# 图书馆管理系统

使用链表存储用户和书籍. 由于一种书籍的每一本可能会借给不同的人, 所以对于每一本书籍都在链表中有一个节点.

```
struct Book
{
    string name;
    int id;
    Book* nxt;
    int rentid;
};

struct User
{
    string name;
    int id;
    User* nxt;
};
```

添加用户和添加书的逻辑类似:

```
void adduser(int id, string name)
{
    auto nown = headuser;
    while(nown->nxt != nullptr && nown->nxt->id < id)
    {
        nown = nown->nxt;
    }
    auto newn = new User;
    newn->nxt = nown->nxt;
    nown->nxt = newn;
    newn->id = id;
    newn->name = name;
    return ;
}

//添加书
void addbook(int id, string name, int rentid)
{
    auto nown = headbook;
    while(nown->nxt != nullptr && nown->nxt->id <= id)
    {
        nown = nown->nxt;
    }
    auto newn = new Book;
```

```
    newn->nxt = nown->nxt;
    nown->nxt = newn;
    newn->id = id;
    newn->rentid = rentid;
    newn->name = name;
    return ;
}
```

输出用户和输出书籍:

```
//输出所有用户
void printUsers()
{
    auto nown = headuser;
    printf("id  name\n");
    while(nown->nxt != nullptr)
    {
        printf("%3d %s\n", nown->nxt->id, nown->nxt->name.c_str());
        nown = nown->nxt;
    }
    printf("\n");
    return ;
}

//输出所有书籍
void printBooks()
{
    auto nown = headbook;
    printf("id  name\n");
    int cnt = 0;
    while(nown->nxt != nullptr)
    {
        if(nown->id != -1 && nown->id != nown->nxt->id)
        {
            printf("%3d %s %3d\n", nown->id, nown->name.c_str(), cnt);
            cnt = 0;
        }
        nown = nown->nxt;
        cnt ++;
    }
    if(nown->id != -1)
    {
        printf("%3d %s %3d\n", nown->id, nown->name.c_str(), cnt);
    }
    printf("\n");
    return ;
}
```

删除用户和删除书籍:

```
//删除一个用户
void deluser(int id)
{
    auto nown = headuser;
    while(nown->nxt != nullptr && nown->nxt->id != id)
    {
        nown = nown->nxt;
    }

    printf("Deleted %s!\n",nown->nxt->name.c_str());
    fflush(stdout);
    auto t = nown->nxt;
    nown->nxt = t->nxt;
    free(t);
    return ;
}

//删除书
int delbook(int id, int cnt)
{
    auto nown = headbook;
    int tcnt = 0;
    while(nown->nxt != nullptr)
    {
        if(nown->nxt->id == id && nown->nxt->rentid == -1)
        {
            auto t = nown->nxt;
            nown->nxt = t->nxt;
            free(t);
            tcnt++;
            if(tcnt == cnt) //删够了
            {
                break;
            }
        }else{
            nown = nown->nxt;
        }

    }

    printf("Deleted %d Books!\n", tcnt);
    fflush(stdout);
    return tcnt;
}
```

修改书和用户的名字:

```
//改一本书的名字
void changebookname(int id, string name)
{
    auto nown = headbook;
    while(nown->nxt != nullptr)
    {
        if(nown->nxt->id == id)
        {
            nown->nxt->name = name;
        }
        nown = nown->nxt;
    }
    return ;
}

//改一个用户的名字
void changeusername(int id, string name)
{
    auto nown = headuser;
    while(nown->nxt != nullptr)
    {
        if(nown->nxt->id == id)
        {
            nown->nxt->name = name;
        }
        nown = nown->nxt;
    }
    return ;
}
```

借还书:

```
//还书
void returnBooks(int userid, int bookid)
{
    auto nown = headbook;
    while(nown->nxt != nullptr)
    {
        if(nown->nxt->id == bookid && nown->nxt->rentid == userid)
        {
            nown->nxt->rentid = -1;
        }
        nown = nown->nxt;
    }
```

```
}

//借书
void rentBooks(int userid, int bookid)
{
    auto nown = headbook;
    bool ok = false;
    while(nown->nxt != nullptr)
    {
        if(nown->nxt->id == bookid && nown->nxt->rentid == -1)
        {
            nown->nxt->rentid = userid;
            ok = true;
            break;
        }
        nown = nown->nxt;
    }
    if(!ok)
    {
        printf("Out of Books!\n");
    }
    return ;
}
```

查询借还书:

```
//查询一本书有谁借了
void findUserByBooks(int bookid)
{
    auto nown = headbook;
    printf("Books %d are rent by: \n", bookid);
    while(nown->nxt != nullptr)
    {
        if(nown->nxt->id == bookid && nown->nxt->rentid != -1)
        {
            printf("User %d\n", nown->nxt->rentid);
        }
        nown = nown->nxt;
    }
    return ;
}

//查一个人借了哪些书
void findBooksbyUser(int userid)
{
    auto nown = headbook;
```

```
    printf("User %d has rent: \n", userid);
    while(nown->nxt != nullptr)
    {
        if(nown->nxt->rentid == userid)
        {
            printf("Book %d, %s\n", nown->nxt->id, nown->nxt-
>name.c_str());
        }
        nown = nown->nxt;
    }
}
```

保存与读取:

```
//保存用户
void saveUsers()
{
    FILE *fp = fopen("users.txt","w");
    auto nown = headuser;
    while(nown->nxt != nullptr)
    {
        fprintf(fp,"%3d %s\n", nown->nxt->id, nown->nxt->name.c_str());
        nown = nown->nxt;
    }
    fprintf(fp,"\n");
    fclose(fp);
    return ;
}

//保存书
void saveBooks()
{
    FILE *fp = fopen("books.txt","w");
    auto nown = headbook;
    while(nown->nxt != nullptr)
    {
        fprintf(fp,"%3d %s %d\n", nown->nxt->id, nown->nxt-
>name.c_str(), nown->nxt->rentid);
        nown = nown->nxt;
    }
    printf("\n");
    fclose(fp);
    return ;
}

//读取用户
void readUsers()
```

```
{
    FILE *fp = fopen("users.txt","r");
    char buf[200];
    int tid;
    while(~fscanf(fp,"%d %s", &tid, buf))
    {
        adduser(tid,string(buf));
    }
    fclose(fp);
    return ;
}

//读取书
void readBooks()
{
    FILE *fp = fopen("books.txt","r");
    char buf[200];
    int tid, trentid;
    while(~fscanf(fp,"%d %s %d", &tid, buf,&trentid))
    {
        addbook(tid,string(buf),trentid);
    }
    fclose(fp);
    return ;
}
```

## 小结

对于结构庞大的代码，事先的规划时必要的. 程序需要怎么组织数据，使用什么数据结构，需要实现什么功能都是要提前设计好的，避免写到一半再进行修改.