

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

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。

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

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

### 【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

## 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);
```

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): █

## // 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

## 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;  
    int song_count;  
} 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 (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

# 流程图



**终端框（起止框）**：表示一个算法的开始或结束。里面的文字一般只是“开始”或“结束”



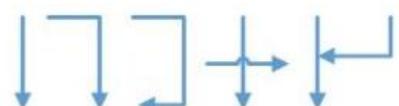
**输入、输出框**：表示一个算法输入和输出的信息。一般来说文字的开头要注明“输入”或“输出”。



**处理框（执行框）**：表示一个赋值、计算等操作。文字注明具体操作。



**判断框**：表示判断某条件是否成立。一般来说，它有两个分支，条件成立与否之后的流程在分支线处标明“是”“否”或“Y”“N”。



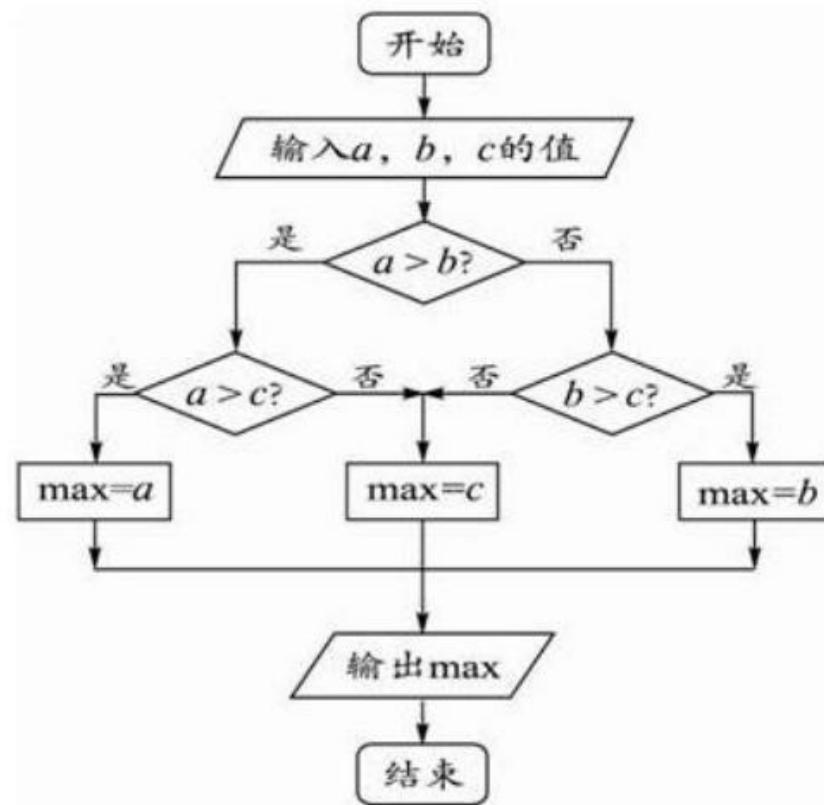
**流程线（指向线）**：流程图各符号之间以有向单向线连接。线一般要求横平竖直，可以有若干个90°的转弯。流程线尽量不要交叉，当两条流程线不得已而交叉时，将其中一条流程线的交叉处用圆弧隔开。

# 流程图

## 求三个数的最大值

| 在线绘制：<https://www.processon.com/>

流程图：



The screenshot shows the ProcessOn website interface. At the top, there is a navigation bar with 'Process On' (highlighted in blue), 'AI', '产品', '下载', '企业服务', '模板社区', '价格', '入门教程', '登录', and '免费注册'. Below the navigation bar, a banner states '专业强大的作图工具, 支持多人实时在线协作, 可用于甘特图、ER图、UML、网络拓扑图、鱼骨图、组织结构图等多种图形绘制'. In the center, there is a workspace where a user is editing a flowchart. The flowchart consists of various shapes: rounded rectangles for steps like '开始', '输入a, b, c的值', '输出max', and '结束'; diamonds for decisions like 'a > b?', 'a > c?', and 'b > c?'; and rectangles for assignments like 'max=a', 'max=c', and 'max=b'. The workspace also includes a sidebar with icons for different shapes and a toolbar at the bottom right. On the right side of the workspace, there are user names 'Amby', 'Mars', 'Geoffrey', and 'Eddie' associated with their respective shapes.



# 期末检查

	检查项	评分标准
期末检查 (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 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.按题目要求实现 2.模块化程序设计 3.算法设计思路
	程序调试	1.正确创建工程文件,路径名命名正确 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 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