

浙江大学

程序设计专题

大程序报告



2019~2020 春夏学期 2020 年 6 月 3 日

报告撰写注意事项

- 1) 图文并茂。文字通顺，语言流畅，无错别字。
- 2) 书写格式规范，排版良好，内容完整。
- 3) 存在拼凑、剽窃等现象一律认定为抄袭；0 分
- 4) 蓝色文字为说明，在最后提交的终稿版本，请删除这些文字。

目 录

1	大程序简介	4
1.1	背景及意义	4
1.2	目标要求	4
1.3	术语说明	4
2	功能需求分析.....	5
3	程序开发设计.....	6
3.1	总体架构设计	6
3.2	功能模块设计	6
3.3	数据结构设计	9
3.4	源代码文件组织设计	12
3.5	函数设计描述	18
4	部署运行和使用说明	31
4.1	编译安装	31
4.2	运行测试	32
4.3	使用操作	33
5	团队合作	40
5.1	任务分工.....	40
5.2	开发计划	40
5.3	编码规范	41
5.4	合作总结	43
5.5	收获感言	44
6	参考文献资料.....	46

图书管理系统大程序设计

1 大程序简介

1.1 选题背景及意义

本次设计任务选题为“图书信息管理系统”设计。

图书馆是人们丰富而宝贵的知识信息源，而基本的图书管理业务若仅凭人工处理每日将耗费大量的人力与财力，且由于人工处理的局限性，图书管理过程常会产生许多错误。利用计算机技术设计图书管理系统将更为准确、便捷、方便地管理图书借阅信息，也便于对图书进行整体分析。因而，本程序设计小组意在设计简单便捷的图书管理系统，为图书馆管理带来便利。

1.2 目标要求

基本实现图书库新建、打开、保存，退出；图书信息增加、排序、查询、修改、删除；图书借书、还书；用户登陆、注册、审核、修改、删除；系统参数管理，背景音乐设置，图形界面用户交互等功能。

1.3 术语说明

1) db 文件名解释：

图书信息库文件 “book_information.db”

用户信息库文件 “user_information.db”

图书用户关系表 “book_user_information.db”

2) 源代码文件名解释：

BMS

1. main.c/main.h

程序入口 Main() 所在的源文件/头文件

2. ds.c/ds.h

各个结构体和链表所用的函数的源文件/头文件

3. file.c/file.h

各个文件库的新增函数和统参数和密码保存及读取函数的源文件/头文件

4. gui.c/gui.h

图形界面用户交互模块的源文件/头文件

5. search.c/search.h

各个搜索所用的函数的源文件/头文件

6. sort.c/sort.h

各排序函数及找出最受欢迎图书、借书最多用户函数的头文件/源文件

imgui

基于 libgraphics 的图形控件的源代码和头文件

libgraphics

图形库 libgraphics 源程序和头文件

3) 宏定义、全局变量名与结构体解释:

宏定义、全局变量名与结构体解释详见“3.3 数据结构设计”。

2 功能需求分析

1) 业务逻辑要求:

符合正常图书管理的操作逻辑,如图书系统只能由管理员进行操作,只有已注册用户才能进行借还书操作,用户只能借还图书库中已有的图书等。

2) 所需基本功能:

- ①有界面友好且易于使用的菜单等;
- ②可实现各文件库的文件信息自链表存入和读入链表的功能;
- ③可实现信息在内存中的链表修改功能(包括新增、删除和插入等);
- ④可实现针对链表的查询功能并能在界面输出显示;
- ⑤能够实现针对书名和用户名的模糊搜索并能在界面输出显示;
- ⑥能够实现针对链表的排序功能;
- ⑦实现背景音乐的设置;
- ⑧实现找出借阅次数最多的五本书、找出结束最多的五名读者等特殊功能。

3) 数据结构需求:

需要向系统输入图书号、书名、用户号、用户名等数据结构信息。

4) 性能要求:

在运行本程序时只要按照正确操作流程就不会出现异常问题，系统稳定性好，安全高效。

3 程序开发设计

3.1 总体架构设计

- 1) main.c 为程序入口, 负责全局变量初始化, 并直接调用 gui.c 中的函数绘制界面和响应用户操作, 其自身代码量较少, 逻辑清晰, 方便修改;
- 2) main.h 为 main.c 的配套头文件, 包含了程序中用到的 C 标准库, 并定义了所有模块都有使用的全局变量 (如系统设置相关变量), 最后以宏定义的形式定义了一些全局的常量; 规定所有的模块都必须包含 main.h;
- 3) gui.c 是用户图形界面模块, 以 libgraphics 和基于 libgraphics 的 imgui 图形库作为基础, 代码中包含整个大程序的框架, 直接服务于用户。其功能分为两个部分: 以一定的内部逻辑绘制界面; 响应用户操作并改变程序状态; 除了 main.c, 所有模块都不会使用 gui.c 中的函数; gui.c 会使用除了 main.c 中主函数之外的所有模块中的函数以实现其功能;
- 4) gui.h 是 gui.c 的配套头文件, 包含了 gui.c 需要的图形库, 并定义了菜单数组、错误标志、状态变量等一系列图形相关的变量;
- 5) ds.c/search.c/sort.c/file.c 分别是数据结构和链表操作模块/搜索功能模块/排序功能模块/文件读写模块的代码文件, 每个代码文件都有对应的同名头文件与之配套; 调用关系是: search.c, sort.c 和 file.c 都调用 ds.c 为自身提供内存数据结构操作支持; search.c, sort.c 和 file.c 的功能相对独立, 互不调用。

3.2 功能模块设计

1) 用户图形界面及编辑功能:

- ①程序启动时, 先注册 mouseHandler(), keyboardHandler(), charHandler(), timerHandler() 四类消息事件的处理函数, 再通过 windowsInit() 函数完成对图形界面的初始化工作;
- ②程序界面绘制采用刷新绘制的方式, 有两类刷新方式: 有消息产生, 事件处理函数处理后立刻刷新; 由定时器发起的高速刷新。二者共同保证了界面信息的实时性;

③刷新屏幕的函数是 `refreshScreen()`，刷新过程中有以下操作：

- 1、调用 `clearFlags()` 清除已有的所有错误标志，准备错误重判；
- 2、调用 `libgraphics` 库提供的 `DisplayClear()` 函数清空屏幕，防止留下残像，同时准备内容重绘；
- 3、更新窗口参数、字体参数和系统时间，其他函数可能要用到这些参数，要为它们更新最新的值；
- 3、调用函数 `drawMenus()`, `drawStatusBar()` 和 `drawInnerContent()` 重绘菜单、状态栏控件；
- 4、调用函数 `drawInnerContent()`，该函数会检查 `graphicsMode` 全局变量，该变量记录了应当显示的页面，根据其值选择调用一下函数其中的一个来绘制对应界面内容：

```
void funcLogin(void);  
void funcBooksManagement(void);  
void funcReadersManagement(void);  
void funcStatInfo(void);  
void funcSettings(void);  
void funcHomepage(void);  
void funcLendBook(void);  
void funcReturnBook(void);
```

5、调用函数 `displayErrMessages()` 显示以上函数更新后的错误及其它提示信息；

④ `keyboardHandler()` 中有快捷键响应逻辑；`timerHandler()` 中有自动保存响应逻辑；

⑤另有一批辅助函数用于实现操作逻辑和错误判断，过程过于繁杂，再次不表，具体作用请见“3.5 函数设计描述”。

2) 检索功能：

①按图书查找：

1、按书号查找：通过对图书信息链表的遍历，找到和输入的号码相同的结构体，并返回指向该结构体的指针。

2、按书名、作者以及关键词的模糊查找：从一个标志指针指向的下一个结构体中的字符串开始，将要进行比较的这两个字符串（输入的字符串为 **A**，在结构体里的字符串为 **B**）中的大写字母改为小写，再通过 `includeJudge` 函数判断 **B** 中是否完全含有 **A**，若有，则表明查找成功。查找成功后，将标志指针赋值为指向该结构体的指针，故而可对同一个输入进行多次查找，找到所有符合条件的

结构体。若失败，则换下一个结构体的字符串或该结构体里的下一个字符串，依次进行。若到了最后的结构体，依旧没有找到，则为检索失败。

3、查书的借阅情况：通过输入的书号，在图书用户关系链表里进行遍历，找到和输入的号码相同的结构体，并返回指向该结构体的指针。

②按用户查找：

1、查用户信息：通过输入的用户号，在用户信息链表里进行遍历，找到和输入的号码相同的结构体，并返回指向该结构体的指针。

2、查用户的借阅信息：先通过输入的用户号，找到该用户所在的结构体，结构体中还包含一个指向用户借阅记录链表的头指针，故而可以通过该用户借阅记录的头指针找到用户的借阅信息。

3) 统计功能：

①通过一个五行两列的二维数组，每一行分别为一类图书的总数和外借数量，五行分别对应五类图书。通过该二维数组，可以实现统计 1、每类图书总数、在库数量和外借数量；2、目前全部书籍总数、在库数量和外借数量的功能。

②通过用户信息结构体中的存储着用户借书数量的一维数组（5 列，每 1 列分别对应 1 类书），可以实现统计该位用户借书的总量以及借某类书的数量的功能。

③通过图书信息结构体中的存储着该书被借次数的整型参数，可以实现统计该书被借阅的次数的功能。

4) 排序功能：

排序功能模块通过采用 `void book_sort(int flag)` 和 `void user_sort(int flag)` 两个函数分别实现对图书信息和用户信息的升降序排序。

5) 文件读写功能：

该功能模块通过事先建立好的已存储有相关信息的图书信息库文件 "book_information.db"、用户信息库文件 "user_information.db"、图书用户关系表 "book_user_information.db" 三个文件，使用下述六个文件操作函数，从而实现信息在文件中的存储和读入功能。

3.3 数据结构设计

main.h

```
#define MAXLEN 100/*全局字符串长度限制*/
#define PASSWD_OFFSET 3/*密码加密偏移参数*/
char currentPassword[9];/*系统参数*/
extern int borrowingDays; /*最长借阅天数*/
extern int borrowingBooks; /*最大借阅本数*/
time_t timep; /*动态刷新的系统时间*/

用途：存储系统设置参数
struct settings
{
    char adminPassword[21]; /*存放密码的密文*/
    int autoSaveSwitch; /*自动保存功能开启标志*/
    int autoSaveTime; /*自动保存时间间隔*/
    int musicSwitch; /*背景音乐功能开启标志*/
    int musicRepeat; /*背景音乐循环开启标志*/
    char musicPath[41]; /*背景音乐的绝对路径*/
}mySettings;
```

ds.h

```
/*图书的全局变量*/
extern struct bookInformation *bookhead; /*图书信息链表的头指针*/
extern struct bookInformation *booktail; /*图书信息链表的尾指针*/
extern int booktype[5][2]; /*存放每一类图书的总数以及外借数量*/
extern const char *booktypeName[5]; /*和图书类类型的字符串*/
extern struct bookInformation *bookformer; /*指向前一个指针*/
extern struct bookInformation *searchflag; /*用于搜索的指针*/

/*用户的全局变量*/
extern struct userInformation *userhead; /*用户信息链表的头指针*/
extern struct userInformation *usertail; /*用户信息链表的尾指针*/
extern struct userInformation *userformer; /*指向前一个指针*/
```

/*书籍和用户关系的全局变量*/

extern struct book_and_userRelation *relationhead; /*图书用户关系链表的头指针*/

extern struct book_and_userRelation *relationtail; /*图书用户关系链表的尾指针*/

extern struct book_and_userRelation *relationformer; /*指向前一个指针*/

用途：存储图书信息

```
struct bookInformation
{
    int bookNumber; /*书号*/
    int booktypeFlag; /*图书类型*/
    int borrowTimes; /*被借阅次数*/
    char bookName[81]; /*书名*/
    char keyword[5][31]; /*关键词，若不足 5 个，则用 '~' 填充*/
    char authorName[3][31]; /*作者名，若不足 3 个，则用 '~' 填充*/
    char publishingHouse[31]; /*出版社名*/
    char publicationDate[11]; /*出版日期，示例：2020-06-01*/
    struct bookInformation *next; /*指向下一个结构体的指针*/
};
```

用途：存储用户的借阅记录

```
struct user_borrowing_record
{
    int bookNumber; /*用户借过的书号*/
    char bookName[31]; /*书名*/
    int returnFlag; /*归还情况（1 位归还，0 为未归还）*/
    struct user_borrowing_record *next; /*指向下一个结构体的指针*/
};
```

用途：存储用户信息

```
struct userInformation
```

```
{
    int userNumber; /*用户号*/
    int borrowtimes[5]; /*借书次数,下标从 0 到 4 分别对应图书类别*/
    char userName[31]; /*用户姓名*/
    int genderFlag; /*性别(0 为女性, 1 为男性*/
    char workUnits[31]; /*工作单位*/
    struct user_borrowing_record *head; /*用户借阅记录链表的头指针*/
    struct user_borrowing_record *tail; /*用户借阅记录链表的尾指针*/
    struct userInformation *Next; /*指向下一个结构体的指针*/
};
```

用途：存储图书用户关系

```
struct book_and_userRelation
{
    int bookNumber; /*书号*/
    int userNumber; /*用户号*/
    int borrowyear; /*借阅日期： 年*/
    int borrowmonth; /*借阅日期： 月*/
    int borrowday; /*借阅日期： 日*/
    int returnyear; /*应还日期： 年*/
    int returnmonth; /*应还日期： 月*/
    int returnday; /*应还日期： 日*/
    struct book_and_userRelation *next; /*指向下一个结构体的指针*/
};
```

gui.h

```
/*窗口相关全局变量定义*/
static double windowsWidth, windowsHeight; /*窗口宽高*/
static double fontHeight; /*全局字体高度*/

/*菜单内容变量的定义*/
static char *menuListFile[] = {"File", "Open | Ctrl-O", "Save |
```

```

Ctrl-S", "Close | Ctrl-D", "Exit    | Ctrl-E"};
static char *menuListView[] = {"View", "Books      | Ctrl-B",
"Readers | Ctrl-R", "Statistic | Ctrl-I"};
static char *menuListOption[] = {"Option", "System Settings", "Log
Out"};
static char *menuListHelp[] = {"Help", "Please read README.md for
more information"};

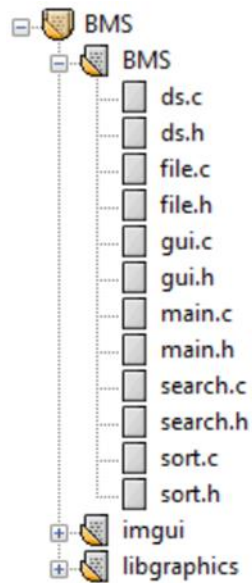
/*定时器相关定义*/
#define timerRefresh 1 /*1 号定时器用作屏幕刷新*/
#define timerAutoSave 2 /*2 号定时器用作文件的自动保存功能*/
#define timerInteval 2 /*屏幕刷新定时器触发时间间隔*/

/*GUI 全局变量 */
enum mode{Login, Readers, Books, Stats, Settings, bookLend,
bookReturn};
enum mode graphicsMode; /*决定主界面内容绘制方式*/
enum boolean isLogin; /*成功登陆标志*/
enum fileStatus{Saved, nSaved, nLoaded}currentFileStatus; /*当前
文件状态*/
bool dateErr; /*日期格式错误标志*/
bool queryErr; /*图书检索输入错误标志（多重输入）*/
bool notFound; /*搜索未找到或结束标志或用户不存在标志*/
bool isExist; /*图书/用户已存在标志*/
bool errPasswd; /*密码输入错误标志*/
bool passwdNotMatch; /*修改密码时两次不匹配的标志*/
bool passwdChangeOK; /*完成密码修改的标志*/
bool isSucceed; /*一般性操作成功标志*/

```

3.4 源代码文件组织设计

1) 文件函数结构



ds.c:

```
void bookAddition_Change(int flag, int bookNumber, int bookType,
char *bookName, char keywordPoiter[][31], int keywordNumber, char
authorPoiter[][31], int authorNumber, char *publishingHouse, char
*publicationDate);
void bookDelete(int bookNumber);
void userAddition_Change(int flag, int userNumber, char *userName,
int gender,
char *workUnits);
void userDelete(int userNumber);
void borrowBooks(int bookNumber, int userNumber, int year, int
month, int day);
void returnBooks(int bookNumber, int userNumber);
int judgeleapyear(int year);
char *passwordEncrypt(char p[], int offset);
```

ds.h:

```
extern struct bookInformation *bookhead;
extern struct bookInformation *booktail;
extern int booktype[5][2];
extern const char *booktypeName[5];
```

```
extern struct bookInformation *bookformer;
extern struct bookInformation *searchflag;
extern struct userInformation *userhead;
extern struct userInformation *usertail;
extern struct userInformation *userformer;
extern struct book_and_userRelation *relationhead;
extern struct book_and_userRelation *relationtail;
extern struct book_and_userRelation *relationformer;
struct bookInformation{.....};
struct user_borrowing_record{.....};
struct userInformation{.....};
struct book_and_userRelation{.....};
以及 ds.c 里面的函数的申明
```

file.c:

```
int bookDocument_add(void);
int userDocument_add(void);
int relationDocument_add(void);
int bookDocument_load(void);
int userDocument_load(void);
int relationDocument_load(void);
void systemSettings_add(void);
void systemSettings_load(void);
void systemSettings_add(void);
void systemSettings_load(void);
int bookSort_add(void);
int userSort_add(void);
```

file.h:

file.c 里面的函数的申明

gui.c:

```
void windowsInit(void);
```

```

void refreshScreen(void);
void drawMenus(void);
void drawStatusBar(void);
void drawInnerContent(void);
void funcLogin(void);
void funcBooksManagement(void);
void funcReadersManagement(void);
void funcStatInfo(void);
void funcSettings(void);
void funcHomepage(void);
void funcLendBook(void);
void funcReturnBook(void);
bool isDateValid(void);
int isQueryValid(void);
void chkPasswd(void);
void bookSearch(void);
void fuzzyLog(void);
void countNumber(void);
void clearCurrentBook(void);
void clearCurrentUser(void);
void clearFlags(void);
void displayErrMsgs(void);
void saveAll(void);
void setPen(void);
void drawHorizontalLine(char *color,double y);
void mouseHandler(int x,int y,int button,int event);
void keyboardHandler(int key,int event);
void charHandler(char ch);
void timerHandler(int timerID);

```

gui.h:

```

static double windowsWidth, windowsHeight;
static double fontHeight;
static char *menuListFile[] = {"File", "Open | Ctrl-O", "Save | Ctrl-S", "Close | Ctrl-D", "Exit | Ctrl-E"};

```

```
static char *menuListView[] = {"View", "Books | Ctrl-B",
"Readers | Ctrl-R", "Statistic | Ctrl-I"};
static char *menuListOption[] = {"Option", "System Settings", "Log
Out"};
static char *menuListHelp[] = {"Help", "Please read README.md for
more information"};
#define timerRefresh 1
#define timerAutoSave 2
#define timerInteval 2
enum boolean{True,False};
enum mode{Login, Readers, Books, Stats, Settings, bookLend,
bookReturn};
enum mode graphicsMode;
enum boolean isLogin;
enum fileStatus{Saved, nSaved, nLoaded}currentFileStatus;
bool dateErr;
bool queryErr;
bool notFound;
bool isExist;
bool errPasswd;
bool passwdNotMatch;
bool passwdChangeOK;
bool isSucceed;
以及 gui.c 中的函数的申明
```

main.c:

```
void Main();
```

main.h:

```
#define MAXLEN 100
#define PASSWD_OFFSET 3
char currentPassword[9];
extern int borrowingDays;
extern int borrowingBooks;
```



```
time_t timep;  
struct settings{.....}mySettings;
```

search.c:

```
struct bookInformation* booknumberSearch(int number);  
struct user_borrowing_record* user_borrowing_record_search(int  
number, struct user_borrowing_record *q);  
struct book_and_userRelation* book_and_userRelation_search(int  
number);  
struct userInformation* usernumberSearch(int number);  
int includeJudge(char *string, char *key);  
char *lowerChange(char *p);  
struct bookInformation* bookname_fuzzySearch(struct  
bookInformation* poiter, char *key);  
struct bookInformation* keyword_fuzzySearch(struct  
bookInformation* poiter, char *key);  
struct bookInformation* author_fuzzySearch(struct  
bookInformation* poiter, char *key);
```

search.h:

search.c 中的函数的申明

sort.c:

```
void book_sort(int flag);  
void user_sort(int flag);  
struct bookInformation *SORT_book(void);  
struct userInformation *SORT_user(void);
```

sort.h:

sort.c 中的函数的申明

2) 多文件构成机制

①.h 文件内存放全局变量的申明和需要被其他文件调用的函数申明；

②每一个.h 文件都使用#define 保护

例如: #ifndef _SORT_H_
#define _SORT_H_
.....
#endif

③.c 文件中定义函数和变量。

④.c 文件中定义的函数若需要被其他文件调用，则在对应的.h 文件中，存放它的申明，并在需要调用他的文件的开头进行预处理#include。

⑤当一个变量需要被不同的文件使用时，将它设置为全局变量，并将其申明写入对应的.h 文件中。

3.5 函数设计描述

gui.c

函数原型: void windowsInit(void);

功能描述: 图形界面的初始化

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 设置窗口标题、窗口宽高，初始化窗口，根据设置开始背景音乐的播放

函数原型: void refreshScreen(void);

功能描述: 刷新屏幕内容

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 清空屏幕内容，更新重要全局变量，调用绘制函数重绘界面

函数原型: void drawMenus(void);

功能描述: 绘制菜单

参数描述：无

返回值描述：无

重要局部变量定义：int menuChoice;

重要局部变量用途描述：保存选中的菜单项

函数算法描述：绘制菜单，根据选中的菜单项改变全局变量从而改变程序目前所处的状态，在下次绘制界面的时候会有体现

函数原型：void drawStatusBar(void);

功能描述：绘制状态栏

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：根据全局变量 graphicsMode 和 currentFileStatus 的值绘制相应的状态栏提示语

函数原型：void drawInnerContent(void);

功能描述：绘制窗口内部功能区内容

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：根据全局变量 graphicsMode 选择相应的子页面绘制函数绘制内部文本框、按钮等控件

函数原型：

void funcLogin(void);

void funcBooksManagement(void);

void funcReadersManagement(void);

void funcStatInfo(void);

void funcSettings(void);

void funcHomepage(void);

void funcLendBook(void);

void funcReturnBook(void);

功能描述：从上到下为绘制登陆界面、图书管理页、用户管理页、统计/排序页、

设置页、主页简明帮助、借书页、还书页

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：绘制对应界面的内容，判断相应的按钮的按下事件，调用其他模块的函数分步实现相应功能

函数原型：**void isInputValid(void);**

功能描述：判断图书管理页添加、修改、删除功能的输入是否有效

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：依次检查页面中每个输入框的内容是否合法，对相应的标志变量置位

函数原型：**int isQueryValid(void);**

功能描述：判断图书管理页搜索功能的输入是否有效

参数描述：无

返回值描述：整型返回值，**1** 表示有效，**0** 表示无效

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：检查是否有不和多重的搜索输入，置位相关标志的同时通过返回值返回搜索输入的有效性

函数原型：**int isQueryValid(void);**

功能描述：判断图书管理页搜索功能的输入是否有效

参数描述：无

返回值描述：整型返回值，**1** 表示有效，**0** 表示无效

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：检查是否有不和多重的搜索输入，置位相关标志的同时通过返回值返回搜索输入的有效性

函数原型: **void chkPasswd(void);**

功能描述: 检查登陆密码是否正确

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 检查登录密码, 加密后与全局设置变量中保存的密码比较, 通过后影响 **isLogin** 的值, 实现登陆过程

函数原型: **void bookSearch(void);**

功能描述: 判断图书搜索类型并实施搜索过程

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 在检查输入有效后, 根据唯一的输入得出是精确搜索还是模糊搜索, 调用 **search.c** 中的函数实现搜索, 并保存模糊搜索的记录

函数原型: **void fuzzyLog(void);**

功能描述: 恢复模糊搜索的历史记录

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 将模糊搜索记录变量中的内容重新拷贝到页面显示变量在

函数原型: **void countNumber(void);**

功能描述: 可选的图书关键词和图书作者计数

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: **ds.c** 中相关函数需要提供输入计数, 本函数会按要求计数, 将结果存入全局变量提供给图书新增、修改相关逻辑使用

函数原型: `int splitTime(int type);`

功能描述: 转换当前的年月日为整型数据

参数描述: `type=0`-年, `type=1`-月, `type=2`-日

返回值描述: 表示年/月/日的整数

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 使用数组查询的办法获取月份数据, 其余数据通过 ASCII 数与整数转换的办法得到

函数原型:

`void clearCurrentBook(void);`

`void clearCurrentUser(void);`

功能描述: 清除当前界面中的图书/读者数据

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 逐个清空图书/读者数据中的每一个输入框

函数原型: `void clearFlags(void);`

功能描述: 清除信息标志位

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 将 `gui.h` 中定义的信息标志逐个清零

函数原型: `displayErrMsgs(void);`

功能描述: 显示错误或成功提示信息

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 先通过 `graphicsMode` 判断当前界面类型, 再对应判断信息标志变量是否置位, 若置位, 在屏幕相应位置显示提示信息

函数原型:

`void saveAll(void);`

`void loadAll(void);`

功能描述: 存取全部文件

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 调用 `file.c` 中的相关函数存取三个数据库文件和两个配置文件, 实现内存和文件的数据互通

ds.c

函数原型: `void bookAddition_Change(int flag,int bookNumber,int bookType,char *bookName, char keywordPoiter[][31],int keywordNumber,char authorPoiter[][31],int authorNumber,char *publishingHouse, char *publicationDate);`

功能描述: 新增图书或修改图书信息

参数描述: `flag=1` 时, 实现图书新增功能; `flag=0` 时, 实现图书信息修改功能
`bookNumber` 等参数均为图书信息结构内的参数

返回值描述: 无返回值

重要局部变量定义: `struct bookInformation *p`

重要局部变量用途描述: 指向图书信息结构的指针

函数算法描述: 链表的新增

函数原型: `void bookDelete(int bookNumber);`

功能描述: 删除图书

参数描述: `bookNumber` 为书号

返回值描述: 无返回值

重要局部变量定义: `struct bookInformation *p`

重要局部变量用途描述: 指向被搜索的图书的信息结构的指针

函数算法描述: 链表的遍历以及删除

函数原型: `void userAddition_Change(int flag,int userNumber, char *userName, int gender, char *workUnits);`

功能描述：新增用户或修改用户信息

参数描述：**flag=1** 时，实现用户新增功能；**flag=0** 时，实现用户修改功能
userNumber 等参数为用户信息结构内的参数

返回值描述：无返回值

重要局部变量定义：**struct userInformation *p**

重要局部变量用途描述：指向用户信息结构的指针

函数算法描述：链表的新增

函数原型：**void userDelete(int userNumber);**

功能描述：删除用户

参数描述：**userNumber** 为书号

返回值描述：无返回值

重要局部变量定义：**struct userInformation *p**

重要局部变量用途描述：指向被搜索的用户的信息结构的指针

函数算法描述：链表的遍历以及删除

函数原型：**void borrowBooks(int bookNumber, int userNumber, int year, int month, int day);**

功能描述：借书,包括增加用户借书记录,增加该书的借阅次数,增加用户借阅次数,增加该类书籍的外借数量,以及新增用户和书籍的关系

参数描述：**bookNumber** 为书号, **userNumber** 为用户号, **year**、**month**、**day** 为借书的年月日

返回值描述：无返回值

重要局部变量定义：**struct userInformation *q ; struct user_borrowing_record *p ; struct bookInformation *s ; struct book_and_userRelation *r;**

重要局部变量用途描述：**q** 为指向用户信息结构的指针；**p** 为指向用户借阅记录结构的指针；**s** 为图书信息结构的指针；**r** 为指向图书与用户关系的指针

函数算法描述：链表的新增

函数原型：**void returnBooks(int bookNumber, int userNumber);**

功能描述：还书,包括修改用户借阅记录中该书籍的状态,减少该类书籍的外借数量,以及删除该图书用户的关系

参数描述：**bookNumber** 为书号, **userNumber** 为用户号

返回值描述：无返回值

重要局部变量定义：`struct userInformation *p ; struct user_borrowing_record *q ; struct bookInformation *s ; struct book_and_userRelation *r;`

重要局部变量用途描述：`p` 为指向用户信息结构的指针；`q` 为指向用户借阅记录结构的指针；`s` 为图书信息结构的指针；`r` 为指向图书与用户关系的指针

函数算法描述：链表的删除

函数原型：`int judgeleapyear(int year);`

功能描述：判断是否为闰年

参数描述：`year` 为借书日期的年份

返回值描述：返回 `1` 为闰年，`0` 为不是闰年

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：无

函数原型：`char *passwordEncrypt(char p[],int offset);`

功能描述：字符串加密函数，加密采用先异或，后凯撒加密的变种方法，凯撒加密中除了自定 `offset` 为偏移量，还添加了元素下标作为偏移量

参数描述：`p` 为输出字符串的首地址；`offset` 为偏移量

返回值描述：返回经过加密后的生成的字符串的首地址

重要局部变量定义：`char *q;`

重要局部变量用途描述：指向一个动态生成的并且与输入字符串同样大小的数组首地址

函数算法描述：采用与 `0101 1010B` 异或，将 `8` 位二进制中的第 `2`、`4`、`5`、`7` 位（从左往右数）翻转

search.c

函数原型：`struct bookInformation* booknumberSearch(int number);`

功能描述：根据书号在图书信息链表里面搜索图书的函数，`former` 指向搜索出的图书的前一个指针

参数描述：`number` 为要搜索的书号

返回值描述：若成功，则返回指向该图书信息结构的指针；若失败，则返回 `NULL`

重要局部变量定义：`struct bookInformation *p;`

重要局部变量用途描述：指向图书信息结构的指针

函数算法描述：链表的遍历

函数原型：

```
struct user_borrowing_record* user_borrowing_record_search(int
number,struct user_borrowing_record *q);
```

功能描述：根据书号在用户借阅记录链表里面查找

参数描述：number 为书号，q 为该用户借阅记录的头指针

返回值描述：若成功找到，则返回指向该借阅记录结构的指针；若失败，则返回 NULL

重要局部变量定义：struct user_borrowing_record *p;

重要局部变量用途描述：指向用户借阅记录的指针

函数算法描述：链表的遍历

函数原型：

```
struct book_and_userRelation* book_and_userRelation_search(int
number);
```

功能描述：根据书号在图书用户关系链表里查找

参数描述：number 为书号

返回值描述：若成功找到，则返回指向该图书用户关系结构的指针；若失败，则返回 NULL

重要局部变量定义：struct book_and_userRelation *p;

重要局部变量用途描述：指向图书用户关系结构的指针

函数算法描述：链表的遍历

函数原型：struct userInformation* usernumberSearch(int number);

功能描述：根据用户号在用户信息链表里面搜索用户的函数，former 指向搜索出的前一个指针

参数描述：number 为用户号

返回值描述：若成功找到，则返回指向该用户信息结构的指针；若失败，则返回 NULL

重要局部变量定义：struct userInformation *p;

重要局部变量用途描述：指向用户信息结构的指针

函数算法描述：链表的遍历

函数原型：int includeJudge(char *string, char *key);

功能描述：判断一个字符串（**string**）是否含有另一个字符串（**key**）

参数描述：**key** 和 **string** 分别为两个字符串的首地址

返回值描述：返回值为 **1**，为有含有；返回值为 **0**，则不含有

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：先将两个字符串中的字母都转为小写，再将 **key** 的第一个字符与 **string** 的第 **i** 个字符进行比较（**i** 从 **0** 开始），若到 **key** 的最后一个字符都相同，则含有；若中途有不用的字符，则 **i++**，再从 **key** 的第一个字符开始比较，直到含有或者 **i** 的值大于（**string** 的长度 - **key** 的长度）。

函数原型：**char *lowerChange(char *p);**

功能描述：将字符串中的大写字母转为小写

参数描述：**p** 为字符串的首地址

返回值描述：转换后的字符串的首地址

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：数组的遍历

函数原型：**struct bookInformation* bookname_fuzzySearch(struct bookInformation* poiter, char *key);**

功能描述：根据书名在图书信息链表里模糊搜索

参数描述：**poiter** 为指向图书信息结构的指针，**key** 为输入的书名搜索词

返回值描述：若成功，则返回指向该结构体的指针；若失败，则返回 **NULL**

重要局部变量定义：**struct bookInformation* p;**

重要局部变量用途描述：指向图书信息结构的指针

函数算法描述：**poiter** 为上一次对该图书进行书名模糊搜索时搜到的指针，故而本次搜索从 **poiter** 指向的下一个结构体开始。用 **includeJudge**，若其返回值为 **1**，则返回一个指向该图书结构体的指针，并且将 **poiter** 变为该指针（通过 **searchflag** 指针）；若失败，则返回 **NULL**

函数原型：**struct bookInformation* keyword_fuzzySearch(struct bookInformation* poiter, char *key);**

功能描述：根据关键字在图书信息链表里模糊搜索

参数描述：**poiter** 为指向图书信息结构的指针，**key** 为输入的关键次搜索词

返回值描述：若成功，则返回指向该结构体的指针；若失败，则返回 **NULL**

重要局部变量定义: `struct bookInformation* p;`

重要局部变量用途描述: 指向图书信息结构的指针

函数算法描述: `poiter` 为上一次对该图书进行关键词模糊搜索时搜到的指针, 故而本次搜索从 `poiter` 指向的下一个结构体开始。用 `includeJudge`, 若其返回值为 `1`, 则返回一个指向该图书结构体的指针, 并且将 `poiter` 变为该指针 (通过 `searchflag` 指针); 若失败, 则返回 `NULL`

函数原型: `struct bookInformation* author_fuzzySearch(struct bookInformation* poiter, char *key);`

功能描述: 根据作者名在图书信息链表里模糊搜索

参数描述: `poiter` 为指向图书信息结构的指针, `key` 为输入的作者名搜索词

返回值描述: 若成功, 则返回指向该结构体的指针; 若失败, 则返回 `NULL`

重要局部变量定义: `struct bookInformation* p;`

重要局部变量用途描述: 指向图书信息结构的指针

函数算法描述: `poiter` 为上一次对该图书进行作者名模糊搜索时搜到的指针, 故而本次搜索从 `poiter` 指向的下一个结构体开始。用 `includeJudge`, 若其返回值为 `1`, 则返回一个指向该图书结构体的指针, 并且将 `poiter` 变为该指针 (通过 `searchflag` 指针); 若失败, 则返回 `NULL`

file.c

函数原型: `int bookDocument_add(void);`

功能描述: 将图书信息写入图书信息库文件

参数描述: 无参数

返回值描述: 正常则返回 `0`, 若文件打开或关闭异常则返回 `1`

重要局部变量定义: `FILE *fp`

重要局部变量用途描述: 指向图书信息库文件"book_information.txt"的指针

函数算法描述: 将链表数据存入文件"book_information.txt"

函数原型: `int userDocument_add(void);`

功能描述: 将用户信息写入用户信息库文件

参数描述: 无参数

返回值描述: 正常则返回 `0`, 若文件打开或关闭异常则返回 `1`

重要局部变量定义: `FILE *fp`

重要局部变量用途描述: 指向用户信息库文件"user_information.txt"的指针

函数算法描述：将链表数据存入文件"user_information.txt"

函数原型：int relationDocument_add(void);

功能描述：将图书用户关系信息写入图书用户关系表

参数描述：无参数

返回值描述：正常则返回 0，若文件打开或关闭异常则返回 1

重要局部变量定义：FILE *fp

重要局部变量用途描述：指向图书用户关系表"book_user_information.txt"的指针

函数算法描述：将链表数据存入文件"book_user_information.txt"

函数原型：int bookDocument_load(void);

功能描述：导出图书信息库文件的数据，便于修改

参数描述：无参数

返回值描述：正常则返回 0，若文件打开或关闭异常则返回 1

重要局部变量定义：FILE *fp

重要局部变量用途描述：指向图书信息库文件"book_information.txt"的指针

函数算法描述：将图书信息库文件"book_information.txt"中的数据加载入链表

函数原型：int userDocument_load(void);

功能描述：导出用户信息库文件的数据，便于修改

参数描述：无参数

返回值描述：正常则返回 0，若文件打开或关闭异常则返回 1

重要局部变量定义：FILE *fp

重要局部变量用途描述：指向图书信息库文件"user_information.txt"的指针

函数算法描述：将图书信息库文件"user_information.txt"中的数据加载入链表

函数原型：int relationDocument_load(void);

功能描述：导出图书用户关系表文件的数据，便于修改

参数描述：无参数

返回值描述：正常则返回 0，若文件打开或关闭异常则返回 1

重要局部变量定义：FILE *fp

重要局部变量用途描述：指向图书用户关系表"book_user_information.txt"

的指针

函数算法描述：将图书用户关系表文件"book_user_information.txt"中的数据加载
入链表

sort.c

函数原型：void book_sort(int flag);

功能描述：图书排序

参数描述：flag==0 时，实现图书升序功能；flag==1 时，实现图书降序功能

返回值描述：无返回值

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：对图书信息链表进行升降序处理

函数原型：void user_sort(int flag);

功能描述：用户排序

参数描述：flag==0 时，实现用户升序功能；flag==1 时，实现用户降序功能

返回值描述：无返回值

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：对用户信息链表进行升降序处理

函数原型：struct bookInformation *SORT_book(void);

功能描述：找出借阅次数最多的五本书，若有并列也一并输出

参数描述：无参数

返回值描述：返回链表的头指针 struct bookInformation *，该链表用于存储所找出的图书的信息

重要局部变量定义：struct bookInformation *searchhead=NULL;

struct bookInformation *searchtail=NULL;

重要局部变量用途描述：searchhead 为该函数所返回的链表的头指针；

searchtail 为该函数所返回的链表的尾部指针；

函数算法描述：另建链表用于存储从图书信息链表中所找出的图书的信息，并返回其头指针

函数原型: `struct userInformation *SORT_user(void);`

功能描述: 找出借书最多的五名读者, 若有并列也一并输出

参数描述: 无参数

返回值描述: 返回链表的头指针 `struct userInformation *`, 该链表用于存储所找出的用户的信息

重要局部变量定义: `struct userInformation *searchhead=NULL;`

`struct userInformation *searchtail=NULL;`

重要局部变量用途描述: `searchhead` 为该函数所返回的链表的头指针;

`searchtail` 为该函数所返回的链表的尾部指针;

函数算法描述: 另建链表用于存储从用户信息链表中所找出的用户的信息, 并返回其头指针

4 部署运行和使用说明

(在 Help.pdf 帮助文档中有相同说明)

4.1 编译安装

1) 用 Dev-C 打开 “./Project/BMS/BooksManagementSystem.dev” 工程文件;

2) 展开左侧工程树中的 BMS, 完成以下设置 (如果仅是测试, 一般不需要设置)

打开 main.h, 编辑:

```
/*数据库文件大小限制, 取决于图书馆规模 */
```

```
/*估算方法: 50*(n+1)*a=size, n 是每一个读者借书记录可到的本数, a  
是读者数, size 是数据库文件大小 (单位: 字节) */
```

```
#define APPRANGE 30000000
```

和

```
/*密码加密偏移参数 */
```

```
/*根据保密需要进行修改 */
```

```
#define PASSWD_OFFSET 3
```

打开 main.c, 编辑:

```
/*读者一本书的最长借阅天数 */  
  
int borrowingDays=30;  
  
/*读者的最大借阅本数 */  
  
int borrowingBooks=5;  
  
/*紧急密码及默认密码,初次登陆及忘记管理员密码后恢复时使用 */  
  
/*修改后请妥善保管;默认为 123456 */  
  
strcpy(mySettings.adminPassword,passwordEncrypt("123456",PASSWD_0  
FFSET));
```

3) 工具栏中点击“全部重新编译”(快捷键 F12),等待编译完成,关闭 Dev-C++ 软件;

4) 运行“./Project/BMS/Outputs/clean.bat”清理输出文件;如有需要,将生成的程序“./Project/BMS/Outputs/BooksManagementSystem.exe”移动到别处;

5) 如果要测试软件功能,将提供的测试数据库

“./Example Database/book_information.db”

“./Example Database/user_information.db”

拷贝到与“BooksManagementSystem.exe”相同的目录下;

6) 如果需要,将背景音乐示例“./Example Database/bgm.wav”拷贝到一绝对路径中不含中文且长度小于 30 个字符的已知路径下;

7) 如果需要,建立“BooksManagementSystem.exe”的快捷方式,完成安装,运行即可。

4.2 运行测试（几个典型的开发案例）

典型案例 1: 沟通问题导致前期数据结构设计存在缺陷

问题: 小组成员之间编写进度不同。小组成员 B 编写数据结构模块,先定义了重要数据结构并编写了数据结构操作相关函数,此时小组成员 A 没有关注。中期小组成员 A 开始 GUI 模块的编写,过程中发现小组成员 B 提供的数据结构存在问题——整数长度不够用,数组定义也有一定的问题(没有考虑‘\0’占据的内存空间)。

解决办法: 简单修改数据类型定义和相关函数的声明、函数头的有关部分,加大

数组长度和整数类型（把 int 换成 long），从而解决问题。

启示：有问题及早沟通；早期的设计越详尽越好；未开始工作的组员要关注工程的动向，有建议要及时提出并作修改。

典型案例 2：图形界面操作逻辑考虑欠缺导致混乱

问题：GUI 框架成型，开始调试，发现存在很多逻辑错误，可能导致内容错乱，严重的会拖垮程序，如重复保存问题，提示信息重叠问题，用户有书未还就能删除用户导致还书时出错等。

解决办法：一方面对代码整体展开调试检查，另一方面针对已知的问题逐个修复。此外，我们设计了更多的测试样例检验系统的稳定性，发觉潜在的 Bug，并将 Beta 测试版和数据样本发给外校的好友进行小范围公测。

启示：对于复杂代码，自顶向下的模块化设计有助于减少 Bug 的产生；慎用全局变量；设计大量的、非常规测试样本来发现更多的问题；借助公众的力量修补更多的漏洞。

典型案例 3：盲目复制+粘贴代码导致畸形链表的产生

问题：GUI 联合数据结构模块测试的过程中发现偶尔会产生字符错位，某一栏目中的内容溢出到别处去。起初怀疑是 GUI 模块的逻辑问题。经过单步调试发现为数据结构模块中一处 for 循环为了复用之前的代码，直接复制而没有修改循环次数，导致指针越界，数据写入其它区域中，严重时还会导致链表损坏，相关函数一旦调用整个程序崩溃。

解决办法：修改循环次数，问题解决。

启示：这种问题及其隐蔽，而且每次实验输入数据不同，现象都不一样，这时设置合理的断点，进行单步调试、观察变量会非常有助于查出错误；对于大量复用的代码，为其编写相应的功能函数才是最好的办法，切忌不加分析盲目复制原有的代码。

4.3 使用操作

（1）管理员登入与登出

登入

在软件刚刚启动时或登出后

- 输入管理员密码，默认为 123456（main.c 中设置）
- 点击“Login”登入（快捷键为回车键），点击“Exit”退出软件

登出

- 登陆后使用菜单“Option” - “Log Out”退出登陆
- “File”-“Exit”可直接退出软件

（2）文件操作

在成功登入后：

- 文件未打开时，使用菜单“File” - “Open”打开文件，此时自动进入常用页面“图书管理页面”
- 文件已打开而未保存（右下角显示“Not Saved!”）时，使用菜单“File” - “Save”保存文件

Note: 除了借/还书功能会在操作后立刻保存外，其余操作需要执行“File” - “Save”才能保存（包括设置），没有保存的内容在关闭软件后会全部丢失！

- 文件已打开时，使用菜单“File” - “Close”关闭文件并返回主页

Note: 执行关闭操作时不会自动保存数据！

（3）功能页面切换

软件登陆且文件打开后，使用“View”和“Option”菜单切换页面：

- “View” - “Books” 打开图书管理页（快捷键 Ctrl+B）
- “View” - “Readers” 打开读者管理页（快捷键 Ctrl+R）
- “View” - “Statistic” 打开统计/排序功能页（快捷键 Ctrl+I）
- “Options” - “System Settings” 打开系统设置页

Note: 切换页面会导致当前页面中的所有信息被清除！

（4）图书管理

使用“View” - “Books”菜单进入图书管理页（快捷键 Ctrl-B）：

添加图书

- 点击“Clear”清空所有输入框（快捷键 F1）
- 遵照下文“（10）附录”中的限制填写图书信息
- 点击“Prev”和“Next”按钮选好图书分类
- 完成后，点击“Add”将图书添加到信息库

Note:

- 书号、书名、出版社和出版日期是必填项
- 关键词、作者是选填项，填写时请按从左至右、从上至下的顺序依次填写，跳格填写会导致意想不到的问题

搜索图书

按书号精确搜索

- 点击“Clear”清空所有输入框（快捷键 F1）
- 输入要查找的书的书号
- 点击“Search”按钮（快捷键 F4），搜索结果（若有）会自动显示在界面中

按关键词模糊搜索

第一次搜索：

- 点击“Clear”清空所有输入框（快捷键 F1）
- 在关键词第一栏中输入要查找的书的关键词（字母大小写不敏感）
- 点击“Search”按钮（快捷键 F4），符合条件的第一本书的信息（若有）会自动显示在界面中

若要继续搜索下一本符合条件的书：

- 点击“Clear”清空所有输入框（快捷键 F1）
- 点击“Fuzzy Log”快速调出上一次搜索的输入（快捷键 F3）（也可以手动输入）
- 点击“Search”（快捷键 F4），符合条件的第二本书的信息（若有）会自动显示在界面中
- 如此可搜索符合条件的第三本书、第四本书……直到整个数据库查找完成

数据库查找完成会给出相关提示，再次执行搜索会从头开始

Note: 如果数据库检索中途更换了搜索关键词，软件会继续从上一次停下的位置继续检索，这会导致数据库前部不被检索；如要检索完整的数据库，请在检索完成后再次点击“Search”，完成数据库前部的检索！

按作者模糊搜索

使用方法与“按关键词模糊搜索”一致，但需要将关键词填入作者第一栏

修改图书

- 使用搜索功能查找并显示待修改图书的完整信息
- 修改需要修改的部分（输入限制与“添加图书”一致）
- 点击“Modify”即可完成修改

删除图书

- 使用搜索功能查找并显示待删除图书的完整信息
- 点击“Delete”即可完成删除

（5）读者（用户）管理

使用“View” - “Readers” 菜单进入读者管理页（快捷键 Ctrl-R）：

添加用户

- 遵照下文“（10）附录”中的限制填写用户信息
- 点击“Select”按钮选好用户性别
- 完成后，点击“Add”将用户添加到信息库

Note:

用户 ID 姓名和公司都是必填项

查询用户数据

- 填写或修改已有的用户 ID，此时页面内容会自动清空
- 点击“Query”，此时对应用户（若存在）的信息会显示在页面上

- 用户数据包括：用户 ID，用户名，用户性别，用户所在公司，借阅最早且尚未归还的 3 本书

修改用户

- 使用“查询用户数据”功能查找并显示待修改用户的完整信息
- 修改需要修改的部分（输入限制与“添加用户”一致）
- 点击“Modify”即可完成修改

删除用户

- 使用“查询用户数据”功能查找并显示待删除用户的完整信息
- 点击“Delete”，若该用户所有借阅图书均已归还，即可删除该用户

Note：修改用户 ID 会自动清空当前界面的所有内容，以方便下一次的查询或修改

（6）借还书操作

借书

- 在图书管理页点击“Clear”清空输入框内容（快捷键 F1）
- 录入读者所需要借阅的书的书号
- 若该书在库，“Lend”按钮显示，点击“Lend”（快捷键 F2）
- 确认屏幕上显示的书籍和借阅限制信息，核对当前时间无误
- 录入借阅该图书的用户 ID
- 若该用户在借的图书书目未超过最大限制（编译前设定），即可点击“Lend”确认将该书借出
- 借阅页面下点击“Cancel”可中断借书流程，返回图书管理页

还书

- 在图书管理页点击“Clear”清空输入框内容（快捷键 F1）
- 录入读者所需要归还的书的书号
- 若该书被借出，“Return”按钮显示，点击“Return”（快捷键 F2）
- 确认屏幕上显示的图书和用户信息无误，检查该用户是否逾期归还
- 点击“Return”按钮确认归还该书
- 归还页面下点击“Cancel”可中断还书流程，返回图书管理页

Note:

若借还书按钮显示“N/A” (Not Available)，说明该书不存在，不能提供借还服务，请检查录入的书号是否有误，及该书是否已经录入数据库中

(7) 统计与排序

使用“View” - “Statistic” 菜单进入统计/排序功能页（快捷键 Ctrl-I）：

- 界面第一部分为分类显示的图书统计信息，每一类的总图书数和已被借出的图书数将被分开显示，二者作差得到在库图书数
- 界面第二部分为排序模块：
 - 点击“Switch” 键可切换排序依据，可选项：书名、用户姓名
 - 点击“Ascending” 执行升序排序，点击“Descending” 执行降序排序
 - 排序结果存放在 *BooksManagementSystem.exe* 同目录下的 *sort_result.txt* 文本文件中，结果被格式化输出，可用文本编辑器（如“记事本” 程序）查看

Note: 排序结果文件请及时转移或备份，下一次排序执行时会将上一次的结果覆盖！

- 界面第三部分为“近期榜单” 功能：
 - 左侧为最受欢迎的三本图书（依据：总借阅次数）
 - 右侧为借书量最多的三名读者

Note: 若存在并列的情况，会按照数据库中的数据顺序显示前三名，执行排序后结果可能发生改变！

(8) 系统设置

使用“Option” - “System Settings” 菜单进入系统设置页：

修改系统设置

自动保存设置

- 点击“Auto-Save ON/OFF” 提示旁边的“ON/OFF” 按钮打开/关闭自动保存功能（缺省关闭）

- 在“Auto-Save Interval”提示旁边的输入框里输入自动保存间隔时间（单位：毫秒）
- 使用“File”-“Save”菜单保存设置

背景音乐设置

- 点击“Music ON/OFF”提示旁边的“ON/OFF”按钮打开/关闭背景音乐功能（缺省关闭）
- 点击“Music Repetition”提示旁边的“ON/OFF”按钮启用/禁用背景音乐循环（缺省禁用）
- 在“Music Path”提示旁边的输入框内输入音乐文件的绝对路径，要求音乐为 wav 格式，路径中不得含有中文字符，路径字符数不超过 30 字
- 使用“File”-“Save”菜单保存设置

修改管理员密码

- 在相应输入框中输入旧密码，新密码并重复新密码以确认（新密码可以设置为空）
- 点击“Change”，若旧密码正确，两次新密码一致，密码修改成功
- 使用“File”-“Save”菜单保存新密码

Note:

- 所有设置项在修改后不会自动保存！请使用“File”-“Save”菜单保存所有设置（包括管理员密码）到配置文件！
- 管理员密码在下一次登陆时生效，其余所有设置在下一次打开软件时生效！

（9）简单帮助信息

- 软件登陆后会在首页显示简单帮助信息
- 在文件打开时
 - 先使用菜单“File”-“Save”保存没有保存的文件
 - 再使用“File”-“Close”即可返回首页查看

（10）附录——软件中各类输入限制

登陆密码：最长 8 位的任意可输入字符

- 图书管理功能：

- 书号：不以 0 开头，最长 8 位的数字
- 书名、出版商：最长 30 位（含空格），数字、大小写英文字母、空格和一般标点符号（下划线除外）
- 关键词、作者：最长 10 位（含空格），数字、大小写英文字母、空格和一般标点符号（下划线除外）
- 出版日期：格式必须为 YYYY-MM-DD
- 读者管理功能及借还书功能：
 - 读者 ID：不以 0 开头，最长 8 位的数字
 - 读者姓名、公司：最长 30 位（含空格），数字、大小写英文字母、空格和一般标点符号（下划线除外）

5 团队合作

5.1 任务分工（互评版略）

略

5.2 开发计划

1) 第一阶段：程序结构统筹（约 5 天）

任务安排：工程框架的建立；项目统筹；全局数据结构的设计与实现；用户界面及菜单结构、程序逻辑的设计（用文字、伪代码、流程图、示意图等手段描述）；文件操作相关模块

P.S. 此阶段关系到项目的整体结构与质量，影响后续编码效率，应集中交流，尽量将问题细化，制定统一的标准方案；注意要代码、图表与文字结合着展开，设计的时候不要空谈想法，不考虑实现；数据结构和文件操作相关函数后续都会用到，需要提前编写。

2) 第二阶段：数据处理功能的实现（约 8 天）

任务安排：编辑功能相关模块；图形用户界面的布局与绘制；检索功能相关模块；统计功能相关模块；排序功能相关模块；帮助进行已有模块的检查

P.S. 此阶段编码量较大，细节问题比较多，为本项目的核心代码，需要反复进行功能验证，花费较多的时间；可能存在少量不可避免的函数交叉使用的情况，请有关人员及时沟通。

3) 第三阶段：图形用户界面的建立与程序的综合调试（约 8 天）

任务安排：图形用户界面的操作逻辑的实现；协助进行综合调试；大家合作进行项目的 Debug；写 1-2 个可以加分的功能

P.S. 该过程的难点在于图形界面相关操作，由于刚学习不久，相关库的使用还在摸索阶段，且有可能出现意外状况，故安排了相对充裕的时间进行编码；由于图形用户界面相关操作较多较杂，主要工作将由同学 A 来完成（如果分给三人编写可能不易管理），期间可能会找同学 B 和同学 C 协助（具体问题的编码）；

4) 第四阶段：项目完善与加分项的争取（约 3 天）

开展简单的自评；使用手册撰写；大程报告撰写

5.3 编码规范

1) 程序版式：

- ①用缩进体现代码结构，只使用空格，不使用 Tab；
- ②每个函数声明、定义结束之后都要加空行；
- ③关键字、逗号、非一行结束符的分号之后要留空格，函数名后不留空格；
- ④在访问指针和引用表达式时，句点或箭头前后不要有空格，指针、地址操作符后不要有空格；
- ⑤‘*’作为申明符时紧靠前一个关键词或者紧靠变量名；
- ⑥函数声明和实现处所有形参名称保持一致；
- ⑦if 条件语句后应紧跟大括号的复合语句；
- ⑧switch 语句中总是包含一个 default 语句；
- ⑨预处理指令不要缩进，从行首开始，即使位于缩进代码块中，预处理指令也应从行首开始；
- ⑩长表达式若有操作符，则要在低优先级操作符处拆分成新行，操作符放在新行之首，新行行首与上一行括号内的第一个字符对齐。

2) 命名约定:

- ①头文件和实现文件分别用 **.h** 和 **.c** 作为文件后缀名,文件名采用小写英文组合,不可采用拼音;
- ②结构体、函数以及变量采用英文大小写混合,以直白准确为原则,可以使用一些常用的变量名如 **cnt** (计数变量)、**flag** (标识变量),除循环变量等特殊情况下,严禁使用如 **a**、**b**、**c**、**e**、**m**、**abc** 等无意义的变量名,不可采用拼音。

3) 注释风格:

- ①不得将一段程序代码注释掉;
- ②注释的位置应与被描述的代码相邻,对于单行代码的注释可以放在该代码的上方或右方,不可放在下方;
- ③重要的、可复用的算法、数据结构加上简明注释,如: **struct bookInformation{...} //存放单本图书信息的结构;**
- ④所有函数都要在声明处添加块注释以说明接口、用途和版本信息,方便自己和组员维护;
- ⑤不太容易理解的变量请添加单行注释;
- ⑥注释可以用中文编写。

4) 头文件:

- ①**#include** 指令后只能紧跟<filename>或 " filename"形式的语句;
- ②所有头文件都应该使用**#define** 防止头文件被多重包含,命名格式为 **_<filename>_h**,如 **_FILE_h**;
- ③头文件应该只用于声明函数、宏等,而不能包含对函数、变量的定义。

5) 健壮性和容错性

- ①程序中不得含有不被执行的代码或不被使用的变量;
- ②在声明一个数组时,需要显式地给出数组大小,或者通过初始化明确数组的大小;
- ③不要使用三级及以上的间接指针,只有当两个指针指向同一数组中的元素时,才可在指针之间进行减法操作。

5.4 合作总结

1) 开发亮点：管理员登入界面、背景音乐、可自选系统是否自动保存更改、管理员 password 加密以及用户可更换密码；

2) 开发挑战点：图形界面绘制和内部链表功能的实现；

3) 应用知识点：链表的相关知识、多文件组织的相关知识、文件读取的相关知识以及图形界面绘制的相关知识等。


4) 合作过程：

①前期准备：


阶段一：2020 年 5 月 12 日：

 【重要】大作业小组分工及编码规范.docx	2020-05-12 0:18	永久	20.2KB
 7.成绩评定及申诉阶段须知.docx	2020-05-02 10:03	永久	184KB
 6.自评互评得分细则模板.xlsx	2020-05-02 10:03	永久	18KB
 5.互评任务注意事项和得分细则.docx	2020-05-02 10:03	永久	244KB
 4.s程序设计专题大程序报告V2020模板.doc	2020-05-02 10:02	永久	624KB
 3.大作业提交须知.docx	2020-05-02 10:01	永久	52.1KB
 2.s2020大程序候选题目-4选1.docx	2020-05-02 10:01	永久	9.79MB
 1.大作业时间节点.png	2020-05-02 10:01	永久	112KB

阶段二：2020 年 5 月 17 日：

 main.zip	2020-05-16 11:25	永久	171KB
 basic_list_function.h	2020-05-16 8:51	永久	646B
 函数参数.docx	2020-05-15 22:15	永久	14.9KB
 global_variable_definition.h	2020-05-15 20:55	永久	2.34KB

阶段三：2020 年 5 月 24 日：

 Document_Base 5.22.rar	2020-05-22 20:42	永久	
 2020_05_20.zip	2020-05-20 21:44	永久	
 Document_Base.c.rar	2020-05-17 21:45	永久	
 Document_Base.rar	2020-05-17 17:11	永久	
 2020_05_07_5点.zip	2020-05-17 16:44	永久	
 图书管理系统 (5.17项目预览).rar	2020-05-17 14:26	永久	
 2020_05_17_main.zip	2020-05-17 0:43	永久	

阶段四：2020 年 6 月 1 日：

大程报告分工

2020年5月30日 18:03

大程序简介（其中术语说明我没太懂是什么东西，或许可以问一下助教和老师？）

功能需求分析

把各自完成的功能模块设计和数据结构设计的详细设计内容完成（编辑功能、检索功能、统计功能、排序功能、文件读取和加分项功能），还有总体设计架构

源代码文件组织设计中的文件函数结构

把各自的函数设计描述写完（按照.h文件内的顺序写，每一条都要，如果没啥可写的就写无）

团队合作（在结合实际的基础上瞎哔哔） 有个人感言

把各自参考的文献写一下

背景音乐的选择（wav格式？）

图书 和用户的测试数据集

运行部署和使用说明

5.5 收获感言

1) 团队心得和体会：

①充分体会到前期沟通的重要性。只有进行了良好并且充分的前期沟通，团队的各项工作才可以较好的展开。我们一开始，忽视了前期沟通的重要性，故而导致产生了各成员在开始时较为迷茫，不知怎样开始的局面，也因此使得我们实际开始的时间较晚以及最后时间内所需完成的工作量较大。

②充分体会到写代码和注释时注重规范的重要性。虽然在一开始，我们团队指定了一部分的代码规范和注释规范，但在实际操作过程中，这些规范渐渐被我们忽视，因此导致最后需要花费大量的精力和时间去统一代码和注释的格式。

③充分体会到一个精确 DDL 的重要性。在代码实现阶段，我们团队制定的 DDL 多为“约*天”，导致组员有拖 DDL 的现象发生，也因此导致实际进度比我们预期的进度慢。

④充分体会到团队的力量和分工的重要性。在团队分工以及协作下，我们成功的将一个在一开始看来不可能完成的 project 完成了。在良好的分工下，每一个人所需要完成的任务渐渐细化，具体化，也让一个看着让人发憊的大作业变得较容易完成。

2) 自我评价:

组员 1: 不得不说,担任大作业小组的组长是对自己的一个挑战。作为组长,需要协调组员的编码进度,保障组员的代码质量,并维护整个工程,在不断加入新功能的同时,保证当前工程的稳健性,更新模块的同时推进自己的 GUI 模块的编写进度。组员编写好的模块常常会出现一些或明显或不明显的漏洞,对于那些明显的漏洞,一般可以及时修复,但不明显的漏洞如果没有在第一时间发现,积累到后期会变得很难解决。为了给整个工程除错,我在了解程序整体架构的同时,深入研读组员编写的功能模块内部的算法与数据结构组织,综合运用分析编译器错误信息、构造测试数据、单步调试、观察变量等方法,最终修复了已知的全部漏洞,并在漏洞的修补过程中积累了调试和沟通经验,加深了对一些编程陷阱的理解,并在阅读组员代码的过程中,认识到注释和代码规范的重要性,学到了一些不错的算法。除此之外,我编写了说明文档,也对规范的团队合作大程序开发有了基本的认知和实践。

组员 2: 在此次大作业的完成过程中,我对自己基本满意。前期,我积极主动的与其他组员进行沟通,主动拉群;阶段一、二和三,及时完成组长分配的工作,提出管理者密码加密的想法并且完成其代码的实现,帮助组长发现程序的 **bug**; 阶段四,主动承担大程报告的任务分配,整合以及排版等工作。但是,仍然存在一些不满意的地方。一、自己的代码没有认真的、多次的检查和测试,导致产生一些低级错误,使得代码整合同学花费了很多精力 **debug**; 二、此次大作业,我们组开始的时间较晚,并且基本只有在周五、周六和周日写代码,使得最终时间较为紧张。总的来说,这一次大作业的经历还是非常的美好。虽然中途有很多的 **bug**,有很多 **debug** 时的烦躁,但当看到最终成品的时候,还是感到非常的自豪和满足。

组员 3: 因为一些其他课程的事情最开始我没能很快着手大程序设计的任务,同时因为前期缺乏与组员的沟通,对程序结构的理解有所偏差导致出现了一些逻辑上的函数设计问题甚至设计了一些多余的函数,值得自我反思。另外,我也认识到在编写代码时规范性与准确性的重要,有时一个小小的错误或许在自我调试时未能发现,但当整合到整个工程时就会暴露出很多问题。写代码尤其是调试的过程固然很让人崩溃,但看到团队最终的结果就觉得一切都算值得。而让我没想到的是,在组员分工协作、细化任务之后,最初看似复杂难以实现的大工程竟也变的简单容易起来。从前我一贯认为个人独立完成会比团队合作更为快速有效,但这次的合作经历确实让我真切体会到了团队

合作的重要性。

6 参考文献资料

1) <https://blog.csdn.net/jijianshuai/article/details/80582211>

2) https://blog.csdn.net/baidu_30000217/article/details/77823084