



**数据结构课程设计实验报告**

课程名称 数据结构

题 目 图书管理

学 院 计算机学院

专 业 软件工程

班 别 4

学 号 3115005372

姓 名 杨宇杰

指导老师 杨劲涛

**2016 年 12月 14 日**

**目录**

[课程设计——图书管理系统 3](#_Toc469742355)

[1. 需求分析 3](#_Toc469742356)

[1.1数据结构分析： 3](#_Toc469742357)

[1.2输出的形式； 3](#_Toc469742358)

[1.3程序所能达到的功能； 3](#_Toc469742359)

[1.4测试数据 4](#_Toc469742360)

[2. 概要设计 4](#_Toc469742361)

[2.1所有数据类型的定义 4](#_Toc469742362)

[2.2主程序的流程 7](#_Toc469742363)

[2.3各程序模块之间的调用关系。 7](#_Toc469742364)

[3. 详细设计 8](#_Toc469742365)

[3.1程序头文件+源代码 8](#_Toc469742366)

[}}3.2函数和过程的调用关系图。 30](#_Toc469742367)

[4. 调试分析 30](#_Toc469742368)

[4.1 调试过程中遇到的问题是如何解决的以及对设计与实现的回顾讨论和分析； 30](#_Toc469742369)

[4.2 算法的时空分析（包括基本操作和其他算法的时间复杂度和空间复杂度的分析）和改进设想； 31](#_Toc469742370)

[4.3 经验和体会。 31](#_Toc469742371)

[5. 用户使用说明 32](#_Toc469742372)

[6. 测试结果 34](#_Toc469742373)

[7. 附录 39](#_Toc469742374)

# 课程设计——图书管理系统

## 1. 需求分析

### 1.1数据结构分析：

图书管理系统中图书管理模块包括图书类型定义：书号(int)，现存量(int)，总存量(int)，出版时间(int),价格(float),书名，作者名为字符型，借阅指针为读者类型；

读者类型定义：ID号(int),姓名为字符型。

B树(2-3树)类型定义：关键字个数和关键字数组为整型，另外还有指向双亲的指针，指向子树的指针；

B树查找结果类型定义：结点指针，关键字序号和查找标志变量为整型。

### 1.2输出的形式；

输出界面以用户于计算机的交互方式进行，在输出窗口上显示“特定的提示信息”之后，由用户按提示在键盘上输入演示程序中规定的运算命令，相应的输入数据和运算结果显示在下面。由于时间和能力有限，该管理系统没有用文件存放数据，所有数据放在内存中存放(后来做了改进版就有了)，但是选做的功能就还没有实现。其基本业务都是以书号为关键字进行，采用了(2-3树)对书号建立索引，以提高效率。

### 1.3程序所能达到的功能；

#### 1)采编入库

新书购入，将书号，书名，作者，册数，出版时间以及价格添加入图书管理系统中，如果这种书在系统中已存在，则只将总库存量增加，每增加一个书号则以凹入表的形式显示B树形状。

#### 2)清除库存

实现某本书的全部信息删除操作，每清除一个书号则以凹入表的形式显示B树形状。

#### 3）图书借阅

如果树的库存量大于零时则执行出借，登记借阅者的图书证号和姓名。

#### 4)图书归还

注销借阅者信息，并改变该书的现存量。

#### 5) 查看图书馆全部图书

用表格输出所有图书的信息。

#### 6) 查看某图书信息

查看指定某一本书的全部信息。

#### 7)查看某本书的借阅者信息

表格输出某本书的全部借阅者信息。

#### 8)读取图书信息

从文件中读取所有图书的信息以及所有借阅者的信息

### 1.4测试数据

入库书号：35, 16, 18, 70, 5, 50, 22, 60, 13, 17, 12 , 45, 25, 42, 15, 90, 30, 7

然后清除：45, 90, 50, 22, 42

其余数据自行设计。由空树开始，每插入删除一个关键字后就显示B树的状态。

## 2. 概要设计

### 2.1所有数据类型的定义

采用KeyType类型(本次实验默认为int)为元素类型实现抽象数据类型BTree。

ADT BTree{

数据对象：T是具有相同特征的数据元素集合。

数据关系：

若D为空集，则称为空树；

（1）树中每个结点最多含有m棵子树；

（2）若根结点不是叶子结点，则至少有2个子树；

（3）除根结点之外的所有非终端结点至少有┌m/2┐棵子树；

（4）每个非终端结点中包含信息：（n，A0，K1，A1，K2，A2，…，Kn，An）。其中：

1）Ki（1<=i<=n）为关键字，且关键字按升序排序；

2）指针Ai（0<=i<=n）指向子树的根结点，Ai-1指向子树中所有结点的关键字均小于Ki，且大于Ki-1；

3）关键字的个数n必须满足：┌m/2┐-1<=n<=m-1。 （5）所有的叶子节点都在同一层，子叶结点不包含任何信息。

#### 基本操作：

result SearchBTree(BTree T, int k);

初始条件:树T存在

操作结果:在m阶B数T上查找关键字k，返回p{pt,i,tag}

Status InsertBTree(BTree &T, int k, BTree q, int i, Record \*recptr);

初始条件:树T存在

操作结果:在B树T上结点p->pt的key[i]和key[i+1]之间插入关键字k

Status DeleteBTree(BTree &T, int k);

初始条件:B树上p结点存在

操作结果:删除B树T上结点p->pt的关键字k

void BTreeTraverse(BTree T, void(\*visit)(BTree));

初始条件:树T存在

操作结果:用visit()函数遍历B树

void DestroyBTree(BTree T);

初始条件:树T存在

操作结果:销毁B树

} ADT BTree

ADT Library{

数据对象：T是具有相同特征的数据元素集合。

数据关系：数据元素同属于一个集合。

#### 基本操作：

void InitLibrary(BTree &L);

操作结果:初始化书库L为空书库

void InsertBook(BTree &L, BookType B, result res);

初始条件:书库L和B存在，result包含B书在书库中的位置或应该插入的位置

操作结果:如果书库中已存在B树，则只将B树库存量增加，否则插入B书到书库L中。

int DeleteBook(BTree &L, BookType B);

初始条件:书库L和B存在

操作结果:如果书库中存在B书，则从书库中删除B书的信息，并返回OK，否则返回ERROR

void BorrowBook(BTree L, BookType B, ReaderType R);

初始条件:书库L存在，B书时书库中的书并且可被读者R借阅

操作结果:借出一本B书，并登记借阅者信息

int ReturnBook(BTree L, int booknum, int IDnum, BookType &B, ReaderType &R);

初始条件:书库L存在

操作结果:若书库L中有读者R借阅B书的记录，则注销该记录，改变B书现存量，并返回OK，书不存在或物该读者记录则返回ERROR

void PrintAllbooks(BTree L);

初始条件:书库L存在

操作结果:显示所有图书的信息

int ShowBookinfo(BTree L, int booknum);

初始条件:书库L存在

操作结果:若书库L中存在书B，则显示B书基本信息并返回OK，否则返回ERROR

void PrintBorrower(ReaderType R);

操作结果:输出某本书所有借阅者的信息

} ADT Library

### 2.2主程序的流程



### 2.3各程序模块之间的调用关系。



## 3. 详细设计

### 3.1程序头文件+源代码

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*头文件定义\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Library.h

#include "windows.h"

#include "stdio.h"

#include "conio.h"

#include "stdlib.h"

#include "malloc.h"

#include "time.h"

#include "string.h"

#include "errno.h"

#define TRUE 1

#define FALSE 0

#define OVERFLOW -1

#define OK 1

#define ERROR 0

#define M 3//定义阶数

#define Super\_Manager\_Code 5372 //超级管理员密码

#define MAX\_NAME\_LEN 20//姓名长度

#define MAX\_BKNAME\_LEN 50

#define OK 1

#define ERROR 0

#define bookinfomationfile "bookinfo.dat" // 图书信息文件

#define borrowerfile "borrower.dat" // 借阅者名单文件

typedef int KeyType;

typedef int Status;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*书和借阅者结点存储定义\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

typedef struct ReaderNode {//借阅者

int IDnum; //ID号

char reader\_name[MAX\_NAME\_LEN]; //姓名

ReaderNode \*rnext; //指向下一个借阅者的指针

}ReaderNode, \*ReaderType;

typedef struct BookNode {

int booknum; //书号

char bookname[MAX\_BKNAME\_LEN]; //书名

char writer[MAX\_NAME\_LEN]; //作者名字

int current; //现存量

int total; //总库存

int published\_year; //出版年份

float price; //价格

ReaderType reader; //借阅者链表指针

}BookNode, \*BookType; //图书类型

typedef BookNode Record; //将书的结点定义为记录

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*B树存储定义\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

typedef struct BTNode { //B树结点类型定义

int keynum; //结点中关键字个数，即结点的大小

KeyType key[M+1]; //关键字，key[0]未用

struct BTNode \*parent; //双亲结点指针

struct BTNode \*ptr[M + 1]; //孩子结点指针数组

Record \*recptr[M + 1]; //记录指针向量，0号单元未用

}BTNode, \*BTree; //B树结点和B树类型

typedef struct result {

BTNode \*pt; //指向找到的结点

int i; //1<=i<=m,在结点中的关键字位序

int tag; //1:查找成功，0:查找失败

}result, \*resultPtr; //B树的查找结果类型

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*B树接口定义\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

result SearchBTree(BTree T, int k);

/\*

初始条件:树T存在

操作结果:在m阶B数T上查找关键字k，返回p{pt,i,tag}

\*/

Status InsertBTree(BTree &T, int k, BTree q, int i, Record \*recptr);

/\*

初始条件:树T存在

操作结果:在B树T上结点p->pt的key[i]和key[i+1]之间插入关键字k

\*/

Status DeleteBTree(BTree &T, int k);

/\*

初始条件:B树上p结点存在

操作结果:删除B树T上结点p->pt的关键字k

\*/

void BTreeTraverse(BTree T, void(\*visit)(BTree));

/\*

初始条件:树T存在

操作结果:遍历B树

\*/

int menu();

/\*

输出选择菜单

\*/

void ShowBTree(BTree T, short x = 8);

/\*

以凹入表的形式输出B树

\*/

int login();

/\*

登陆界面

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*图书馆接口定义\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void InitLibrary(BTree &L);

/\*

初始化一棵空树

\*/

void InsertBook(BTree &L, BookType B, result res);

/\*

插入新的书籍

\*/

int DeleteBook(BTree &L, BookType B);

/\*

删除已存在的书籍，不存在则提示不存在

\*/

int CanBorrow(BTree L, BookType B, ReaderType R);

/\*

判断指定书籍能否出借

\*/

void BorrowBook(BTree L, BookType B, ReaderType R);

/\*

书籍出借

\*/

int ReturnBook(BTree L, int booknum, int IDnum, BookType &B, ReaderType &R);

/\*

书籍归还

\*/

void PrintH();

/\*

输出表格的头部

\*/

void PrintB(BookType B);

/\*

输出指定书籍的信息

\*/

void PrintT();

/\*

输出表格的尾部

\*/

void PrintAll(BTree p);

/\*

输出B树结点的全部关键字信息

\*/

void PrintBook(BookType B);

/\*

以表格形式输出一本书的信息

\*/

void PrintAllbooks(BTree L);

/\*

输出所有书的信息

\*/

int ShowBookinfo(BTree L, int booknum);

/\*

输出指定某本书的信息

\*/

void PrintBorrower(ReaderType R);

/\*

输出某本书的借阅者信息

\*/

void Welcome(int color);

/\*

显示欢迎界面

\*/

void Creat(BTree &L);

/\*

从文件中读取图书信息

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*B树基本操作实现\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

BTree\_Operation.cpp

#include "Library.h"

#include "Library.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*B树接口实现\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int Search(BTree p, int k) {

//在B树p中查找关键字k的位置i，使得p->node[i].key<=K<p->node[i+1].key

int i;

for (i = 0; i < p->keynum&&p->key[i + 1] <= k; i++);

return i;

}

result SearchBTree(BTree T, int k) {

//在m阶B树T上查找关键字k，返回(pt,i,tag)

//若查找成功，则特征值tag=1,指针pt所致结点中第i个关键字等于k;否则

//特征值tag=0，等于k的关键字记录应插入在指针pt所指结点中第i-1个和第i个关键字间

result r;

int i = 1, found = 0;

BTree p = T, q = NULL;

while (p != NULL && 0 == found) {

i = Search(p, k);//在p->key[1..keynum]中查找p->key[i-1]<k<=p->p->key[i]

if (i > 0 && p->key[i] == k)

found = 1;//找到待查关键字

else {

q = p;

p = p->ptr[i];

}

}

if (1 == found) {//查找成功

r.pt = p;

r.i = i;

r.tag = 1;

}

else {//查找不成功，返回key的插入位置i

r.pt = q;

r.i = i;

r.tag = 0;

}

return r;

}

void split(BTree &q, int s, BTree &ap) {

//将q结点分裂成两个结点，前一半保留，后一半移入新结点ap

int i, n = q->keynum;

ap = (BTNode\*)malloc(sizeof(BTNode));//生成新结点ap

ap->ptr[0] = q->ptr[s];

for (i = s + 1; i <= n; i++) {//后一半移入ap结点

ap->key[i-s] = q->key[i];

ap->ptr[i-s] = q->ptr[i];

ap->recptr[i - s] = q->recptr[i];

}

ap->keynum = n - s;//计算ap的关键字个数

ap->parent = q->parent;

for (i = 0; i < n - s; i++) {

if (ap->ptr[i])

ap->ptr[i]->parent = ap;//将ap所有孩子结点指向ap

}

q->keynum = s - 1;//q结点的前一半保留，修改keynum

}

void newroot(BTree &T, BTree p, int k, BTree ap,Record \*recptr) {//生成新的根结点

T = (BTNode\*)malloc(sizeof(BTNode));

T->keynum = 1;

T->ptr[0] = p;

T->ptr[1] = ap;

T->key[1] = k;

T->recptr[1] = recptr; //T的子树ap的父亲指针

if (p != NULL) p->parent = T;

if (ap != NULL) ap->parent = T;

T->parent = NULL;//新根的双亲是空指针

}

void Insert(BTree &q, int i, int k, BTree ap,Record \*recptr) {//k和ap分别插到q->key[i+1]和q->ptr[i+1]

//并插入关键字为k的记录recprt

int j, n = q->keynum;

for (j = n; j >= i; j--) {

q->key[j + 1] = q->key[j];//关键字指针向后移一位

q->ptr[j + 1] = q->ptr[j];//孩子结点指针向后移一位

q->recptr[j + 1] = q->recptr[j];

}

q->key[i+1] = k;//赋值

q->ptr[i+1] = ap;

q->recptr[i + 1] = recptr;

if (ap != NULL) ap->parent = q;

q->keynum++;//关键字数+1

}

Status InsertBTree(BTree &T, int k, BTree q, int i,Record \*recptr) {

//在B树T上q结点的key[i]和key[i+1]之间插入关键字k

//若引起结点过大,则沿双亲指针进行必要的结点分裂调整,使T仍是m阶的B树

int s,finished = 0, neednewroot = 0;

BTree ap = NULL;

if (NULL == q)//q为空，则新建根结点

newroot(T, NULL, k, NULL, recptr);

else {

while (!finished) {

Insert(q, i, k, ap,recptr);//key和ap分别插到q->key[i]和q->ptr[i]

if (q->keynum < M) finished = 1;//插入完成

else {//分裂q结点

s = (M + 1) / 2;

split(q, s, ap);

k = q->key[s];

recptr = q->recptr[s];

if (q->parent != NULL) {

q = q->parent;

i = Search(q, k);//在双亲结点中查找x的插入位置

}

else finished=OVERFLOW;

}

}//while

if (finished==OVERFLOW)//T是空树或者根结点已分裂为q和ap结点

newroot(T, q, k, ap,recptr);//生成含信息(q,x,ap)的新的根结点T

}

return OK;

}

void TakePlace(BTree &q, int &i) {

//\*q结点的第i个关键字为k，用q的后继关键字代替q，且令q指向后继所在结点

BTree p = q;

q = q->ptr[i];

while (q->ptr[0]) q = q->ptr[0]; //查找p的后继

p->key[i] = q->key[1]; //记录代替

p->recptr[i] = q->recptr[1];

i = 1; //代替后应该删除q所指结点的第1个关键字

}

void Del(BTree q, int i) {

//删除q所指结点第i个关键字及其记录

for (; i < q->keynum; i++) {//关键字和记录指针前移

q->key[i] = q->key[i + 1];

q->recptr[i] = q->recptr[i + 1];

}

q->keynum--;//关键字数目减1

}

int Borrow(BTree q) {

//若q的兄弟结点关键字大于(M-1)/2,则从兄弟结点上移最小（或最大）的关键字到双亲结点，

//而将双亲结点中小于(或大于)且仅靠该关键字的关键字下移至q中，并返回OK,否则返回ERROR

int i;

BTree p = q->parent, b;//p指向q的双亲结点

for (i = 0; p->ptr[i] != q; i++);

if (i >= 0 && i + 1 <= p->keynum&&p->ptr[i + 1]->keynum > (M - 1) / 2) {

//若q的右兄弟关键字个数大于(M-1)/2

b = p->ptr[i + 1];//b指向右兄弟结点

q->ptr[1] = b->ptr[0];//子树指针也要同步移动

q->key[1] = p->key[i + 1];//向父结点借第i+1个关键字

q->recptr[1] = p->recptr[i + 1];

p->key[i + 1] = b->key[1];//b第一个关键字上移到父结点

p->recptr[i + 1] = b->recptr[1];

for (i = 1; i <= b->keynum; i++) {//b第一个关键字上移，需把剩余记录前移一位

b->key[i] = b->key[i + 1];

b->recptr[i] = b->recptr[i + 1];

b->ptr[i - 1] = b->ptr[i];

}

}

else if (i > 0 && p->ptr[i - 1]->keynum > (M - 1) / 2) {

//若q的左兄弟关键字个数大约(M-1)/2

b = p->ptr[i - 1]; //b指向左兄弟结点

q->ptr[1] = q->ptr[0];

q->ptr[0] = b->ptr[b->keynum];

q->key[1] = p->key[i];//从父结点借第i个关键字

q->recptr[1] = p->recptr[i];

p->key[i] = b->key[b->keynum];//将b最后一个关键字上移到父结点

p->recptr[i] = b->recptr[b->keynum];

}

else return ERROR;//无关键字大于(M-1)/2的兄弟

q->keynum++;

b->keynum--;

for (i = 0; i <= q->keynum; i++) {

if (q->ptr[i])

q->ptr[i]->parent = q;//刷新q的子结点的双亲指针

}

return OK;

}

void Combine(BTree &q) {

int i, j;

BTree p = q->parent, b = NULL;//p指向q的父亲

for (i = 0; p->ptr[i] != q; i++);//查找q在父亲p中的子树位置

if (i == 0) {//若为0，则需合并为兄弟的第一个关键字

b = p->ptr[i + 1];

for (j = b->keynum; j >= 0; j--) {//将b的关键字和记录后移一位

b->key[j + 1] = b->key[j];

b->recptr[j + 1] = b->recptr[j];

b->ptr[j + 1] = b->ptr[j];

}

b->ptr[0] = q->ptr[0];//合并

b->key[1] = p->key[1];

b->recptr[1] = p->recptr[1];

}

else if (i > 0) {//若q在父亲的子树位置大于0，需合并为兄弟b的最后一个关键字

b = p->ptr[i - 1];

b->key[b->keynum + 1] = p->key[i];//合并

b->recptr[b->keynum + 1] = p->recptr[i];

b->ptr[b->keynum + 1] = q->ptr[0];

}

if(i==0||i==1)//若i为0或1，需将父结点p关键字前移一位

for (; i < p->keynum; i++) {

p->key[i] = p->key[i + 1];

p->ptr[i] = p->ptr[i + 1];

p->recptr[i] = p->recptr[i + 1];

}

p->keynum--;

b->keynum++;

free(q);

q = b; //q指向修改的兄弟结点

for (i = 0; i <= b->keynum; i++)

if (b->ptr[i]) b->ptr[i]->parent = b;//刷新b的子结点的双亲指针

}

Status DeleteBTree(BTree &T, KeyType k)

// 在m阶B树T上删除关键字k及其对应记录，并返回OK。

// 如T上不存在关键字k，则返回ERROR。

{

KeyType x = k;

BTree q, b = NULL;

int finished = FALSE, i = 1;

result res = SearchBTree(T, k); // 在T中查找关键字k

if (res.tag == 0) return ERROR; // 未搜索到

else

{

q = res.pt; // q指向待删结点

i = res.i;

if (q->ptr[0]) TakePlace(q, i); // 若q的子树不空，(非底层结点)

// 则以其后继代之，且令q指向后继所在结点

Del(q, i); // 删除q所指向结点中第i个关键字及记录

if (q->keynum >= (M - 1) / 2 || !q->parent) // 若删除后关键字个数不小于(m-1)/2或q是根节点

{

finished = TRUE; // 删除完成

if (q->keynum == 0) T = NULL; // 若q的关键字个数为0 ，则为空树

}

while (!finished)

{

if (Borrow(q)) finished = TRUE; // 若q的相邻兄弟结点关键字大于(m-1)/2,则从该

// 兄弟结点上移一个最大（或最小）关键字到

// 父节点，从父节点借一关键字到q

else { // 若q相邻兄弟关键字个数均等于┌m /2┑-1

Combine(q); // 将q中的剩余部分和双亲中的相关关键字合并至q的一个兄弟中

q = q->parent; // 检查双亲

if (q == T && T->keynum == 0) // 若被删结点的父节点是根T且T的关键字个数为0

{

T = T->ptr[0]; // 新根

T->parent = NULL;

free(q); // 删除原双亲结点

finished = TRUE;

}

else if (q->keynum >= M / 2) finished = TRUE;

} // 合并后双亲关键字个数不少于(m-1)/2，完成

}

}

return OK;

}

void BTreeTraverse(BTree T, void (\*visit)(BTree )) {

//遍历B树T，对每个结点调用visit函数

if (!T) return;

visit(T);

for (int i = 0; i <= T->keynum; ++i) {

if (T->ptr[i]) BTreeTraverse(T->ptr[i], visit);

}

}

void ShowBTree(BTree T, short x)

// 递归以凹入表形式显示B树T,每层的缩进量为x，初始缩进量为8

{

if (!T) return;

int i;

printf("\n");

for (i = 0; i <= x; i++) putchar(' '); // 缩进x

for (i = 1; i <= T->keynum; i++)

{

printf("%d,", T->key[i]);

}

for (i = 0; i <= T->keynum; i++) // 递归显示子树结点关键字

ShowBTree(T->ptr[i], x + 7);

}

int menu() {//菜单

int choice;

printf("\n\n\t\t\t|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|\n");

printf("\t\t\t|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|\n");

printf("\t\t\t \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n");

printf("\t\t\t| 图书馆管理系统 |\n");

printf("\t\t\t| |\n");

printf("\t\t\t| 1.采编入库 2.清除库存 |\n");

printf("\t\t\t| |\n");

printf("\t\t\t| 3.借阅图书 4.归还图书 |\n");

printf("\t\t\t| |\n");

printf("\t\t\t| 5.查看图书馆全部图书 6.查看某图书信息 |\n");

printf("\t\t\t| |\n");

printf("\t\t\t| 7.查看某书借阅者信息 8.读取图书信息 |\n");

printf("\t\t\t| |\n");

printf("\t\t\t| 0.退出 |\n"); printf("\t\t\t|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n");

printf("\t\t\t|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|\n");

printf("\t\t\t|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|\n");

printf("\t\t\t| 15软件工程(4)班 |\n");

printf("\t\t\t| 3115005372 |\n");

printf("\t\t\t| 杨宇杰 |\n");

printf("\t\t\t|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*S\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|\n");

do {

printf("\t\t\t请选择功能（输入0-7任意一个数字）:");

scanf\_s("%d", &choice);

} while (choice < 0 || choice>7);//避免非法输入

return choice;

}

void gotoxy(HANDLE hOut, int x, int y) {//定位光标

COORD pos;

pos.X = x; //横坐标

pos.Y = y; //纵坐标

SetConsoleCursorPosition(hOut, pos);

}

int login() {//登陆界面

HANDLE hOut = GetStdHandle(STD\_OUTPUT\_HANDLE);//定义显示器句柄变量

int secret\_code, i;

printf("\n\n\t\t\t \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf("\t\t\t| |\n");

printf("\t\t\t| |\n");

printf("\t\t\t| 请输入图书管理员密码！ |\n");

printf("\t\t\t| 密码：\_\_\_\_ |\n");//密码为5372

printf("\t\t\t|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n");

gotoxy(hOut, 43, 6);

scanf\_s("%d", &secret\_code);

fflush(stdin);

printf("\n\t\t\tLoding：");//进入动画

for (i = 0; i<11; i++) {

Sleep(50);

printf("█ ");

}

printf("\n\t\t\t载入完成，进入下一级系统");

Sleep(1000);

system("cls");//清屏

if (secret\_code == Super\_Manager\_Code)//判断是不是管理员

return 1;

else

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*图书基本操作\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Library.cpp

#include"Library.h"

void InitLibrary(BTree &L) {

L = NULL;

}

void InsertBook(BTree &L, BookType B, result res) {

//书库L已存在，res包含B书在书库L中的位置或应该插入的位置

//如果书库中已存在B书，则只将B书的库存量增加，否则插入B书到书库L中

if (res.tag == 0) InsertBTree(L, B->booknum, res.pt, res.i, B);

//书库中不存在该书，则插入

else {

BookType b = res.pt->recptr[res.i];

b->current = b->current + B->total;//现存量增加

b->total = b->total + B->total;//总库存增加

}

}

int DeleteBook(BTree &L, BookType B) {

//如果书库中存在B书，则从书库中删除B书的信息，并返回OK，否则返回ERROR

if (DeleteBTree(L, B->booknum)) return OK;//删除成功

else return ERROR;//删除失败

}

int CanBorrow(BTree L, BookType B, ReaderType R) {

//如果书库中存在B书，若B书现存量大于0，则可出借返回OK，否则返回ERROR

if(B->current>0) return TRUE;//现存量大于零

else return FALSE;//其他情况均不允许出借

}

void BorrowBook(BTree L, BookType B, ReaderType R)

// 书库L存在，B书是书库中的书并且可被读者R借阅（已通过CanBorrow()判断）

// 借出一本B书，登记借阅者R的信息，改变现存量，记录借书日期，最迟应还日期等信息。

{

ReaderType r;

if (!B->reader) B->reader = R; // 无其他借阅者，则直接登记

else {

for (r = B->reader; r->rnext; r = r->rnext);

r->rnext = R; // 否则到借阅者链表尾，登记

}

B->current--; // 现存量减1

}

int ReturnBook(BTree L, int booknum, int IDnum, BookType &B, ReaderType &R) {

//booknum为还书书号，IDnum是借阅者的ID号

//若书库中不存在书号为booknum的书，则搜索出错，返回-1

//若有借阅记录，则注销该记录，并用B和R返回图书信息和借阅者信息并返回1

//若没有r借阅的记录，则用B返回图书信息，并返回0

result res = SearchBTree(L, booknum); //在B树上按书号搜索

if (res.tag == 0) return -1;

B = res.pt->recptr[res.i];

ReaderType p = res.pt->recptr[res.i]->reader, pre = p;

while (p) {

if (pre == p&&p->IDnum == IDnum) {

R = p;

B->current++;//现存量加一

return 1;

}

if (p->IDnum == IDnum) {//多个借书者

R = p;

pre->rnext = p->rnext;

B->current++;//现存量+1

return 1;

}

pre = p;

p = p->rnext;

}

return 0;

}

void PrintH() {//表格头部格式

printf("\n");

printf("|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*图书基本信息\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|\n");

printf("| 书号 | 书名 | 作者 | 现存 | 总库存 | 出版年份 | 定价 |\n");

}

void PrintB(BookType B) { //显示B书基本信息

printf("|--------|--------------------|---------------|------|--------|----------|------|\n");

printf("| %-4d |%-20s|%-15s|", B->booknum,B->bookname,B->writer);

printf(" %-5d| %-4d | %-6d |%-6.1f|\n", B->current, B->total, B->published\_year, B->price);

}

void PrintT() {//表格底部格式

printf("|--------|--------------------|---------------|------|--------|----------|------|\n");

}

void PrintAll(BTree p) {//显示B树每个结点的记录信息

int i;

for (i = 1; i <= p->keynum; ++i) {

PrintB(p->recptr[i]);

}

}

void PrintBook(BookType B) {//以表格形式显示一本书的基本信息

PrintH();

PrintB(B);

PrintT();

printf("\n");

}

void PrintBorrower(ReaderType R) {//以表格形式显示一本书的借阅者信息

ReaderType p = R;

printf("|------------------------|\n");

while (p) {

printf("| %-5d | %-10s |\n", p->IDnum, p->reader\_name);

p = p->rnext;

}

printf("|------------------------|\n");

}

void PrintAllbooks(BTree L) {//显示书库L所有图书信息

PrintH();

BTreeTraverse(L, PrintAll);

PrintT();

}

int ShowBookinfo(BTree L, int booknum) {

//若书库L中存在书号为booknum的书，则现在该书的所有信息并返回OK，书库L不存在则

//返回ERROR

result res = SearchBTree(L, booknum);

if (!res.tag) return ERROR;

BookType B = res.pt->recptr[res.i];

PrintBook(B);

return OK;

}

void gotoxy(short x, short y)

// 移动光标到（x，y）坐标，25>x>=0,80>y>=0

{

COORD point = { x, y };

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), point);

}

void SetColor(unsigned short TextColor)

// 设置字体和背景颜色

{

HANDLE hCon = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hCon, TextColor);

}

void Welcome(int color)

// 显示欢迎界面

{

SetColor(color);

gotoxy(2, 5);

printf(" ╭╮╭╮╭──╮╭╮　　╭──╮╭──╮╭╭╮╮╭──╮\n");

printf(" │││││╭─╯││　　│╭─╯│╭╮││　　││╭─╯\n");

printf(" │╰╯││╰─╮││　　││　　│││││││││╰─╮\n");

printf(" │││││╭─╯││　　││　　│││││╭╮││╭─╯\n");

printf(" │　　││╰─╮│╰─╮│╰─╮│╰╯││││││╰─╮\n");

printf(" ╰╰╯╯╰──╯╰──╯╰──╯╰──╯╰╯╰╯╰──╯\n");

SetColor(11);

printf(" ╔══════════════╗\n");

printf(" ║ 欢迎进入图书管理系统 ║\n");

printf(" ╚══════════════╝\n");

SetColor(color+1);

}

void Creat(BTree &L) {

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// 创建函数。当显示菜单时按8即可进入本创建函数。

// L为书库。

// 从指定文件读入图书信息进行图书入库。

// 然后从文件读入借阅者名字，用随机数产生借阅证号和借阅书号，进行图书借阅，并记录日志，

// 接着读入预约者名字，用随机数产生借阅证号和图书证号，进行图书预约，并记录日志

// 完成后即可返回主菜单，进行还书、预约或其他测试。

FILE \*fp;

BookType B;

result res;

errno\_t err;

ReaderType R;

//fp = fopen\_s(bookinfomationfile, "r"); // 打开图书信息文件

if ((err = fopen\_s(&fp, "bookinfo.dat", "r")) != 0)

{

printf("不能打开图书信息文件,请确认%s文件已放到盘根目录...", bookinfomationfile);

getchar();

return;

}

printf("\n下面将从文件%s读入图书信息。按任意键继续...", bookinfomationfile);

getchar();

while (!feof(fp)) // 当未到文件结束

{

B = (BookType)malloc(sizeof(BookNode)); // 申请图书空间

fscanf(fp, "%d %s %s %d %d %d %f", &B->booknum, B->bookname, B->writer, &B->current,

&B->total, &B->published\_year, &B->price);// 读入图书数据

B->reader = NULL; // 读者和借阅者指针置空

res = SearchBTree(L, B->booknum); // 查找插入位置

if (!res.tag) // 书库中不存在该书，则插入

{

InsertBook(L, B, res); // 插入

printf("\n\n\n 插入下面的图书");

PrintBook(B); // 显示该图书

printf("\nB树的状态\n\n");

ShowBTree(L); // 显示插入后B树的状态

}

}

fclose(fp);

printf("\n\n读入的所有图书信息");

PrintAllbooks(L); // 显示所有图书

printf("\n文件图书信息读入完毕!下面从文件读入借书者信息，按任意键继续...\n");

getchar();

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

system("cls");

int booknum, i = 0;

fp = fopen(borrowerfile, "r"); // 打开借阅者名单文件

if (!fp)

{

printf("\n\n\n不能打开读者信息文件,请确认%s文件已放到盘根目录...", borrowerfile);

getch();

return;

}

while (i++ <4000 && !feof(fp))

{

R = (ReaderType)malloc(sizeof(ReaderNode));

fscanf(fp, "%6s", &R->reader\_name); // 从文件读入借阅者名字

R->IDnum = rand() % 100; // 用随机数产生借阅证号

R->rnext = NULL; // 后续借阅者预约者指针置空

booknum = rand() % 100 + 1; // 随机产生借阅书号

res = SearchBTree(L, booknum); // 搜索二叉树

if (res.tag) // 如果找到该书号

{

B = res.pt->recptr[res.i];

if (CanBorrow(L, B, R)) // 判断该借阅者是否可以借阅该书

{

BorrowBook(L, B, R); // 如果可以则借出一本

}

}

}

fclose(fp);

PrintAllbooks(L); // 显示所有图书（可以看到现存量减少）

printf("\n借书信息读入完毕，下面将从文件读入预约者信息。按任意键继续....");

getch();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*主函数\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

main.cpp

#include"Library.h"

int main() {

BTree L = NULL;

BookType B;

result res;

ReaderType R;

int booknum, IDnum, k, book\_num;

char flag;

int i;

for (i = 0; i <= 10; i++){ // 显示欢迎界面

Welcome(i % 16);

Sleep(100);

}

Sleep(800);

system("cls");//清屏

if (login()) {

for (;;) {

switch (menu())

{

case 1://采编图书

B = (BookType)malloc(sizeof(BookNode));

B->reader = NULL;

printf("请输入要入库的书号：\n");

fflush(stdin);

scanf\_s("%d", &B->booknum);

getchar();

res = SearchBTree(L, B->booknum);

if (!res.tag) {

fflush(stdin);

printf("请输入该书书名：\n");

fflush(stdin);

gets\_s(B->bookname);

printf("请输入该书作者：\n");

fflush(stdin);

gets\_s(B->writer);

printf("请输入入库数：\n");

fflush(stdin);

scanf\_s("%d", &B->current);

printf("请输入出版年份：\n");

fflush(stdin);

scanf\_s("%d", &B->published\_year);

printf("请输入书本价格：\n");

fflush(stdin);

scanf\_s("%f", &B->price);

B->total = B->current;

InsertBook(L, B, res);

printf("用凹入表形式显示B树如下：\n");

ShowBTree(L, 8);

break;

}

else {//图书已存在，只增加库存

printf("书库已有该书记录，请输入增加的册数：\n");

fflush(stdin);

scanf\_s("%d", &B->total);

InsertBook(L, B, res);

B = res.pt->recptr[res.i];

printf("用凹入表形式显示B树如下：\n");

ShowBTree(L);

printf("新书入库操作完成！\n");

system("pause");

break;

}

case 2://清空库存

printf("请输入要删除的图书书号：\n");

fflush(stdin);

scanf\_s("%d", &booknum);

res = SearchBTree(L, booknum);

if (res.tag) {

B = res.pt->recptr[res.i];

PrintBook(B);

printf("\n您确认删除上面的图书<Y确认，其余按键返回主菜单>?");

fflush(stdin);

getchar();

flag = getchar();

if (flag == 'Y' || flag == 'y') {

DeleteBook(L, B);

printf("\n清除完毕！\t删除后用凹入表显示B树如下：\n");

ShowBTree(L, 8);

}

}

else {

printf("\n书库不存在此书，删除失败！\n");

}

printf("\t");

system("pause");

break;

case 3://借阅图书

R = (ReaderType)malloc(sizeof(ReaderNode));

R->rnext = NULL;

printf("请输入借阅图书书号：\n");

fflush(stdin);

scanf\_s("%d", &booknum);

getchar();

res = SearchBTree(L, booknum);

if (res.tag) {

B = res.pt->recptr[res.i];

printf("请输入您的姓名：\n");

fflush(stdin);

gets\_s(R->reader\_name);

printf("请输入您的ID号：\n");

fflush(stdin);

scanf\_s("%d", &R->IDnum);

if (CanBorrow(L, B, R)) {

BorrowBook(L, B, R);

printf("\n借书完成！");

}

else {

printf("\n库存不足，借阅失败！");

free(R);

}

}

else {

printf("\n库存无此书，借阅失败！");

free(R);

}

printf("\t");

system("pause");

break;

case 4://归还图书

printf("\n请输入图书号：\n");

scanf\_s("%d", &booknum);

printf("请输入您的ID号：\n");

scanf\_s("%d", &IDnum);

k = ReturnBook(L, booknum, IDnum, B, R);

if (k == 1) {

printf("\n还书完成！");

R = NULL;

}

else if (k == 0) {

printf("\n还书失败，系统没有您借该书记录！");

}

else {

printf("\n书库中不存在此书!");

}

printf("\t");

system("pause");

break;

case 5://输出总库存

printf("图书馆总库存信息如下：\n");

PrintAllbooks(L);

break;

case 6://查找某种图书

printf("请输入要查找的图书的编号：\n");

scanf\_s("%d", &book\_num);

ShowBookinfo(L, book\_num);

break;

case 7://输出某图书的借阅者信息

printf("请输入要查找的图书的编号：\n");

scanf\_s("%d", &book\_num);

res = SearchBTree(L, book\_num);

if (res.tag) {

B = res.pt->recptr[res.i];

printf("借阅者信息如下：\n");

PrintBorrower(B->reader);

}

else {

printf("\n书库不存在此书，查找失败！\n");

}

printf("\t");

system("pause");

break;

case 8:Creat(L);

system("cls");

break;

case 0: //退出管理系统

exit(0);

default:

break;

}

}

}

else {

printf("密码输入错误！\n");

system("pause");

}

return 0;

}

### }}3.2函数和过程的调用关系图。



## 4. 调试分析

### 4.1 调试过程中遇到的问题是如何解决的以及对设计与实现的回顾讨论和分析；

此次的课程设计的重难点应该在于B树的定义以及删除操作，由于我设计性实验选的也是B树，有几个操作函数大同小异，不过要在原来的基础上增加一些东西，例如在原有的关键字的基础上加上与关键字对应的图书结点，因此算法需要一些小改动。

一开始程序跑不起来，一看居然68个错误，可是大致看下来都是不太正常的提示，所以在这个地方卡了蛮久的时间，好在经过不断的调试，发现是头文件里各个数据结构的顺序放倒了，更改顺序后只剩下寥寥几个小错误，大大增长了把程序做好的希望。

此次实验的不足在于这个图书管理系统的存储是建立在内存上的，故程序退出数据得不到保存，每次都得重新输入（经过改进已经解决这个问题）。每个功能比较独立，相互联系不算多，仅仅是完成各部分需求的算法，采编入库，清除库存和显示信息相对逻辑关系强一点。由于个人能力的局限以及时间上的紧迫，此次的选做要求没有实现，系统功能的不够完整和联系不够紧密是我以后要提高的地方。

### 4.2 算法的时空分析（包括基本操作和其他算法的时间复杂度和空间复杂度的分析）和改进设想；

算法的时空分析和改进设想

设B树的阶为m，（书号）关键字个数为N

1. SearchBTree最好情况为O(1),最坏时比较的结点数不超过log[m/2]((N+1)/2)+1；
2. 因为B树总是在最后一层插入，因此InsertBTree操作比较的结点数也不超过log[m/2]((N+1)/2)+1，
3. DeleteBTree最好情况为被删关键字在最下层结点而且删除后该结点关键字个数不小于m/2，时间复杂度O(1)；最坏情况为删除后需要合并父节点直到到根节点，需比较2 \*（log[m/2]((N+1)/2)+1）次；
4. InsertBook、DeleteBook时间复杂度与B树的插入和删除基本相同。
5. BorrowBook、ReturnBook除了需要SearchBTree，还与该书借阅人数成正比。

### 4.3 经验和体会。

1. 按照指导书给出的实习报告规范的步骤，先进行需求分析、概要设计，再进行详细设计，能使实际问题从抽象到具体，能从整体上把握程序的功能方向。
2. 通过划分成几个模块，单独编写每个模块的基本操作，再组合在一起，细分了程序的功 能，降低设计的难度，方便了程序的调试、修改和组合。
3. 本次设计实在被B树的指针折磨透了，我深深的体会到：编码时必须对程序的算法了如 指掌，对可能出现的每一种情况都处理周到，不然对于复杂的（像B树的删除）函数将会无所适从。
4. 编码时就应该尽量把程序写正确，避免产生错误，这样可能远比调试时才来发现问题、 解决问题用的时间少得多，效率高得多。
5. 测试时要注意测试的完备性
6. 编码时要形成一种自己的风格，同时要注意这种风格应该是大众所接受的，要保持程序的可读性。

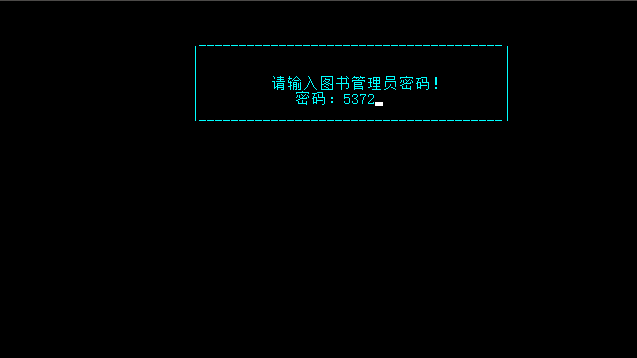
## 5. 用户使用说明

1. 本程序的运行环境为Windows 10操作系统，编译环境是VS 2015。
2. 执行文件为BTree\_Library.exe

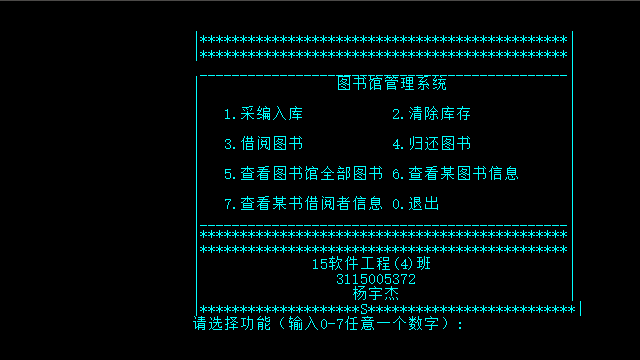
打开BTree\_Library.exe，显示欢迎界面



1. 输入管理员密码：5372



1. 进入图书管理系统



1. 选择1新书入库后提示输入入库书号，读入书号后程序会查找是否已存在，如已存在则提示输入新增加数量，如不存在则提示输入图书其他书名、著者、等信息，插入后显示B树状态。
2. 选择2清除库存后，提示输入删除书号，如果存在该书则提示是否确认删除，如无该书则提示不存在。删除后显示B树状态。
3. 选择3图书出借后，提示输入书号，如该书存在则显示该书，提示输入借阅证号和借阅者姓名，判断如果能借阅则输出借阅成功，否则输出不能借阅；如无该书则提示不存在该书。
4. 选择4图书归还后，提示输入归还书号和借阅证号，如存在该借阅记录则提示还书成功，否则提示不存在该书或无该读者借书记录。
5. 选择5查看图书馆所有图书后，以表格形式输出图书管理系统的总库存。
6. 选择6查看某图书信息后，以表格形式输出该图书的库存情况以及作者等相关信息。
7. 选择7查看某树借阅者信息后，以表格形式输出该图书的所有借阅者的信息。
8. 选择8将文件中的图书内容和借阅者资料导入
9. 选择0则退出系统。

## 6. 测试结果

列出你的测试结果，包括输入和输出。这里的测试数据应该完整和严格，最好多于需求分析中所列。

测试数据如下：

// 书号 书名 著者 现存量 总库存 出版年份 价格

35 灯下漫笔 鲁迅 39 39 1997 19.5

16 BASIC语言 谭浩强 38 38 1980 18.0

18 C程序设计 谭浩强 40 40 2007 26.0

70 阿Q正传 鲁迅 28 28 1984 13.5

5 计算机组成原理 白中英 56 56 2008 39.0

50 意志力训练手册 美·弗兰克 46 46 2005 20.0

22 打出速度-五笔教程 导向科技 45 45 2003 18.0

60 幽默大师 南琦 48 48 2007 38.5

13 鲁迅小说选 鲁迅 39 39 1976 18.3

17 计算机组成原理题库 白中英 36 36 2008 18.5

12 坟 鲁迅 55 55 1973 8.6

45 学会说话 南琦 57 57 2004 18.5

25 准风月谈 鲁迅 60 60 1973 13.9

42 超越自己 南琦 45 45 2007 15.0

15 鲁迅杂文选 鲁迅 30 30 1973 18.9

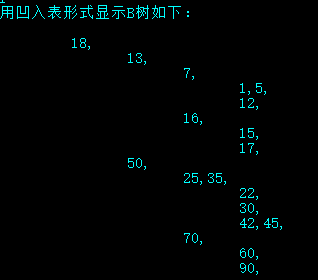
90 鲁迅日文作品集 鲁迅 45 45 1981 15.5

30 数字电子简明教程 余孟尝 55 55 2005 35.0

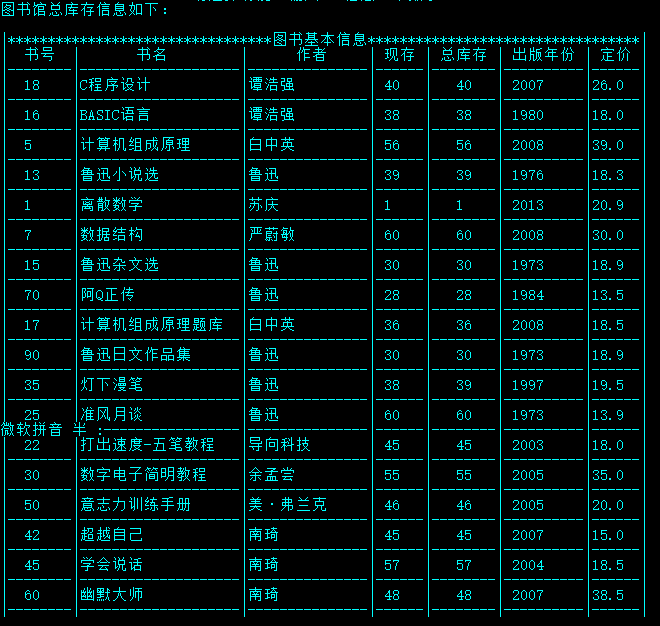
7 数据结构(C语言版) 严蔚敏 60 60 2008 30.0

1 离散数学 苏庆 1 1 2013 20.9

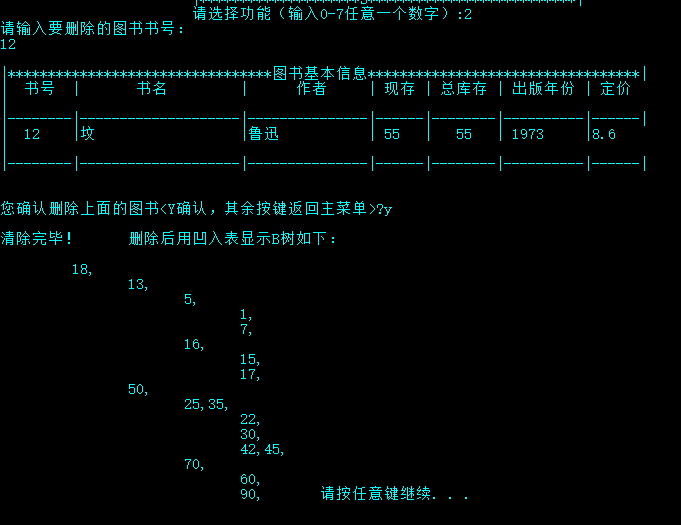
将全部数据输入后用凹入表形式显示B树：



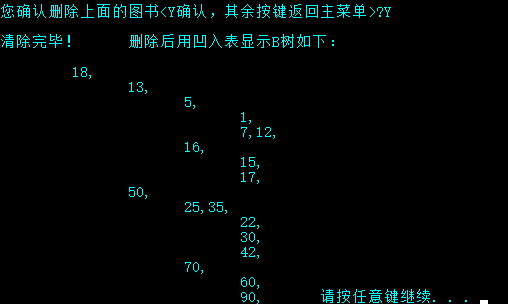
使用功能5：用表格输出图书馆总库存：



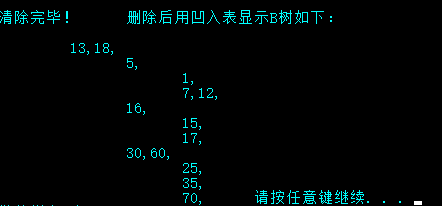
选择功能2：清空某一本书的库存,



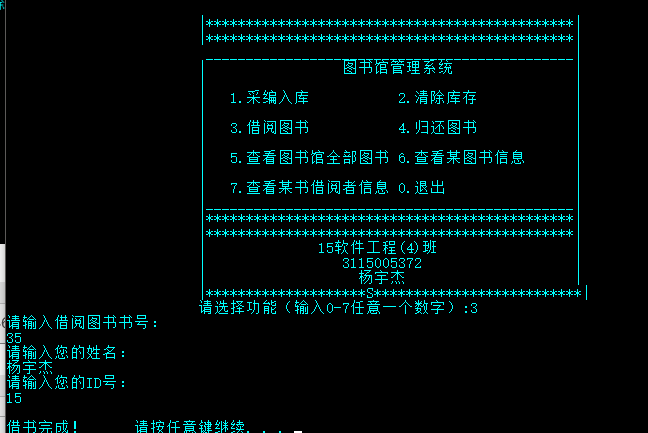
清除45后的凹入表：



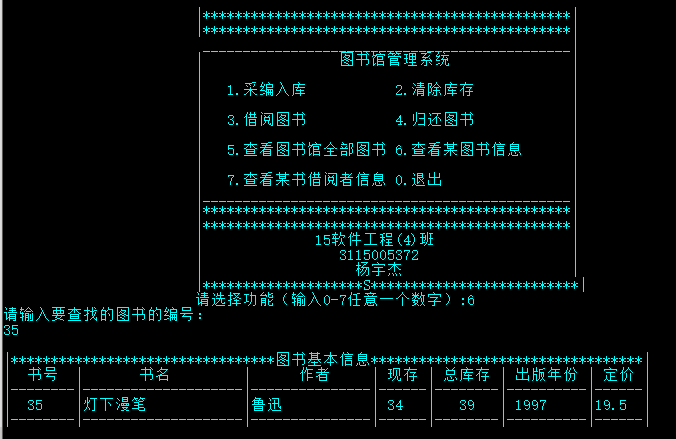
用功能2依次清除：45, 90, 50, 22, 42后的凹入表表示如下：



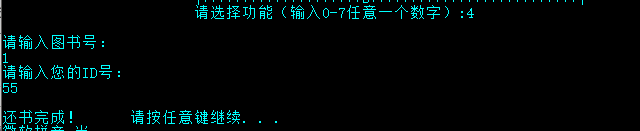
选择功能3：借阅图书，输入要借的图书号，借阅者的姓名和ID号



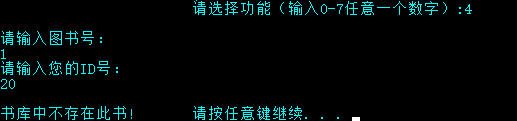
选择功能6：查看某一本图书的信息，并以表格形式输出。



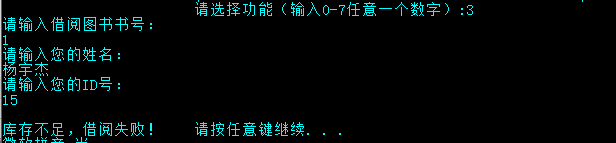
选择功能4：归还图书



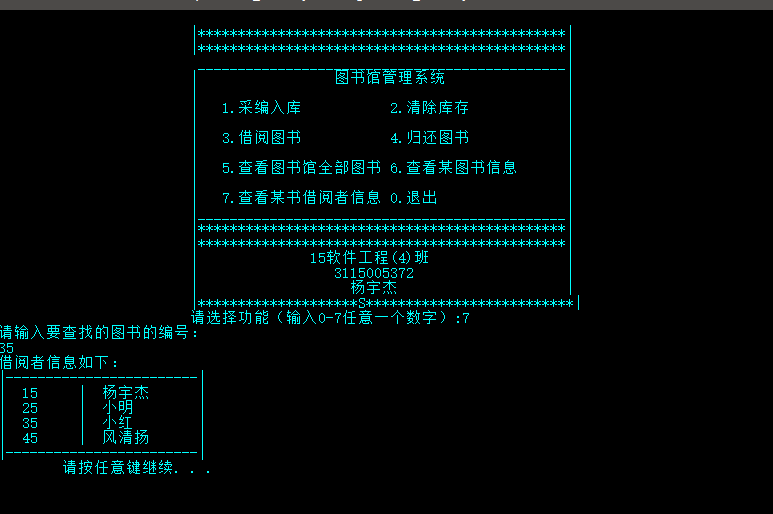
当图书号不存在时，输出书库中不存在此书



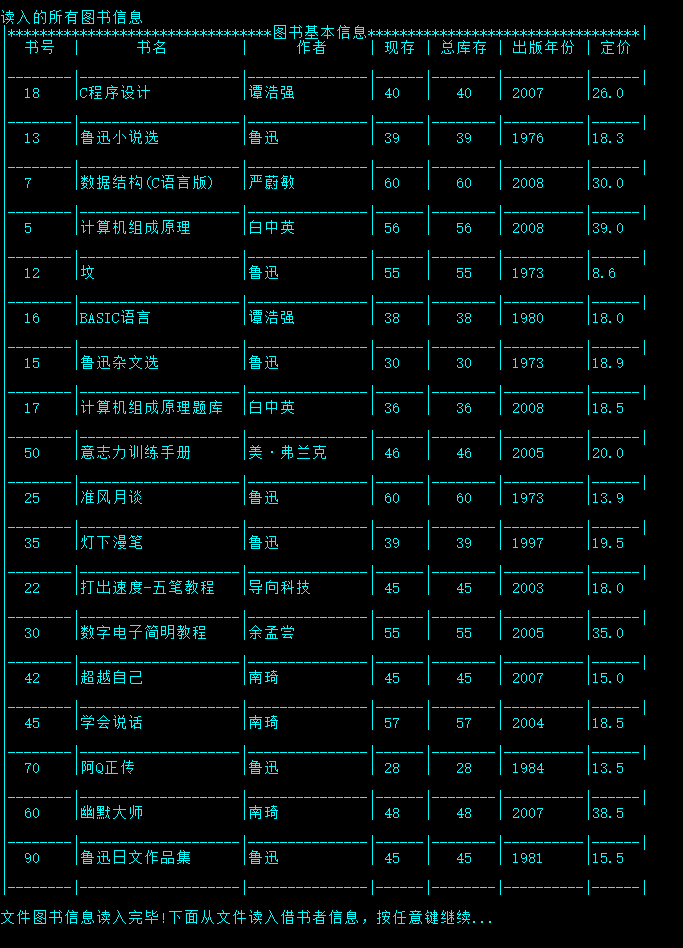
当借书时该图书的库存已空，则提示库存不足，借阅失败！



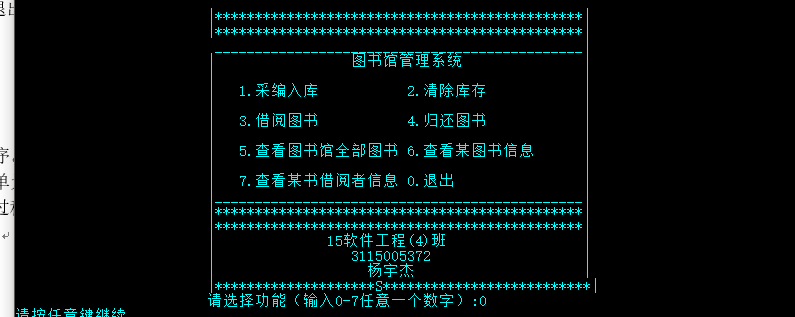
选择功能7：查看某本书的借阅者信息



输入功能8：从文件中导入图书信息和借阅者信息



输入功能0：退出系统



## 7. 附录

带注释的源程序。如果提交源程序软盘，可以只列出程序文件名清单。

程序文件名清单：

Library.h

Library.cpp

BTree\_Operation.cpp

main.cpp

bookinfo.dat

borrower.dat

程序文件名截图如下：

