哈尔滨工业大学（深圳）
HARBIN INSTITUTE OF TECHNOLOGY

实验报告

开课学期：2025 秋季

课程名称：计算思维与实践

实验名称：实验 11 音乐播放器 I

实验性质：设计型

实验学时：2 地点：T2507

学生学号：2025311313

学生姓名：陈靖韡

评阅教师：

报告成绩：

实验与创新实践教育中心制

2025 年 12 月

一、实验目的

1. 掌握单向链表的基本概念及常见操作；

2. 理解链表在动态数据管理中的应用场景；

3. 掌握文件读写操作，实现数据的持久化存储；

4. 提升程序设计与综合调试能力。

二、实验内容

设计并实现一个简单的音乐播放器歌单管理系统，使用单向链表存储歌曲信息，实现以下功能：

1. 从文件中读取歌曲信息，并建立歌单链表；
2. 支持用户添加新的歌曲信息；
3. 支持用户删除指定歌曲；
4. 支持按歌曲名查找并“播放”歌曲；
5. 显示当前歌单列表；
6. 将更新后的歌单信息写回文件。

三、系统设计（40 分）

3.1 数据结构设计

请说明歌曲信息结点所包含的数据成员，例如歌曲名、歌手、歌曲时长，以及指向下一个结点的指针，并给出结构体定义。

链表组织形式说明，所采用的链表类型（单/双向链表）、是否设置头结点，以及链表中各结点之间的逻辑关系。

答：歌曲信息节点的成员变量有：

1. id，歌曲的序号；

2. title，歌曲的名称；

3. artist，歌曲的作者；

4. filepath，歌曲的文件存储路径；

5. next，存储下一个节点的地址。

结构体定义如下：

```
// 歌曲节点结构体
typedef struct Song {
int id;
char title[100];
char artist[50];
char filepath[300];
struct Song* next;
} Song;
```

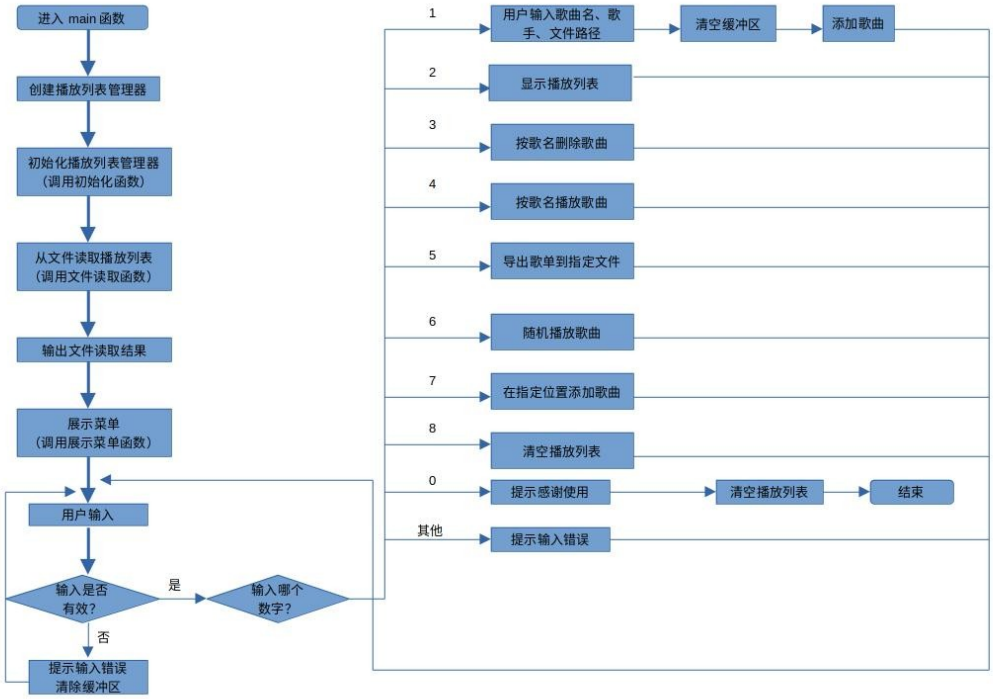
本实验还定义了播放列表管理器结构，用于管理歌曲链表的第一个节点、尾节点、当前节点和歌曲个数（即节点数量）。

```
// 播放列表管理器结构
typedef struct PlaylistManager{
Song* head; // 头节点
Song* tail; // 尾节点
Song* current;
int song_count;
} PlaylistManager;
```

本实验采用不带头节点的单链表，除了第一个节点和最后一个节点，所有的节点都有唯一的前驱和唯一的后继，由上一首歌曲可以唯一确定地找到下一首歌曲，从而获取到整个播放列表。

3.2 程序总体结构设计

说明主函数 main() 的功能，各子函数之间的调用关系，以及程序的整体运行流程。请绘制程序流程图或模块调用关系图。



四、函数设计（40 分）

4.1 函数接口定义

列出程序中主要函数的函数名、参数说明、返回值及其功能描述。

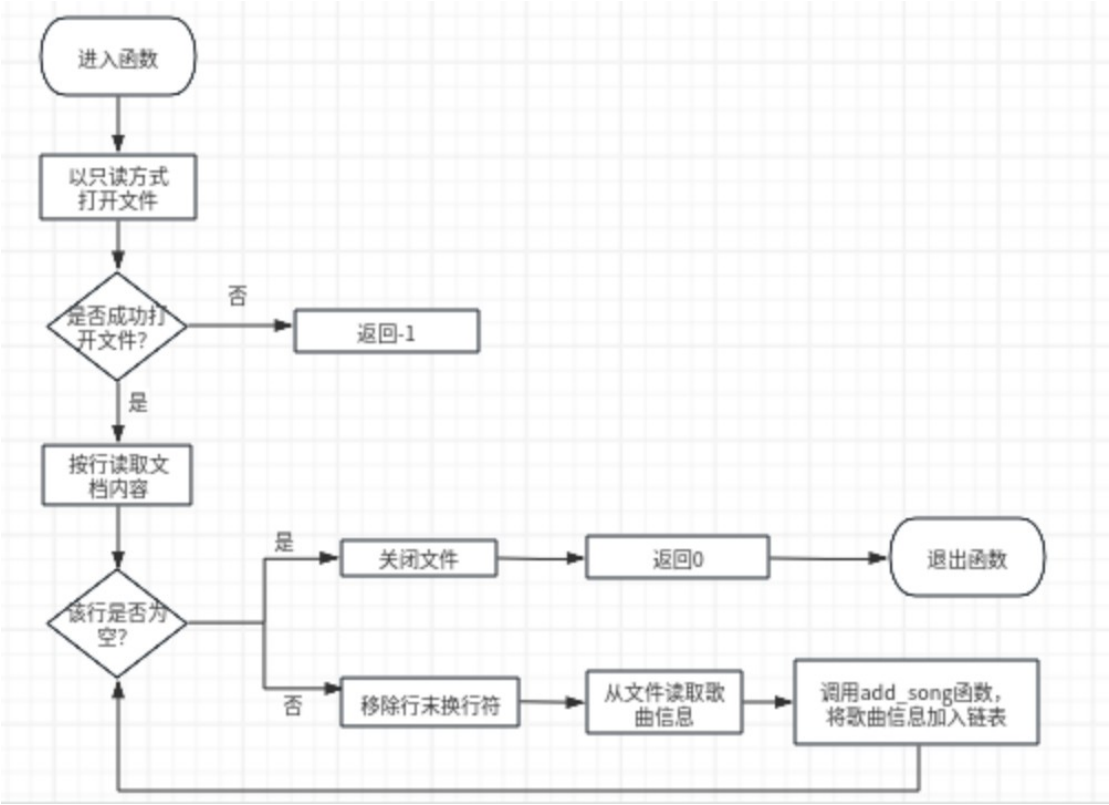
函数名	参数说明	返回值	功能说明
load_songs_from_file	播放列表管理器 manager, 文件名 filename	成功返回 0,失败返回-1	从 filename 文件中读取已经保存的播放列表，并将其加入链表
add_song	播放列表管理器 manager，歌名 title，歌手 artist，文件路径 filepath	无	通过用户手动输入，在链表末尾添加歌曲
delete_songs_by_title	播放列表管理器 manager，歌名 title	成功返回 0,失败返回-1	根据用户输入的歌名删除歌曲

函数名	参数说明	返回值	功能说明
play_song_by_title	播放列表管理器 manager，歌名 title	成功返回 0,失败返回-1	根据用户输入的歌名播放歌曲

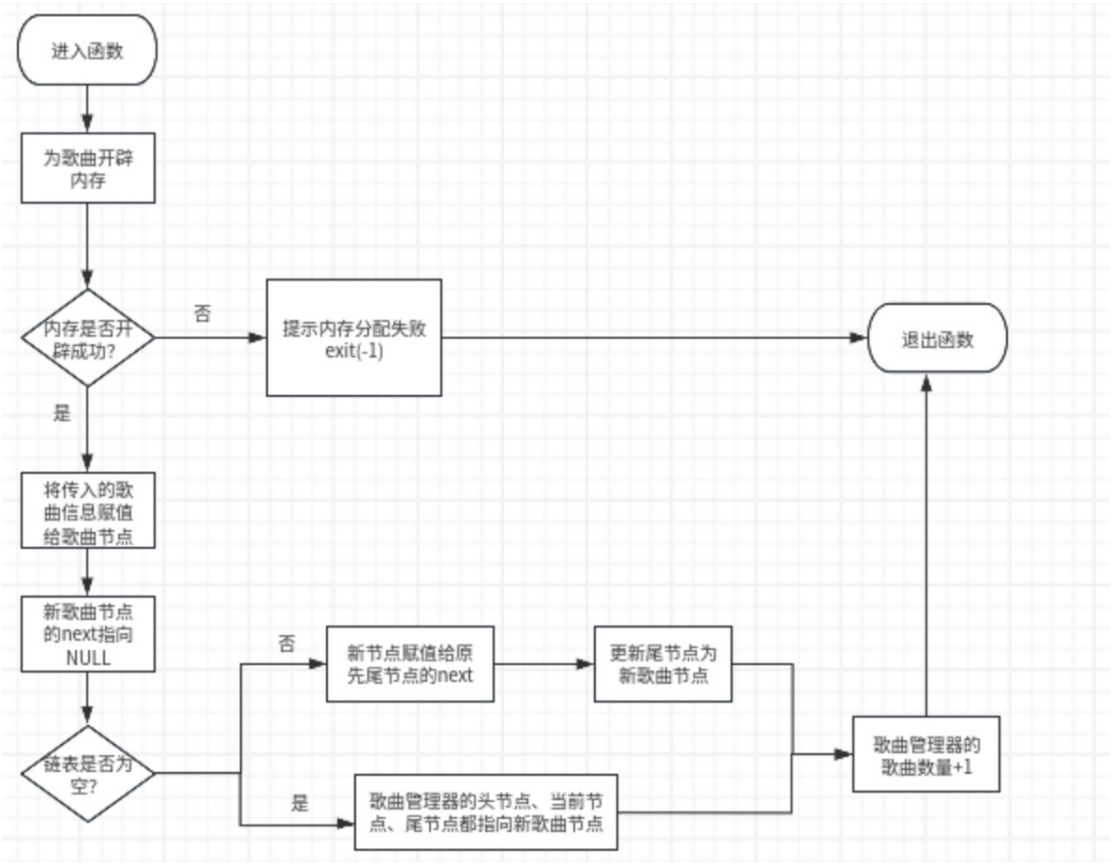
4.2 主要函数功能说明

分别说明以下函数的设计思路和实现逻辑：

1. 从文件读取歌曲信息并建立链表的函数；



2. 添加歌曲信息的函数；



3. 删除指定歌曲的函数；

I.歌曲管理器的当前节点 current 指向头节点；

II.创建 prior 节点，保存 current 的前驱；

III.进入循环：当前节点不指向空 且 当前歌曲名称与用户输入名称不同，则让 prior 指向 current，再让 current 指向其后继；

IV.循环结束后,

- 如果 current 指向 NULL，提示未找到歌曲并返回-1；
- 如果找到歌曲且删除的是头节点，需要更新头节点后再提示“已删除歌曲”并释放 current 的内存，并将歌曲管理器统计的歌曲数量-1，最后返回 0;
- 如果找到歌曲且删除的不是头节点，直接让 prior 指向 current 的 next，提

示“已删除歌曲”，再释放 current 的内存，并将歌曲管理器统计的歌曲数量-1 即可,最后返回 0。

4. 按歌曲名查找并播放歌曲的函数；

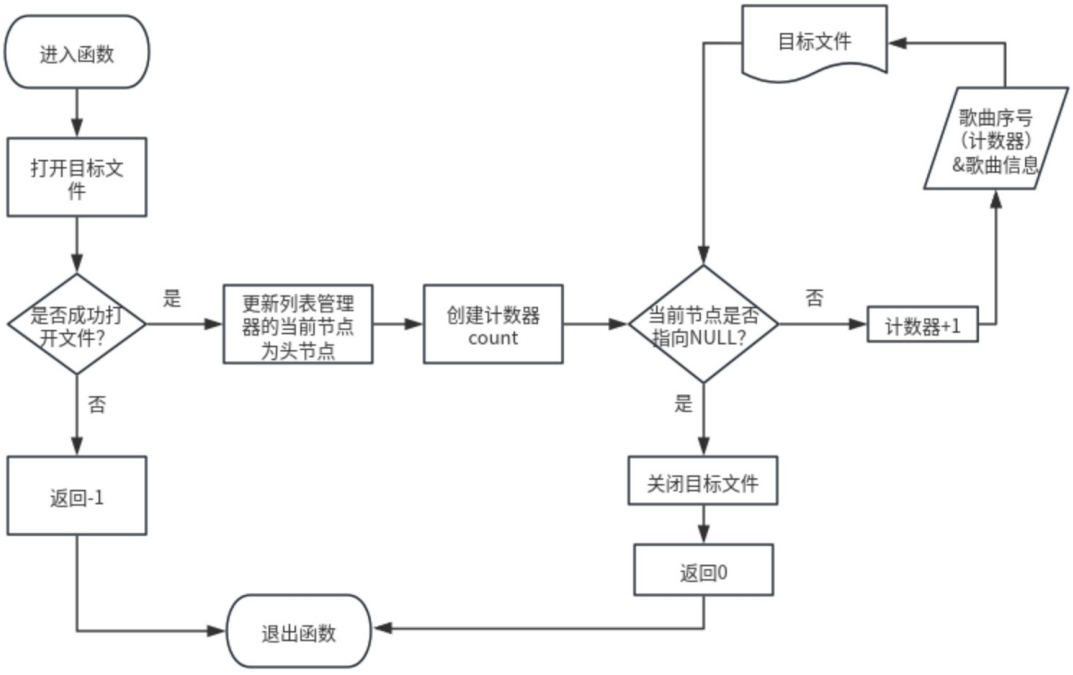
- 从头遍历各个节点，直到找到目标歌曲或者遍历到 NULL。如果遍历到 NULL，提示未找到歌曲并返回-1; 如果找到目标歌曲，则调用 play_audio( )函数播放目标歌曲，并提示成功播放即可，最后返回 0。

5. 显示歌单列表的函数；

I 输出播放列表有多少首歌曲；

II 从头到尾遍历各个节点，每次都输出该节点对应的歌曲信息（歌名，歌手，文件路径）。

6. 将链表数据写回文件的函数。



（每个函数需说明：实现思路 + 关键算法流程）

五、实验结果与测试（20 分）

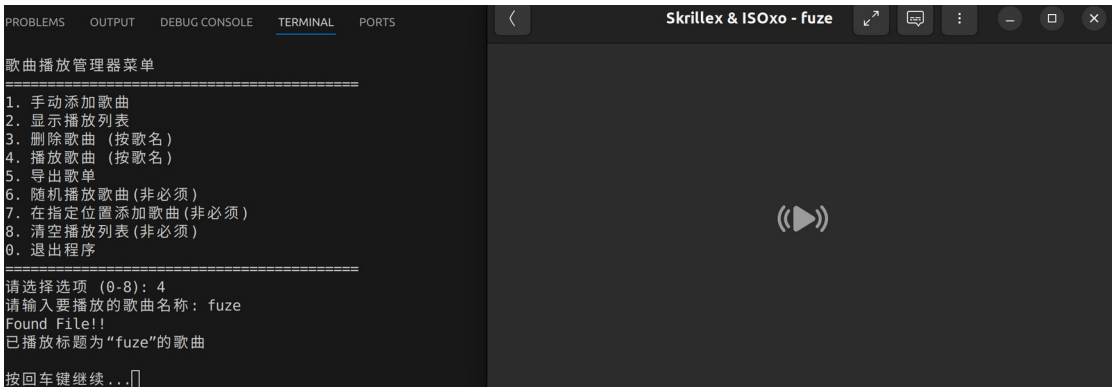
5.1 测试用例设计

请设计并说明测试用例，包括正常输入、边界情况（如空歌单、仅一个结点）以及异常情况测试。

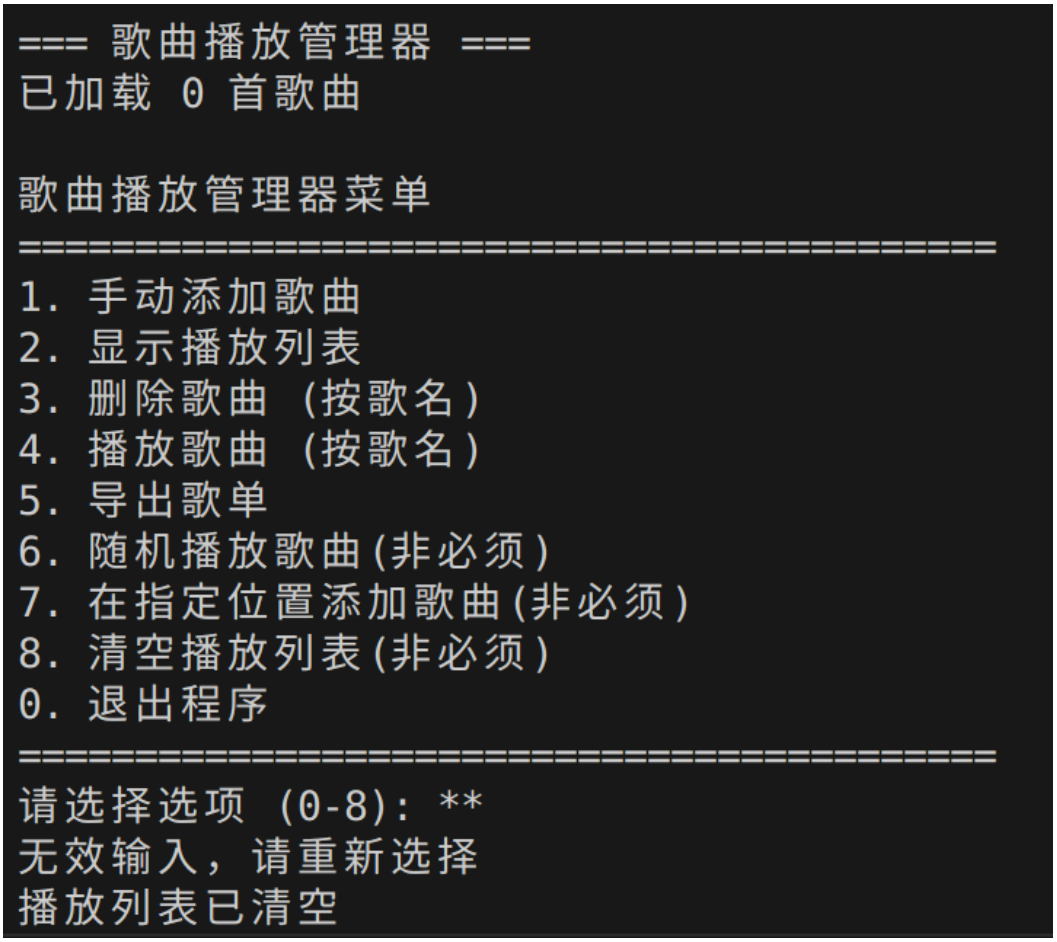
5.2 测试结果展示

展示程序运行结果，可通过截图或文字形式说明各功能是否正确实现。

正常输入：



异常输入：

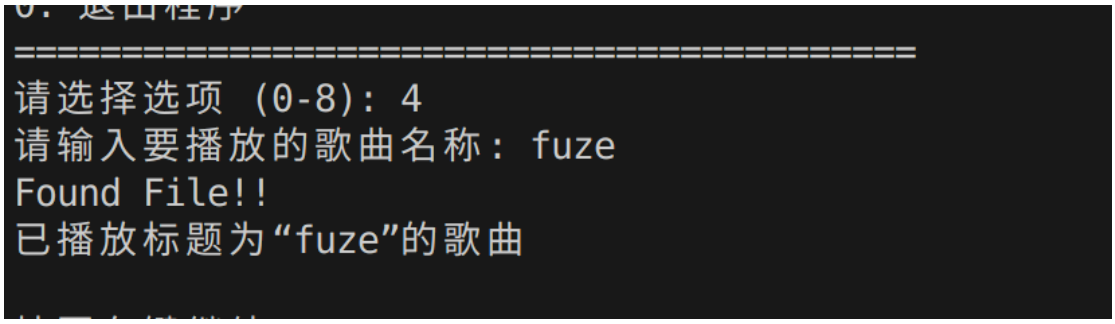


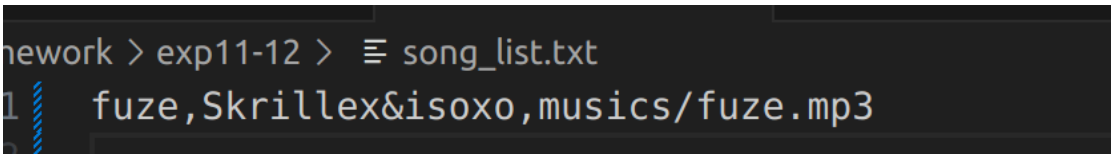
边界情况：将 song_list.txt 清空后执行操作

```
=== 歌曲播放管理器 ===
已加载 0 首歌曲

歌曲播放管理器菜单
============================================
1. 手动添加歌曲
2. 显示播放列表
3. 删除歌曲 (按歌名)
4. 播放歌曲 (按歌名)
5. 导出歌单
6. 随机播放歌曲(非必须)
7. 在指定位置添加歌曲(非必须)
8. 清空播放列表(非必须)
0. 退出程序
============================================
请选择选项 (0-8): 2
播放列表 (0首歌曲)
============================================

按回车键继续...
```

```
--------------------------------------------
请选择选项 (0-8): 1
请输入歌曲名: fuze
请输入歌手: Skrillex&isoxo
请输入文件路径: musics/fuze.mp3
```

```
--------------------------------------------
请选择选项 (0-8): 2
播放列表 (1首歌曲)
========================================
1. fuze - Skrillex&isoxo        (musics/fuze.mp3)

按回车键继续...4
```





六、问题与解决方法

说明实验过程中遇到的问题、原因分析以及对应的解决方法。

答：

问题：在编写 load_songs_from_file 函数时，我发现无法正确读取目标文件中的歌曲列表，每次运行程序都显示“已读取 0 首歌曲”，播放列表也没有存入歌曲。

原因分析：经过断点 debug 测试，我发现程序在使用 fscanf 函数读取文件时，无法正确从文件按照目标格式读取歌曲信息并存入变量。

```
char title[100], artist[50], filepath[300];
while (fscanf(fp, "%s,%s,%s", titlle, artist, filepath) == 3) …..
```

在 debug 中我发现文件中一行的歌曲信息全部存入了 title 这个数组，其他数组都为空，导致这个循环条件始终为假，进而导致后面的代码都没有执行，没有成功读取到歌曲信息，所以 manager→count 始终为 0。

解决方法：改用 fgets 函数按行读取+sscanf 函数，两个函数配合使用，进行

更精确地按格式读取。

七、课程总结与建议

总结：对数据结构与算法有了更深的理解，初步接触到了计算机应该学习的内容。