

cin >> tv[count1].actor;

cout << "集数：";

cin >> tv[count1].number;

while (cin.fail()) {//防止输入错误

cin.clear();

cout << "输入错误请重新输入！" << endl;

cin.ignore(10000, '\n');

cin >> tv[count1].number;

}

cout << "剧情：";

cin >> tv[count1].intro;

//评价信息二维数组对应位置初始化信息

for (int j = 0; j < MAXUSER; j++) { //初始化结构体类型的二维数组存放电视剧评价的各个指标

esitimateOfTv[count1][j].userName = "未输入";

esitimateOfTv[count1][j].plot = 0;

esitimateOfTv[count1][j].behavior = 0;

esitimateOfTv[count1][j].levelOfDirector = 0;

esitimateOfTv[count1][j].levelOfWriter = 0;

esitimateOfTv[count1][j].photographer = 0;

esitimateOfTv[count1][j].music = 0;

esitimateOfTv[count1][j].art = 0;

esitimateOfTv[count1][j].effect = 0;

esitimateOfTv[count1][j].overallEstimate = "未输入";

esitimateOfTv[count1][j].average = 0;

}

count1++; //电视剧数量++

(\*countOfTv)++; //主函数电视剧数量++

//为新建的电视剧新建一个文件用于存储用户对该电视剧的评价

fstream newTv("C:\\Users\\86188\\Documents\\Tencent Files\\971049909\\FileRecv\\热门电视剧评价及推荐系统 (7)\\热门电视剧评价及推荐系统\\热门电视剧评价及推荐系统\\" + tv[count1].name + ".txt",fstream::out);

if (!newTv) {

cout << "不能打开评论信息文件" << endl;

return ;

}

newTv.close();

}

else //电视剧数量超过40

{

cout << "电视剧信息已满，无法再增加信息！";

}

}

//修改电视剧信息

void manager::modify(InfoOfTv\* tv)

{

//参数 tv电视剧数组

//无返回值

cout << "请输入想要修改的电视剧名称：" << endl;

string nameOfTv;

cin >> nameOfTv;

int flag = 0; //用于显示电视剧名称是否在数组之中

int pos; //用于存储当前电视剧在电视剧基本信息数组中的位置

for (int i = 0; i < N; i++) { //查找

if (nameOfTv == tv[i].name) {

pos = i;

flag = -1;

}

if (flag == -1) { //找到信息退出for循环

break;

}

}

if (flag != 0) {

//找到了就返回基本信息

cout << endl;

cout << "《" << tv[pos].name << "》的基本信息是：" << endl;

cout << "名称:" << tv[pos].name << endl;

cout << "类型:" << tv[pos].type << endl;

cout << "导演:" << tv[pos].director << endl;

cout << "主演:" << tv[pos].actor << endl;

cout << "集数:" << tv[pos].number << endl;

cout << "简介:" << tv[pos].intro << endl;

//并且在偏好数组中给这个电视剧加1

}

int a;

cout << "请输入你想要修改的内容" << endl;

cout << "1 、名称 2 、类型 3、 导演 4、主演 5、集数 6、简介 " << endl;

cout << "请输入你的选择";

cin >> a;

while (cin.fail()||(a!=1&&a!=2&& a != 3 && a != 4 && a != 5 && a != 6) ) {

cin.clear();

cout << "输入错误请重新输入！" << endl;

cin.ignore(10000, '\n');

cin >> a;

}

//判断输入内容

if (a == 1)

{

cout << "名称：" << endl;

cin >> tv[pos].name;

}

if (a == 2)

{

cout << "类型：" << endl;

cin >> tv[pos].type;

}

if (a == 3)

{

cout << "导演：" << endl;

cin >> tv[pos].director;

}

if (a == 4)

{

cout << "主演：" << endl;

cin >> tv[pos].actor;

}

if (a == 5)

{

cout << "集数：" << endl;

cin >> tv[pos].number;

while (cin.fail()) {

cin.clear();

cout << "输入错误请重新输入！" << endl;

cin.ignore(10000, '\n');

cin >> tv[pos].number;

}

}

if (a == 6)

{

cout << "简介：" << endl;

cin >> tv[pos].intro;

}

cout << "你已成功修改信息！";

}

//删除电视剧信息

void manager::deletetv(InfoOfTv\* tv, Esitimate esitimateOfTv[N][MAXUSER], int\* countOfTv,int\* posOfDelete)

{

//参数 tv电视剧数组 esitimateOfTv评价二维数组 countOfTv主函数电视剧个数 posOfDelete删除位置

//无返回值

cout << "请输入你想要删除的电视剧" << endl;

string nameOfTv;

cin >> nameOfTv;

int flag = 0; //用于显示电视剧名称是否在数组之中

int pos; //用于存储当前电视剧在电视剧基本信息数组中的位置

for (int i = 0; i < N; i++)

{ //查找

if (nameOfTv == tv[i].name) {

pos = i;

flag = -1;

}

if (flag == -1) {

break;

}

}

if (flag != 0) {

int i;

//硬盘中 相应被删除的电视剧评价的txt文件也要被删除

string nameOfFile = "C:\\Users\\86188\\Documents\\Tencent Files\\971049909\\FileRecv\\热门电视剧评价及推荐系统 (7)\\热门电视剧评价及推荐系统\\热门电视剧评价及推荐系统\\" + tv[pos].name + ".txt";

remove(nameOfFile.c\_str());

//内存中 用后面的覆盖之前的

for (i = pos; i < N-1; i++)

{

tv[i] = tv[i + 1];

for (int j = 0; j < MAXUSER; j++) {

esitimateOfTv[i][j] = esitimateOfTv[i + 1][j];

}

//用户倾向数组也要进行删除

}

\*posOfDelete = pos;

count1--;

(\*countOfTv)--; //主函数电视剧数量--

cout << "删除电视剧成功！";

}

else

{

cout << "没有该电视剧信息！";

}

}

//按照名称返回电视剧信息

void manager::searchofmanager(InfoOfTv\* tv)

{

//参数 tv电视剧数组

//无返回值

cout << "请输入想要查找的电视剧名称：" << endl;

string nameOfTv;

cin >> nameOfTv;

int flag = 0; //用于显示电视剧名称是否在数组之中

int pos; //用于存储当前电视剧在电视剧基本信息数组中的位置

for (int i = 0; i < N; i++) { //查找

if (nameOfTv == tv[i].name) {

pos = i;

flag = -1;

}

if (flag == -1) {

break;

}

}

if (flag != 0) {

//找到了就返回基本信息

cout << endl;

cout << "《" << tv[pos].name << "》的基本信息是：" << endl;

cout << "名称:" << tv[pos].name << endl;

cout << "类型:" << tv[pos].type << endl;

cout << "导演:" << tv[pos].director << endl;

cout << "主演:" << tv[pos].actor << endl;

cout << "集数:" << tv[pos].number << endl;

cout << "简介:" << tv[pos].intro << endl;

//并且在偏好数组中给这个电视剧加1

}

else

{

cout << "没有该电视剧信息哦~";

}

}

//查看所有电视剧评价

void manager::esitimateFromuser(Esitimate esitimateOfTv[N][MAXUSER], InfoOfTv\* tv)

{

//参数 esitimateOfTv评价二维数组 tv电视剧数组

//无返回值

int i;

cout << "用户对电视剧的评价是：" << endl;

for (int j = 0; j < N; j++) //输出电视剧评价的二维数组

{

if (tv[j].name[0]== '\0')

{

continue;

}

cout << "《" << tv[j].name << "》:" << endl;

for (i = 0; i < MAXUSER; i++)

{

if (esitimateOfTv[j][i].userName == "未输入")

{

continue;

}

else

{

cout << esitimateOfTv[j][i].userName << ": " << esitimateOfTv[j][i].overallEstimate << " ta的总体评价是：" << esitimateOfTv[j][i].average << endl;

}

}

cout << endl;

}

}

//输出特定指标的排序

void outputByLevel(int pos[N], InfoOfTv\* tv) {

//参数 pos临时电视剧地址数组 tv电视剧数组

//无返回值

cout << "请输入需要查看的分数段：" << endl;

cout << "1、1~10名 2、11~20名 3、21~30名 4、31~40名" << endl;

int choice,level;

cin >> choice;

while ((choice != 1 && choice != 2 && choice != 3 && choice != 4) || cin.fail()) {

cout << "输入错误，请重新输入！" << endl;

cin.clear();

cin.ignore(1000, '\n');

cin >>choice;

}

//保证输入正确

if (choice == 1) {

level = 10;

}

else if (choice == 2) {

level = 20;

}

else if (choice == 3) {

level = 30;

}

else

{

level = 40;

}

for (int i = level-10; i < level; i++) {

cout << "第" << i + 1 << "名:" << endl;

cout << "名称:" << tv[pos[i]].name << endl;

}

}

//按照排名分数段查看

void manager::rank(int choice,Esitimate esitimateOfTv[N][MAXUSER], InfoOfTv\* tv)

{

// 参数 choice 指标变量 esitimateOfTv评价二维数组 tv电视剧数组

//无返回值

double averageOfTv[N]; //存放特定指标的平均分

int pos[N]; //临时存放电视剧位置的数组，用于排序

double sum = 0.00;

if (choice == 1) {

for (int i = 0; i < N; i++) {

for (int j = 0; j < MAXUSER; j++) {

sum += esitimateOfTv[i][j].plot;

}

averageOfTv[i] = sum / MAXUSER;

pos[i] = i;

sum = 0.00;

}

rankIndex(pos, averageOfTv);

outputByLevel(pos, tv);

}

else if (choice == 2) {

for (int i = 0; i < N; i++) {

for (int j = 0; j < MAXUSER; j++) {

sum += esitimateOfTv[i][j].behavior;

}

averageOfTv[i] = sum / MAXUSER;

pos[i] = i;

sum = 0.00;

}

rankIndex(pos, averageOfTv);

outputByLevel(pos, tv);

}

else if (choice == 3) {

for (int i = 0; i < N; i++) {

for (int j = 0; j < MAXUSER; j++) {

sum += esitimateOfTv[i][j].levelOfDirector;

}

averageOfTv[i] = sum / MAXUSER;

pos[i] = i;

sum = 0.00;

}

rankIndex(pos, averageOfTv);

outputByLevel(pos, tv);

}

else if (choice == 4) {

for (int i = 0; i < N; i++) {

for (int j = 0; j < MAXUSER; j++) {

sum += esitimateOfTv[i][j].levelOfWriter;

}

averageOfTv[i] = sum / MAXUSER;

pos[i] = i;

sum = 0.00;

}

rankIndex(pos, averageOfTv);

outputByLevel(pos, tv);

}

else if (choice == 5) {

for (int i = 0; i < N; i++) {

for (int j = 0; j < MAXUSER; j++) {

sum += esitimateOfTv[i][j].photographer;

}

averageOfTv[i] = sum / MAXUSER;

pos[i] = i;

sum = 0.00;

}

rankIndex(pos, averageOfTv);

outputByLevel(pos, tv);

}

else if (choice == 6) {

for (int i = 0; i < N; i++) {

for (int j = 0; j < MAXUSER; j++) {

sum += esitimateOfTv[i][j].music;

}

averageOfTv[i] = sum / MAXUSER;

pos[i] = i;

sum = 0.00;

}

rankIndex(pos, averageOfTv);

outputByLevel(pos, tv);

}

else if (choice == 7) {

for (int i = 0; i < N; i++) {

for (int j = 0; j < MAXUSER; j++) {

sum += esitimateOfTv[i][j].art;

}

averageOfTv[i] = sum / MAXUSER;

pos[i] = i;

sum = 0.00;

}

rankIndex(pos, averageOfTv);

outputByLevel(pos, tv);

}

else if(choice==8){

for (int i = 0; i < N; i++) {

for (int j = 0; j < MAXUSER; j++) {

sum += esitimateOfTv[i][j].effect;

}

averageOfTv[i] = sum / MAXUSER;

pos[i] = i;

sum = 0.00;

}

rankIndex(pos, averageOfTv);

outputByLevel(pos, tv);

}

else

{

for (int i = 0; i < N; i++) {

for (int j = 0; j < MAXUSER; j++) {

sum += esitimateOfTv[i][j].average;

}

averageOfTv[i] = sum / MAXUSER;

pos[i] = i;

sum = 0.00;

}

rankIndex(pos, averageOfTv);

outputByLevel(pos, tv);

}

}

#include<string>

#include<conio.h>

#include"管理员.h"

#include"用户.h"

using namespace std;

void menuOfStatus(); //身份选择菜单

void menuOfManager(); //管理员菜单

void menuOfRegister(); //用户登录注册菜单

void menuOfUser(); //用户菜单

void menuOftv(); //用户对于一个电视剧的选择

int main()

{

/\*

注册管理员

\*/

cout << "请输入管理员的姓名：";

string name;

cin >> name;

cout << "请设置管理员的密码（6位）：";

string password;

cin >> password;

while (password.length() != 6) { //检测密码位数

cout << "输入位数不正确，请重新输入！" << endl;

cin >> password;

}

manager manager(name,password); //创建管理员对象

/\*

主函数中的变量

\*/

int countOfUser = 0; //用户个数

int countOfTv = 0; //电视剧个数

int posOfDelete=N; //管理员删除电视剧时电视剧在tv数组中的位置 ，用于调整用户偏好数组

int choice1, choice2, choice3,choice4,choice5; //对应1、2、3、4、级菜单的选项

user\* userPtr[MAXUSER]; //用户数组指针

InfoOfTv tv[N]; //用于把文件中的数据存入内存

Esitimate esitimateOfTv[N][MAXUSER]; //每一个电视剧的评价

/\*

读取电视剧数据文件和电视剧评价文件

\*/

//打开电视剧数据文件

fstream docOfTv("C:\\Users\\86188\\Documents\\Tencent Files\\971049909\\FileRecv\\热门电视剧评价及推荐系统 (7)\\热门电视剧评价及推荐系统\\热门电视剧评价及推荐系统\\三十部电视剧简介缩略版.txt");

if (!docOfTv) {

cout << "不能打开电视剧信息文件" << endl;

return 0;

}

//将文件读入内存

for (int i = 0; i < N; i++) {

docOfTv >> tv[i].name;

if (tv[i].name[0] == '\0') {

break;

}

docOfTv >> tv[i].type;

docOfTv >> tv[i].director;

docOfTv >> tv[i].actor;

docOfTv >> tv[i].number;

docOfTv >> tv[i].intro;

countOfTv++;

}

docOfTv.close();

//打开电视剧评价文件

ifstream esitimateList[N]; //每一个电视剧评价用一个文件存储 流对象存储在一个数组里 地址和电视剧在tv数组里的一样

for (int k = 0; k < countOfTv; k++) {

esitimateList[k].open("C:\\Users\\86188\\Documents\\Tencent Files\\971049909\\FileRecv\\热门电视剧评价及推荐系统 (7)\\热门电视剧评价及推荐系统\\热门电视剧评价及推荐系统\\"+tv[k].name+".txt");

//fstream类只有在out方法下才可以创建文档 或者使用ofstream

if (!esitimateList[k]) {

cout << "不能打开评论信息文件" << endl;

return 0;

}

}

//将电视剧评价信息读入内存

for (int i = 0; i < countOfTv; i++) {

for (int j = 0; j < MAXUSER; j++) {

esitimateList[i] >> esitimateOfTv[i][j].userName;

esitimateList[i] >> esitimateOfTv[i][j].plot;

esitimateList[i] >> esitimateOfTv[i][j].behavior;

esitimateList[i] >> esitimateOfTv[i][j].levelOfDirector;

esitimateList[i] >> esitimateOfTv[i][j].levelOfWriter;

esitimateList[i] >> esitimateOfTv[i][j].photographer;

esitimateList[i] >> esitimateOfTv[i][j].music;

esitimateList[i] >> esitimateOfTv[i][j].art;

esitimateList[i] >> esitimateOfTv[i][j].effect;

esitimateList[i] >> esitimateOfTv[i][j].overallEstimate;

esitimateList[i] >> esitimateOfTv[i][j].average;

}

}

//为电视剧评价信息中暂时为空的数据赋予初值

for (int i = 0; i < N; i++) {

for (int j = 0; j < MAXUSER; j++) {

if (esitimateOfTv[i][j].userName[0] == '\0') {

//如果导入的评价信息为空 把相关评价设置为0

esitimateOfTv[i][j].userName = "未输入";

esitimateOfTv[i][j].plot = 0;

esitimateOfTv[i][j].behavior = 0;

esitimateOfTv[i][j].levelOfDirector = 0;

esitimateOfTv[i][j].levelOfWriter = 0;

esitimateOfTv[i][j].photographer = 0;

esitimateOfTv[i][j].music = 0;

esitimateOfTv[i][j].art = 0;

esitimateOfTv[i][j].effect = 0;

esitimateOfTv[i][j].overallEstimate = "未输入";

esitimateOfTv[i][j].average = 0;

}

}

}

for (int i = 0; i < countOfTv; i++) {

esitimateList[i].close();

}

/\*

主函数菜单模块

\*/

do

{

menuOfStatus();

cin >> choice1;

while (cin.fail()) {

cin.clear();

cout << "输入错误请重新输入！" << endl;

cin.ignore(10000, '\n');

cin >> choice1;

}

switch (choice1) {

case 1: {

//身份选择是管理员

//管理员身份需要先通过密码

int time = 0; //只有三次机会输入

char c[7]; //存放管理员输入密码的临时数组

do

{

int count = 0; //防止输入过多

if (time == 0) {

cout << "请输入密码: " << endl;;

}

else

{

cout << endl << "密码错误，请再次输入密码！你还有"<<3-time<<"次机会: " << endl;

}

while (count < 6)

{

c[count] = getch(); //getch可以不显示输入的字符

if (c[count] == '\r') //回车键表示\r\n

{

break;

}

cout << "\*";

count++;

}

c[6] = '\0';

if (!manager.checkPassword(c)) {

time++;

}

else //密码正确跳出do while循环

{

break;

}

} while (time<3);

if (time == 3) { //3次错误直接退出程序

return 0;

}

//管理员菜单

do

{

menuOfManager();

cin >> choice2;

while (cin.fail()) {

cin.clear();

cout << "输入错误请重新输入！" << endl;

cin.ignore(10000, '\n');

cin >> choice2;

}

switch (choice2) {

case 1:

//增加电视剧信息

manager.Add(tv,esitimateOfTv,&countOfTv);

break;

case 2:

//修改电视剧信息

manager.modify(tv);

break;

case 3:

//删除电视剧信息

manager.deletetv(tv,esitimateOfTv,&countOfTv,&posOfDelete);

break;

case 4:

//按照名称返回电视剧信息

manager.searchofmanager(tv);

break;

case 5:

//查看所有电视剧的评价

manager.esitimateFromuser(esitimateOfTv, tv);

break;

case 6:

//按照排名分数段查看

cout << "请输入需要分段排行的项目：" << endl;

cout << "1、情节 2、演员表现力 3、导演水平 4、编剧水平 5、摄影 6、音乐 7、美术 8、特效 9、总体评价" << endl;

int choiceOfManager;

cin >> choiceOfManager;

while ((choiceOfManager != 1 && choiceOfManager != 2 && choiceOfManager != 3 && choiceOfManager != 4 && choiceOfManager != 5 && choiceOfManager != 6 && choiceOfManager != 7 && choiceOfManager != 8 && choiceOfManager != 9)||cin.fail()) {

cout << "输入错误，请重新输入" << endl;

cin.clear();

cin.ignore(10000, '\n');

cin >> choiceOfManager;

}

manager.rank(choiceOfManager,esitimateOfTv, tv);

break;

case 0:

//返回

break;

default:

cout << "输入选项错误！请重新输入" << endl;

break;

}

} while (choice2);

break;

}

case 2:

//身份选择是普通用户

do{

menuOfRegister(); //显示登陆菜单

cin >> choice5;

while (cin.fail()) {

cin.clear();

cout << "输入错误请重新输入！" << endl;

cin.ignore(10000, '\n');

cin >> choice5;

}

switch (choice5){

case 1: {

//注册模块

cout << "请输入用户名" << endl;

string name;

cin >> name;

//查重模块

int flag = 0;

for (int i = 0; i < countOfUser; i++) {

if (userPtr[i]->getName() == name) {

cout << "用户名已被使用！" << endl;

flag = -1;

}

}

if (flag == 0) {

userPtr[countOfUser] = new user(name); //创建用户类

countOfUser++;

}

break;

}

case 2: {

//登陆模块

cout << "请输入用户名：" << endl;

string userName;

cin >> userName;

//在用户数组中搜索是否有该用户

int flag = 0; //用于显示用户是否在数组之中

int pos; //用于存储当前登录用户在用户类数组中的位置

for (int i = 0; i < countOfUser; i++) {

if (userName == userPtr[i]->getName()) {

pos = i;

flag = -1;

}

}

if (flag != 0) {

cout << "欢迎" << userPtr[pos]->getName() <<"!!!"<< endl;

}

else

{

//登录失败返回上一级菜单

cout << "未检测到该用户！" << endl;

break;

}

do {

//登录成功显示以下菜单

menuOfUser();

cin >> choice3;

while (cin.fail()) {

cin.clear();

cout << "输入错误请重新输入！" << endl;

cin.ignore(10000, '\n');

cin >> choice3;

}

switch (choice3) {

case 1: {

//系统推荐用户观看电视剧

int max = userPtr[pos]->recommendTv(&posOfDelete);

cout << "我们推荐您看《" << tv[max].name << "》哦~" << endl;

break;

}

case 2:

//返回前10名电视剧的基本信息

userPtr[pos]->TopTenTv(esitimateOfTv, tv);

break;

case 3:

//按照相关指标进行前10名电视剧的排行

cout << "请输入您想检索的指标：" << endl;

cout << "1、情节 2、演员表现力 3、导演水平 4、编剧水平 5、摄影 6、音乐 7、美术 8、特效" << endl;

int choice;

cin >> choice;

while ((choice != 1 && choice != 2 && choice != 3 && choice != 4 && choice != 5 && choice != 6 && choice != 7 && choice != 8)||cin.fail()) {

cout << "输入错误，请重新输入！" << endl;

cin.clear();

cin.ignore(1000, '\n');

cin >> choice;

}

userPtr[pos]->TopTenTvByIndex(choice, esitimateOfTv, tv);

break;

case 4:

//用户按照电视剧名称查找

//调用用户类 searchTv方法

int posOfThis; //标记当前搜索的电视剧的位置

if (userPtr[pos]->searchTv(tv,&posOfThis)) {

//找到电视剧进入下一级菜单

do{

menuOftv();

cin >> choice4;

while (cin.fail()) {

cin.clear();

cout << "输入错误请重新输入！" << endl;

cin.ignore(10000, '\n');

cin >> choice4;

}

switch (choice4)

{

case 1:

//观看电视剧

cout << "观看成功！" << endl;

cout <<endl<< "请问您想对本电视剧进行评价吗？（Y/N）" << endl;

char esitimateOrNot;

cin >> esitimateOrNot;

if (esitimateOrNot == 'y' || esitimateOrNot == 'Y') {

userPtr[pos]->esitimateTv(posOfThis,pos,esitimateOfTv,tv);

}

else if (esitimateOrNot == 'N' || esitimateOrNot == 'n') {

break;

}

else

{

cout << "输入有误！" << endl;

}

break;

case 2:

//观看其他用户的评价

userPtr[pos]->esitimateFromOtherUser(posOfThis, esitimateOfTv, tv);

break;

case 0:

break;

default:

cout << "输入错误！请重新输入" << endl;

break;

}

} while (choice4);

}

else

{

cout << "没有找到这个电视剧哦~" << endl;

break;

}

break;

case 0:

//返回

break;

default:

cout << "输入有误！请重新输入" << endl;

break;

}

} while (choice3);

}

//登录模块以及登录后的所有菜单

case 0:

//返回

break;

default:

cout << "输入选项错误！请重新输入" << endl;

break;

}

} while (choice5);

case 0:

cout << "谢谢使用本程序！！！" << endl;

break;

default:

cout << "输入选项错误！请重新输入" << endl;

break;

}

} while (choice1);

/\*

内存数据输出至外部文件

方法就是把原来文件中的内容用trunc方法清空

\*/

//将内存中的数据输出

fstream newDocOfTv("C:\\Users\\16046\\Desktop\\编程\\热门电视剧评价及推荐系统\\热门电视剧评价及推荐系统\\三十部电视剧简介缩略版.txt", ios::out|ios::trunc);

if (!newDocOfTv) {

cout << "不能打开电视剧信息文件" << endl;

return 0;

}

for (int i = 0; i < countOfTv; i++) {

newDocOfTv << tv[i].name<< " ";

newDocOfTv << tv[i].type << " ";

newDocOfTv << tv[i].director << " ";

newDocOfTv << tv[i].actor << " ";

newDocOfTv << tv[i].number << " ";

newDocOfTv << tv[i].intro << " ";

}

newDocOfTv.close();

fstream newEsitimateList[N]; //每一个电视剧评价用一个文件存储 流对象存储在一个数组里 地址和电视剧在tv数组里的一样

for (int k = 0; k < countOfTv; k++) {

newEsitimateList[k].open("C:\\Users\\16046\\Desktop\\编程\\热门电视剧评价及推荐系统\\热门电视剧评价及推荐系统\\" + tv[k].name + ".txt", fstream::out|fstream::trunc);

//fstream类只有在out方法下才可以创建文档 或者使用ofstream

if (!newEsitimateList[k]) {

cout << "不能打开评论信息文件" << endl;

return 0;

}

}

//将电视剧评价信息读入内存

for (int i = 0; i < countOfTv; i++) {

for (int j = 0; j < MAXUSER; j++) {

newEsitimateList[i] << esitimateOfTv[i][j].userName<<" ";

newEsitimateList[i] << esitimateOfTv[i][j].plot << " ";

newEsitimateList[i] << esitimateOfTv[i][j].behavior << " ";

newEsitimateList[i] << esitimateOfTv[i][j].levelOfDirector << " ";

newEsitimateList[i] << esitimateOfTv[i][j].levelOfWriter << " ";

newEsitimateList[i] << esitimateOfTv[i][j].photographer << " ";

newEsitimateList[i] << esitimateOfTv[i][j].music << " ";

newEsitimateList[i] << esitimateOfTv[i][j].art << " ";

newEsitimateList[i] << esitimateOfTv[i][j].effect << " ";

newEsitimateList[i] << esitimateOfTv[i][j].overallEstimate << " ";

newEsitimateList[i] << esitimateOfTv[i][j].average << " ";

}

}

for (int i = 0; i < N; i++) {

newEsitimateList[i].close();

}

return 0;

}

void menuOfStatus() {

cout << "请问你的身份是？" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*1、管理员\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*2、用 户\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*0、退 出\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "请输入您的选项:" << endl;

}

void menuOfManager() {

cout << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*1、增加电视剧\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*2、修改电视剧信息\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*3、删除电视剧信息\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*4、按照名称返回电视剧信息\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*5、查看所有电视剧的评价\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*6、按照排名分数段查看\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*0、返 回\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "请输入您的选项:" << endl;

}

void menuOfRegister() {

cout << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*1、注 册\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*2、登 录\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*0、返 回\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "请输入您的选项：" << endl;

}

void menuOfUser() {

cout << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*1、查看系统推荐电视剧\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*2、查看前10名电视剧基本信息\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*3、按照相关指标查询前10名电视剧基本信息\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*4、按照名称返回电视剧信息\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*0、返 回\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "请输入您的选项:" << endl;

}

void menuOftv() {

cout << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*1、观看该电视剧\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*2、查看其它用户对它的评价\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*0、返 回\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "请输入您的选项:" << endl;

}

**五、测试数据及其结果分析**

运行后，程序的主菜单界面如下。

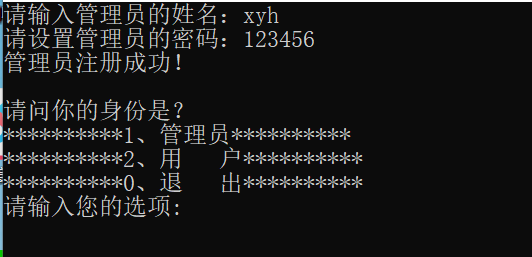


图3.主菜单

请输入所要进行的操作：

功能1：对象注册/登录（用户）

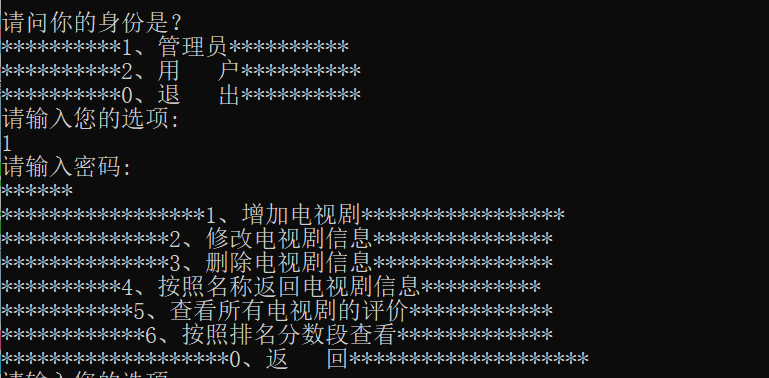


图4. 登录界面（管理员）

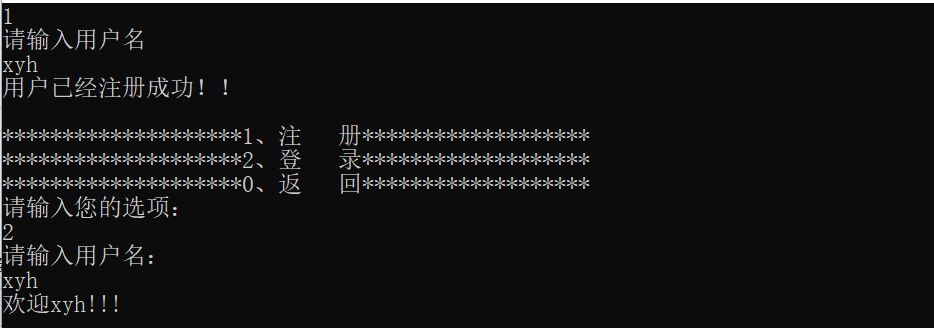


图5.注册界面（用户）

功能2：登录对应身份的账号

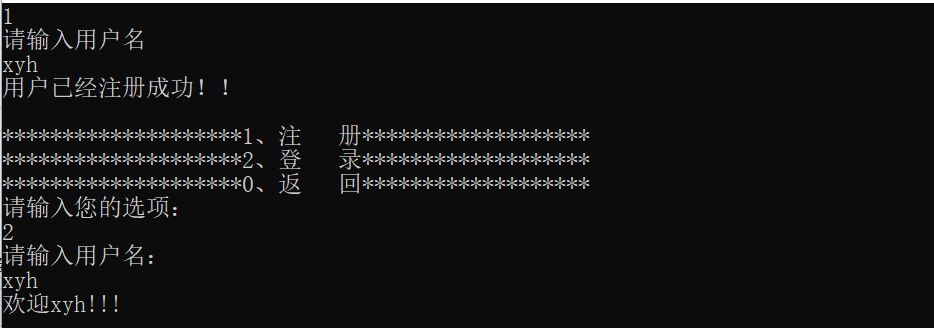


图6. 用户登录界面

功能3：以管理员身份使用

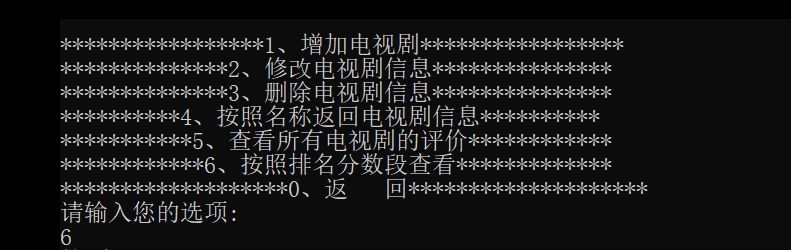


图7. 管理员可用功能

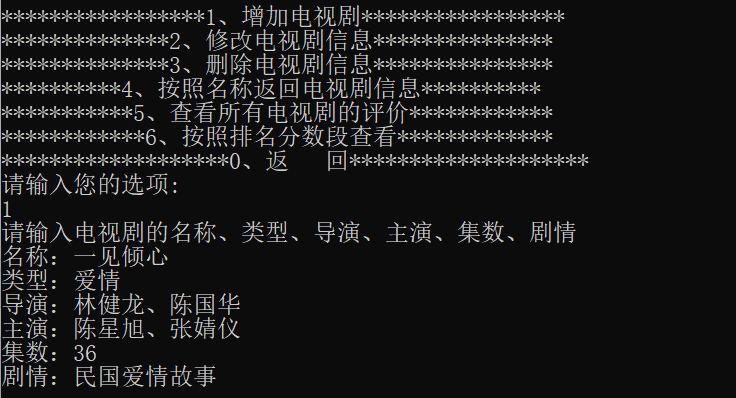


图8.增加电视剧信息

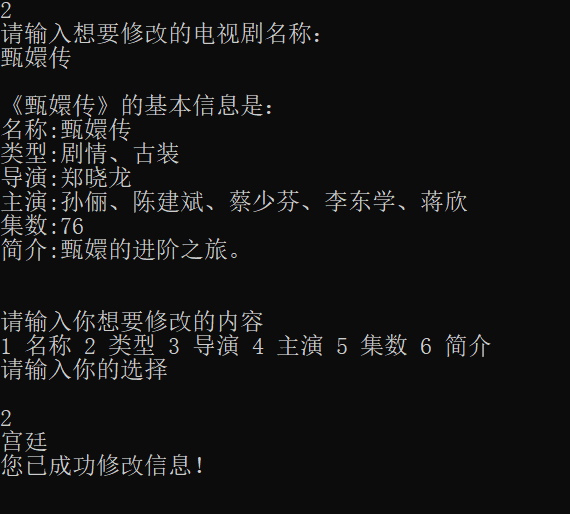


图9. 修改电视剧信息

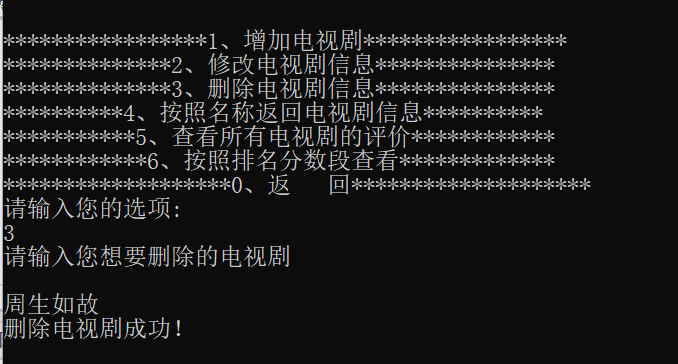


图10. 删除电视剧信息

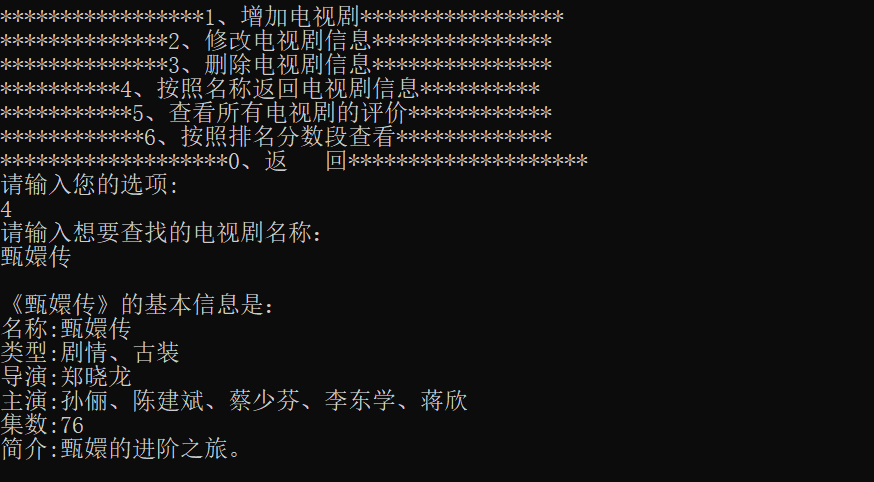


图11. 按照名称返回电视剧信息

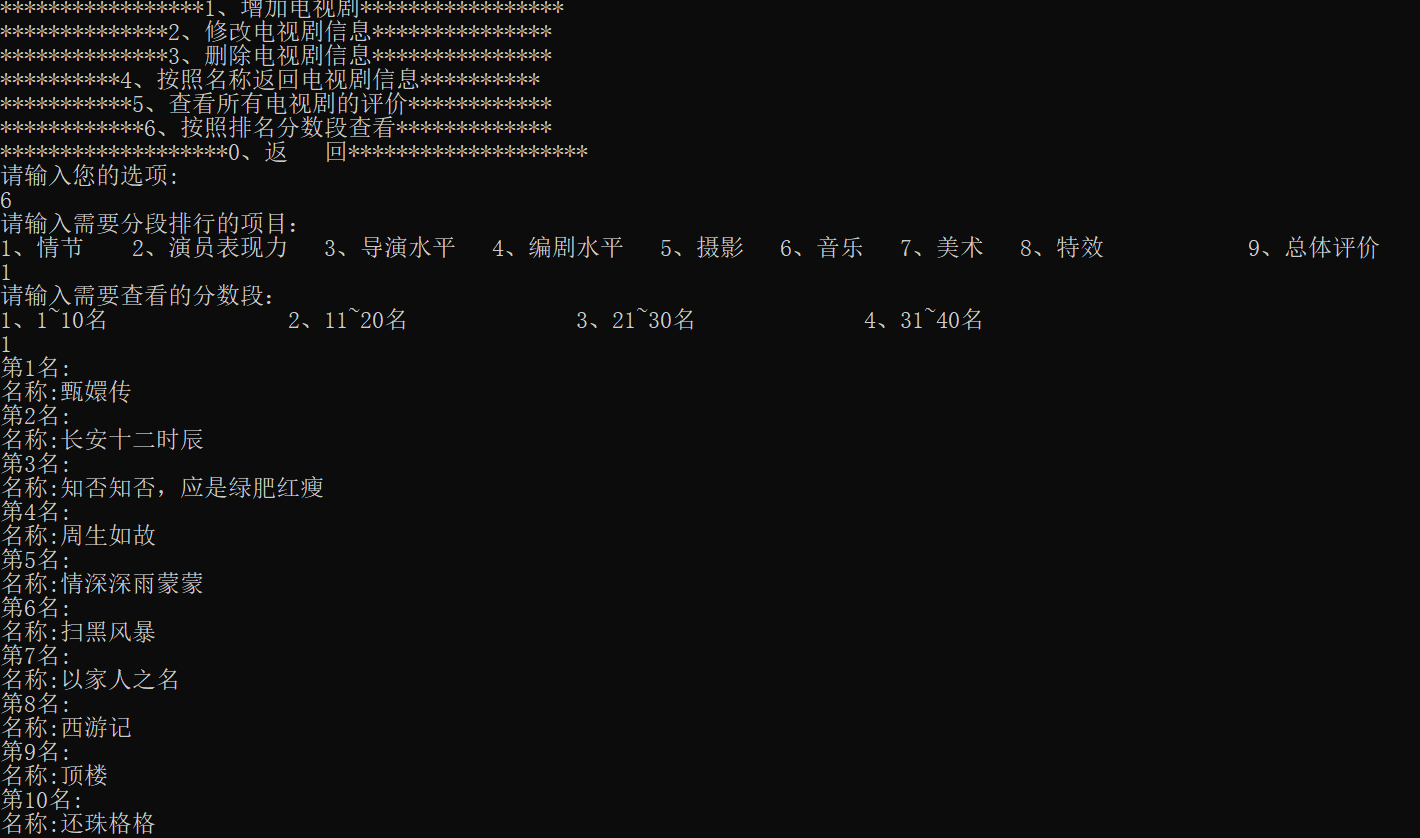


图12. 按照排名分数段查看

功能4：以用户身份使用

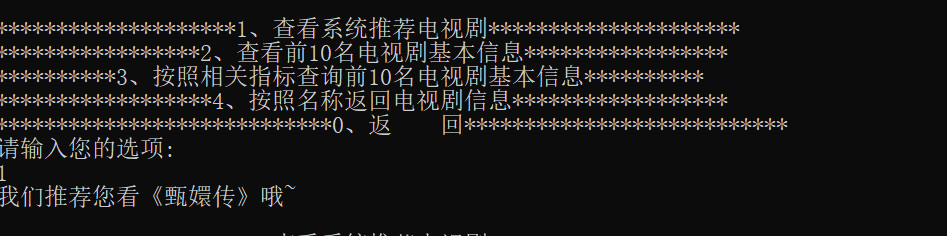
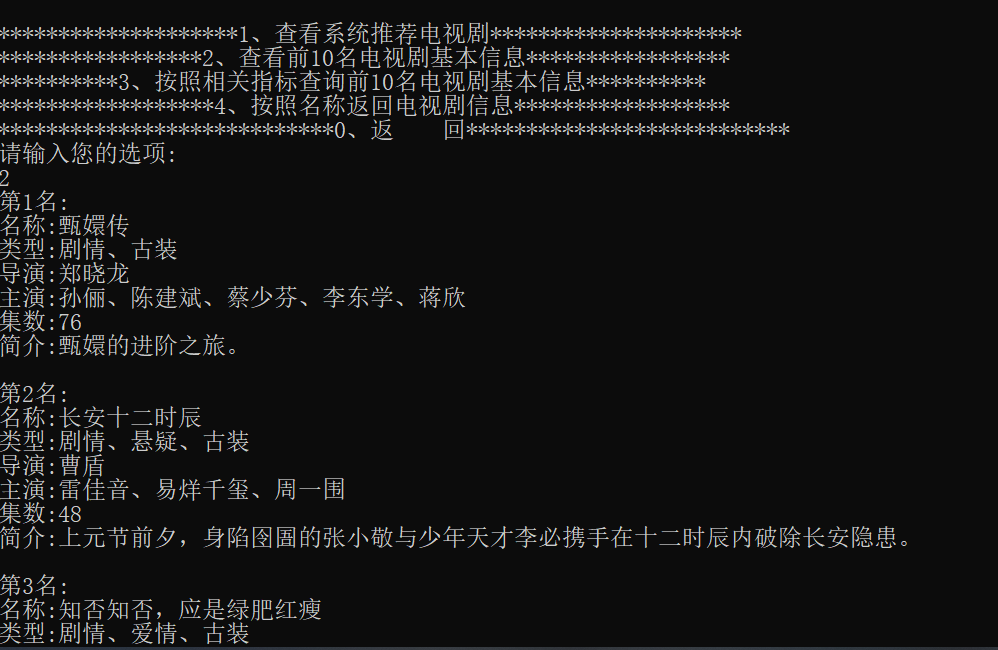
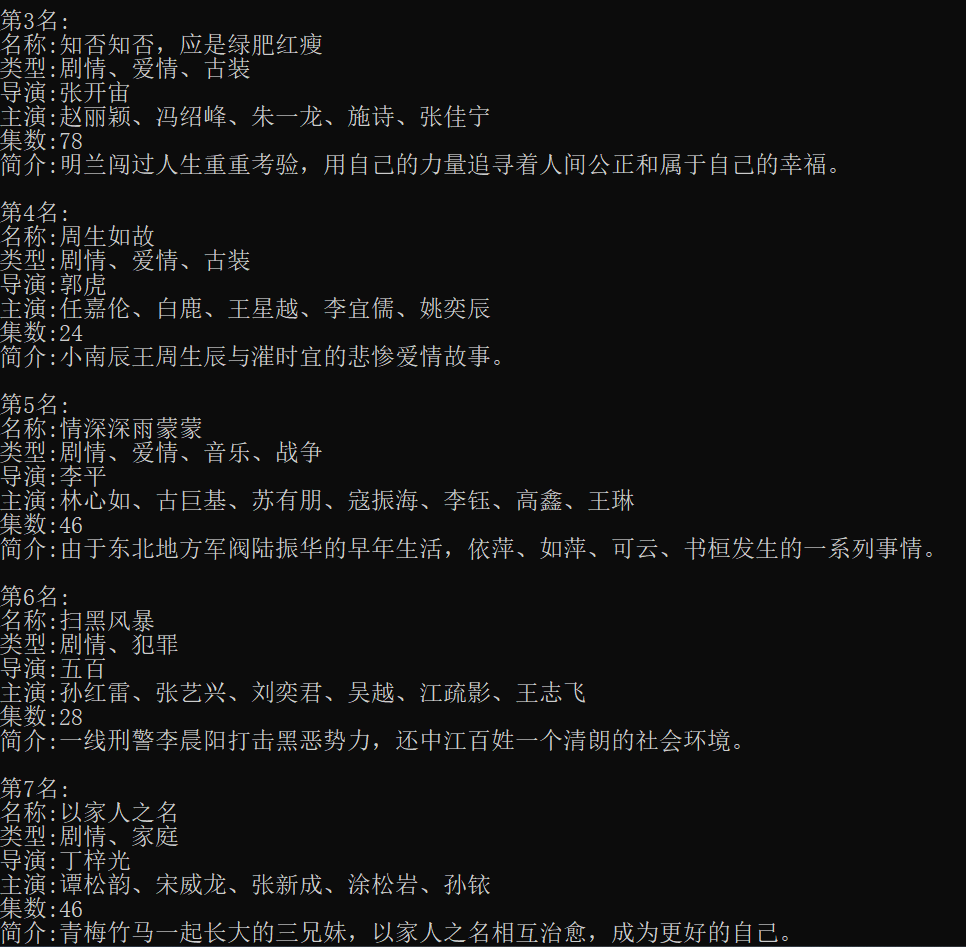


图13.查看系统推荐电视剧





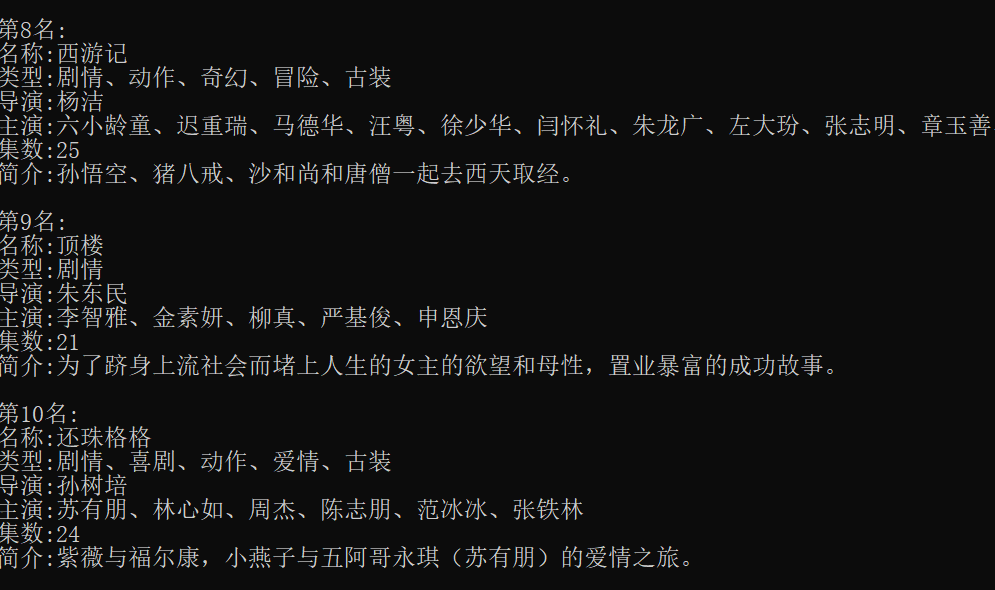


图14. 查看前十名电视剧基本信息

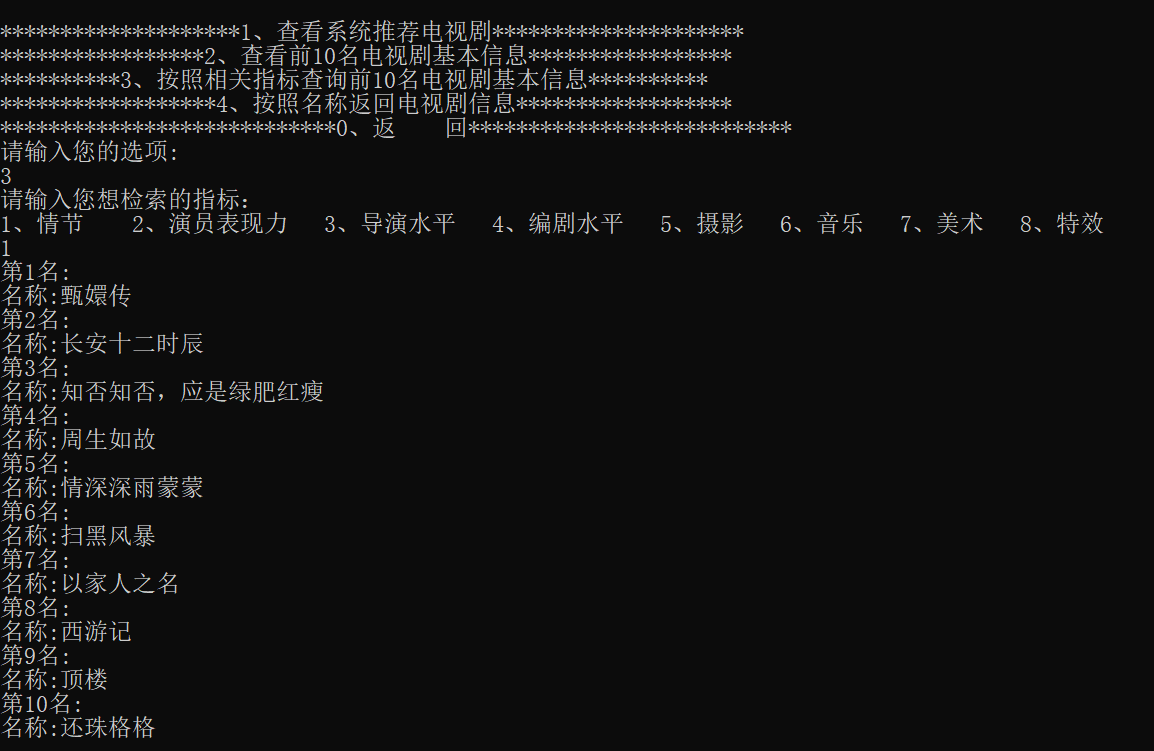


图15. 按照相关指标查询前10名电视剧基本信息

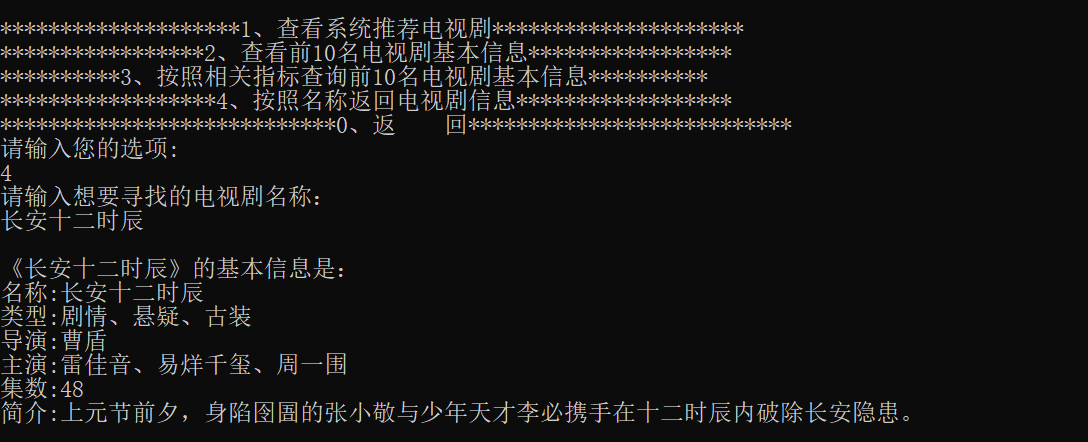


图16. 按照名称返回电视剧信息

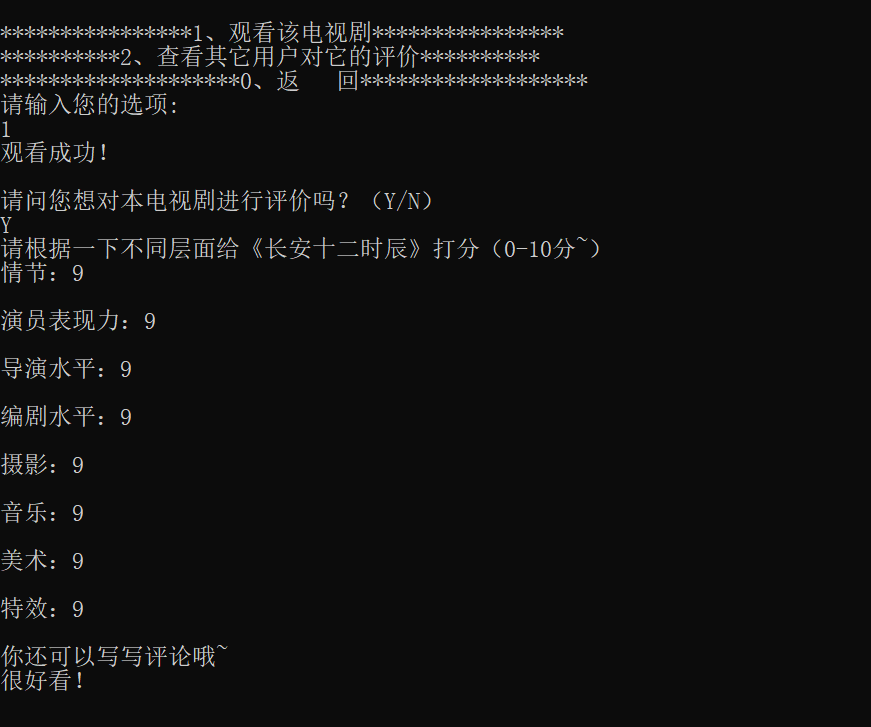


图17. 观看该电视剧并对该电视剧进行评价

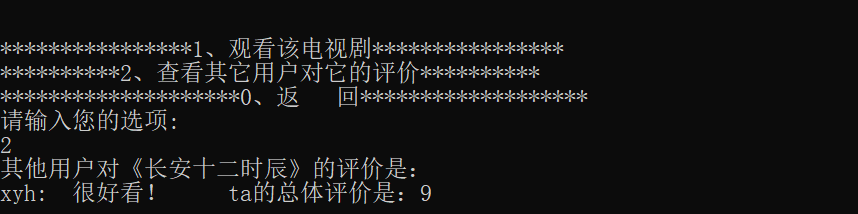


图18 . 查看用户对他的评价

**六、调试过程中的问题**

1、cin错误输入对于程序健壮性的影响

例图向int型数据中输入字符型会导致cin流错误，为了防止用户错误输入，在所有的cin输入里增加了检测模块

2、管理员删除操作的逻辑漏洞

管理员执行删除电视剧操作会影响到内存很多数据

一开始只是单纯删除了tv数组中的数据 没有考虑到对评价二维数组以及用户偏好数组的影响

并且没有考虑到删除相关电视剧的txt文件

3、文件流输出的相关问题

建立新文件fstream流使用ios::out方法可以创建不存在的文件

Ostrea流可以直接创建不存在的文件

Ios：：trunc方法可以将文件中已有的内容删除

1. **课程总结**

本次程序设计周是学习编程后第一次真正的进行编程，以做出一个符合要求的小程序。尽管是小组合作完成这个任务，工作量依旧不算小。还需要考虑和伙伴们的配合，高质量快速的完成自己任务的同时尽量让代码整合更容易一些，不给后期集中收尾时增加负担。每一个变量或者结构体的命名都需要注意要让伙伴也可以轻易识别，让大家都可以轻易知道你的编程思路。尽管如此，由于编程风格和习惯的不同，后期整合的时候还是比想象中要费劲，可能这就是这次实验安排我们小组合作完成的意义吧，毕竟一个人永远不可能独自完成那些庞大的程序设计，一定要和他人合作的。我们在具有独自编程解决问题的能力以后还要培养自己高效和别人合作的能力。

具体的编程实现过程中，我们先一起思考并列出来整个程序功能的框架图，然后再具体分配任务，独立编写时遇到困难也经常三个人一起讨论，以使我们的代码更加完美的实现所要求的功能。面对我们三个都不能轻易解决的问题时，例如用户评价，讨论无果时我们会先分开各自查找资料或者寻求帮助，如果有什么突破口会第一时间进行交流。在这个过程中，不仅锻炼了我们缜密的思维和坚持不解的毅力，更磨练了一个队伍的团结互助的精神。问题解决后，我们由衷的体会到合作的快乐，只有通过大家一起努力才能将课程设计的所有环节都顺利的完成。