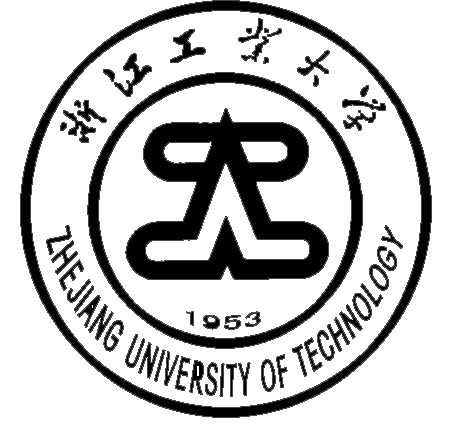


C++程序设计课程设计

实验报告

**2019/2020(2)**



实验题目 歌手比赛系统

学生姓名 王宇

学生学号 202103151320

学生班级 软工04

任课教师 廖锋峰

提交日期

**计算机科学与技术学院**

**歌手比赛系统 实验报告**

1. **大型实验的内容**

歌手比赛系统（SCS:Singer Contest System）可以提供对于歌唱比赛信息的管理，要求完成的主要功能有比赛场次管理，歌手信息维护，观众使用系统，管理员进行后台管理等。可以完成多场比赛的管理，单独各场比赛选手的排序、搜索、投票以及个人信息和成绩的修改，歌手的晋级等功能。要求使用学习过的C/C++程序设计的知识完成图书管理系统的设计与实现。

1. **运行环境**

**软件环境：**

开发软件：Visual Studio 2019

操作系统：Windows 10

**硬件环境：**

处理器：Intel(R) Core(TM) i7-10875H CPU @ 2.30GHz 2.30 GHz

内存：16.00GB

系统类型：64位操作系统

1. **实验课题分析**

**3.1 歌手比赛系统的主要功能**

**歌手比赛系统（SCS）**的主要功能为：普通用户对比赛信息的查看和对喜欢的歌手进行投票；管理员可以对比赛信息做出修改和进行维护。大致分为三大模块。

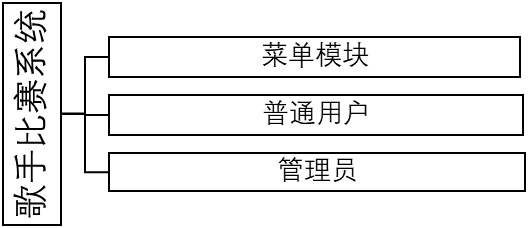


图1 系统结构图

模块具体功能为：

1. **菜单模块**：

进入程序即显示菜单，并先从文件读入所有的数据，用户选择普通用户或管理员身份登陆，普通用户只需任意输入一个ID即可，管理员需要正确输入密码。之后可以进行相应用户组的操作。并且程序正常退出的时候，会保存所有修改后的数据，录入文件，以便下次运行时再使用。

1. **普通用户**：

①可以查看所有场次的比赛的歌手各项信息，包括编号、姓名、评委投票、获得的观众投票数、综合得分等，

②用户可以在一场比赛下通过歌手姓名来搜索歌手，支持模糊搜索，如果有多个结果，会显示所以匹配的结果。

③用户可以对比赛的歌手按照某一项分数（如：单个评委投票，综合得分，投票数等）查看排序后的歌手列表。

④用户可以对喜欢的歌手进行投票，每场未结束的比赛，一个用户可以投出一票。

1. **管理员**：

①具备普通用户①—③的操作选项。

②可以在原来的比赛基础上，进行晋级操作，比如将原先16人的比赛进入下一阶段，晋级8强后继续比赛。

③可以修改比赛的名称，也可以重新开放一场比赛。

④可以删除某一场比赛。

⑤可以对一场比赛的歌手进行新增歌手，修改歌手信息，删除歌手信息等操作。

**3.2 系统分析及设计**

根据功能的实现，系统大致有三个大类组成。

1. 链表类：其中有通过C++模板实现的双向链表类—myList
2. 数据类：其中有contest 类和singer 类，用于存储比赛信息和歌手信息
3. 操作类：其中有basic类、user类和admin类，其中user类和admin类由basic类派生而来，其中basic类具有普通用户和管理员共有的操作。

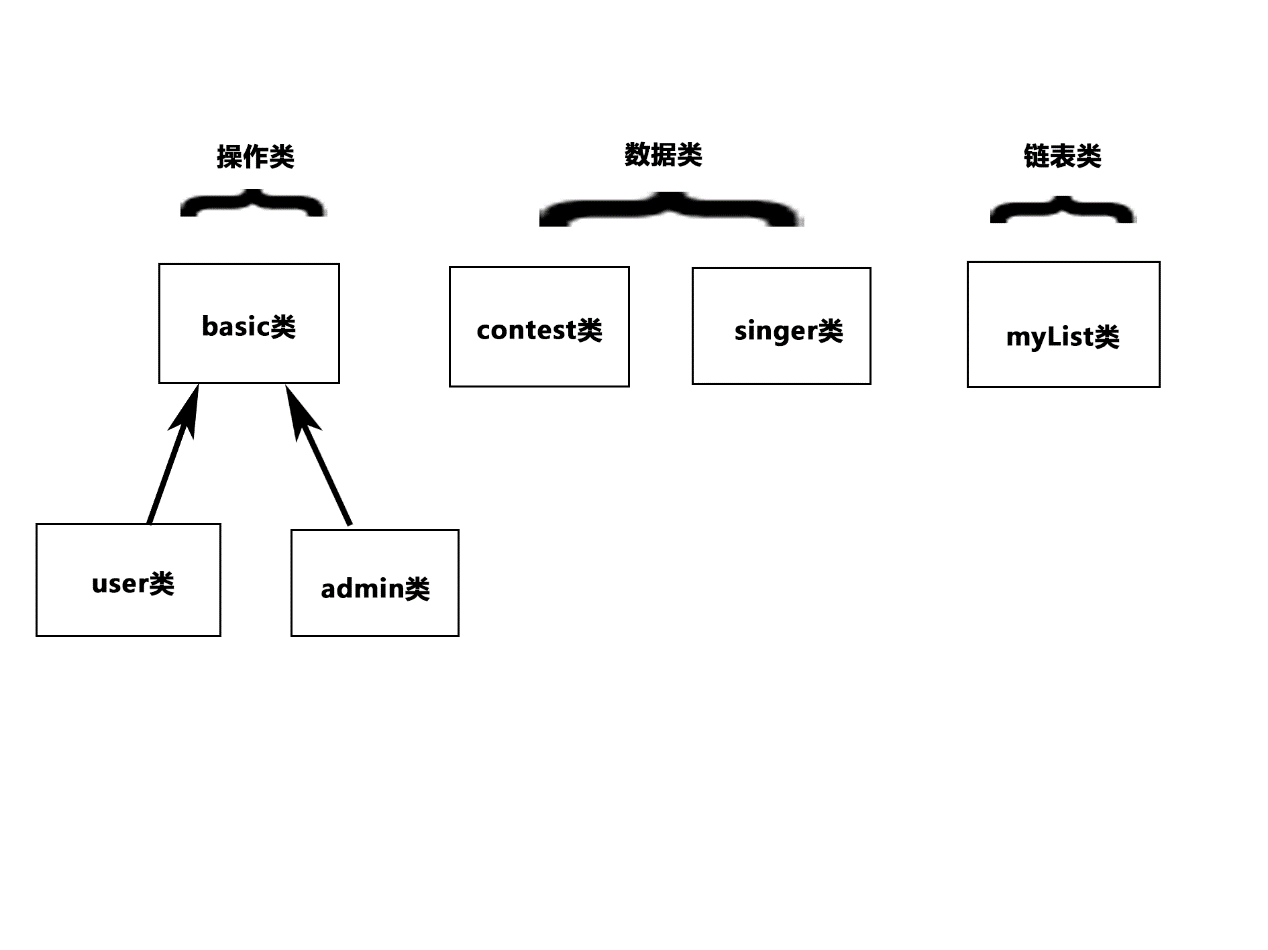


图2 系统主要类结构图

1. **系统的实现**

**SCS系统主要包含了contest类（比赛信息类），singer类（歌手信息类），basic类（用户基本操作类）以及在其基础上派生的user类（普通用户的操作类）和admin类（管理员操作类）。**

**具体声明如下：**

* **contest类：**

class contest {

public:

    contest\* prev;//用于链表中，指向前一场比赛

    contest\* next;//用于链表中，指向后一场比赛

    myList<singer> contestantList;//参赛者链表

    contest(string,int,int);

    void readContestantList(string);//从文件读入该场比赛的参赛人

    void printContestantList();//输出参赛者列表

    void  display();//输出比赛信息

    string getContestName();//获取比赛名称

    int getContestantCount();//获取该比赛的参赛人数

    int getFinished();//获取比赛状态信息

    void setContestName(string);//设置比赛名称

    void setContestantCount(int);//设置比赛的参赛人数

    void setFinished(int);//设置比赛状态

    friend ofstream& operator <<(ofstream&,contest&);//输出比赛信息的友元函数

private:

    string contestName;//比赛名称

    int contestantCount;//参赛者人数

    int isFinished;//设置比赛是否已经结束，0表示进行中，1表示已结束

};

* **singer类**

class singer {

public:

    singer\* prev;//用在链表中，指向其前一个元素

    singer\* next;//用在链表中，指向其后一个元素

singer();

    void setNumber(string);//设置编号

    void setName(string);//设置姓名

    void setScore(int, double);//设置评委i+1的打分(0-9)

    void countScores();//计算最高最低分，总得分，平均分

    void setVotes(int);//设置观众投票数

void setPromoted(int);

void display();//输出歌手信息

    string getNumber()const;//返回编号

    string getName()const;//返回姓名

    double getScore(int)const;//int 为1-10，则返回评委1-10打分，11返回最高分，12返回最低分，13返回总分，14返回平均分，15返回投票数，16返回综合得分

    int getVotes()const;//返回观众投票数

    int getPromoted()const;//返回是否晋级

    friend void checkPromoted(myList<singer>&,int);//检查选手在比赛中是否可以晋级

    friend ofstream& operator <<(ofstream&, const singer&);//友元，用于输出

    double& operator [](int);//友元，可以通过下标返回歌手的各项得分。

private:

    string number;//歌手编号

    string name;//歌手姓名

    double score[10];//十个评委打分

    double maxScore;//评委最高分

    double minScore;//评委最低分

    double totalScore;//评委总得分（“不”去掉最高最低分的）

    double averageScore;//平均得分（去掉最高最低后的）

    double votes;//获得的用户投票数

    double finalScore;//综合得分

    int isPromtoted;//设置该选手是否晋级,0表示未晋级，1表示已晋级

};

* **baisc类**

class basic {

public:

    basic();

    void showSingers(contest\*);//显示歌手信息，参数表示比赛场次

    void search(contest\*);//搜索功能,支持模糊搜索

};

* **user类**

class user :public basic {

public:

    user(string,int);

    ~user();

    void voteSinger(int, singer\*);//给歌手投票

private:

    string id;//用户的ID

    bool\* voted;//记录当前用户各场次的投票情况

};

* **admin类**

class admin :public basic {

public:

    void addContest(myList<contest>&, int&);//追加比赛信息

    void modifyContestData(myList<contest>&, int&);//修改比赛数据

    void deleteContest(myList<contest>&, int&);//删除比赛信息

    void addSingerData(contest\*);//追加歌手数据

    void modifySingerData(contest\*);//修改歌手数据

    void deleteSingerData(contest\*);//删除歌手数据

};

**（2）链表的使用**

对于经常需要增删的数据，数组的使用已经显然跟不上效率，对数组进行操作，时间复杂度以及程序复杂度陡增，需要采用一个兼顾效率和使用方便性的数据结构，而链表十分适合。

故本次作业采用**双向链表**。

具体结构声明如下：

template <typename T>

class myList {

public:

    myList();

    ~myList();

    int getSize();//返回链表中元素个数

    void addNode(T&);//在链表末尾添加元素

    void printList();//遍历链表并输出。

    void swapNodes(int ,int);//交换链表中两个元素位置

    void deleteNode(int);//链表中删除元素

    T\* getNode(int);//获得一个元素

private:

    T\* listHead;//链表头，其若不为NULL，则指向链表的第一个元素

    T\* listEnd;//链表尾

    int size;//链表大小

};

本题链表考虑到比赛信息和歌手信息都可以通过链表来存储，而比赛与歌手属于两个不同的类，故通过**C++模板**来实现双向链表。

将每一个contest（比赛）或者singer（歌手）作为链表的节点，并在contest和singer均分别定义了contest\* prev和contest\* next以及singer\*prev和singer\* next成员来链接。

以实例化为myList<contest> ML的链表为例。

ML中的listHead成员中存储的指针指向链表第一个元素，从而找到这个链表，找到链表中的第一个元素后，该元素的next成员指向下一个元素……由此，便可以找到链表中的所有元素。

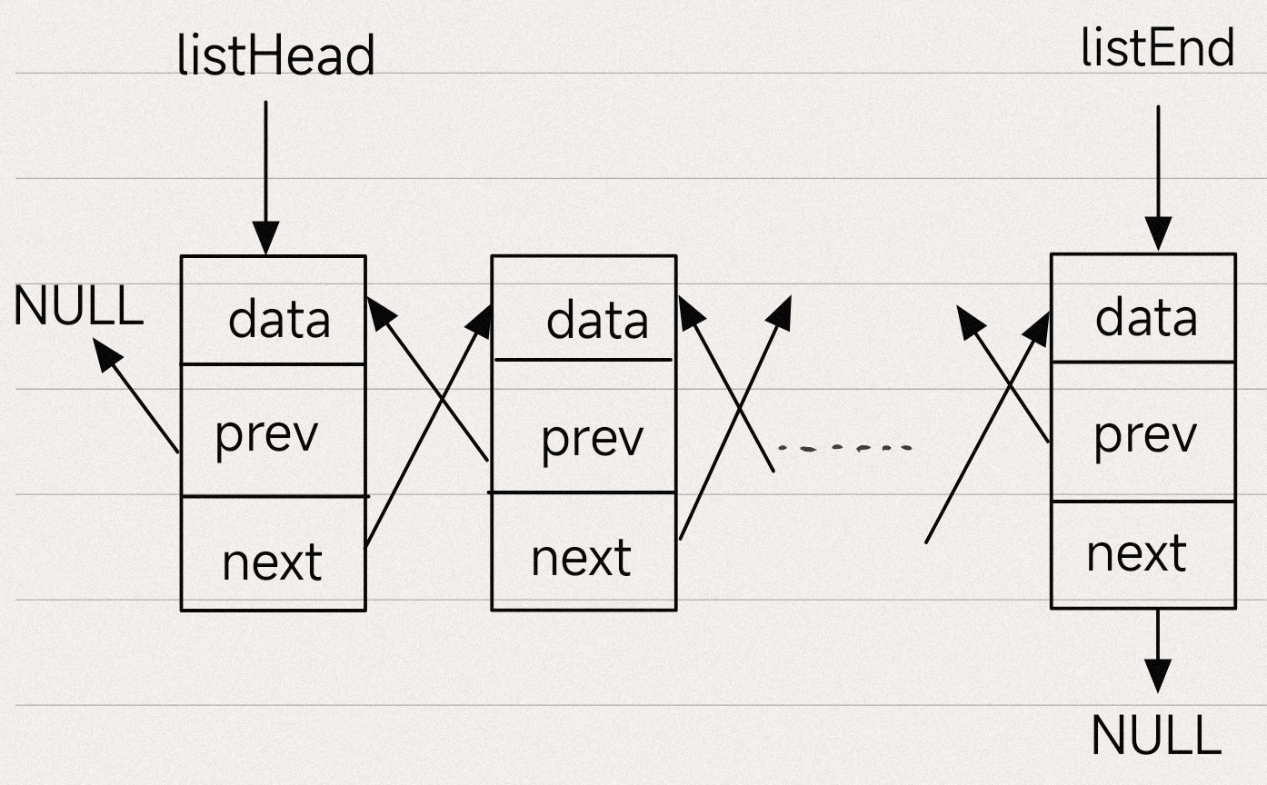


图3 链表结构示意图

用户的操作都要依赖链表的操作。

链表各函数功能简略讲解：

1. **addNode()**：增加结点，考虑到本题的情况，只设计了在链表末尾追加元素，故设置了数据成员 listEnd，用于指向链表的最后一个元素，所以插入元素时只需通过listEnd找到最后一个元素进行连接即可。
2. **deleteNode()**：删除结点函数，可以删除链表中的元素，由于采用的是双向链表，故十分容易找到欲删除的结点的前驱结点和后继结点，删除结点之后可以重新连接前后两个结点，使链表保持连续。
3. **swapNodes(int,int)：**交换两个结点，本以为只是简单的改变两个要交换的元素的前后元素的指向便可完成交换。后来发现如果**要交换的两个元素是相邻元素**的时候，会出现问题，思考半天不得一个普遍解法，无奈只好对相邻元素的交换进行了特判，导致该段代码略显冗长。
4. **getSize() , printList() , getNode(int)**:三个函数功能分别为获取链表元素个数，输出链表，获取链表某一特定结点。实现较为简单。

**（3）用户功能的实现**

basic类中：

普通用户和管理员均具备的功能：

1. **将歌手数据进行排序，显示排序后的列表。流程图见下：**

提示输入错误

开始

显示可选的排序方式

用户选择是否合法

提示输入错误

否

是

显示升序与降序的选择

用户选择是否合法

否

是

调用mySort函数对链表进行排序

输出排序后的链表

结束

流程图I 歌手排序查看

1. **搜索某一歌手**

如果一场比赛歌手人数过多的话，想要查看某一特定歌手的信息会变得十分不方便，因此，设计了搜索功能，支持通过歌手姓名模糊搜索歌手。将用户输入的关键字，与存储的歌手数据的姓名进行匹配，如果用户关键字是歌手姓名的子串的话，便将该歌手的信息输出。

user类中：

由basic类派生而来，具备basic类的全部功能，此外，额外有为歌手投票的功能。

1. **为某位歌手投票**

现今很多比赛为了让观众更具有参与感，都设立了**观众投票**功能。本系统也紧跟时代，设置了观众投票功能，并且观众的投票，会影响选手最后的综合得分。流程图见下方。

提示比赛已结束

无法投票

开始

用户选择比赛场次

用户选择是否合法

提示输入错误

否

是

比赛是否进行中？

否

是

显示选手列表，提示用户选择

为相应歌手投票数+1

结束

用户选择是否合法

提示输入错误

是

否

提示用户已经

投过票了

否

用户该场比赛是否尚未投票？

重新计算该歌手的综合得分

流程图II 为歌手投票

admin类中：

由basic类派生，具备basic类全部功能。

1. **追加比赛**

比赛是阶段性的，诸如唱歌比赛这类，是可以有选手的晋级与淘汰的。

管理员可以选择一场仍处于进行中的比赛，将其状态设置为已结束，并以该比赛为基础生成新的比赛，将晋级选手的信息保存到新的比赛中。

以下为流程图：

提示比赛已结束

无法进行新增

开始

管理员选择比赛场次

用户选择是否合法

提示输入错误

否

是

比赛是否进行中？

否

是

输入新增比赛的名字

复制晋级选手到新增的比赛

结束

提示输入错误

是

否

流程图III 管理员新增比赛

检查新增名字是否合法

新增的比赛添加到现有的比赛列表

1. **修改比赛信息**

管理员可以修改某场比赛的名称，也可以重新开放已结束的比赛和强制结束正在进行的比赛。

1. **删除比赛**

管理员可以删除一场比赛，借助链表中提供的deleteNode函数，实现删除。

1. **增加歌手信息**

管理员可以在某场比赛中追加歌手的信息。

1. **修改歌手信息**

管理员可以修改歌手的编号，姓名，评委打分，获得投票数等信息。系统提示管理员进行相应数据的修改的输入，同时会显示数据原先的值，方便管理员进行修改。

1. **删除歌手信息**

管理员可以删除歌手的信息。

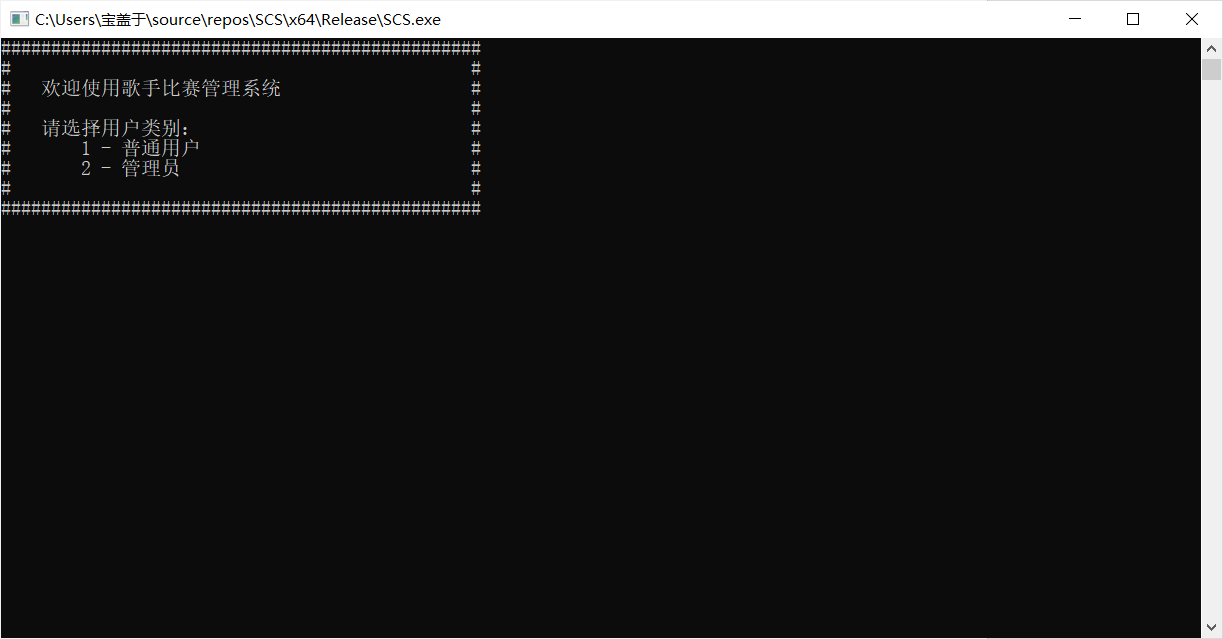
 **（4）操作菜单的设计**

图4 程序运行初始界面

程序运行时初始界面如图4所示。

并且运行中设置了简明的多级菜单，信息提示得当，比如在用户进行操作时会显示当前正在对哪一场比赛进行操作。适当时机使用system(“cls”);函数进行清屏，不再显示无关信息，方便用户阅读，也使得程序的输出不是那么混乱。

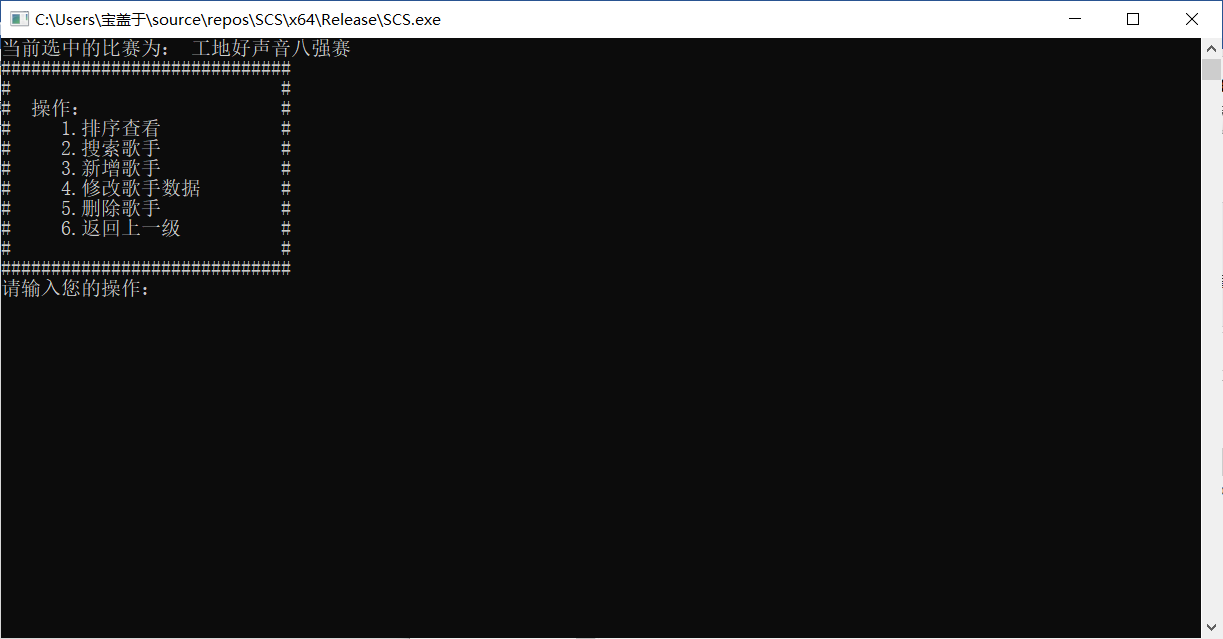


图6 使用截图（二）

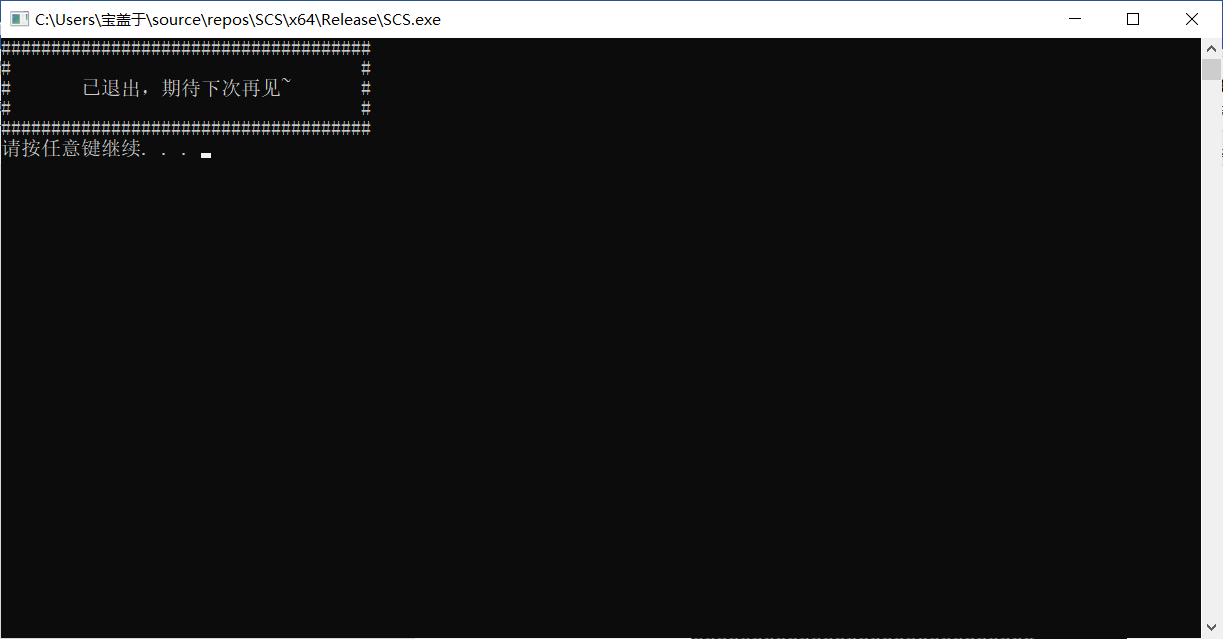


图7 退出系统



图5 使用截图（一）

至此，整个程序已经完成了

1. **实现过程中遇到的问题**
   * 1. **链表元素交换时，出现链表元素丢失的问题。**

在调试swapNodes函数时，发现有时候交换两个元素之后再输出正常，有时候交换后再输出会出现元素丢失的问题。

开始以为是交换后元素重新连接没有写好，后来发现只在交换两个相邻元素时出现，分析代码之后明白假如要交换相邻的p1和p2时，p1.next本来是指向p2的，而交换p1与p2之后，p2理应指向p1.next的对象，这就导致了p2.next最后指向了p2自身，导致链表输出出现问题。

所以在swapNode函数中加入特判，特殊处理交换相邻元素的情况。

特判代码（部分）：

    if (p1==p2Prev)     //特殊处理要交换两个相邻元素的情况

    {

        p1->prev = p2;

        p1->next = p2Next;

        if (p1Prev != NULL)

            p1Prev->next = p2;

        p2->prev = p1Prev;

        p2->next = p1;

        if (p2Next != NULL)

            p2Next->prev = p1;

    }

1. **实验总结**

本次实验作业从无到有共计花费3天时间（完成前期规划和代码编写）。自评已经全部实现题目的功能要求，且采用了双向链表来进行数据的管理，提升了 程序的效率。且多处设置了输入检测，具备一定的鲁棒性。信息反馈较为完善，用户体验较好。

当然缺点还是有的，最明显的，是经验的不足，对程序整体的架构规划不是特别好，前期工作没有做好，对程序维护可能有一定的影响；程序的输入检测虽然有，但不是特别强，如果用户不能好好输入，程序仍有可能崩溃；程序一些功能还可以更加完善，目前只是简单地实现，如所谓的模糊搜索只是简单的子串匹配。

经过这次实验，体会到以后还有很长的路要走；同时，这次实验将这学期学到的面向对象的编程进行了一次大练习，让我加深了对类的派生，重载，继承等操作的理解，获益颇丰。从高中起，我的信息技术老师就告诉我要多练不能只看，如果只是看书，那知识永远只会留在纸上。

1. **附录：源代码**