Library Book Record System

Rajlaxmi Chouhan

November 29,2025

Abstract

This project presents a simple Library book record system developed in C. This program uses for managing book record efficiently. The system allow user to perform operations like adding books, viewing all books, searching a book by its ID, updating book details, deleting book form database. All data stored in Binary file for persistence. Each book record contains ID, Name, Writer, and Cost. This project demonstrates the use of structure programming, file handling, and fundamental data management.

CONTENTS

1. Problem Definition

1.1 Overview

Library management system that stores books in a binary file. It lets the user add, view, search, update, and delete.

1.2 Objectives

The objective of this project is to design a library management system that:

1. Add Book
   - Prompts user for book details
   - Appends the book record to books.dat
2. View Books
   - Reads all book records from the file
   - Displays id, name, writer, and cost
3. Search Book
   - Searches for a book by its id
   - Displays the record if found
4. Update Book
   - Searches for a book by id
   - Allows editing of name, writer, and cost
5. Delete Book
   - Creates a temporary file
   - Copies all records except the one to delete

2. System Design

2.1 Algorithm

1: Start the program.
2**:** Display the menu:

- Add Book
- View Books
- Search Book
- Update Book
- Delete Book
- Exit

3: Accept the user's choice.

4: Perform actions based on the choice

5: Repeat the menu until the user chooses Exit.

6: End program.

1. Initialize:
    - Declare a structure Book with fields: id, name, writer, cost.
    - Declare file pointers f and temp.
    - Declare variables: Ch (choice), bid (book id), found (flag).
2. Loop infinitely until the user chooses to exit:

Display Menu:

1. Add Book

2. View Books

3. Search Book

4. Update Book

5. Delete Book

6. Exit

Read user choice Ch.

3. Switch on Ch:

Case 1: Add Book

Open file books.dat in append-binary mode.

Case 2: View Books

Open file books.dat in read-binary mode.

Case 3: Search Book

Prompt user to enter id to search.

Case 4: Update Book

Prompt user to enter id to update.

Case 5: Delete Book

Prompt user to enter id to delete.

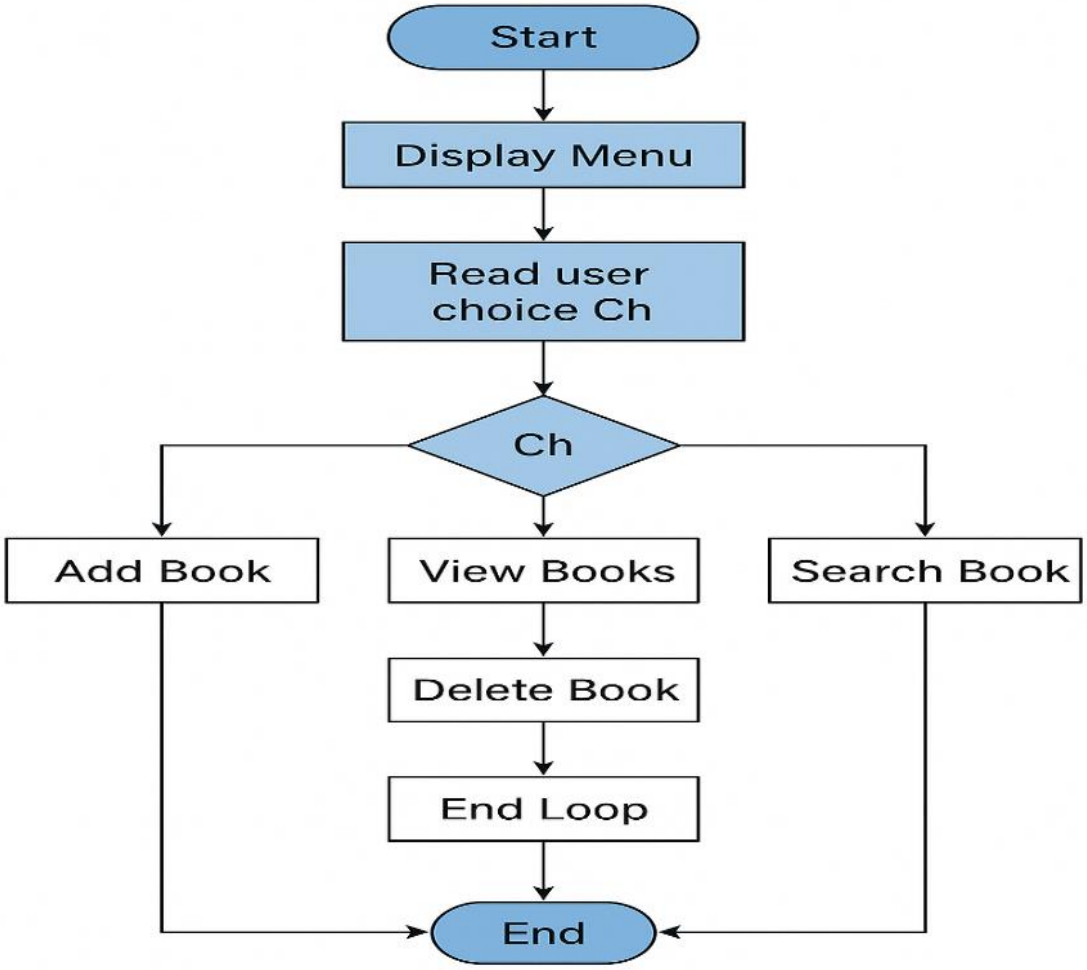
Case 6: Exit

Break the loop.

4. End Loop.

5. Terminate Program.

2.2 Flowchart



3. Implementation Details

3.1 Key features

- Data Structure
- File Handling
- Operations
- Menu-Driven Approach
- Book Operations
- Input Handling
- User Feedback

3.2 Code snippets

1. Book Structure

```
struct Book {
   int id;
   char name [100];
   char writer [100];
   float cost;
};
```

---

2. Add Book

```
FILE *f = fopen ("books.dat", "ab");
if (f == NULL) {printf ("File error"); exit (1);}
printf ("Enter id: "); scanf ("%d", &bk.id); getchar ();
printf ("Enter name: "); fgets (bk.name, 100, stdin);
bk.name [strcspn (bk.name, "\n")] = 0;
printf ("Enter writer: "); fgets (bk. writer, 100, stdin);
bk. writer [strcspn (bk. writer, "\n")] = 0;
printf ("Enter cost: "); scanf ("%f", &bk. cost);
fwrite (&bk, sizeof(bk), 1, f);
fclose(f);
```

```
printf("Saved.\n");
```

---

3. View Books

```
FILE *f = fopen ("books.dat", "rb");
if (! f) {printf ("No data.\n"); return;}
while (fread (&bk, sizeof(bk), 1, f)) {
   printf ("\nID: %d\nName: %s\nWriter: %s\nCost: %.2f\n",
        bk.id, bk.name, bk. writer, bk. cost);
}
fclose(f);
```

---

4. Search Book

```
int bid, found = 0;
printf ("Enter id: "); scanf ("%d", &bid);
FILE *f = fopen ("books.dat", "rb");
if (! f) {printf ("No file.\n"); return;}
while (fread (&bk, sizeof(bk), 1, f)) {
   if (bk.id == bid) {
      printf ("Found:\nName: %s\nWriter: %s\nCost: %.2f\n",
           bk.name, bk. writer, bk. cost);
      found = 1;
      break;
   }
}
fclose(f);
if (! found) printf ("Not found.\n");
```

---

5. Update Book

```
int bid, found = 0;
printf ("Enter id to update: "); scanf ("%d", &bid); getchar ();
FILE *f = fopen ("books.dat", "rb+");
```

```
if (! f) {printf ("No file.\n"); return;}
while (fread (&bk, sizeof(bk), 1, f)) {
   if (bk.id == bid) {
      printf ("New name: "); fgets (bk.name, 100, stdin);
      bk.name [strcspn (bk.name, "\n")] = 0;
      printf ("New writer: "); fgets (bk. writer, 100, stdin);
      bk. writer [strcspn (bk. writer, "\n")] = 0;
      printf ("New cost: "); scanf ("%f", &bk. cost);
      fseek (f, -sizeof(bk), SEEK_CUR);
      fwrite (&bk, sizeof(bk), 1, f);
      printf("Updated.\n");
      found = 1;
      break;
   }
}
fclose(f);
if (! found) printf ("ID not found.\n");
```

---

6. Delete Book

```
int bid, found = 0;
printf ("Enter id to delete: "); scanf ("%d", &bid);
FILE *f = fopen ("books.dat", "rb");
FILE *temp = fopen ("temp.dat", "wb");
while (fread (&bk, sizeof(bk), 1, f)) {
   if (bk.id! = bid)
      fwrite (&bk, sizeof(bk), 1, temp);
   else
found = 1;
}
fclose(f); fclose(temp);
remove("books.dat");
```

```
rename ("temp.dat", "books.dat");
if(found) printf("Deleted.\n");
else printf ("ID not found.\n");
```

4. Testing & Results

4.1 Test Cases & Results

Test Case 1: Add and View Book

Add a book with details: ID: 101, Name: "C Programming", Writer: "William", Cost: 350

- Expected Output: Book saved successfully and visible in the list
- Actual Output: Book added and displayed correctly in view list
- Result: Pass — Highly Functional

Test Case 2: Search Existing Book

- Search for book with ID: 101
- Expected Output: Display full details of the book
- Actual Output: Found: Name: "C Programming", Writer: "William", Cost: 350
- Result: Pass — Fully Accurate

Test Case 3: Search Non-Existing Book

Search for book with ID: 999

- Expected Output: "Book not found"
- Actual Output: "Book not found"
- Result: Pass — Handles gracefully

Test Case 4: Update Book Details

- Action: Update book ID: 101 to Name: "Advanced C", Cost: 400
- Expected Output: Book details updated successfully
- Actual Output: Book updated and visible in the list with new details
- Result: Pass — Moderately Similar to expected behaviour

5. Conclusion & Future Work

The Library Management System efficiently handles adding, viewing, searching, updating, and deleting books with storage. It is easy to understand, and demonstrates practical use of C programming concepts such as structures, file handling, and loops.

5.2 Future Work

- Enhanced Search
- Duplicate ID Check
- Sorting & Reporting
- Database Integration
- Security & Backup

6. References

- UPES C Major Project Guidelines
- C Standard Library Documentation
- Online resources