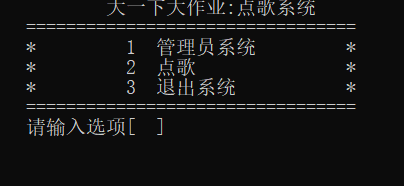


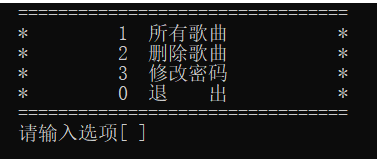
C语言程序设计综合训练报告

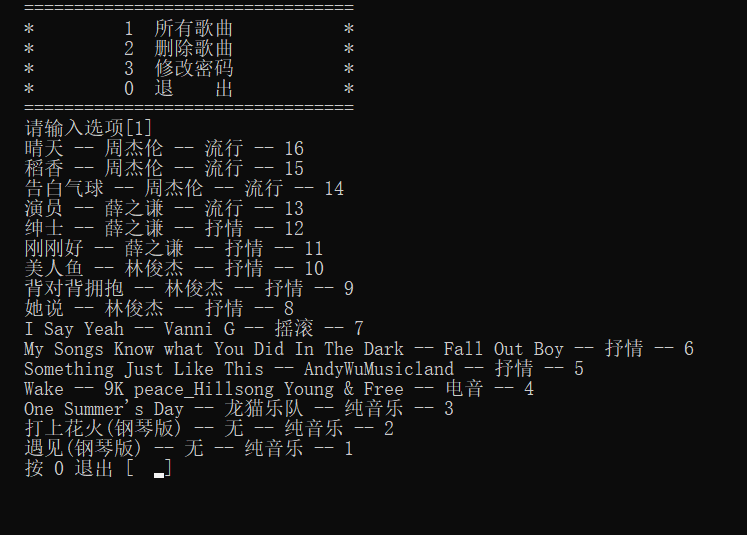
题目

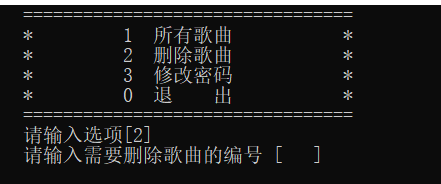
点歌系统

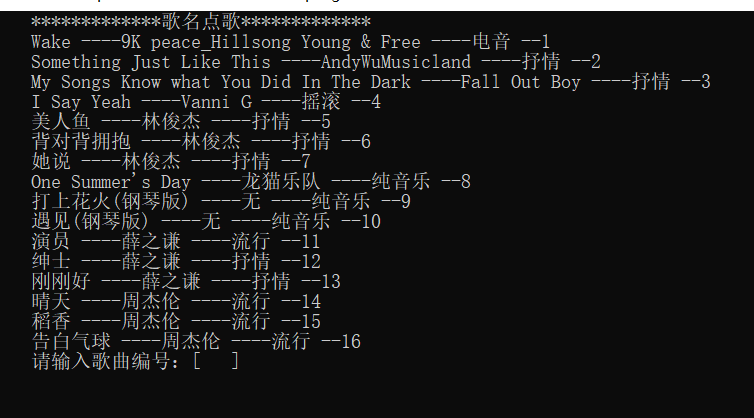
程序代表性截图

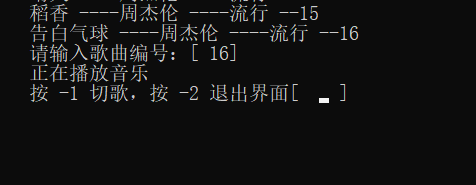












团队名称

疼校

完成日期

2022/6/16

基本信息：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 成员 | | | | |
| 序号 | 姓名 | 班级 | 学号 | 联系方式 |
|  | 杨宸 | 计UC211 | 202115030120 |  |
|  | 杨昊 | 计UC211 | 202115030121 |  |
|  | 杨钊瑞 | 计UC211 | 202115030122 |  |

工作分配：

|  |  |
| --- | --- |
| 成员 | 具体工作 |
| 杨宸 | 主要负责main.c, EXITSYSTEM.c, MainSystem.c 编写; |
| 杨昊 | 主要负责SongSys.c, Song.txt 编写; |
| 杨钊瑞 | 主要负责 AdminSys.c, admin.txt musiclib 编写; |

# 目录

[目录 5](#_Toc106876768)

[一、 设计任务 6](#_Toc106876769)

[1、 训练目标 6](#_Toc106876770)

[2、 系统功能 6](#_Toc106876771)

[二、 系统使用说明 6](#_Toc106876772)

[1、 文件组成 6](#_Toc106876773)

[2、 操作说明 7](#_Toc106876774)

[三、 程序设计与实现 8](#_Toc106876775)

[1、 程序结构（结构图） 8](#_Toc106876776)

[2、 核心数据结构（结构体定义及说明） 10](#_Toc106876777)

[23、 函数说明（每个函数） 11](#_Toc106876778)

[四、 设计难点与解决方法 16](#_Toc106876779)

[五、 程序效果与不足之处 16](#_Toc106876780)

[六、 设计心得与体会 17](#_Toc106876781)

# 设计任务

## 训练目标

1. **模拟KTV点歌系统：通过ｃ语言设计程序并应用数据结构实现点歌系统的基本功能。**
2. **用户可按歌名查找某首歌曲或按歌手名查找其所有歌曲，点歌后播放音乐。管理员可删除歌曲，全部歌曲的路径用一个单独的文件存储，包含歌名，歌手名，歌曲类型，相对路径。**
3. **用户体验良好，操作方便。**
4. **符合大作业的基本要求。**

## 系统功能

1. 用户功能：点歌（查找+播放）、查询所有歌曲。
2. .管理员功能：添加歌曲、删除歌曲、修改歌曲、修改密码、退出点歌系统。



# 系统使用说明

## 文件组成

* Main.c
* MainSystem.c
* SongSys.c
* EXITSYSTEM.c
* AdminSys.c
* Admin.txt
* Muscilib
* Song.txt;

## 操作说明

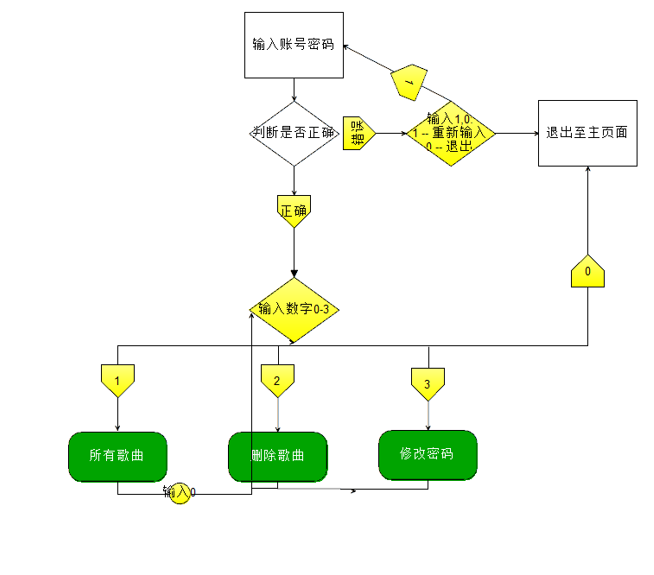
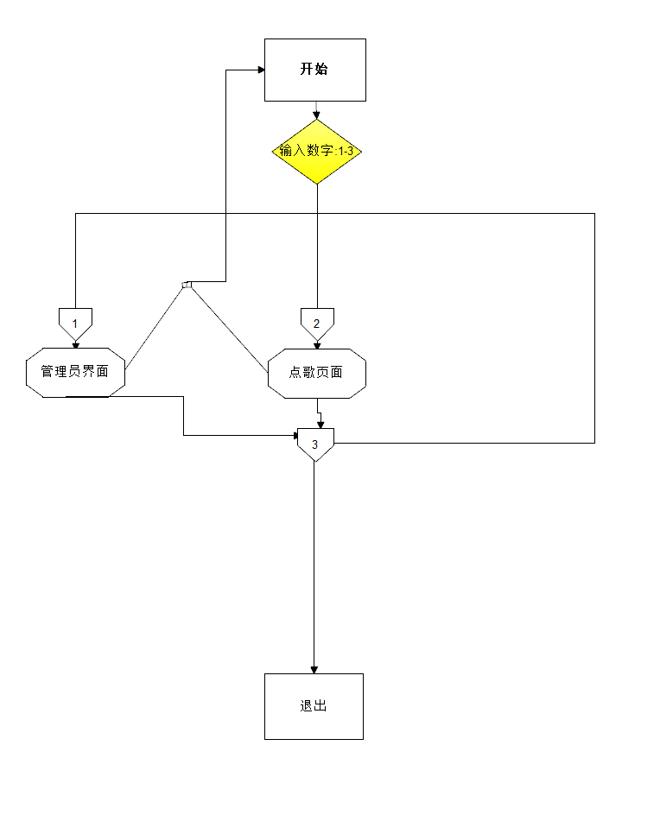
进入点歌系统主页面，通过输入数字（1 – 3），选择进入不同的页面，

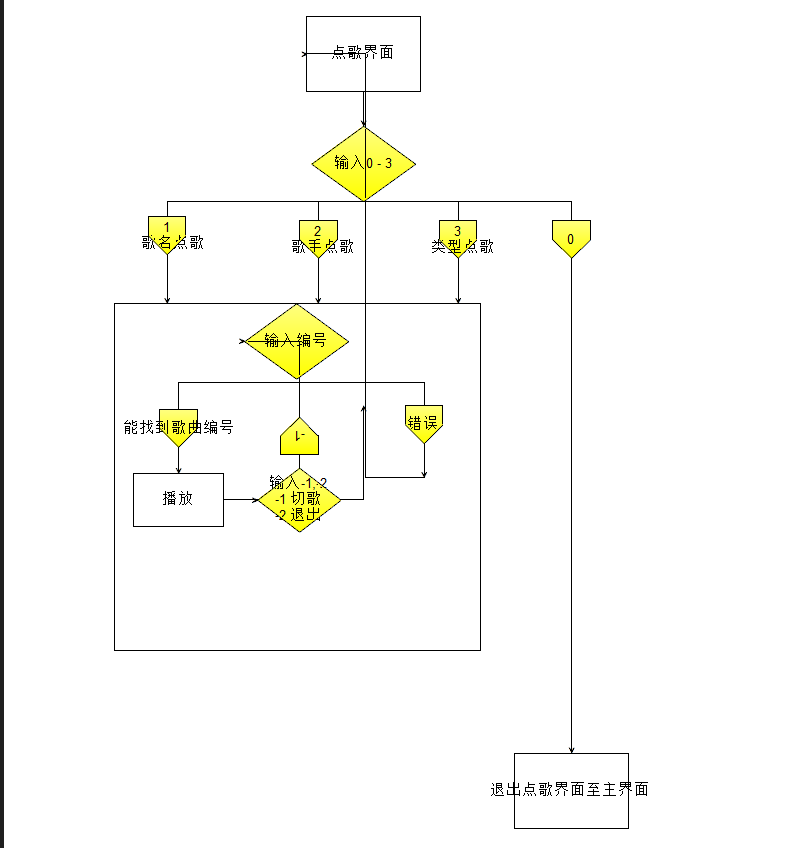
进入管理员界面，首先要输入账号密码，正确后方可进入。进入管理员界面后，通过输入功能对应的编号，选择进入不同的页面。

点歌功能等类似。

# 程序设计与实现

## 程序结构（结构图）





## 核心数据结构（结构体定义及说明）

1. // 管理员结构体；
2. typedef struct adm {
3. char AdminName[ADMINACCLENG]; // 管理员名字；
4. char AdminPass[ADMINPASSLENG]; // 管理员密码；
5. struct adm\* Next;
6. }Admin, \*AdminNode;
7. Admin FirstAdmin;
8. typedef struct song {
9. char Name[100]; // 歌曲名字
10. char SongN[100]; // 歌曲歌手名字
11. char Type[100]; // 歌曲类型
12. char address[100]; // 歌曲地址
13. struct song\* NextSong; // 歌曲指针 头插法 <----
14. int Num; // 普通用户界面的歌曲编号；
15. //struct song\* AdNextSong;
16. int SeNum; // 管理员界面的歌曲编号
17. } Song, \*SongPointer;
18. // head -->  SongList;
19. SongPointer SShead;

## 函数说明（每个函数）

void MainMenu()  
函数功能：显示主界面;  
参数：无  
返回：无返回值  
算法描述：

先显示点歌系统主界面内容。 采用getch()来获取键盘输入，当输入为’1’ – ‘33’时，getch()将获得的字符赋值给CHOICE。

通过采用每0.1秒循环打印’\*’，来提高用户的体验感，减少用户焦虑。最后结束。

函数原型：

void AdminLogin();  
函数功能：

管理员登录  
参数：无  
返回：无  
算法描述：

用getch()来获取账号输入字符，之后再打印到显示器；获取密码字符，打印’\*’;

之后判断与管理员账号密码是否般配。若般配，则登入成功，反之，登入失败，选择重新登入或退出。

函数原型：

void AdminMainSys()  
函数功能：

显示管理员界面；  
参数：无  
返回：无  
算法描述：

打印管理员界面功能。输入字符，选择进入哪一个功能界面。

函数原型：

void ChangePass()  
函数功能：

改管理员密码  
参数：无  
返回：无  
算法描述：

先输入旧密码，再输入新密码。若旧密码正确，就修改密码，否则修改错误。

成功修改，通过文件操作，将新密码复制到管理员结构体密码，并返回到管理员登入界面。

函数原型：

void AllAdSong()  
函数功能：

打印歌。  
参数：无  
返回：无  
算法描述：

通过链表遍历。

按 歌名 歌手名 歌曲类型 歌曲内部管理编号打印到显示器。

函数原型：

void DeleteSong()  
函数功能：

删除歌；  
参数：无  
返回：无  
算法描述：

输入数字，并且忽略’\n’，内部调用ListDel(int a);

函数原型：

void ListDel(int number)  
函数功能：

删除歌曲；  
参数：number  
返回：无  
算法描述：

采用前后指针遍历有头链表，遍历链表寻找内部数字number，之后删除number所对应的歌曲；

函数原型：

void SongMainSys()  
函数功能：

打印点歌界面;  
参数：无  
返回：无  
算法描述：

打印点歌需要的功能和编号。

采用getch()函数获取字符，若字符再’1’ – ‘3’ 之间则将字符转为数字赋给SONGCHOICE ,并且打印字符。之后用switch..case 来寻找SONGCHOICE所对应的部分，并执行对应类型的歌曲排序及打印的函数，之后输入STOP ，调用PlayMusic(int number)函数播放音乐，并可以输入-2 / -1 来进行切歌 / 停止 音乐。

函数原型：

int PlayMusic(int number,int StorSt  
函数功能：

播放音乐;  
参数：number, StorSt  
返回：0 – 错误 1 – 成功播放;  
算法描述：

结构体指针遍历，寻找 要播放的音乐的结构体地址；用字符串来复制，剪切成windows程序设计音乐播放API的形式, 并进行播放。

函数原型：

void ImportMusic()  
函数功能：

导入音乐库；  
参数：无  
返回：无  
算法描述：

进行文件操作，打开”Song.txt” ,”r”; 将文件内容赋值到链表。

函数原型：

void PrintResult(int singchoice)  
函数功能：

打印歌曲排序结果；  
参数：singchoice  
返回：无  
算法描述：

根据singchoice 判定switch..case ，然后用遍历链表，及那个其打印下来。

函数原型：

void SortSong()  
函数功能：

对歌曲进行排序；  
参数：无  
返回：无  
算法描述：

先是头结点前移，用OnePointTwo()函数进行排序；

之后再用AddHead()增加头结点，用PrintResult()打印排序结果；

函数原型：

void SortSnger()  
函数功能：

对歌手进行排序；  
参数：无  
返回：无  
算法描述：

先是头结点前移，用OnePointTwo()函数进行排序；

之后再用AddHead()增加头结点，用PrintResult()打印排序结果；

函数原型：

void SortType()  
函数功能：

对类型进行排序；  
参数：无  
返回：无  
算法描述：

先是头结点前移，用OnePointTwo()函数进行排序；

之后再用AddHead()增加头结点，用PrintResult()打印排序结果；

函数原型：

SongPointer OnePointTwo(SongPointer sshead, int singchoice)   
函数功能：

链表一分为二  
参数：sshead, singchoice  
返回：SongPointer  
算法描述：

归并排序，三个指针，每次Left Mid向下一个遍历，Right遍历两个，之后Mid指向NULL，然后调用ListMerge()函数

函数原型：

SongPointer ListMerge(SongPointer head, SongPointer mid, int singchoice)  
函数功能：

两个链表排序;  
参数：head， singchoice  
返回：

SongPointer  
算法描述：

新建一个头结点，通过singchoice选择对什么排序，之后两个链表作比较，插入到新的链表中。最后返回新链表的头结点。

函数原型：

SongPointer AddHead(SongPointer sshead)  
函数功能：

增加头结点；  
参数：sshead  
返回：SongPointer;  
算法描述：

创建一个空链表，空链表指向NULL，之后用头插法，将空链表指向下一个，返回SongPointer。

# 设计难点与解决方法

1. Vscode没办法插入”winmm.lib”。解决方法： dev C++：  
           工具 --> 编译选项 --> 在编译器命令行加入：-lwinmm  
       vscode:  
           使用 gcc自定义链接库，要在task.json中添加编译参数“-lwinmm”
2. 自己建的头文件---- .h，要是在超过两个文件里导入会报错。解决方法：用#ifdef #define #endif
3. SongSys.c中切歌 会不再[ ] 这个范围内，这是由于缓冲区有字符，对输入输出缓冲区不了解。解决方法：%\*c 忽略’\n’ 和getchar(), 就是输入后，为了保险起见，刷新一下缓冲区。
4. Song.txt文件输入到链表中，有’\n’ 使输入错误。解决方法：%[^\n]%\*c 用通配符，忽略’\n’， 并把’\n’忽略掉。另一个方法是做完后想到的 fwrite();
5. 有无头结点的链表的排序，归并排序，有头结点链表错误。解决方法：有头结点去头结点，在排序，排序后再加一个头结点。

# 程序效果与不足之处

程序有的输入输出采用相同输入输出函数，但是输出方式不一样。

Song.txt输入到链表有点复杂。

多文件编译(t.h, t.c)不规范，本来想用CMAKE，环境没配好，就采用了.c -- .c方式来多文件编译。

由于刚开始写的时候，分工不明确，导致后期分工时，每个文件的风格不太相同，导致看代码显得复杂。

关于extern int x ，由于没采用(t.h – t.c)这种多文件编译方式，好像这个没用上；

# 设计心得与体会

学到了好多课上没有讲到的库和函数，像<conic.h>中的getch()，winmm.lib <windows.h>。

…