燕山大学课程三级项目报告

计算机应用基础与程序设计
（三级项目）

图书管理系统的设计与开发

项目小组	计算机科学与技术 2 班第 1 组		指导教师	牛景春
小组成员	学号	姓名	项目成绩	所做工作
	202211040044	李汪洋		编写主体代码、报告制作
	202211040030	潘亚航		答辩 PPT 制作、部分代码编写、资料搜集
	202211040029	牛旭晨		部分代码编写、资料搜集
开发日期	2022 年 11 月 20 日至 2022 年 11 月 29 日			

摘要

[摘　要]

本文主要分为摘要、目录、正文、心得及体会、附录 5 个部分。本报告主要记录了本小组在设计该系统的过程中的设计方案以及心得体会等内容，重点阐述了本组对图书管理系统的整体设计思路。

图书管理系统是生活中常见的一种管理系统设计，具有广泛的应用。本小组尝试以 C 语言实现一个小型的图书管理系统，具有增加图书、更改图书、删除图书、查询图书、保存和读取图书馆文件等功能。在实现过程中，采用了链表、指针、函数等 C 语言编程方法，并且使用了动态内存分配和字符串库等功能。本小组还实现了 ISBN-13 校验算法，以校验录入图书的 ISBN 号。最终形成了一个较为完备的小型图书管理系统。

[关键词]　C 语言；图书管理系统；链表；动态内存分配；文件 IO；ISBN-13 校验算法。

目录

正文

一、研究内容的基本原理

项目的主要工作是实现一个图书管理系统，主要考察对 C 语言的运用。项目指导书还要求采用链表、函数等方式实现该系统。

二、所采用的研究方法和工具

本次三级项目采用小组成员合作的方法进行。小组成员在课外时间进行面对面讨论和线上交流，合作完成该项目。

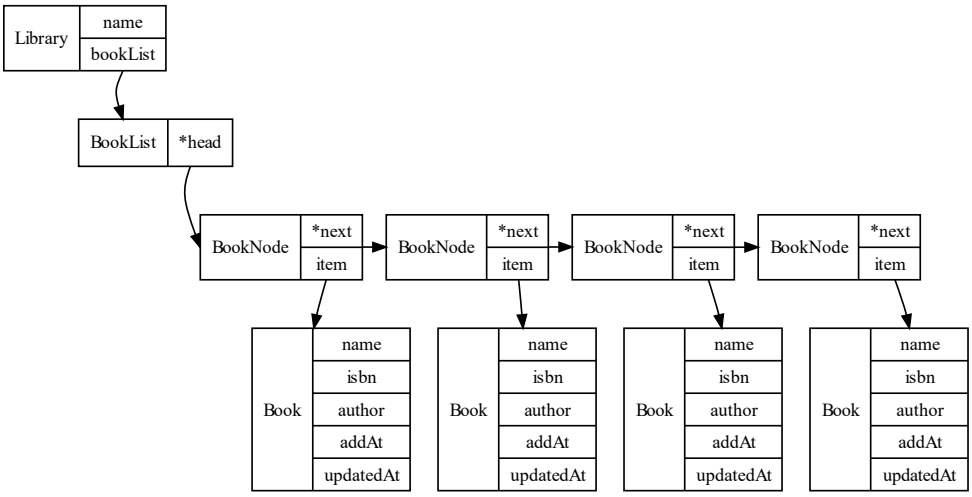
小组成员通过查阅 C 语言教科书和网上的资料，以 DevC++工具完成该项目的开发工作，以 Word 和 Powerpoint 软件完成项目报告和答辩 PPT 的编写工作。

三、项目的方案设计

通过小组成员的讨论，决定将本项目分为两个部分设计，即数据结构设计和算法设计。

（一）数据结构设计

在数据结构方面，为了满足图书管理系统整体设计要求，小组成员合作讨论出以下数据结构设计：



其中，Library 结构代表图书馆，持有 name 和 bookList 属性。name 属性代表图书馆的名称，bookList 属性代表图书馆中存放的书籍列表。BookList 结构体代表一个单向链表，存放有一个 head 属性，指向链表的第一个节点。BookNode 结构体代表链表的节点，存放有代表书籍的 Book 结构体和指向下一个链表节点的 next 指针。这四个结构体就是本项目的所有数据结构，用 C 语言代码实现如下：

```c
#define BOOK_NAME_LENGTH 21
#define BOOK_ISBN_LENGTH 18
#define BOOK_AUTHOR_LENGTH 21
#define LIBRARY_NAME_LENGTH 20

// Book Item
typedef struct {
```

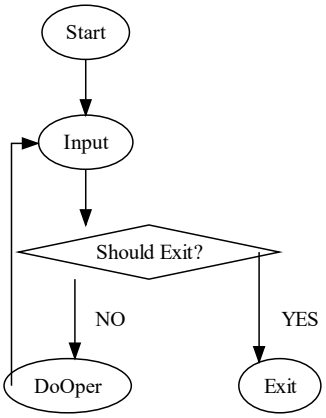
```
    char name[BOOK_NAME_LENGTH]; // Book name
    char isbn[BOOK_ISBN_LENGTH]; // Book ISBN number
    char author[BOOK_AUTHOR_LENGTH]; // Book Author
    long long addAt; // When the book was added to the library
    long long updatedAt; // When the book was last updated
} Book;

// BookNode in BookList
typedef struct _BookNode {
    Book item; // The book in this node
    struct _BookNode* next;
} BookNode;

// BookList (a List)
typedef struct {
    BookNode* head; // Head pointer of this List
} BookList;

typedef struct {
    char name[LIBRARY_NAME_LENGTH];
    BookList bookList;
} Library;
```

（二）算法设计

经过小组讨论，我们分为三个部分完成算法设计。第一是界面设计，第二是命令设计，第三是辅助函数设计。其中，界面设计主要设计用户交互界面，处理程序的输入和输出。命令设计主要实现具体的功能，处理对数据结构的操作。辅助函数设计主要设计各种会被频繁使用的公共函数。下面我们将分类详细介绍各个部分的具体设计。

1. 界面设计

程序与人交互的地方就是程序的界面，本项目所要设计的图书管理系统需要进行多种操作，输入很多数据。因此，结合小组成员的代码能力，我们设计出了一种不断循环读取命令并处理的循环式交互界面。



在具体实现中，我们采用了 while 循环语句加上 switch 语句的方式来实现，通过 scanf 语句实现输入单个字母的命令，switch 语句判断该执行哪些命令。具体代码实现如下：

```
1. char command;
2. 
3.  int shouldExit = 0;
4.  while(shouldExit == 0){
5.   printf("========== Book Management System ==========\n");
6.   printf(" - [l] Show books in current library\n");
7.   printf(" - [a] Add a new book\n");
8.   printf(" - [d] Delete a book from the library.\n");
9.   printf(" - [c] Change book data by id\n");
10.  printf(" - [q] Query book by its name\n");
11.  printf(" - [i] Query book by its ISBN\n");
12.  printf(" - [s] Save book data to disk\n");
13.  printf(" - [L] Load book data from disk\n");
14.  printf("============================================\n");
15.  printf(" - [v] Show version information\n");
16.  printf(" - [x] Exit this system\n");
17.  printf("============================================\n");
18.  printf(" + Current Library: %s\nPress C to change the nam
   e.\n\n", library->name);
19. 
20.  fflush(stdin);
21.  switch(command){
22.   // Command Operations...
23.  }
24.  if(shouldExit == 0) pressEnterToContinue();
25. }
```

2. 命令设计

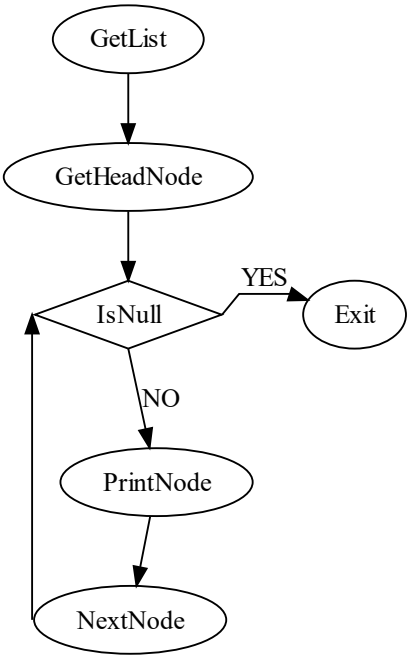
在命令设计中，小组成员综合考虑了项目要求和实际编码能力，提出了以下几个命令：

- 列出当前存在的图书
- 增加图书
- 删除图书
- 更改图书数据
- 通过图书名称查询图书
- 通过 ISBN 号查询图书
- 从文件中加载图书馆
- 将图书馆保存到文件中

在实际程序设计中，我们在 main 函数中存储一个当前 Library 的指针，把每个功能设计成一个或多个独立的函数，在调用函数时将 Library 指针作为参数传入到函数中进行处理。接下来我将一一介绍每个函数的设计思路和具体实现。

列出当前存在的图书

该功能本质上就是遍历链表并输出每个链节的过程。流程图如下：



具体代码实现如下：

```
1. /*
2.  * void timestampToTime(long long, char*, char*, int)
3.  * - Convert timestamp to date string in the given format
4.  */
5. void timeStampToTime(long long timestamp, char* dest, char*
    format, int sizeofDest){
6.  struct tm tm;
7.  time_t tick = (time_t) timestamp;
8.  tm = *localtime(&tick);
9.  strftime(dest, sizeofDest, format, &tm);
10.}
11.
12./*
13. * void printBook(Book*)
14. * - print book on stdout
15. */
16.void printBook(Book* book){
17. char addTime[20], updatedTime[20];
18. timeStampToTime(book->addAt, addTime, "%Y-%m-%d %H:%M:%S",
    sizeof(addTime));
19. timeStampToTime(book->updatedAt, updatedTime, "%Y-%m-%d %H
   :%M:%S", sizeof(updatedTime));
20. printf("%-*s|%-*s|%-*s|%-20s|%-20s\n", BOOK_NAME_LENGTH-
   1, book->name, BOOK_AUTHOR_LENGTH-
```

```
    1, book->author, BOOK_ISBN_LENGTH-
    1, book->isbn, addTime, updatedTime);
}

#define PRINT_HEAD printf("==============================
    ============================================================
    =============\n");\
 printf("%-3s %-*s %-*s %-*s %-20s %-
    20s\n","ID", BOOK_NAME_LENGTH - 1, "BOOK NAME", BOOK_AUTHOR
    _LENGTH - 1, "AUTHOR", BOOK_ISBN_LENGTH - 1, "ISBN", "ADD T
    IME", "UPDATE TIME");\
 printf("++++++++++++++++++++++++++++++++++++++++++++++++++++++
    ++++++++++++++++++++++++++++++++++++++++++++++++++++++\n");
#define PRINT_TAIL printf("==============================
    ============================================================
    =============\n");
/*
 * void printBookList(Library*)
 * - Print a table of books
 */
void printBookList(Library* library){
 int sum=1;
 BookNode* node = library->bookList.head;
 PRINT_HEAD
 while(node != NULL){
  printf("%-3d ", sum);
  printBook(&node->item);
  node = node->next;
  sum++;
 }
 PRINT_TAIL
}
```

我们将这段代码分成了三个小函数，第一个函数 timestampToTime 是用来将时间戳转换为人类可读的日期字符串，第二个函数 printBook 是用来打印单个书籍信息，在这里使用了 printf 格式化输出，让输出更加美观，第三个函数 printBookList 是遍历图书馆中的书籍并打印的操作，其逻辑就是上面的思维导图的逻辑。

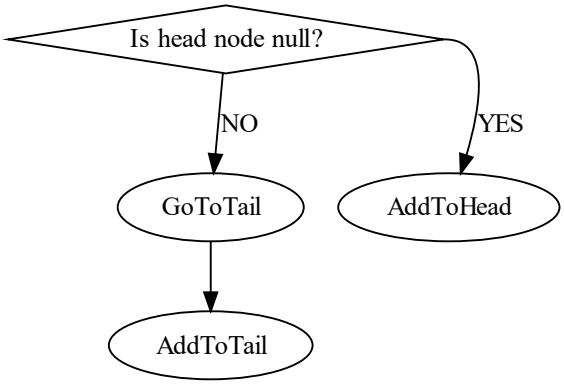
我们还定义了两个宏，即 PRINT_HEAD 和 PRINT_TAIL，它们负责打印表头和表尾。将这两个功能独立出来，有利于重复使用和减少冗余代码。

以下是这个功能的运行截图：

```
==========================================================================================================
ID  BOOK NAME           AUTHOR              ISBN             ADD TIME            UPDATE TIME
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1   Ideological Morality|Liu Chengyin       |978-7-04-056621-5|2022-11-28 16:57:00 |2022-11-28 16:57:00
2   FastReading         |Cao Fenglong       |978-7-100-17467-1|2022-11-28 16:57:22 |2022-11-28 16:57:22
3   Further Mathematics |Tong Ji            |978-7-04-039662-1|2022-11-28 16:57:28 |2022-11-28 16:57:28
4   Engineering         |Liu Yongshan       |978-7-5761-0021-1|2022-11-28 16:57:34 |2022-11-28 16:57:34
==========================================================================================================
```

增加图书

该功能的流程图如下：



实现代码如下：

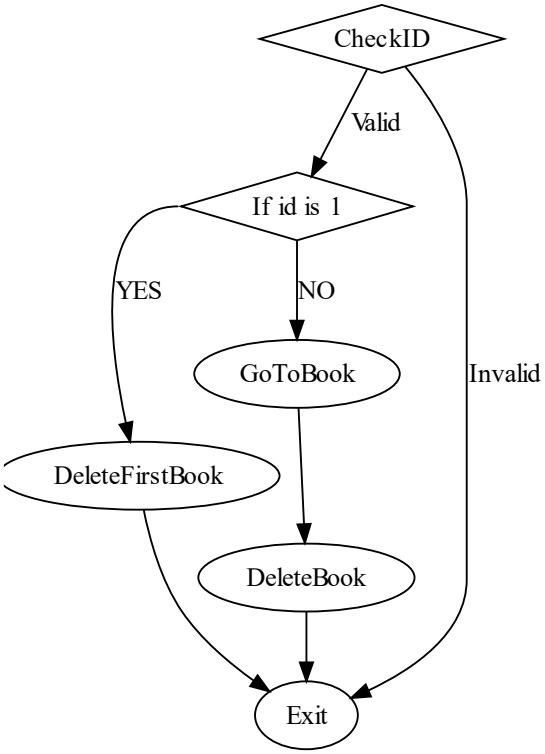
```
1. /*
2.  * void addBook(Library*, Book*)
3.  * - Add book into Library
4.  */
5. void addBook(Library* library, Book* book){
6.  BookNode* node = library->bookList.head;
7.  if(library->bookList.head == NULL){ // When the head is NULL, add book as head.
8.   library->bookList.head = (BookNode*) malloc(sizeof(BookNode)); // Allocate memory for BookNode
9.   library->bookList.head->item = *book;
10.  library->bookList.head->next = NULL; // Initalize the next property with NULL. That ensures no error will occurred when looping the list
11. }else{
12.  BookNode* p;
13.  for(; node!=NULL; p=node, node=node->next); // Go to the last node of the list
14.  node = (BookNode*) malloc(sizeof(BookNode)); // Allocate memory for BookNode
15.  node->next = NULL; // Initalize the next property with NULL. That ensures no error will occurred when looping the list
16.  p->next = node; // Append the BookNode in the end of the list
17.  node->item = *book;
18. }
19.}
```

在这个功能中，我们把情况分为两种来处理，一种是链表头为空，另外一种是链表头不为空。做这样的情况分类是因为，插入链表尾部需要上一个节点，而这两种情况中一个没有

上一个节点，另一个有上一个节点。此外，在创建新节点的过程中我们使用了动态内存分配功能，分配了一个新的内存空间，避免因调用完成后新节点变量超出作用域而被自动释放掉。

删除图书

此功能也是一个遍历链表的操作，与增加图书的操作刚好相反。流程图和代码如下：



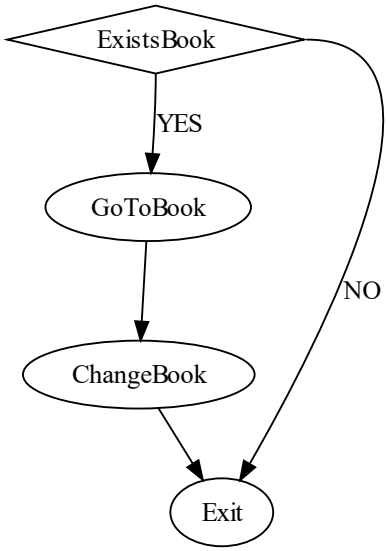

```
/* int deleteBook(Library*, int)
 * - Delete a book from the library
 *   Return the id of the deleted book
 *   Return -1 if the book isnot found
 */
int deleteBook(Library* library, unsigned int index){
 BookNode* node = library->bookList.head, *p=node;
 if(node == NULL) return -1;
 if(index == 1){
  p = library->bookList.head;
  library->bookList.head = library->bookList.head->next;
  free(p);
  return index;
 }
 int now = 1;
 while(now != index){
  p = node;
  node = node->next;
  if(node == NULL) return -1;
  now++;
```

```
21. }
22. p->next = node->next;
23. free(node);
24. return index;
25.}
```

在删除函数中，我们使用索引 ID 来指示将要被删除的书。在删除函数的最后，我们使用了 free 函数来释放不再被使用的内存空间，避免内存泄露。

更改图书数据

此功能主要涉及到对链表中链节内容的修改，也是对链表的遍历操作。流程图和实现代码如下：




```
1. /*
2.  * int changeBook(Library*, Book*, int)
3.  * - change the book by id
4.  *   return the id of the changed book
5.  *   return -1 if the id is invaild
6.  */
7. int changeBook(Library* library, Book* book, int id){
8.  if(existsBook(library, id) == -1) return -1;
9.  int sum = 1;
10. BookNode* node = library->bookList.head;
11. while(sum!=id){
12.  node = node->next;
13.  sum++;
14. }
15. long long addTimestamp = node->item.addAt;
16. node->item = *book;
17. node->item.updatedAt = get_timestamp();
18. node->item.addAt = addTimestamp;
19. return id;
```

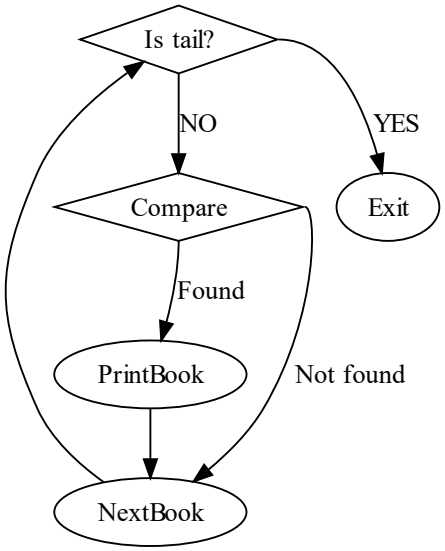
```
20.}
```

在此函数中我们使用了帮助函数 existsBook。该函数的主要功能是检查给定 ID 的书籍是否存在。该函数的源代码如下：

```
1. /*
2.  * int existsBook(Library*, int)
3.  * - Return the id of the book. If not found, return -1
4.  */
5. int existsBook(Library* library, int id){
6.  if(id < 1) return -1;
7.  BookNode* node = library->bookList.head;
8.  int sum = 1;
9.  while(sum != id){
10.  if(node == NULL) return -1;
11.  node = node->next;
12.  sum++;
13. }
14. return sum;
15.}
```

通过图书名称查询图书

本功能涉及到遍历链表，并使用字符串库中的 strstr 函数进行字符串匹配，最终输出所有匹配到的书籍。流程图和实现代码如下：




```
1. /*
2.  * Book* getBookByName(Library* char*)
3.  * - Get book by given book name
4.  */
5. Book* getBookByName(Library* library, char* name){
6.  BookNode* node = library->bookList.head;
7.  if(node == NULL) return NULL;
8.  Book* book = NULL;
```

```
9.  PRINT_HEAD
10. while(node != NULL){
11.  if(strstr(node->item.name, name) != NULL){
12.   book = &node->item;
13.   printf("%-3d ", getIDByBook(library, book));
14.   printBook(book);
15.  }
16.  node = node->next;
17. }
18. PRINT_TAIL
19. return book;
20.}
```

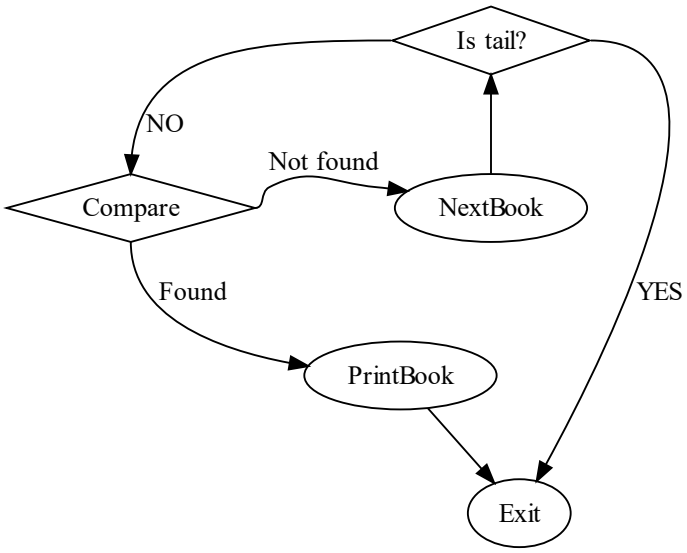
在代码中，我们采用 strstr 函数来查找目标字符串在书名中的位置，如果存在就输出书籍信息，不存在就切换到下一本书，直到到达链表末端。该功能的运行截图如下：

```
Please input book name:g
=============================================================================================================
ID  BOOK NAME           AUTHOR              ISBN             ADD TIME            UPDATE TIME
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1   Ideological Morality|Liu Chengyin       |978-7-04-056621-5|2022-11-28 16:57:00 |2022-11-28 16:57:00
2   FastReading         |Cao Fenglong       |978-7-100-17467-1|2022-11-28 16:57:22 |2022-11-28 16:57:22
4   Engineering         |Liu Yongshan       |978-7-5761-0021-1|2022-11-28 16:57:34 |2022-11-28 16:57:34
=============================================================================================================
```

通过 ISBN 号查找图书

该功能与上一个功能“根据书名查找图书”基本相同，不一致的地方在于该功能使用 strcmp 进行字符串的比较，只存在相等和不相等两种情况，并且由于 ISBN 号的唯一性，该函数只会输出一本图书的信息，而不会输出多本图书。

流程图和实现代码如下：




```
1. /*
2.  * Book* getBookByISBN(Library*, char*)
3.  * - Get book by given ISBN code
4.  */
5. Book* getBookByISBN(Library* library, char* isbn){
```

```
6.   BookNode* node = library->bookList.head;
7.   if(node == NULL) return NULL;
8.   Book* book = NULL;
9.   while(node != NULL){
10.   if(strcmp(node->item.isbn, isbn) == 0){
11.    book = &(node->item);
12.    break;
13.   }
14.   node = node->next;
15.  }
16.  return book;
17.}
```

该功能的运行截图如下：

从文件中加载图书馆

图书馆的持久化是本项目的亮点之一。下面我将从文件结构和算法两方面来陈述持久化的方法和操作。

文件结构

用来保存图书馆的文件结构可以用以下结构体表示：

```
1. #pragma pack(1)
2.
3. typedef struct {
4.     char name[B];
5.     char author[A];
6.     char isbn[I];
7.     long long addAt;
8.     long long updatedAt;
9. } Book;
10.
11.typedef struct {
12.    int B;
13.    int A;
14.    int I;
15.    int L;
16.    char library_name[L];
17.    int n;
18.    Book bookList[n];
19.} DataFile;
```

从文件头开始：

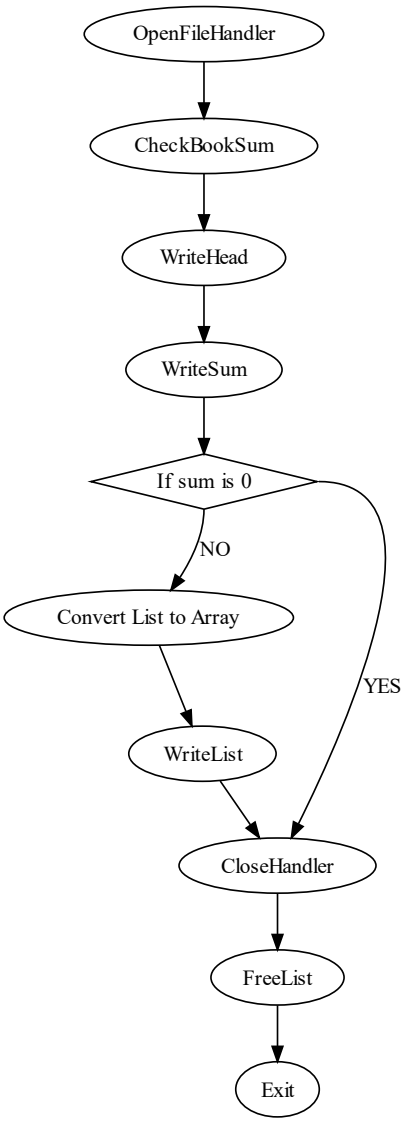
1. 4 个 int 大小的数据，分别是 图书名称字段长度 B 图书作者字段长度 A 图书 ISBN 字段长度 I 图书馆名称字段长度 L

2. 一个长度为 L 的以\0 结尾的字符串，为图书馆名称
3. 一个 int 大小的数据，为书籍数量 n
4. n 个大小为(B+A+I+2*long long)的结构体

以上结构在文件中均按 1 字节对齐，即成员之间没有空隙。

保存算法

保存算法的流程图如下：



代码如下：

```
/*
 * void saveToFile(Library*, char*)
 * - save Library* data to given destination
 *
 *   FILE STRUCTURE
 *   + BOOK_NAME_LENGTH BOOK_AUTHOR_LENGTH BOOK_ISBN_LENGTH
LIBRARY_NAME_LENGTH (4*sizeof(int))Byte
```

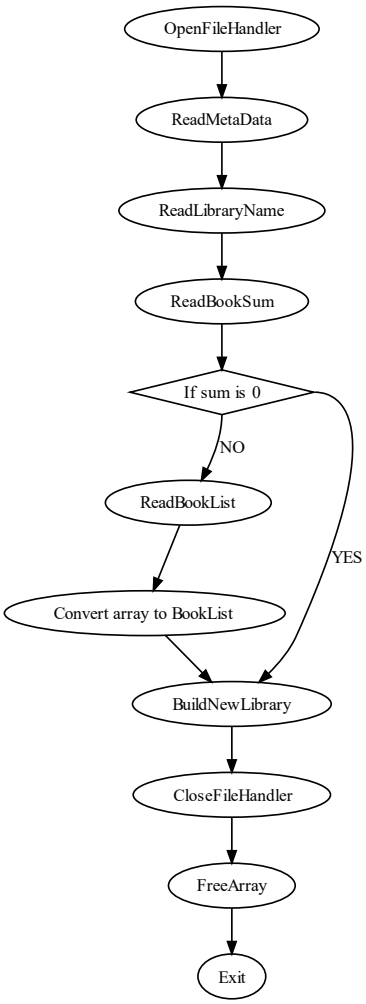
```
 *    + Library Name (20Byte)
 *    + Book numbers (sizeof(int)Byte)
 *    + n*Book (n*sizeof(Book)Byte)
 */
int saveToFile(Library* library, char* dest){
 FILE* f = fopen(dest, "wb+");
 if(f == NULL) return 0;
 BookNode* node = library->bookList.head;
 int sum = 0;
 while(node != NULL){
  sum++;
  node = node->next;
 }
 if(sum == 0){
  int head[] = {BOOK_NAME_LENGTH, BOOK_AUTHOR_LENGTH, BOOK_
ISBN_LENGTH, LIBRARY_NAME_LENGTH};
  fwrite(head, sizeof(int), 4, f);
  fwrite(library->name, sizeof(char), LIBRARY_NAME_LENGTH,
f);
  fwrite(&sum, sizeof(int), 1, f);
  fclose(f);
  return 1;
 }
 node = library->bookList.head;
 Book* bookList = (Book*) calloc(sum, sizeof(Book));
 for(int i=0; i<sum; i++){
  bookList[i] = node->item;
  node = node->next;
 }

 int head[] = {BOOK_NAME_LENGTH, BOOK_AUTHOR_LENGTH, BOOK_I
SBN_LENGTH, LIBRARY_NAME_LENGTH};
 fwrite(head, sizeof(int), 4, f);
 fwrite(library->name, sizeof(library->name), 1, f);
 fwrite(&sum, sizeof(int), 1, f);
 //fwrite(bookList, sizeof(Book), sum, f);
 for(int i=0; i<sum; i++){
  fwrite(bookList[i].name, sizeof(char), BOOK_NAME_LENGTH,
f);
  fwrite(bookList[i].author, sizeof(char), BOOK_AUTHOR_LENG
TH, f);
  fwrite(bookList[i].isbn, sizeof(char), BOOK_ISBN_LENGTH,
f);
  fwrite(&bookList[i].addAt, sizeof(long long), 1, f);
```

```
45.  fwrite(&bookList[i].updatedAt, sizeof(long long), 1, f);
46. }
47. fclose(f);
48. free(bookList);
49. return 1;
50.}
```

在这个函数中，综合利用了 C 语言标准库中的文件 IO 相关 API 以及动态内存分配相关 API，在操作过程中，将链表转化为动态数组，方便后续的写入操作，在写入操作完成后及时释放动态数组，防止内存泄露的发生。

读取算法

读取算法的流程图如下：



具体实现代码如下：

```
1. Library* loadFromFile(char* dest){
2.  FILE* f = fopen(dest, "r");
3.  if(f == NULL) return NULL;
4.  Library* library = (Library*) malloc(sizeof(Library));
5.  library->bookList.head = NULL;
6.  int head[4];
```

```
7.  fread(head, sizeof(head), 1, f);
8.  char* libraryString = (char*) malloc(sizeof(char)*head[3])
    ;
9.  fread(libraryString, sizeof(char), head[3], f);
10. strfit(library->name, libraryString, LIBRARY_NAME_LENGTH);
11. free(libraryString);
12. int sum;
13. fread(&sum, sizeof(int), 1, f);
14. if(sum == 0) return library;
15. Book* bookList = (Book*) malloc(sizeof(Book)*sum);
16. for(int i=0; i<sum; i++){
17.  char* t;
18.  t = (char*) malloc(sizeof(char*)*head[0]);
19.  fread(t, sizeof(char), head[0], f);
20.  strfit(bookList[i].name, t, BOOK_NAME_LENGTH);
21.  free(t);
22.  t = (char*) malloc(sizeof(char*)*head[1]);
23.  fread(t, sizeof(char), head[1], f);
24.  strfit(bookList[i].author, t, BOOK_AUTHOR_LENGTH);
25.  free(t);
26.  t = (char*) malloc(sizeof(char*)*head[2]);
27.  fread(t, sizeof(char), head[2], f);
28.  strfit(bookList[i].isbn, t, BOOK_ISBN_LENGTH);
29.  free(t);
30.  fread(&bookList[i].addAt, sizeof(long long), 1, f);
31.  fread(&bookList[i].updatedAt, sizeof(long long), 1, f);
32. }
33. fclose(f);
34. library->bookList.head = (BookNode*) malloc(sizeof(BookNod
    e));
35. library->bookList.head->next = NULL;
36. library->bookList.head->item = bookList[0];
37. BookNode *p=library->bookList.head;
38. for(int i=1; i<sum; i++, p=p->next){
39.  p->next = (BookNode*) malloc(sizeof(BookNode));
40.  p->next->item = bookList[i];
41.  p->next->next = NULL;
42. }
43. free(bookList);
44. return library;
45.}
```

与保存函数一样，在本函数中综合运用了文件 IO 和动态内存分配等功能，将文件中的数据一步步读取到内存中，最后构造出一个 Library 指针并返回。在主函数中调用完成本函数之后，保存的 Library 指针将会被替换为本函数返回的指针，原来的指针则会被释放，避

免内存泄露。

以上是对每个具体的命令的设计思路和具体实现的介绍，有了以上功能代码的支持，再编写一系列胶水代码将这些功能代码和交互界面代码耦合起来，就构成了一个功能完备的图书管理系统。

3. 辅助函数设计

在本系统的编码过程中，有时会频繁用到某一段代码，为了简化这些代码，减少代码的冗余，我们把这些代码段抽出来，使其形成一个独立的函数，方便调用。下面简要介绍几个辅助函数：

输入图书函数

该函数封装了从用户输入生成 Book 结构体的过程。值得一提的是，在此处我们可以通过宏 TP_FORCE_RIGHT_ISBN 来决定是否启用强制 ISBN 校验的功能。如果启用该功能，在输入错误的 ISBN 时，程序会拒绝录入图书。如果未启用该功能，程序将只显示一个警告，但依然录入该图书。

函数源代码如下：

```
/*
 * int inputBook(Book*)
 * input the book data
 */
int inputBook(Book* b){
 printf(" - Please input book name (in %d characters): ", BOOK_NAME_LENGTH-1);
 stdinRead(b->name, BOOK_NAME_LENGTH);
 printf(" - Please input the author of this book: ");
 stdinRead(b->author, BOOK_AUTHOR_LENGTH);
 printf(" - Please input the ISBN code of this book: ");
 stdinRead(b->isbn, BOOK_ISBN_LENGTH);
#ifndef TP_FORCE_RIGHT_ISBN
 if(!checkISBN(b->isbn)) printf("Warning: ISBN invalid!\n");
#else
 if(!checkISBN(b->isbn)){
  printf("Error: ISBN invalid!\n");
  return 0;
 }
#endif
 b->addAt = get_timestamp();
 b->updatedAt = get_timestamp();
 return 1;
}
```

暂停函数

该函数主要代替了 Windows 系统中的 pause 命令的功能。通过用该函数替换 pause 命

令，可以提高程序的可移植性，减少对特定系统的依赖。

```
/*
 * void pressEnterToContinue()
 * - Generate a message
 */
void pressEnterToContinue(){
 printf("Press ENTER to continue.");
 char a[10];
 stdinRead(a, 10);
}
```

安全的输入函数

该函数主要封装了从标准输入读取指定长度的字符串的功能。避免了使用 scanf 和 gets 时可能造成的缓冲区溢出。本程序没有使用 VC++内置的 scanf_s 函数，主要因为 scanf 函数以空格和换行为分隔符。

```
/*
 * int stdinRead(char*, int);
 * - Read the stdin with given length
 */
void stdinRead(char* data, int size){
 fgets(data, size, stdin);
 char* p;
 while(p=strchr(data, '\n'), p!=NULL) *p = '\0';
 fflush(stdin);
}
```

该图书管理系统的完整代码详见附录 B。

心得及体会

本次三级项目为编写一个小型管理系统，我们小组的选题为图书管理系统。做一个三级项目对于刚刚进入大学学习的我们大一新生来说极具挑战性。但是我们小组的每个成员都没有畏惧，每个人都迎难而上，永不言弃。每天我们都在宿舍里一起讨论项目的进展和实现细节，不时提出一些全新的思路和方法，找出项目中的亮点和不足。

在本次项目中，我创新性地提出了使用文件 IO 来进行保存和读取图书馆的功能，然而，使用文件 IO 涉及到了内存和文件系统的细节，难度比较大，但是我们没有放弃，通过广泛的查找资料，查看网络上的代码和教程，最终攻克了文件 IO 和动态内存分配的问题，实现了这个功能。

在实现代码的过程中，难免会遇到出人意料的 Bug，在解决这些 Bug 的过程中，我们小组成员学习了如何使用 GDB 进行动态调试，如何在代码中设置断点，如何使用步过和步进等调试功能……我们相信，这些技能都将为我们将来的学习提供帮助。

在处理标准输入的过程中，我们遇到了 scanf 和 gets 不能够指定字符串长度的问题，如果不能妥善的处理好这个问题，将导致缓冲区溢出的错误。为了解决这一问题，我们研究了 C 语言库中关于标准输入和输出的函数，最终选定了 fgets 作为安全的 scanf 和 gets 的替代，同时我们还学习到了文件操作有关的知识，这帮助了我们更深入地了解 C 语言和 C 标准库。

在项目进行的过程中，每个人都展现了对项目的责任感和设计项目的积极性。最后完成这个项目之后，每个人都感到非常的自豪，因为我们通过自己的努力完成一件事情，证明了自己的能力。

附录

A 参考文献

ISBN-13 校验算法的原理 https://zhuanlan.zhihu.com/p/342937543

C 参考手册 https://zh.cppreference.com/w/c

动态内存管理 https://zh.cppreference.com/w/c/memory

空终止字节字符串 https://zh.cppreference.com/w/c/string/byte

文件输入/输出 https://zh.cppreference.com/w/c/io

日期和时间工具 https://zh.cppreference.com/w/c/chrono

B 完整程序源代码

```
1. /*
2.  * The EndOfCourse project for C Program Language in the YS
   U
3.  *
4.  * Date: 2022/11/22
5.  * Author: SkyRain PYH NXC
6.  * Instruction Teacher: NJC
7.  *
8.  * Name: Book Management System
9.  */
10.
11.#include <stdio.h>
12.#include <stdlib.h>
13.#include <string.h>
14.#include <time.h>
15.#include <errno.h>
16.
17.
18.#define BOOK_NAME_LENGTH 21
19.#define BOOK_ISBN_LENGTH 18
20.#define BOOK_AUTHOR_LENGTH 21
21.#define LIBRARY_NAME_LENGTH 20
22.
23.// Book Item
24.typedef struct {
25. char name[BOOK_NAME_LENGTH]; // Book name
26. char isbn[BOOK_ISBN_LENGTH]; // Book ISBN number
27. char author[BOOK_AUTHOR_LENGTH]; // Book Author
28. long long addAt; // When the book was added to the library
29. long long updatedAt; // When the book was last updated
30.} Book;
31.
32.// BookNode in BookList
33.typedef struct _BookNode {
34. Book item; // The book in this node
35. struct _BookNode* next;
36.} BookNode;
37.
38.// BookList (a List)
39.typedef struct {
40. BookNode* head; // Head pointer of this List
41.} BookList;
```

```
42.
43.typedef struct {
44. char name[LIBRARY_NAME_LENGTH];
45. BookList bookList;
46.} Library;
47.
48.
49./*
50. * Library* createLibrary(char*)
51. * - create a Library object in the given name
52. *   This function will check the length of the name
53. * - Return allocated Library*
54. */
55.Library* createLibrary(char* name){
56. Library* library; // Define the pointer that is to be retu
    rnd
57. library = (Library*) malloc(sizeof(Library)); // Dynamicly
     allocate memory for Library
58. memcpy(library->name, name, sizeof(char) * (strlen(name) >
     LIBRARY_NAME_LENGTH ? LIBRARY_NAME_LENGTH : strlen(name) )
    ); // String length limit for the name property
59. library->bookList.head = NULL; // Initalize the head prope
    rty with NULL. That ensures no error will occurred when loo
    ping the list.
60. return library;
61.}
62.
63./*
64. * void freeLibrary(Library*)
65. * - free library
66. */
67.void freeLibrary(Library* library){
68. BookNode *node = library->bookList.head, *n;
69. while(node!=NULL){
70.  n = node->next;
71.  free(node);
72.  node = n;
73. }
74. free(library);
75.}
76.
77./*
78. * void addBook(Library*, Book*)
79. * - Add book into Library
```

```
80. */
81.void addBook(Library* library, Book* book){
82. BookNode* node = library->bookList.head;
83. if(library->bookList.head == NULL){ // When the head is NU
    LL, add book as head.
84.  library->bookList.head = (BookNode*) malloc(sizeof(BookNo
    de)); // Allocate memory for BookNode
85.  library->bookList.head->item = *book;
86.  library->bookList.head->next = NULL; // Initalize the nex
    t property with NULL. That ensures no error will occurred w
    hen looping the list
87. }else{
88.  BookNode* p;
89.  for(; node!=NULL; p=node, node=node->next); // Go to the
    last node of the list
90.  node = (BookNode*) malloc(sizeof(BookNode)); // Allocate
    memory for BookNode
91.  node->next = NULL; // Initalize the next property with NU
    LL. That ensures no error will occurred when looping the li
    st
92.  p->next = node; // Append the BookNode in the end of the
    list
93.  node->item = *book;
94. }
95.}
96.
97./*
98. * void saveToFile(Library*, char*)
99. * - save Library* data to given destination
100.  *
101.  *   FILE STRUCTURE
102.  *   + BOOK_NAME_LENGTH BOOK_AUTHOR_LENGTH BOOK_ISBN_LENG
    TH LIBRARY_NAME_LENGTH (4*sizeof(int))Byte
103.  *   + Library Name (20Byte)
104.  *   + Book numbers (sizeof(int)Byte)
105.  *   + n*Book (n*sizeof(Book)Byte)
106.  */
107. int saveToFile(Library* library, char* dest){
108.  FILE* f = fopen(dest, "wb+"); // Open file handler
109.  if(f == NULL) return 0; // Error process
110.  BookNode* node = library->bookList.head;
111.  int sum = 0;
112.  // Get book number
113.  while(node != NULL){
```

```
114.     sum++;
115.     node = node->next;
116.   }
117.   if(sum == 0){ // Book number == 0
118.     int head[] = {BOOK_NAME_LENGTH, BOOK_AUTHOR_LENGTH, BOO
   K_ISBN_LENGTH, LIBRARY_NAME_LENGTH};
119.     fwrite(head, sizeof(int), 4, f); // Write Meta
120.     fwrite(library->name, sizeof(char), LIBRARY_NAME_LENGTH
   , f); // Write Library name
121.     fwrite(&sum, sizeof(int), 1, f); // Write Book number
122.     fclose(f);// Close handler
123.     return 1; // Return
124.   }
125.   node = library->bookList.head;
126.   Book* bookList = (Book*) calloc(sum, sizeof(Book)); // A
   lloc memory
127.   for(int i=0; i<sum; i++){
128.     bookList[i] = node->item;
129.     node = node->next;
130.   }
131.
132.   int head[] = {BOOK_NAME_LENGTH, BOOK_AUTHOR_LENGTH, BOOK
   _ISBN_LENGTH, LIBRARY_NAME_LENGTH};
133.   fwrite(head, sizeof(int), 4, f);
134.   fwrite(library->name, sizeof(library->name), 1, f);
135.   fwrite(&sum, sizeof(int), 1, f);
136.   //fwrite(bookList, sizeof(Book), sum, f);
137.   for(int i=0; i<sum; i++){ // Write book
138.     fwrite(bookList[i].name, sizeof(char), BOOK_NAME_LENGTH
   , f);
139.     fwrite(bookList[i].author, sizeof(char), BOOK_AUTHOR_LE
   NGTH, f);
140.     fwrite(bookList[i].isbn, sizeof(char), BOOK_ISBN_LENGTH
   , f);
141.     fwrite(&bookList[i].addAt, sizeof(long long), 1, f);
142.     fwrite(&bookList[i].updatedAt, sizeof(long long), 1, f)
   ;
143.   }
144.   fclose(f);
145.   free(bookList);
146.   return 1;
147.  }
148.
149.  /*
```

```
150.   * int checkISBN(char*)
151.   * - This function check if the ISBN is valid.
152.   *   Returns 1 if valid
153.   *   else returns 0 if  isbn string is too short or it is
    invalid
154.   */
155.  int checkISBN(char* isbnCode){
156.   if(strlen(isbnCode)<13) return 0;
157.   int sum=0, len=strlen(isbnCode);
158.   char c;
159.   for(int i=0,j=1; i<len-1; i++){
160.    if(isbnCode[i] >= '0' && isbnCode[i] <= '9'){
161.     if(j%2==1){
162.      sum+=isbnCode[i]-'0';
163.     }else{
164.      sum+=3*(isbnCode[i]-'0');
165.     }
166.     j++;
167.    }
168.   }
169.   c = isbnCode[len-1];
170.   sum=10-(sum%10);
171.   if(sum == 10) return c == 'X' || c == 'x';
172.   else return sum == c-'0';
173.  }
174.
175.  // Deprecated
176.  Library* loadFromFilev1(char* dest){
177.   FILE* f = fopen(dest, "r");
178.   if(f == NULL) return NULL;
179.   Library* library = malloc(sizeof(Library));
180.   library->bookList.head = NULL;
181.   int head[4];
182.   fread(head, sizeof(head), 1, f);
183.   fread(library->name, sizeof(char), LIBRARY_NAME_LENGTH,
   f);
184.   int sum;
185.   fread(&sum, sizeof(int), 1, f);
186.   if(sum == 0) return library;
187.   Book* bookList = (Book*) malloc(sizeof(Book)*sum);
188.   fread(bookList, sizeof(Book), sum, f);
189.   fclose(f);
190.   library->bookList.head = (BookNode*) malloc(sizeof(BookN
   ode));
```

```
191.   library->bookList.head->next = NULL;
192.   library->bookList.head->item = bookList[0];
193.   BookNode *p=library->bookList.head;
194.   for(int i=1; i<sum; i++, p=p->next){
195.    p->next = (BookNode*) malloc(sizeof(BookNode));
196.    p->next->item = bookList[i];
197.    p->next->next = NULL;
198.   }
199.   free(bookList);
200.   return library;
201.  }
202.
203.  void strfit(char* d, char* s, int size){
204.   // Fit string to given size
205.   if(strlen(s)+1 <= size){
206.    memcpy(d, s, sizeof(char)*strlen(s)+1);
207.   }else{
208.    memcpy(d, s, size);
209.    d[size-1] = '\0';
210.   }
211.  }
212.
213.  /*
214.   * Library* loadFromFile(char*)
215.   * - load Library from given path.
216.   *   return NULL if errors occurred
217.   */
218.  Library* loadFromFile(char* dest){
219.   FILE* f = fopen(dest, "r");
220.   if(f == NULL) return NULL;
221.   Library* library = (Library*) malloc(sizeof(Library));
222.   library->bookList.head = NULL;
223.   int head[4];
224.   fread(head, sizeof(head), 1, f);
225.   char* libraryString = (char*) malloc(sizeof(char)*head[3
    ]);
226.   fread(libraryString, sizeof(char), head[3], f);
227.   strfit(library->name, libraryString, LIBRARY_NAME_LENGTH
    );
228.   free(libraryString);
229.   int sum;
230.   fread(&sum, sizeof(int), 1, f);
231.   if(sum == 0) return library;
232.   Book* bookList = (Book*) malloc(sizeof(Book)*sum);
```

```
233.   for(int i=0; i<sum; i++){
234.    char* t;
235.    t = (char*) malloc(sizeof(char*)*head[0]);
236.    fread(t, sizeof(char), head[0], f);
237.    strfit(bookList[i].name, t, BOOK_NAME_LENGTH);
238.    free(t);
239.    t = (char*) malloc(sizeof(char*)*head[1]);
240.    fread(t, sizeof(char), head[1], f);
241.    strfit(bookList[i].author, t, BOOK_AUTHOR_LENGTH);
242.    free(t);
243.    t = (char*) malloc(sizeof(char*)*head[2]);
244.    fread(t, sizeof(char), head[2], f);
245.    strfit(bookList[i].isbn, t, BOOK_ISBN_LENGTH);
246.    free(t);
247.    fread(&bookList[i].addAt, sizeof(long long), 1, f);
248.    fread(&bookList[i].updatedAt, sizeof(long long), 1, f);
249.   }
250.   fclose(f);
251.   library->bookList.head = (BookNode*) malloc(sizeof(BookN
   ode));
252.   library->bookList.head->next = NULL;
253.   library->bookList.head->item = bookList[0];
254.   BookNode *p=library->bookList.head;
255.   for(int i=1; i<sum; i++, p=p->next){
256.    p->next = (BookNode*) malloc(sizeof(BookNode));
257.    p->next->item = bookList[i];
258.    p->next->next = NULL;
259.   }
260.   free(bookList);
261.   return library;
262.  }
263.
264.
265.  /* int deleteBook(Library*, int)
266.   * - Delete a book from the library
267.   *   Return the id of the deleted book
268.   *   Return -1 if the book isnot found
269.   */
270.  int deleteBook(Library* library, unsigned int index){
271.   BookNode* node = library->bookList.head, *p=node;
272.   if(node == NULL) return -1;
273.   if(index == 1){
274.    p = library->bookList.head;
275.    library->bookList.head = library->bookList.head->next;
```

```
276.     free(p);
277.     return index;
278.   }
279.   int now = 1;
280.   while(now != index){
281.     p = node;
282.     node = node->next;
283.     if(node == NULL) return -1;
284.     now++;
285.   }
286.   p->next = node->next;
287.   free(node);
288.   return index;
289. }
290. 
291. /*
292.  * long long get_timestamp()
293.  * - Get the timestamp now
294.  */
295. long long get_timestamp()
296. {
297.   time_t seconds = time(NULL); //The function time(NULL) r
    eturns the time since the Epoch (00:00:00 UTC, January 1, 1
    970), measured in seconds.
298.   return (long long)seconds;
299. }
300. 
301. /*
302.  * void timestampToTime(long long, char*, char*, int)
303.  * - Convert timestamp to date string in the given format
304.  */
305. void timeStampToTime(long long timestamp, char* dest, cha
    r* format, int sizeofDest){
306.   struct tm tm;
307.   time_t tick = (time_t) timestamp;
308.   tm = *localtime(&tick);
309.   strftime(dest, sizeofDest, format, &tm);
310. }
311. 
312. /*
313.  * void printBook(Book*)
314.  * - print book on stdout
315.  */
316. void printBook(Book* book){
```

```
317.   char addTime[20], updatedTime[20];
318.   timeStampToTime(book->addAt, addTime, "%Y-%m-%d %H:%M:%S
   ", sizeof(addTime));
319.   timeStampToTime(book->updatedAt, updatedTime, "%Y-%m-%d
   %H:%M:%S", sizeof(updatedTime));
320.   printf("%-*s|%-*s|%-*s|%-20s|%-20s\n", BOOK_NAME_LENGTH-
   1, book->name, BOOK_AUTHOR_LENGTH-
   1, book->author, BOOK_ISBN_LENGTH-
   1, book->isbn, addTime, updatedTime);
321.  }
322.
323.  #define PRINT_HEAD printf("==========================
   ============================================================
   ===============\n");\
324.   printf("%-3s %-*s %-*s %-*s %-20s %-
   20s\n","ID", BOOK_NAME_LENGTH - 1, "BOOK NAME", BOOK_AUTHOR
   _LENGTH - 1, "AUTHOR", BOOK_ISBN_LENGTH - 1, "ISBN", "ADD T
   IME", "UPDATE TIME");\
325.   printf("++++++++++++++++++++++++++++++++++++++++++++++++
   ++++++++++++++++++++++++++++++++++++++++++++++++++++++++\n"
   );
326.  #define PRINT_TAIL printf("==========================
   ============================================================
   ===============\n");
327.  /*
328.   * void printBookList(Library*)
329.   * - Print a table of books
330.   */
331.  void printBookList(Library* library){
332.   int sum=1;
333.   BookNode* node = library->bookList.head;
334.   PRINT_HEAD
335.   while(node != NULL){
336.    printf("%-3d ", sum);
337.    printBook(&node->item);
338.    node = node->next;
339.    sum++;
340.   }
341.   PRINT_TAIL
342.  }
343.
344.  /*
345.   * int getIDByBook(Library*, Book*)
346.   * - return the id
```

```
 */
int getIDByBook(Library* library, Book* book){
 int id = 1;
 BookNode* node = library->bookList.head;
 while(&node->item != book){
  node = node->next;
  if(node == NULL) return -1;
  id++;
 }
 return id;
}

/*
 * int existsBook(Library*, int)
 * - Return the id of the book. If not found, return -1
 */
int existsBook(Library* library, int id){
 if(id < 1) return -1;
 BookNode* node = library->bookList.head;
 int sum = 1;
 while(sum != id){
  if(node == NULL) return -1;
  node = node->next;
  sum++;
 }
 return sum;
}

/*
 * int changeBook(Library*, Book*, int)
 * - change the book by id
 *   return the id of the changed book
 *   return -1 if the id is invaild
 */
int changeBook(Library* library, Book* book, int id){
 if(existsBook(library, id) == -1) return -1;
 int sum = 1;
 BookNode* node = library->bookList.head;
 while(sum!=id){
  node = node->next;
  sum++;
 }
 long long addTimestamp = node->item.addAt;
 node->item = *book;
```

```
391.   node->item.updatedAt = get_timestamp();
392.   node->item.addAt = addTimestamp;
393.   return id;
394.  }
395.
396.  /*
397.   * int stdinRead(char*, int);
398.   * - Read the stdin with given length
399.   */
400.  void stdinRead(char* data, int size){
401.   fgets(data, size, stdin);
402.   char* p;
403.   while(p=strchr(data, '\n'), p!=NULL) *p = '\0';
404.   fflush(stdin);
405.  }
406.
407.  /*
408.   * int inputBook(Book*)
409.   * input the book data
410.   */
411.  int inputBook(Book* b){
412.   printf(" - Please input book name (in %d characters): ",
    BOOK_NAME_LENGTH-1);
413.   stdinRead(b->name, BOOK_NAME_LENGTH);
414.   printf(" - Please input the author of this book: ");
415.   stdinRead(b->author, BOOK_AUTHOR_LENGTH);
416.   printf(" - Please input the ISBN code of this book: ");
417.   stdinRead(b->isbn, BOOK_ISBN_LENGTH);
418.  #ifndef TP_FORCE_RIGHT_ISBN
419.   if(!checkISBN(b->isbn)) printf("Warning: ISBN invalid!\n
   ");
420.  #else
421.   if(!checkISBN(b->isbn)){
422.    printf("Error: ISBN invalid!\n");
423.    return 0;
424.   }
425.  #endif
426.   b->addAt = get_timestamp();
427.   b->updatedAt = get_timestamp();
428.   return 1;
429.  }
430.
431.  /*
432.   * Book* getBookByISBN(Library*, char*)
```

```
433.   * - Get book by given ISBN code
434.   */
435.  Book* getBookByISBN(Library* library, char* isbn){
436.   BookNode* node = library->bookList.head;
437.   if(node == NULL) return NULL;
438.   Book* book = NULL;
439.   while(node != NULL){
440.    if(strcmp(node->item.isbn, isbn) == 0){
441.     book = &(node->item);
442.     break;
443.    }
444.    node = node->next;
445.   }
446.   return book;
447.  }
448.
449.  /*
450.   * Book* getBookByName(Library* char*)
451.   * - Get book by given book name
452.   */
453.  Book* getBookByName(Library* library, char* name){
454.   BookNode* node = library->bookList.head;
455.   if(node == NULL) return NULL;
456.   Book* book = NULL;
457.   PRINT_HEAD
458.   while(node != NULL){
459.    if(strstr(node->item.name, name) != NULL){
460.     book = &node->item;
461.     printf("%-3d ", getIDByBook(library, book));
462.     printBook(book);
463.    }
464.    node = node->next;
465.   }
466.   PRINT_TAIL
467.   return book;
468.  }
469.
470.  /*
471.   * void pressEnterToContinue()
472.   * - Generate a message
473.   */
474.  void pressEnterToContinue(){
475.   printf("Press ENTER to continue.");
476.   char a[10];
```

```
477.   stdinRead(a, 10);
478.  }
479.
480.  /*
481.   * The enterance of this program
482.   */
483.  int main(){
484.   Library* library = createLibrary("YSULib");
485.   char command;
486.
487.   int shouldExit = 0;
488.   while(shouldExit == 0){
489.  #ifdef __WIN32__
490.    system("cls");
491.  #endif
492.    fflush(stdin);
493.    printf("========== Book Management System ==========\n"
   );
494.    printf(" - [l] Show books in current library\n");
495.    printf(" - [a] Add a new book\n");
496.    printf(" - [d] Delete a book from the library.\n");
497.    printf(" - [c] Change book data by id\n");
498.    printf(" - [q] Query book by its name\n");
499.    printf(" - [i] Query book by its ISBN\n");
500.    printf(" - [s] Save book data to disk\n");
501.    printf(" - [L] Load book data from disk\n");
502.    printf("============================================\n"
   );
503.    printf(" - [v] Show version information\n");
504.    printf(" - [x] Exit this system\n");
505.    printf("============================================\n"
   );
506.    printf(" + Current Library: %s\nPress C to change the n
   ame.\n\n", library->name);
507.
508.    while(scanf("%c", &command), command=='\n' || command==
   ' '); // Get rid of `pause`
509.    fflush(stdin);
510.    switch(command){
511.     case 'l':{
512.         printBookList(library);
513.         break;
514.        }
515.     case 'a':{
```

```
516.         Book b;
517.         printf("\n");
518.         if(inputBook(&b)){
519.         addBook(library, &b);
520.         }else{
521.          printf("Input invalid!\n");
522.         }
523.         break;
524.        }
525.
526.     case 'v':{
527.         printf("TP. The Final Version\n");
528.         break;
529.        }
530.     case 'x':{
531.         printf("Bye.\n");
532.         shouldExit = 1;
533.         break;
534.        }
535.     case 'L':{
536.         printf("Please input the path of the data file: ")
    ;
537.         char path[50];
538.         stdinRead(path, 50);
539.         Library* l = loadFromFile(path);
540.         if(l == NULL){
541.          printf("Path or file is invalid.\n");
542.         }else{
543.          freeLibrary(library);
544.          library = l;
545.          printf("Successfully loaded.\n");
546.          printBookList(library);
547.         }
548.         break;
549.        }
550.     case 'i':{
551.         printf("Please input book ISBN:");
552.         char isbn[BOOK_ISBN_LENGTH];
553.         stdinRead(isbn, BOOK_ISBN_LENGTH);
554.         Book* b = getBookByISBN(library, isbn);
555.         if(b == NULL) printf("Book does not exist!\n");
556.         else{
557.          PRINT_HEAD
558.          printf("%-3d ", getIDByBook(library, b));
```

```
559.          printBook(b);
560.          PRINT_TAIL
561.         }
562.         break;
563.        }
564.     case 'q':{
565.         printf("Please input book name:");
566.         char name[BOOK_NAME_LENGTH];
567.         stdinRead(name, BOOK_NAME_LENGTH);
568.         if(getBookByName(library, name) == NULL) printf("B
    ook does not exist!\n");
569.         break;
570.        }
571.     case 'd':{
572.         printBookList(library);
573.         printf("\n\nPlease input the book id you want to d
    elete: ");
574.         int index;
575.         scanf("%d", &index);
576.         fflush(stdin);
577.         if(deleteBook(library, index) == -1){
578.          printf("Invalid index is given!\n");
579.         }else{
580.          printf("Successfully deleted.\n");
581.         }
582.         break;
583.        }
584.     case 'c':{
585.         printBookList(library);
586.         printf("Please input the book id that you want to
    change: ");
587.         int id;
588.         scanf("%d", &id);
589.         fflush(stdin);
590.         if(existsBook(library, id) == -1){
591.          printf("Book id is invalid! \n");
592.          //pressEnterToContinue();
593.          break;
594.         }
595.         Book book;
596.         printf("\n");
597.         if(inputBook(&book)){
598.         changeBook(library, &book, id);
599.         }else{
```

```
600.           printf("Input invalid!\n");
601.          }
602.          break;
603.         }
604.
605.     case 's':{
606.          printf("Please input the path of the data file: ")
   ;
607.          char path[50];
608.          stdinRead(path, 50);
609.          if(saveToFile(library, path) == 0){
610.           printf("Failed to open file: Errno[%d] %s\n", err
   no, strerror(errno));
611.          }else{
612.           printf("Save success!\n");
613.          }
614.          break;
615.         }
616.     case 'C':{
617.          printf("Please input the name you like: ");
618.          stdinRead(library->name, LIBRARY_NAME_LENGTH);
619.          break;
620.         }
621.
622.     default:{
623.        printf("Please input correct command!\n");
624.       }
625.    }
626.    if(shouldExit == 0) pressEnterToContinue();
627.   }
628.   return 0;
629.  }
```