Final Report

XJCO1921 Programming Project

A library information system

Username : sc17yq

Name : Yi Qiao

**Contents**

**1.Modular structure**......................................................................................1

**2.Descriptions about this project**................................................................2

2.1.What this project have implemented..................................................2

2.2.Description of the design strategy.......................................................3

**3.Test cases and results**...............................................................................3

3.1.Test two list functions works................................................................4

3.2.Test register in two different ways.......................................................5

3.3.Test if the add and remove functions...................................................7

3.4.Test if the borrow and return functions................................................9

3.5.Try some invalid data..........................................................................11

**4.Personal reflection**.....................................................................................12

**5.Some deficiency**.........................................................................................13

1. **Modular structure**:

**main.c**: containing the main function.

// to create the whole structure for the program.

int main(int argc, char \*argv[]);

**read.c** and **read.h**: read files into linked lists and write the final information into the files. It will also define the structures and repeating structure for students and the librarian.

// read files into linked lists for student.txt and book.txt respectively.

extern void readStud();

extern void readBook();

// switch structures, the users can choose the options. The repeating module.

extern int operateLibrarian(int order);

extern int operateStudent(int order);

// save linked list into files.

extern void saveInStuFile();

extern void saveInBookFile();

// release the linked lists.

extern void freeStud();

extern void freeBook();

**print.c** and **print.h**: print prompt message so that the code in other modules can have higher readability.

// print helf information

extern void printHelp();

// print the headers for two tables.

extern void studHeaderPrint();

extern void bookHeaderPrint();

// functions printing operations

extern void printIdentity();

extern void printLibrarian();

extern void printStudent();

**student.c** and **student.h**: including some operations for students.

// create node to student linked list.

extern void addStud(char\* name, char\* id, char\* passw, int quan);

// return the student node we are searching for, which is quiet useful in latter functions.

extern Stud\* searchStud(char\* name, char\*id);

// set the node sNow as the student now log in.

extern void setUser(char\* name, char\* id);

// ask the user to input their name, id number and password.

extern void signUp();

// if the user is not in the list, ask if they want to log in as a new user.

extern int logIn();

// to search all the students in the linked list to make sure if the recent user have singned up.

extern int identityVertify(char\* name, char\* id, char\* passw);

// to ask if the user if willing to sign up.

extern int registerConfirm(char\* name, char\* id, char\* passw);

// the operation for students and the librarian to list all books.

extern void listBook();

// return the book searching for.

extern Book\* searchBook(char\* name, char\* booknum);

// return the next book searching for.

extern Book\* searchNextBook(char\* name, char\* booknum);

// the operation for students to search a book and print its information.

extern void searchBookFunc();

// the operation for students to borrow a book.

extern void borrowBook();

// the operation for students to return a book.

extern void returnBook();

// while the operation of borrowing or returning is successful, record it into the file record.txt.

void recordOper(int order, char\* name, char\* num);

**librarian.c** and **librarian.h**: including some operations for the librarian.

// create nodes to the book linked list

extern void addBook(char\* name, char\* booknum, char\* auth, int quan, int totquan);

// the operation for the librarian to list all users.

extern void listUser();

// the operation for the librarian to add books.

extern void addBookInTheList();

// the operation for the librarian to remove books.

extern void removeBook();

1. **Descriptions about this project**
   1. What this project have implemented:

I have chosen the project to build a library information system, which can meet several commands from both students and the librarian.

Functions for students: register, borrow books, return books, list all the books.

Functions for the librarian: list all students, list all users, add books, remove books.

All the information changed will be written into files respectively, whose names are ‘student.txt’, ‘book.txt’ and ‘record.txt’.

* 1. Description of the design strategy:

Methods to plan the work at a high level:

At first, we need to know what to implement exactly, and get those key functions. Next, we need to figure out the flowchart of this project, which means we need to stand in users shoes, so that we can get a precise requirement. Then the most important thing to plan a project is learn about the knowledge maybe used in it. As to this specific one, the knowledge is the linked list. After knowing many types of operations of it, like adding a node at the end of the list, removing a given node or search a node, we can easily find the corresponding method for each function.

Design iterations used:

1. This project can provide several option for users or the librarian to operate. Every time they operated successfully, the asking messages should appear again until they choose to quit this system. To get this goal, the author used switch structure functions operateLibrarian() and operateStudent() in a while loop.
2. Read the file and add the nodes to the linked list one by one.
3. Every time to show the options there should be a printed list on the screen.

Specific application features associated with each iteration:

1. Switch functions:

The main function will call these two functions and get the return value based on the choice from the users. Namely, if the choice is not ”quit”, the return value will always be 1 and this operate function will work again.

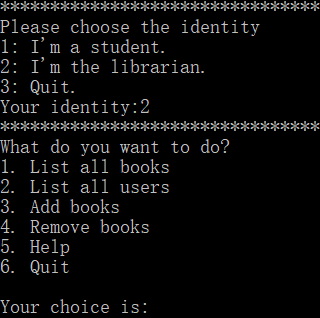
1. Every time add a new node to the linked list, the functions readStud() and readBook() need to call addStud() and addBook().
2. Options:

operateLibrarian() and operateStudent() will call printLibrarian() and printStudent() respectively.

1. **Test cases and results**

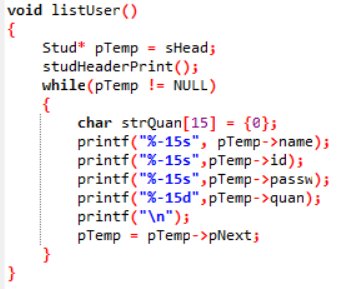
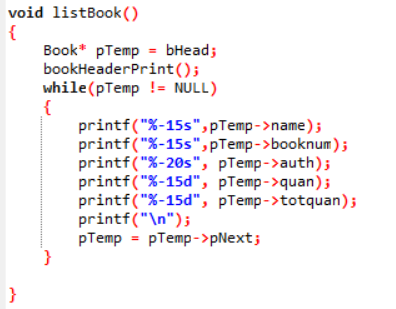
Five non-trivial test cases：

First of all, I would like to show the information list of this program.



* 1. For the librarian, test if the two list functions works.

Test data:



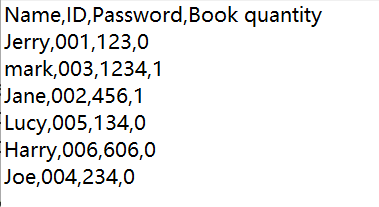
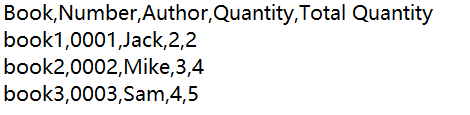
Required input:

Your choice is:1

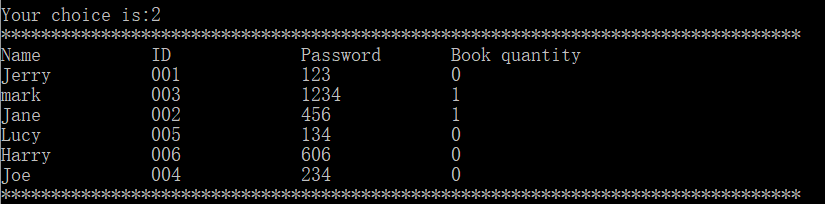
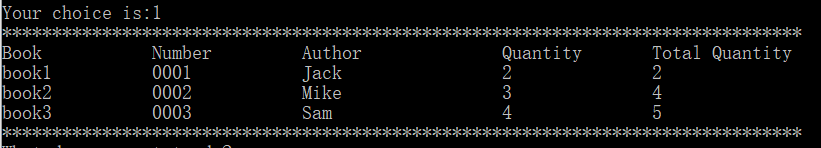
Your choice is:2

Expected output:

List all the information in the corresponding file, with right headers and neat format. And there is the content of file book.txt and student.txt.

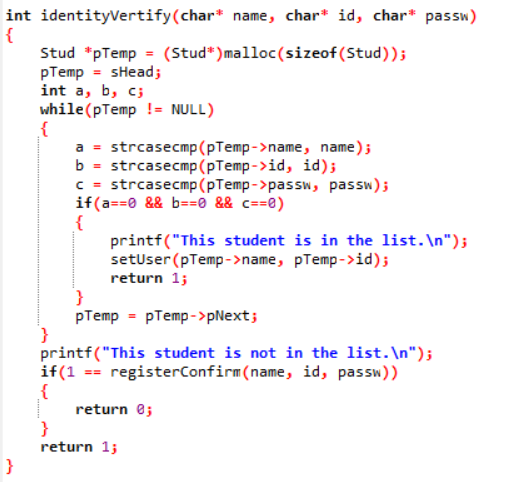
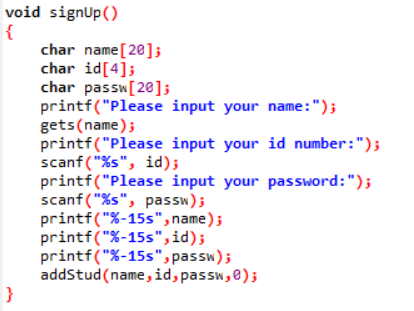


Actual result:



* 1. For the students, test if they can register in two different ways.

Test data:

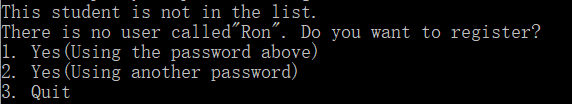


Required input:

Please input your name:Ron

Please input your id number:009

Please input your password:king

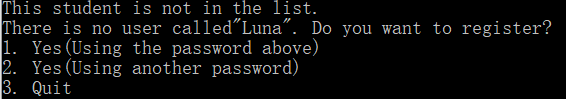


Your choice is:1

Please input your name:Luna

Please input your id number:008

Please input your password:123



Your choice is:2

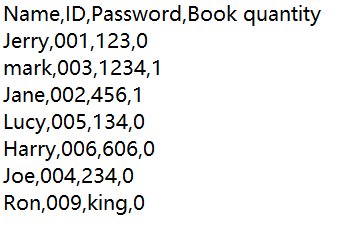
Please input the password:moon

Expected output:

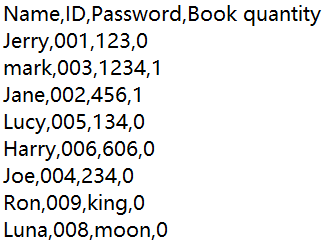
In the student file, will add two new users with zero book borrowed.

Actual result:

[9L4TZNT[@}{~@RQVRU%_%I



After the second time register:



3.3 For the librarian, test if the add and remove functions works, and check the information in the corresponding file.

Test data:

void addBookInTheList();

void removeBook();

These two functions are too long, so check them in the code is more convenient.

Required input:

Your choice is:3(add not existed books)

Please input the name of the book:book6

Please input the number of the book:0006

Please input the author of the book:Snape

Please input the adding quantity of the book:2

Your choice is:3(add existed books)

Please input the name of the book:book2

Please input the number of the book:0002

This book is already in the list.

Please input the adding quantity of the book:2

Your choice is:4(decrease the quantity of a book)

Please input the name of the book:book2

Please input the number of the book:0002

What do you want to do with this kind of book:

1. Remove the entire kind.

2. Decrease the quantity of it.

Your choice is:2

Please input the decreasing quantity of the book:3

Your choice is:4(remove the entire kind of book)

Please input the name of the book:book6

Please input the number of the book:0006

What do you want to do with this kind of book:

1. Remove the entire kind.

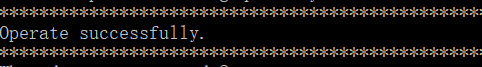
2. Decrease the quantity of it.

Your choice is:1

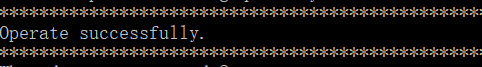
Expected output:

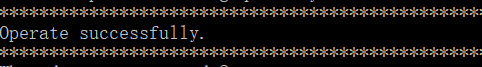
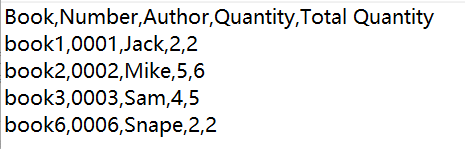
The file book.txt will change with the operation of adding and removing books.

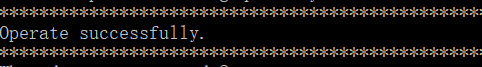
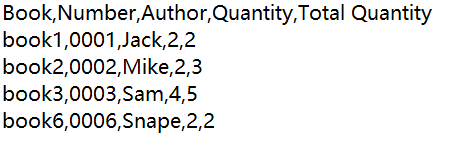
Actual result:

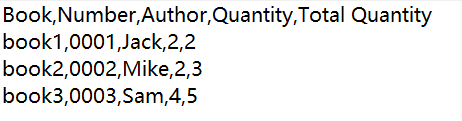
(add not existed books)

![3($H9PP{W25SILM]~TA]([X](data:image/png;base64,)

(add existed books)

(decrease the quantity of a book)

(remove the entire kind of book)



3.4 For the students, test if the borrow and return functions works, and check the information in the corresponding file.

Test data:

void borrowBook();

void returnBook();

Required input:

Your choice is:3(borrow)

Please input the name of the book:book3

Please input the number of the book:0003

Your choice is:4(return)

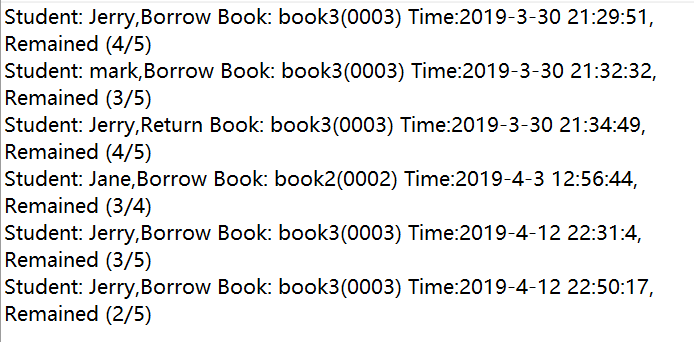
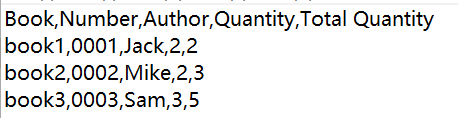
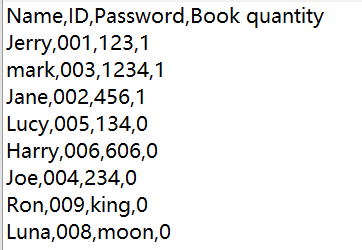
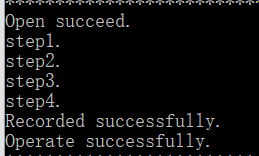
Please input the name of the book:book3

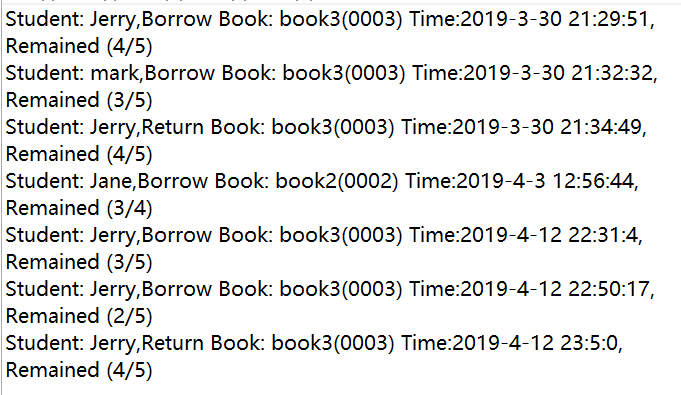
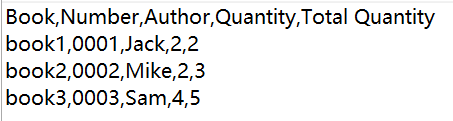
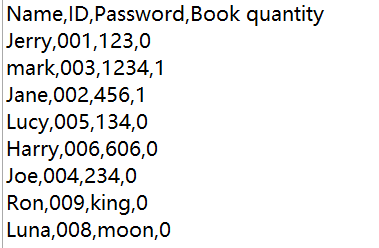
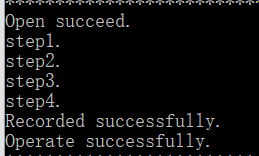
Please input the number of the book:0003

Expected output:

The file record.txt will show the new record information if borrowing or returning. And the file of the student’s and books’ will change correspondingly.

Actual result:





3.5. Try some invalid data.

Test data:

Nearly all the code.

Required input:

Operation option: 7

Identity: 4

Borrow not existed book:

Your choice is:3

Please input the name of the book:book4

Please input the number of the book:0004

Return book which is not borrowed:

Your choice is:4

Please input the name of the book:book1

Please input the number of the book:0001

Remove not existed book:

Please input the name of the book:book4

Please input the number of the book:0004

What do you want to do with this kind of book:

1. Remove the entire kind.

2. Decrease the quantity of it.

Your choice is:1

Expected output:

Operation option: Invalid data.

Identity:Invalid data.

Borrow not existed book: This book is not in the list. Please check the name and number carefully.(And recycle the option list.)

Return book which is not borrowed: All this kind of book is already here.(And recycle the option list.)

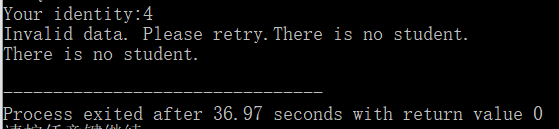
Remove not existed book: Can not find this book.(And recycle the option list.)

Actual result:

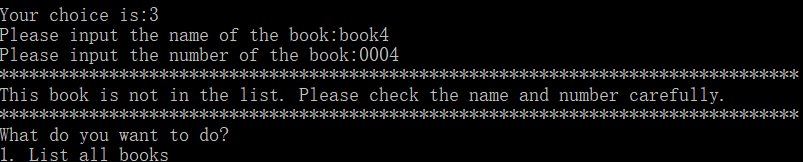
Operation option:

JOY6)9G5BM~(YCW%ZZTIF_M

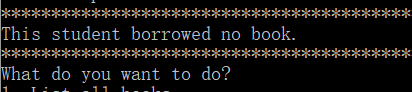
Identity:



Borrow not existed book:

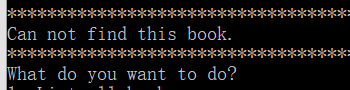


Return book which is not borrowed:



This is because the test used the student Jerry who did not borrowed any book, so the situation is also suitable.

Remove not existed book:



Testing summary:

As the pictures shown above, nearly all the tests passed as we expected.

1. **Personal reflection**
2. Things went well with this project:

At first, the structure of this program is clear and readable. Then in this project, the basic functions can work well and have some abilities to recognize errors. And it can tolerate some mistakes by users’ irregular operations like input some invalid data. During testing, some flaws were corrected so that the program will work more smoothly.

1. The hardest part of this work and the way to address this for the future:

The part of file operations is hardest. Because this need us to combine the knowledge of file and linked list. If we want to read something from a file to a linked list, it needs us to understand clearly how the pointers works and which step they are in. In the future, I will practice more about linked list and try to analysis memory allocation, so that my understanding of such structures would be more precise.

1. Acquired knowledge:

It is “Linked list”, as I have said for many time before. It forced me to understand more about how the memory is allocated for our functions and variables, and I will consider more about saving space and improve the efficiency of my program. Also the technique of debugging also helped a lot. By debug step by step, some implicit errors will show their true color.

1. **Some deficiency:**
2. When the books in the file is deleted one by one till the end, there wold be something wrong, because the removeBook() function did not considered when the linked list only have two or one node.
3. In the recordOper() function, there must be some printed information like “step”. They can help the program go on. Though I am not sure why the program will stop in the half way without them.
4. There always some printed information which make the interface like a mess. It is not easy to make it perfect, so I remained them there.