

FEEL THE MEANING OF THE TRIP

青 / 春 / 不 / 老 / 梦 / 想 / 永 / 在



DREAM

MY DREAM WILL NEVER STOP

计算思维与实践

实验11&12 链表与文件

TAKE YOU ON A TRIP
GO!



哈爾濱工業大學(深圳)
HARBIN INSTITUTE OF TECHNOLOGY, SHENZHEN

探索 从未停止

目录

CONTENT

01

实验目的

03

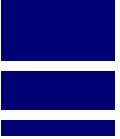
实验报告

02

链表与文件

04

期末检查



实验目的

- 掌握单向链表/双向链表的实现及查找、插入、删除操作
- 掌握文件存取数据操作



实验内容

实验11 音乐播放器I

设计一个音乐播放器的歌单，使用**单向链表**来存储和管理歌曲。实现以下功能：

- 从文件读取歌曲信息，并建立歌单链表；
- 用户**添加或删除**歌曲信息；
- 用户通过歌曲名**查找**并播放歌曲；
- 显示**歌单列表
- 将更新后的歌单信息写回到**文件**。

【输入描述】

程序启动时，从名为**song_list.txt**的文件中读取歌曲信息，创建链表。文件中的每一行包含一首歌曲和对应的信息（例如：“For_Elise,Beethoven,for_elise.mp3”）。

用户选择添加歌曲功能，接下来从命令行读入歌曲信息，**插入**歌单**尾部**。

用户选择删除歌曲功能，接下来从命令行读入歌曲名，从歌单中**删除**歌曲。

用户选择播放歌曲功能，接下来从命令行读入歌曲名，**播放**该歌曲。

用户选择导出歌单功能，接下来导出歌单到**文件**中。

【输出描述】

并把插入、删除操作结束后的歌单信息写回文件**song_list_result.txt**。

实验内容

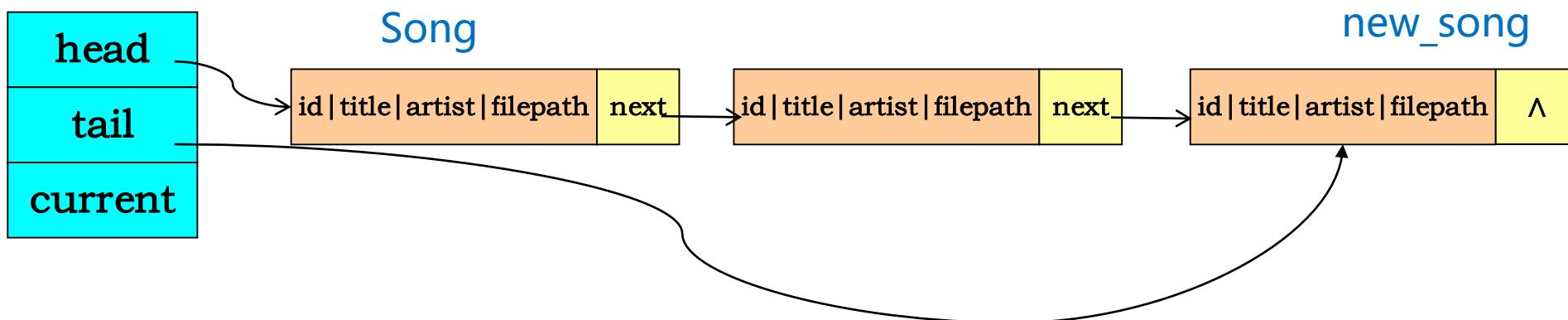
// 定义歌曲节点结构（单向链表）

```
typedef struct Song{  
    int id;  
    char title[100];  
    char artist[50];  
    char filepath[300];  
    struct Song* next;  
}Song;
```

// 定义歌单状态结构

```
typedef struct PlaylistManager{  
    Song* head;  
    Song* tail;  
    Song* current;  
} PlaylistManager;
```

PlaylistManager





实验内容

函数原型：

```
void add_song(PlaylistManager* manager, const char* title, const char* artist, char* filepath);
void display_playlist(PlaylistManager* manager);
int delete_song(PlaylistManager* manager, const char* title);
int play_song(PlaylistManager* manager, const char* title);
void export_playlist(PlaylistManager* manager, char* filepath);
void play_song_random(PlaylistManager* manager);
int add_song_by_position(PlaylistManager* manager, int
position, const char* title, const char* artist, char* filepath);
void clear_playlist(PlaylistManager* manager);
```

// 歌曲播放操作 (Windows)

```
void play_audio_windows(const char* filename) {
    char command[256];
    snprintf(command, "start \"\" \"%s\"", filename);
    ret = system(command);
    if (ret != 0) {
        printf("播放失败，请检查文件是否存在或格式是否支持。\\n");
    }
}
```

实验内容

```
链表音乐播放器管理器
=====
1. 人工添加歌曲
2. 显示播放列表
3. 删除歌曲 (按标题)
4. 播放歌曲 (按标题)
5. 导出歌单
6. 随机播放歌曲(非必做)
7. 在指定位置添加歌曲(非必做)
8. 清空播放列表(非必做)
0. 退出程序
=====
请选择操作 (0-8):
```

1

```
无法打开文件: song_list.txt
== 链表音乐播放器管理器 ==
已加载 0 首示例歌曲

链表音乐播放器管理器
=====
1. 人工添加歌曲
2. 显示播放列表
3. 删除歌曲 (按标题)
4. 播放歌曲 (按标题)
5. 导出歌单
6. 随机播放歌曲(非必做)
7. 在指定位置添加歌曲(非必做)
8. 清空播放列表(非必做)
0. 退出程序
=====
请选择操作 (0-8): 1
请输入歌曲标题: music1
请输入作者: author1
请输入歌曲路径: music1.mp3
已添加歌曲: author1 - music1
```

2

```
正在从文件加载歌曲...
已添加歌曲: Beethoven - ForElise
已添加歌曲: Pachelbel - Canon
从文件成功加载 2 首歌曲
== 链表音乐播放器管理器 ==
已加载 2 首示例歌曲

链表音乐播放器管理器
=====
1. 人工添加歌曲
2. 显示播放列表
3. 删除歌曲 (按标题)
4. 播放歌曲 (按标题)
5. 导出歌单
6. 随机播放歌曲(非必做)
7. 在指定位置添加歌曲(非必做)
8. 清空播放列表(非必做)
0. 退出程序
=====
请选择操作 (0-8): 2
* 播放列表 (2 首歌曲)
=====
1. ForElise - Beethoven (musics/ForElise.mp3)
2. Canon - Pachelbel (musics/Canon.mp3)
```

3

```
链表音乐播放器管理器
=====
1. 人工添加歌曲
2. 显示播放列表
3. 删除歌曲 (按标题)
4. 播放歌曲 (按标题)
5. 导出歌单
6. 随机播放歌曲(非必做)
7. 在指定位置添加歌曲(非必做)
8. 清空播放列表(非必做)
0. 退出程序
=====
请选择操作 (0-8): 3
请输入待删除歌曲名
请输入要删除的歌曲标题: music1
已删除歌曲: author1 - music1 (path: music1.mp3)
共删除 1 首标题为 "music1" 的歌曲
```

实验内容

链表音乐播放器管理器

- 1. 人工添加歌曲
- 2. 显示播放列表
- 3. 删 除歌曲 (按标题)
- 4. 播放歌曲 (按标题)
- 5. 导出歌单
- 6. 随机播放歌曲(非必做)
- 7. 在指定位置添加歌曲(非必做)
- 8. 清空播放列表(非必做)
- 0. 退出程序

请选择操作 (0-8): 4

请输入要播放的歌曲标题: Canon

Founded File! 'wmplayer' 不是内部或外部命令，也不是可运行的程序
或批处理文件。

已播放标题为 "Canon" 的歌曲

4

1. 人工添加歌曲

2. 显示播放列表

3. 删 除歌曲 (按标题)

4. 播放歌曲 (按标题)

5. 导出歌单

6. 随机播放歌曲(非必做)

7. 在指定位置添加歌曲(非必做)

8. 清空播放列表(非必做)

0. 退出程序

请选择操作 (0-8): 2

? 播放列表 (3 首歌曲)

- 1. ForElise - Beethoven (musics/ForElise.mp3)
- 2. songNew - authorNew (songNew.mp3)
- 3. Canon - Pachelbel (musics/Canon.mp3)

6

- 1. 人工添加歌曲
- 2. 显示播放列表
- 3. 删 除歌曲 (按标题)
- 4. 播放歌曲 (按标题)
- 5. 导出歌单
- 6. 随机播放歌曲(非必做)
- 7. 在指定位置添加歌曲(非必做)
- 8. 清空播放列表(非必做)
- 0. 退出程序

请选择操作 (0-8): 7

请输入歌曲标题: songNew

请输入作者: authorNew

请输入歌曲路径: songNew.mp3

请输入歌曲插入位置: 1

5

1. 人工添加歌曲

2. 显示播放列表

3. 删 除歌曲 (按标题)

4. 播放歌曲 (按标题)

5. 导出歌单

6. 随机播放歌曲(非必做)

7. 在指定位置添加歌曲(非必做)

8. 清空播放列表(非必做)

0. 退出程序

新加的歌曲保存到文件

请选择操作 (0-8): 5

请输入要导出的目标文件名: song_list.txt

已保存 3 首歌曲到文件: song_list.txt

- 1 ForElise,Beethoven,musics/ForElise.mp3
- 2 songNew,authorNew,songNew.mp3
- 3 Canon,Pachelbel,musics/Canon.mp3

7

实验内容

实验12 音乐播放器II

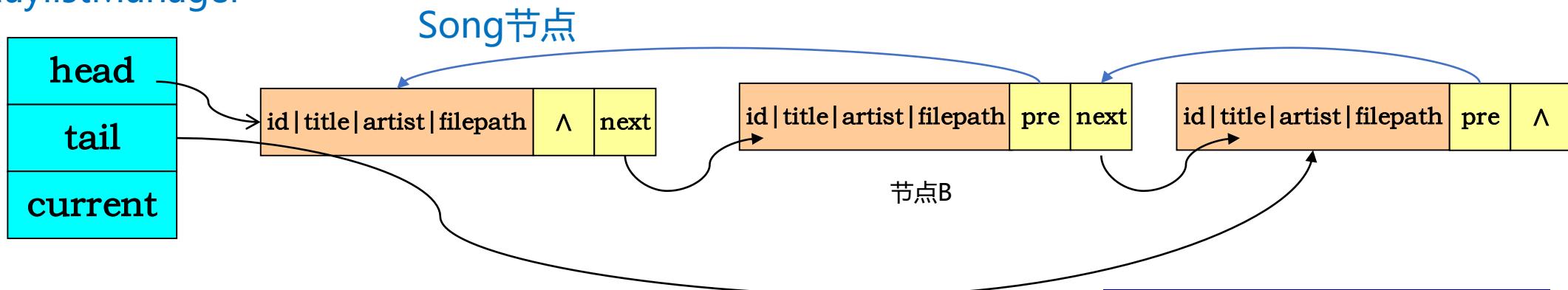
在实现实验11的基础上，使用**双向链表**来存储和管理歌曲。实现以下功能：

- 从文件读取歌曲信息，并建立歌单链表；
- 用户添加或删除歌曲信息；
- 用户通过歌曲名查找并播放歌曲；
- 显示歌单列表（正向+**逆向**）
- 切换到**上/下**一首歌
- 将更新后的歌单信息写回到文件。
- 根据歌名/作者/时长排序（**选做**）

```
// 定义歌曲节点结构（双向链表）
typedef struct Song{
    int id;
    char title[100];
    char artist[50];
    char filepath[300];
    struct Song* pre;
    struct Song* next;
}Song;

// 定义歌单状态结构
typedef struct PlaylistManager{
    Song* head;
    Song* tail;
    Song* current;
} PlaylistManager;
```

PlaylistManager



实验内容

功能界面

链表音乐播放器管理器

- ```
=====
1. 添加歌曲
2. 删除歌曲 (按标题)
3. 播放歌曲 (按标题)
4. 显示播放列表 (正向)
5. 显示播放列表 (逆向)
6. 导出歌单
7. 切换到下一首歌
8. 切换到上一首歌
9. 随机播放歌曲(非必做)
10. 在指定位置添加歌曲(非必做)
11. 清空播放列表(非必做)
12. 按照歌曲名排序(非必做)
0. 退出程序
=====
```

请选择操作 (0-12):

## 函数原型:

```
void add_song(PlaylistManager* manager, const char* title,
const char* artist, char* filepath);
int delete_song(PlaylistManager* manager, const char* title);
int play_song(PlaylistManager* manager, const char* title);
void display_playlist(PlaylistManager* manager);
void display_playlist_reverse(PlaylistManager* manager);
void export_songlist(PlaylistManager* manager, char* filepath);
void play_song_next(PlaylistManager* manager);
void play_song_previous(PlaylistManager* manager);
void play_song_random(PlaylistManager* manager);
int add_song_by_position(PlaylistManager* manager, int
position, const char* title, const char* artist, char* filepath);
void clear_songlist(PlaylistManager* manager);
void sort_songlist_by_name(PlaylistManager* manager);
```



# 实验报告

|              | 评分项     | 评分标准 |
|--------------|---------|------|
| 实验报告<br>(5分) | 系统设计    | 2分   |
|              | 函数设计    | 2分   |
|              | 系统结果与测试 | 1分   |

## □ 系统设计

- 1.数据结构设计
- 2.系统总体结构设计

## □ 函数设计

- 1.函数接口定义
- 2.函数功能说明

## □ 系统结果与测试

- 1.测试用例设计
- 2.测试结果说明

## □ 问题与解决方法

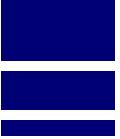
## □ 课程总结与建议

详细信息请查看实验报告模版



# 期末检查

|              | 检查项  | 评分标准                                                           |
|--------------|------|----------------------------------------------------------------|
| 期末检查<br>(5分) | 基本功能 | 1.编译通过，正确执行正常顺序流程，实现所有功能<br>2.对输入正常值、边界值、异常值可以进行判断和处理          |
|              | 程序设计 | 1.排序、查找算法<br>2.模块化程序设计                                         |
|              | 程序调试 | 1.正确创建工程文件,路径名命名正确<br>2.启动调试，单步或断点或打印语句调试程序，读懂调试结果并解释watch窗口内容 |
|              | 编码规范 | 1.代码整洁，层次清晰，易读，可维护<br>2.符合命名规范                                 |



# 作业提交

---

□ 实验11、12需提交：

- ① 完整工程文件
- ② pdf格式电子版**实验报告（实验11和12任选其一，按模板提交）**；
- ③ 压缩成.zip文件后，提交到收作业平台: **http://10.249.12.98:8000**
- ④ 初始用户名、密码均为**学号**；
- ⑤ DDL: **2025年12月31日**，截止之后**不接受补交**；
- ⑥ 作业提交截止时间内，可以重新提交作业，不限次数；

FEEL THE MEANING OF THE TRIP

青 / 春 / 不 / 老 / 梦 / 想 / 永 / 在



DREAM

MY DREAM WILL NEVER STOP

TAKE YOU ON A TRIP  
GO!

请同学们开始实验



哈爾濱工業大學(深圳)

HARBIN INSTITUTE OF TECHNOLOGY, SHENZHEN

探索 从未停止