Library Management System

Code Documentation

Submitted by:
- Sameer Ahmed 24-101318
- Omar Yasser 24-101451
- Kareem Ahmed 24-101379
- Mohamed Ayman 24-101127
- Kirollos Ihab 24-101383
- Youssef Khaled 24-101359
- Abdullah Abdulrhman 24-101329

Contents

1 Introduction

This document provides a comprehensive explanation of the Library Management System code. It includes the structure, functionality of each section, and a user guide for available commands. The system is implemented in C and is designed to manage books, members, and borrowing transactions efficiently.

2 Code Overview

The program serves as a Library Management System that tracks books, members, and borrowing records. The core functionalities include adding books and members, recording borrow transactions, and providing various data queries for management purposes.

2.1 Global Variables

The program utilizes several global arrays to store data for books, members, and borrowing transactions:

- `book_ids`, `book_copies`, `book_borrowed_counts`: Store book details like IDs, number of copies, and borrow counts.
- `member_ids`, `member_borrowed_counts`: Store member details like IDs and the count of borrowed books.
- `borrow_book_ids`, `borrow_member_ids`, `borrow_dates`: Store borrowing transaction details including book IDs, member IDs, and dates.

2.2 Input Procedure

Input data is provided via the terminal in a structured format. Below is an example illustrating the expected input format:

```
Books:
101 5
102 3
103 7
104 2
105 0
Members:
201000
202000
203000
Borrowed_Books:
101 201000 01/12/2024
101 202000 01/12/2024
102 202000 03/12/2024
103 201000 03/12/2024
104 202000 01/12/2024
^Z
```

The input begins with book data, followed by member data, and finally borrowing transaction data. Each section is marked with a specific header.

3 Addition Commands

3.1 Add_Book

This function adds a new book to the library's collection. It assigns the provided book ID and the number of copies to the respective arrays and increments the total book count.

```
void add_book(int id, int copies) {
    book_ids[book_count] = id;
    book_copies[book_count] = copies;
    book_borrowed_counts[book_count] = 0;
    book_count++;
}
```

Listing 1: Add Book Function

3.2 Add_Member

This function registers a new library member. It assigns the provided member ID to the members array, initializes their borrow count to 0, and increments the total member count.

```
void add_member(int id) {
    member_ids[member_count] = id;
    member_borrowed_counts[member_count] = 0;
    member_count++;
}
```

Listing 2: Add Member Function

3.3 Add_Borrow

This function logs a borrowing event. It records the book ID, member ID, and borrowing date in their respective arrays, and increments the count of borrowed books and members who borrowed books. Additionally, it updates the total borrow counts for the specific book and member.

```
void add_borrow(int book_id, int member_id, const char *date) {
    borrow_book_ids[borrow_count] = book_id;
    borrow_member_ids[borrow_count] = member_id;
    strcpy(borrow_dates[borrow_count], date);
    borrow_count++;
    for (int i = 0; i < book_count; i++) {
        if (book_ids[i] == book_id) {
            book_borrowed_counts[i]++;
            break;
        }
    }
    for (int i = 0; i < member_count; i++) {
        if (member_ids[i] == member_id) {
            member_borrowed_counts[i]++;
            break;
        }
    }
}
```

Listing 3: Add Borrow Function

4 Query Commands

4.1 Number_Books

Displays the total number of books in the library by simply printing the value stored in the variable `book_count`, which is updated whenever a book is added or removed from the system.

```
void number_books() {
    printf("%d\n", book_count);
}
```

Listing 4: Number Books Function

Input:

`Number_Books`

Output:

`5`

4.2 Number_Members

Displays the total number of members in the library by printing the value of `member_count`, which keeps track of all registered library members.

```
void number_members() {
    printf("%d\n", member_count);
}
```

Listing 5: Number Members Function

Input:

`Number_Members`

Output:

`3`

4.3 Book_ID_Min

Finds the book with the smallest ID by iterating through the array `book_ids` and comparing each ID to the current minimum. The smallest ID is then printed. If there are no books, it outputs `none`.

```
void book_id_min() {
    if (book_count == 0) {
        printf("none\n");
        return;
    }
    int min_id = book_ids[0];
    for (int i = 1; i < book_count; i++) {
        if (book_ids[i] < min_id) {
            min_id = book_ids[i];
        }
    }
    printf("%d\n", min_id);
}
```

Listing 6: Book ID Min Function

Input:

`Book_ID_Min`

Output:

101

4.4 Books_Available

Lists all books with available copies for borrowing by checking if the number of borrowed copies is less than the total copies for each book. If no such books exist, it outputs `none`.

```
void books_available() {
    int found = 0;
    for (int i = 0; i < book_count; i++) {
        if (book_copies[i] > book_borrowed_counts[i]) {
            printf("%d\n", book_ids[i]);
            found = 1;
        }
    }
    if (found == 0) printf("none\n");
}
```

Listing 7: Books Available Function

Input:

`Books_Available`

Output:

101
102
103
104

4.5 Most_Borrowed

Identifies the most borrowed book(s) by finding the maximum value in `book_borrowed_counts`. It then prints the IDs of books that match this maximum value. If no books have been borrowed, it outputs `none`.

```
void most_borrowed() {
    if (book_count == 0) {
        printf("none\n");
        return;
    }
    int max_borrows = 0;
    for (int i = 0; i < book_count; i++) {
        if (book_borrowed_counts[i] > max_borrows) {
            max_borrows = book_borrowed_counts[i];
        }
    }
    if (max_borrows == 0) {
        printf("none\n");
        return;
```

```
    }
    for (int i = 0; i < book_count; i++) {
        if (book_borrowed_counts[i] == max_borrows) {
            printf("%d\n", book_ids[i]);
        }
    }
}
```

Listing 8: Most Borrowed Function

Input:

`Most_Borrowed`

Output:

`101`

4.6 List_Book_Borrowers

Lists all members who have borrowed a specific book by iterating through the `borrow_book_ids` array and checking for matches with the given book ID. It then prints the corresponding member IDs.

```
void list_book_borrowers(int book_id) {
    int found = 0;
    for (int i = 0; i < borrow_count; i++) {
        if (borrow_book_ids[i] == book_id) {
            printf("%d\n", borrow_member_ids[i]);
            found = 1;
        }
    }
    if (found==0) printf("none\n");
}
```

Listing 9: List Book Borrowers Function

4.7 List_Member_Books

Lists all books borrowed by a specific member by iterating through `borrow_member_ids` and matching the given member ID. The corresponding book IDs are then printed.

```
void list_member_books(int member_id) {
    int found = 0;
    for (int i = 0; i < borrow_count; i++) {
        if (borrow_member_ids[i] == member_id) {
            printf("%d\n", borrow_book_ids[i]);
            found = 1;
        }
    }
    if (found == 0) printf("none\n");
}
```

Listing 10: List Member Books Function

Input:

`List_Member_Books 201000`

Output:

```
101
103
```

4.8 Members_Less

Lists members who have borrowed fewer than a specified number of books by iterating through `member_borrowed_cou` and printing the IDs of members who meet this criterion. If no such members exist, it outputs `none`.

```c
void members_less(int n) {
    int found = 0;
    for (int i = 0; i < member_count; i++) {
        if (member_borrowed_counts[i] < n) {
            printf("%d\n", member_ids[i]);
            found = 1;
        }
    }
    if (found == 0) printf("none\n");
}
```

Listing 11: Members Less Function

Input:

```
Members_Less 3
```

Output:

```
201000
203000
```

4.9 Books_Unborrowed

Lists all books that have never been borrowed by checking if the borrow count for each book in `book_borrowed_count` is zero. If no such books exist, it outputs `none`.

```c
void books_unborrowed() {
    int found = 0;
    for (int i = 0; i < book_count; i++) {
        if (book_borrowed_counts[i] == 0) {
            printf("%d\n", book_ids[i]);
            found = 1;
        }
    }
    if (found == 0) printf("none\n");
}
```

Listing 12: Books Unborrowed Function

Input:

```
Books_Unborrowed
```

Output:

```
105
```

4.10 Books_Borrowed_Days

Counts the number of unique days on which books were borrowed by comparing the dates in `borrow_dates`. It checks for duplicates before incrementing the count of unique days.

```
void books_borrowed_days() {
    int unique_days = 0;
    for (int i = 0; i < borrow_count; i++) {
        int is_unique = 1;
        for (int j = 0; j < i; j++) {
            if (strcmp(borrow_dates[i], borrow_dates[j]) == 0) {
                is_unique = 0;
                break;
            }
        }
        if (is_unique) unique_days++;
    }
    printf("%d\n", unique_days);
}
```

Listing 13: Books Borrowed Days Function

Input:

Books_Borrowed_Days

Output:

2

4.11 Books_Per_Member

Displays the number of books borrowed by each member by iterating through `member_borrowed_counts` and printing each member ID along with their corresponding count.

```
void books_per_member() {
    for (int i = 0; i < member_count; i++) {
        printf("%d %d\n", member_ids[i], member_borrowed_counts[i]);
    }
}
```

Listing 14: Books Per Member Function

Input:

Books_Per_member

Output:

201000 2
202000 3
203000 0

4.12 Overlapping_Borrowers

Identifies members who borrowed the same book on the same day by checking for matching `borrow_book_ids` and `borrow_dates` in the borrow records. It prints the IDs of overlapping borrowers.

```
void overlapping_borrowers(int book_id) {
    int found = 0;
    for (int i = 0; i < borrow_count; i++) {
        if (borrow_book_ids[i] == book_id) {
            for (int j = i + 1; j < borrow_count; j++) {
                if (borrow_book_ids[j] == book_id && strcmp(borrow_dates[i],
                   borrow_dates[j]) == 0) {
                    printf("%d\n%d\n", borrow_member_ids[i], borrow_member_ids[j]);
                    found = 1;
                }
            }
        }
    }
    if (found==0) printf("none\n");
}
```

Listing 15: Overlapping Borrowers Function

Input:

```
Overlapping_Borrowers
```

Output:

```
201000
202000
```

4.13 Quit

Terminates the library management system by breaking the loop and finishing the program.

```
 while (gets(input)) {

        if (strcmp(input, "Quit") == 0) break;
        //The other commands
        }
        printf("Thanks!")
```

Listing 16: Quit Function

Input:

```
Quit
```

Output:

```
Thanks!
```