南京郵電大學

课程设计报告

（ 2017 / 2018 学年 第 二 学期）

题 目：图书管理系统

专业： 网络工程

学生姓名： 陈慧

班级学号： **B16070404**

指导教师： 朱海婷，张媛（研究生助教）

指导单位： 南京邮电大学

日期： **2018** 年 **7** 月 **2** 日

评分细则	评分项	成绩
	遵守机房规章制度（5 分）	
	上机时的表现（5 分）	
	学习态度（5 分）	
	程序准备情况（5 分）	
	程序设计能力（10 分）	
	团队合作精神（5 分）	
	课题功能实现情况（10 分）	
	算法设计合理性（10 分）	
	用户界面设计（10 分）	
	报告书写认真程度（5 分）	
	内容详实程度（10 分）	
	文字表达熟练程度（10 分）	
	回答问题准确度（10 分）	
简短评语	教师签名： 年月日	
评分等级		
备注	评分等级有五种：优秀、良好、中等、及格、不及格	

图书管理系统

一、课题内容和要求

1. 设计题目：图书管理系统

2. 设计目的

巩固树的相关知识(建立、插入、删除)，灵活运用 C 语言的三种基本程序控制结构和函数来实现系统。

3. 设计内容

图书管理的基本业务活动：对一本书的采编入库，清除库存，借阅和归还等。设计一个图书管理系统，将上述业务活动借助于计算机系统完成。

4. 设计要求

1) 每种书的登记内容至少包括书号、书名、作者现存量和总库量等五项。

2) 系统应实现的操作及其功能如下：

i. 采编入库：新购入一种书，经分类和确定书号之后登记到图书账目中去。

ii. 清除库存：某种书已无保留价值，将它从图书账目中注销。

iii. 借阅：如果一种书的现存量大于零，则借出一本，登记借阅者的图书证号和归还期限。

iv. 归还：注销对借阅者的登记，改变该书的现存量。

3) 上机能正常运行，并写出课程设计报告

5. 增加功能说明

为了让设计的图书管理系统更加人性化，功能更加齐全，我利用链表结构特意在系统中增加了两个功能：显示当前图书馆库存、按照书名、书号、作者对库存进行查询。

二、需求分析

在图书管理系统中，对管理员而言，他们有权限进行新书的采编入库，每当有新书入库，需要经分类和确定书号之后登记到图书账目中，每种书的登记内容包括书号、书名、作者、现存量和总库量五项，以方便对图书的管理和查询。

对于借阅者而言，他们可以很方便的通过我们的系统进行图书的借阅和归还。借阅时，如果一本书的现存量大于零，则借出一本，相应书目的存储量减一，否则输出提示信息。同时登记借阅者的图书证号和归还期限，以此保证借出图书的安全性。当借阅者归还后，注销对借阅者的登记，相应书目的存储量加一。

并且图书管理员可以定期或不定期对图书信息进行入库、修改、删除等操作。对于已无保留价值的书，可以实现将该书从图书管理库存中彻底删除的功能，时刻保持馆藏图书的新鲜力。

在产品性能方面，需要通过 c 语言的编程，让界面美观，用户体验舒适，系统运营流程准确流畅。具体功能完成流程图，如图 1 所示

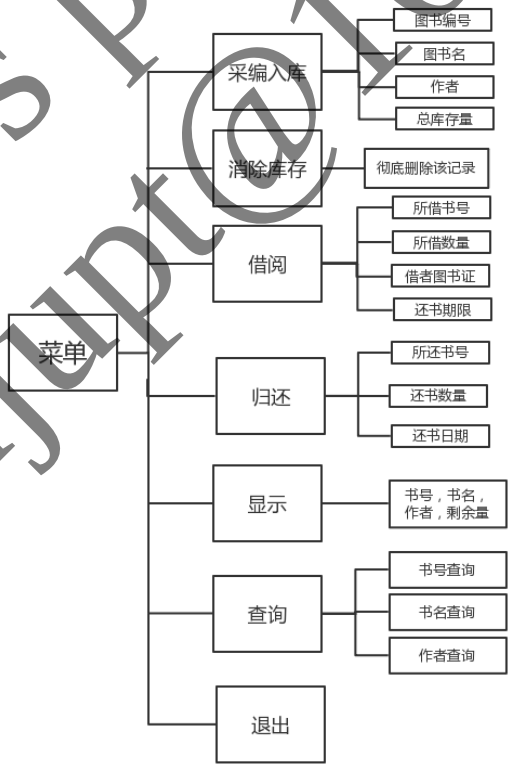


图 1 图书管理系统总流程图

三、概要设计

在该图书管理系统中，为了方便图书管理员对图书的管理，需要设计一个书本的数据结构，来记录一种书的书号、书名、作者、现存量、总量等基本信息，对这些信息进行打包，命名为 b[N]，作为书的数据类型。具体数据结构如下：

```
typedef struct
{
    int num;                //书号
    char name[20];          //书名
    char writer[20];        //书作者
    int left;               //图书现存量
    int storage;            //该图书总量

}DataType;
```

我采用树形结构进行管理操作。在实际操作中，B 树查询快，增删结点相对于链表或者顺序表效率更好，因此用来存储大量图书信息十分合适。系统中 B 树的存储结构如下：

```
typedef struct node
{
    struct node *parent;        //指向父结点的指针
    int keynum;                 //结点关键值
    DataType key[M + 1];        //一个结点中最多存放的书的个数
    struct node *ptr[M + 1];    //指向孩子结点的指针
}node;
```

此外，为了记录借阅者的信息，还定义了一个借阅着的数据结构，记录其借阅证号和预计归还日期。对于每次查找的结果，为了方便后面的使用，设定一个数据结构来存放找到的结点的相关信息。查找结果的存储结构体如下：

```
typedef struct
{
    node *ptr;      //结点指针，指向查找结果的结点
    int pos;        //具体位置标示
    int find;       //书专用编号，判断该书是否录入，进而判断是否找到
}result;
```

对于实现题目中要求的各功能，在代码中设定函数声明如下：

```
void assist();
result SearchBT(node *root, int key); //搜索树，用于各函数中查找相应结点位置
void Insert(node *root, int pos, node *child, DataType key);    // 直接插入结点
void Split(node *p, int *key, int pos, node **child);    //分裂结点
```

```
void output(node *root, int floor);        //以凹入表的形式输出树
void assist(node *root);     //查询函数
void borrow(node *root);     //借书函数
void dele(node *root);     //消除库存
void welcome();     //欢迎函数，方便回到选择界面
void input();     //采编入库
```

四、详细设计

1. 采编入库

```
void input()     //采编入库
{
      int y;     //判断是否要继续输入
      printf("请依次输入图书信息:\n");          //打印提示信息
      printf("-----------------------------------------------------------------------\n");
      do{
            for(int i=amount;i<N;i++)
                {
                  printf("请输入图书编号:");
                  scanf("%d",&b[i].num);          //b[N]书的结构体数组
                  printf("请输入图书名:");
                  scanf("%s",b[i].name);
                  printf("请输入作者:");
                  scanf("%s",b[i].writer);
                  printf("请输入总库存量:");
                  scanf("%d",&b[i].storage);
                  b[i].left=b[i].storage;                //目前剩余量=总量
                  amount++;                              //储存总量+1
                  SearchInsert(&root,b[i]); //搜索并且插入结点，调用函数找到插入
结点的位置
                  printf("第%d 个信息已经输完是否继续?按任意数字键继续，按 0
结束\n",amount);
                  scanf("%d",&y);
                  break;
                }
      }while(y!=0);
}
```

图书管理员录入信息即将图书信息按照提示依次录入，写进之前定义的全局变量 b[N]中，之后的操作就可以直接使用 b[N]这个书本结构体变量中的信息。

对于搜索并且插入结点，调用函数找到插入结点的位置这个函数，我使用的是B 树中的插入操作。插入一个元素时，首先检查 B 树中是否存在，如果不存在，即在子结点处结束，然后在子结点中插入该新的元素。注意：如果子结点空间足够，需要向右移动该子结点中大于新插入关键字的元素，如果空间满了以致没有足够的空间去添加新的元素，则将该结点进行“分裂”，将一半数量的关键字元素分裂到新的其相邻右结点中，中间关键字元素上移到父结点中（当然，如果父结点空间满了，也同样需要“分裂”操作），而且当结点中关键元素向右移动了，相关的指针也需要向右移。如果在根结点插入新元素，空间满了，则进行分裂操作，这样原来的根结点中的中间关键字元素向上移动到新的根结点中，因此导致树的高度增加一层。具体分裂插入代码如下所示：

```
//插入结点的时候，先直接插入到叶结点，
//再判断是否要分裂结点
void InsertBT(node **T, DataType key, node *p, int pos)
{
    node *child = NULL;                    //建立新的结点指针，为后面的结点分裂做准备
    node *just;
    while (p)
    {
        if (p->keynum < M)                          //结点 p 中的书的种类〈M
      {
            Insert(p, pos, child, key);          //直接插入结点
            return;
      }
      else {                                  //超过该结点的最大承受书类型数目
      Split(p, &key, pos, &child);          //产生新的 key, child...
      just = p;
      p = p->parent;                                 //p 指向他的父结点
      if (p)
      {
            pos = 0;
            while (pos<p->keynum && p->key[pos+1].num<=key.num)
                  {
                        pos++;
                  }     / /符合一条树分支的限额设置，pos 相应加 1
      }
      }
}
      node *root = (node *)malloc(sizeof(node));
```

```
        root->keynum = 1;
        root->key[1] = key;
        root->ptr[0] = just;
        root->ptr[1] = child;
        root->parent = NULL;

        (*T)->parent = child->parent = root;
        *T = root;
}
    void Split(node *p, DataType *key, int pos, node **child) //分裂结点
    {
        DataType tempkey[M + 2];
        node *temptr[M + 2];
        int i, j;
        i = j = 0;
        for (i=0; i<=M+1; i++) {
            if (i == pos + 1) {
                tempkey[i] = *key;
                temptr[i] = *child;
            }
            else {
                tempkey[i] = p->key[j];
                temptr[i] = p->ptr[j++];
            }
        }                                       //将分裂结点之前的结点安置好
        *child = (node *)malloc(sizeof(node));      //为新的子结点分配空间
        (*child)->parent = p->parent;
        int mid = (M + 1) / 2 + 1;
        p->keynum = mid - 1;
        (*child)->keynum = M - (mid - 1);       //计算各分支结点的书类数
        for (i=0; i<=M+1; i++) {
            if (i < mid) {
                p->key[i] = tempkey[i];
                p->ptr[i] = temptr[i];
            }
            else if (i == mid) {
                (*child)->ptr[0] = temptr[i];
                *key = tempkey[i];
            }
            else {
                (*child)->key[i - mid] = tempkey[i];
                (*child)->ptr[i - mid] = temptr[i];
            }
        }
    }
```

2. 消除库存

在 B 树的删除结点操作中，首先查找 B 树中需删除的元素,如果该元素在 B 树中存在，返回 result 类型的结果，包括该结点的位置，指针，和是否找到的标志信息 find=0/1。若找到则将该元素在其结点中进行删除，如果删除该元素后，首先判断该元素是否有左右孩子结点，如果有，则上移孩子结点中的某相近元素(“左孩子最右边的节点”或“右孩子最左边的节点”)到父节点中；如果没有，则直接删除后。

实际操作的流程图如图 2 所示：

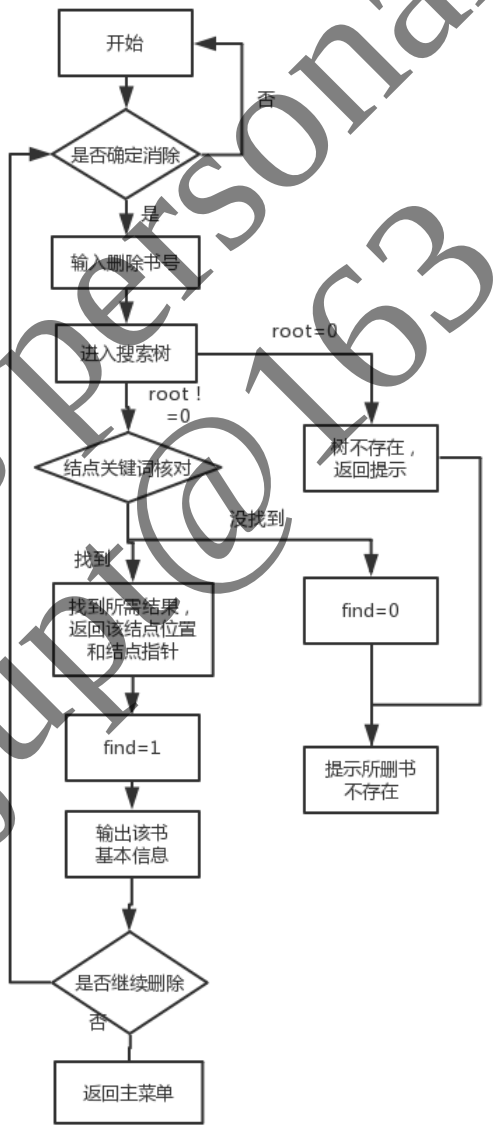


图 2 消除库存函数流程图

还书主要代码如下：

```
 printf("请输入你要删除的书的书号");
 scanf("%d",&num);
 d=SearchBT(root,num);//返回结果矩阵
 if(!d.find)    printf("你要找的书不存在!");
 else
{
 printf("书号        书名        作者        剩余量        总库存\n");
 printf("%d%7s%7s%8d%12d\n\n",d.ptr->key[d.pos].num,d.ptr->key[d.pos].na
 me,d.ptr->key[d.pos].writer,d.ptr->key[d.pos].left,d.ptr->key[d.pos].storage);
printf("确定要删除吗?请输入 yes 或者 no.\n");
scanf("%s",t);
if(!strcmp(t,"yes"))
{
d.ptr->key[d.pos].storage=0;     //存储量置 0
for(int i=0;i<amount;i++)
{
if(b[i].num==num)
tag=i;
}
b[tag]=b[amount-1];     //把该书目的所有书一本一本的删除
amount--;
for(int x=0;x<amount;x++)
{
SearchInsert2(&root,b[x]);
}
  }
  }
```

3. 借阅

在借书系统中，程序需要首先判断图书管理员是否将书本信息录入系统，没有则输出提示信息。当有库存时，通过数组指针中的相关关键词的比较，确定是否找到所需图书，找到则返回该结点的相关位置信息，询问借阅者所借图书数量，若满足借阅能力，则借阅成功，该书的库存量相应减一；若存书不足，则输出提示信息。总体借阅流程图如图 3 所示：

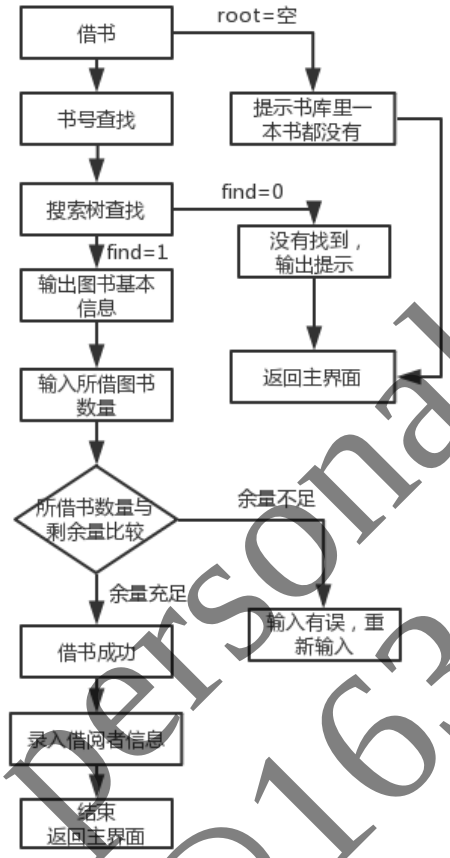


图 3 借阅流程

4. 还书

在还书系统中，同样调用搜索树的函数，查找该书是否属于该图书馆，不属于则输出提示信息，返回主界面。属于则询问要还书数量，判断还书数量是否超出之前借阅的数量，若超出，则提示输入有误，在还书范围，则将该书的库存量增加，还书成功。全部过程，都使用的指针数组进行操作。

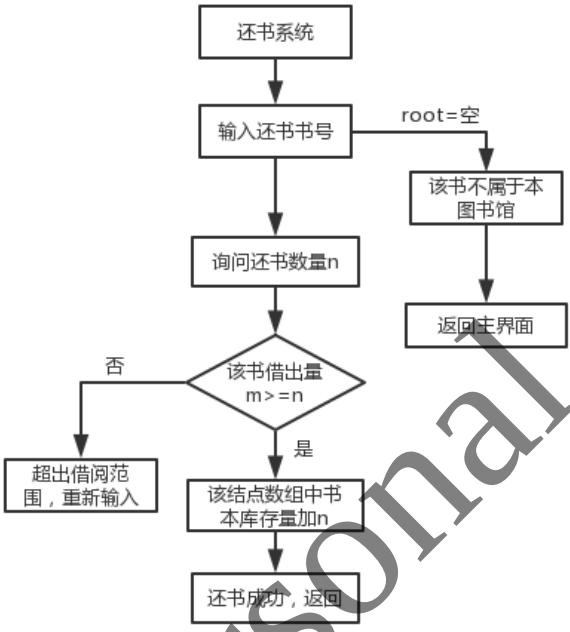


图 4 还书流程图

5. 显示查询

为了让改图书管理系统更加方便，操作性能更强，我增加了显示图书馆库存和查询书目的功能。

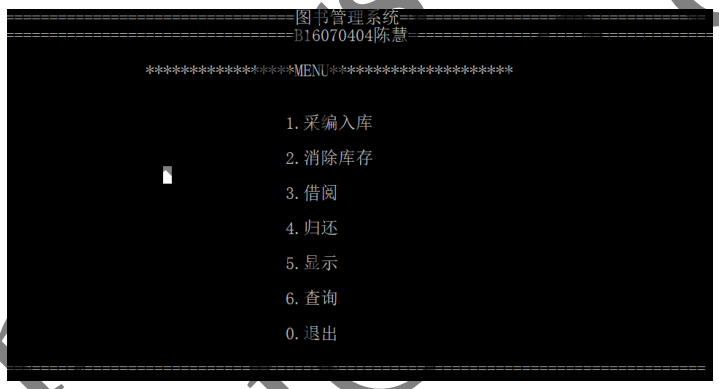
显示时，我使用凹入表的形式输出树中所有节点信息。查询图书时，使用中序遍历的方法，以此检索所有图书，当找到书号和所需书号一致时则输出。为了丰富查找形式，我设置了三种查询方式，书号查询，书名查询，作者查询。并且按照作者查询时，我将输出函数放入 for 循环查找里面，以此将同一作者的所有书目都找出来。部分代码如下：

```
case 3: printf("请输入你要查询的作者名：\n\n");
        scanf("%s",&b2.writer);
        printf("书号\t 书名\t 作者\t 剩余量\t 总库存\n");
        for(i=0;i<amount;i++)
        {
         if(strcmp(b2.writer,b[i].writer)==0)
         {
          num=b[i].num;            //将查到的该作者的数组中的书号找到
        d=SearchBT(root,num); //按照书号查找，省去重新编写函数的工作
```

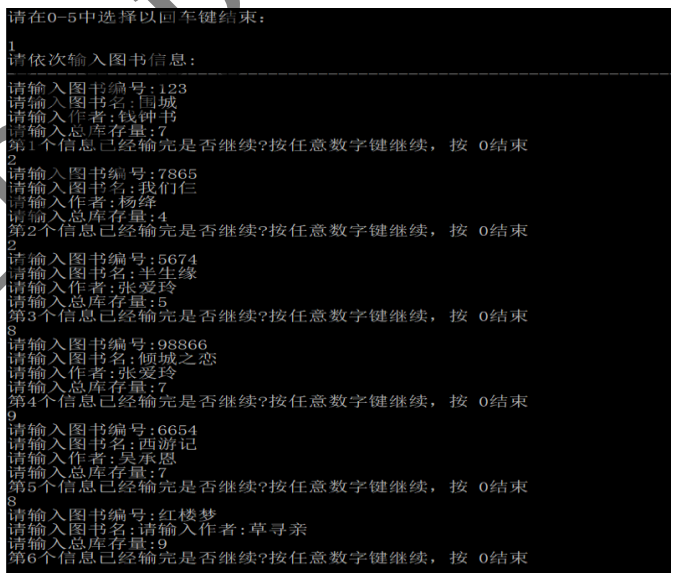
```
  if(d.find&&d.ptr->key[d.pos].storage!=0)
{  printf("%d\t%s\t%s\t%d\t%d\n\n",d.ptr->key[d.pos].num,d.ptr->key[d.pos].
name,d.ptr->key[d.pos].writer,d.ptr->key[d.pos].left,d.ptr->key[d.pos].storage
);                    //找到该结点则输出
} } }
   if(i>amount)       //没有找到则输出错误信息
   printf("你要找的作者暂时没有书储存于该系统!\n\n");
   printf("你还要继续查找吗?选择 yes 或者 no.\n");
   scanf("%s",a);
   break;
```

五、测试数据及其结果分析

1. 主界面显示，在界面设计中，使用 puts()函数来输出字符串，没有格式控制，里面的参数可以直接是字符串或者是存放字符串的字符数组名，这样能够更加方便的将系统所需按钮清晰简明的放入页面。



2. 采编入库，由图书管理员手动输入图书编号，作者，库存量



3. 显示馆存

```
请在0-5中选择以回车键结束：

5
        序列 书号----------------------书名   作者  剩余量
        [1]  0-----------------        红楼梦   草寻亲  9
        [2]123-----------------        围城   钱钟书  7
        [3]5674----------------        半生缘   张爱玲  5
        [4]6654----------------------- 西游记   吴承恩  7
        [5]7865-----------------       我们仨   杨绛  4
        [6]98866----------------       倾城之恋   张爱玲  7
```

4. 借书，只有当书号在库存中确实存在且借书量小于等于总库存时，才能显示借书成功，同时借阅者也需要登记图书真好和预计还书日期。

```
请在0-5中选择以回车键结束：

3
请输入这本书的书号:123
书号      书名      作者      剩余量   总库存
  123      围城    钱钟书        7        7

请输入您要借这本书的数量(小与等于7)
5
借书成功
请输入您的图书证号：B16070404
请输入您的还书日期：7月5日
```

5. 查询图书

```
请在0-5中选择以回车键结束：

6
==========================================================================
                ***************查询*********************

                                1.书号查询

                                2.书名查询

                                3.作者查询

==========================================================================
```

6. 按书号查询

```
请在1-3中选择以回车键结束：

1
请输入你要查询的书的书号：

123
书号      书名      作者      剩余量   总库存
123       围城      钱钟书   2        7

你还要继续查找吗?选择yes或者no.
yes
```

7. 按书名查询

```
请在1-3中选择以回车键结束：

2
请输入你要查询的书的书名：

围城
书号      书名      作者      剩余量   总库存
123       围城      钱钟书   2        7

你还要继续查找吗?选择yes或者no.
yes
```

8. 按作者查询，显示同一作者所有书籍，及剩余量，库存量

```
请在1-3中选择以回车键结束：

3
请输入你要查询的作者名：

张爱玲
书号    书名    作者    剩余量  总库存
5674    半生缘  张爱玲  5       5

98866   倾城之恋        张爱玲  7       7

你还要继续查找吗?选择yes或者no.
no
```

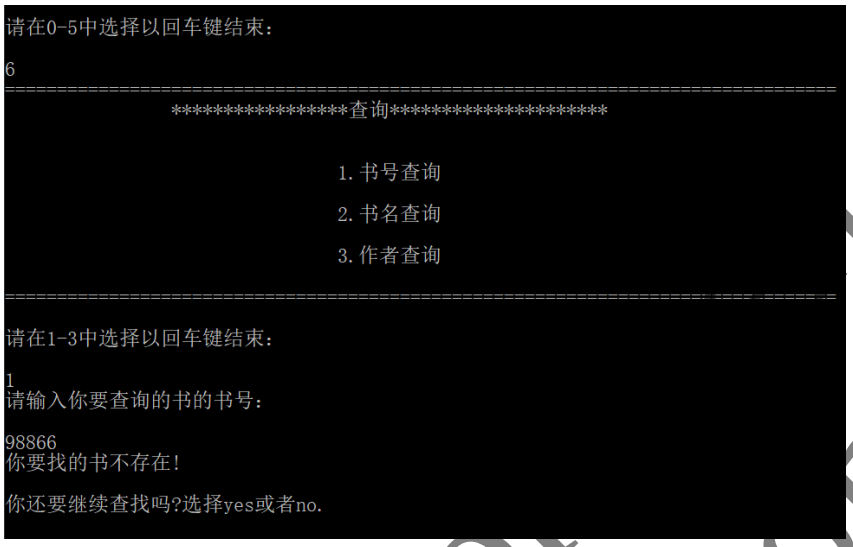
9. 还书，超过借书总量则提示输入错误，否则还书成功

```
请在0-5中选择以回车键结束：

4
请输入这本书的书号:123
书号      书名      作者      剩余量      总库存
123     围城  钱钟书      2           7

请输入您要还书的数量:
6
输入有误，请重新输入
4
还书成功
```

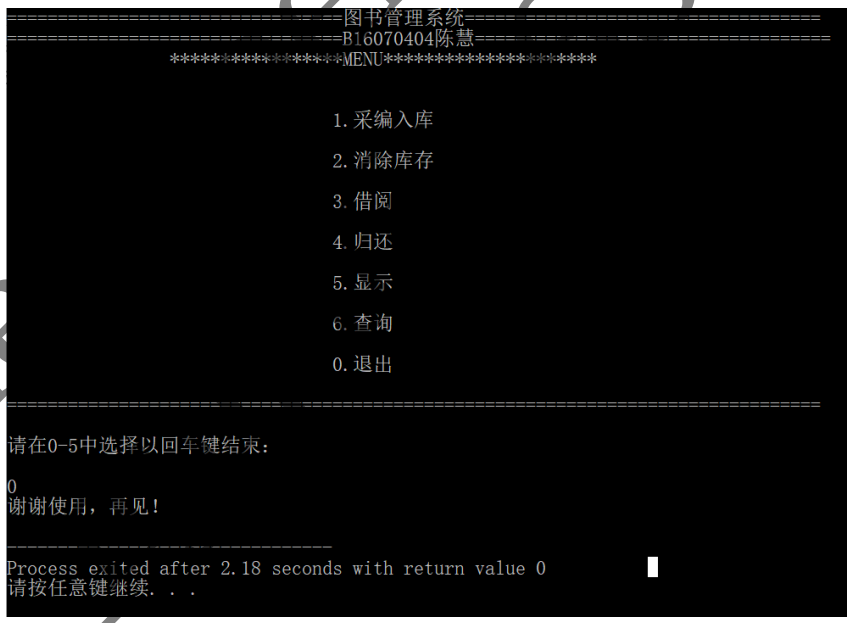
10. 消除库存，确认删除，删除成功后返回上层菜单

```
请在0-5中选择以回车键结束：

2
确定要消除库存吗?:

1.确定

0.返回

请选择:
1
请输入你要删除的书的书号98866
书号      书名      作者      剩余量      总库存
98866倾城之恋  张爱玲      7           7

确定要删除吗?请输入yes或者no.
yes
是否继续删除
请键入yes或no以回车键结束
yes
请按任意键以回车键结束返回上层菜单:
================================B16070404陈慧================================

                    **********MENU**********
```

再次查询当前该书库存时，发现“你要找的书不存在”



11. 最后退出



综上，题目中所要求内容都成功实现，并增加了两种功能，显示和查询，也让系统更加人性化。

六、调试过程中的问题

问题 1：之前设定的是一个局部变量 amount（用来记录录入图书的总量），在每一个模块里都设定了一个 amount，再调用采编函数中的 count 值（每加一本书，count 值增加）来赋值。运行过程中，发现当调用了借阅函数后，再调用显示函数时，并不能很好的实现动态删减功能。后来，想到设定一个全局变量 amount，用来保存图书总量，这样整个流程里就一个 amount 值，调用任意函数，都是此时此刻真实的馆存数量。之前学习 C 语言时，只是听老师讲了全局变量和局部变量的区别，通过这个问题，体会到了不同变量的不同功能和对于整个程序的不同作用。

问题 2：对于每本书的存储，之前就想着单纯的用一个链表，每录入一本书，就在链表的后面增加一个表项。但是调试时发现，用这种简单的存储方法，根本无法解决问题，系统中的借阅，归还，查询等功能完全不能实现，要想实现也需要繁杂繁琐的代码，和题目中完成功能简明便捷的目标不符合。后来我使用了 B 树进行存储，根据定义数据结构中的关键字标记，确定每录入一个书目，应该存放在哪个位置。这样对于后面的借阅，查找，归还等有很大的便利，也能很成功的完成题目中所需要求。通过这个调试，我发现 B 树查询快，增删结点相对于链表或者顺序表效率更好，因此用来存储大量图书信息十分合适。

问题 3：当我想得到全面的找的树结点的信息时，我发现仅仅返回一个 int 型或者 char 型的变量是远远不够的，要知道该结点位于树中的哪个位置，指向该结点的指针又是什么，在树中是否能找到所需结点。由此，我想到，可以再定义一个数据结构 result，让每次搜索树函数得到的最终结果，都返回 result 型的变量数组，这样每次找到的结果都能很好的传达后续流程所需所有信息。极大的方便了程序的进行。

改进设想：在查询模块中，原本只是单纯的按照书号进行查询，无疑这是一个很不方便的功能。在和老师的讨论中，我增加了按照书名查询和作者查询这两个功能。即根据不同的输入，使用同一个接口，进入查找函数中进行搜索，找到后返回搜索结果。

七、课程设计总结

本次的数据结构课程设计需要我们针对具体的实际项目来进行需求分析，概要设计，测试分析等。经过一周的数据结构课程设计，我觉得我有了很多的心得体会，作为一个之前对代码摸不着头脑的女生，第一次感受到了代码的魅力。

首先，在编写函数之前要充分利用图书资源和网络资源，在借鉴别人想法的基础上，让自己的设计更加合理规范；其次，应该更详细的考虑实际情况，使程序更切合实际，更具有实用性，像我本次的图书管理系统，适当的增加一些功能，就可以让用户体验更加便利；更多的，我觉得是那种自学的能力和积极和老师同学交流想法的能力。通过这次课程设计练习，我更深刻地理解了数据结构中的重要存储结构 B 树的精髓和线性链表的使用。完成整个程序设计后，我觉得我对于线性链表的使用变得更加熟练，对于 B 树的创建，删除，增加结点，遍历，打印输出等功能有了更深刻的认识。同时也加深了对数据结构的理解。

与此同时，经过这次课程设计，我深刻认识到算法在程序设计中的重要性，一个完整的程序总是由若干个函数构成的，这些相应的函数体现了算法的基本思想。特别是怎样将理论与实践相结合，把书本上的的内容应用到我们做的程序上。在各个子模块实施的详细功能中，特别是各个子模块之间的接口，一定要相当清晰，让他们达到相互协调的作用。其次我熟悉了数据结构的知识，学会了很多关于程序设计的经验和技巧，知道了程序调试的一般方法。重要的是通过本次程序设计，我逐步具备了走向程序员的基本素质。知道如何在困难重重时一步一步细心发现问题，解决问题。

编程是一件枯燥乏味工作，但是只要认真专研，我们会从中学到很多在课本上学不到，也无法在课堂上掌握的知识，同时也能从中感受到编程的乐趣。兴趣是可以培养的，只要坚持下去，面对困难我们总能够找到解决问题的方法。

最后，我觉得针对我们这个专业，专门留出一周的时间，让我们亲自动手，一个人一个组完成一个实际问题的编程实现，是一件很有意义的事情。让我们将书本知识和理论实践结合起来，为后期的编程打下坚实的基础。同时，也要感谢老师这一周的辛苦付出。