Library Management System

Introduction:

This project is all about building a simple yet useful user-friendly library management program using C. The idea is to create a tool that makes it easy to organize books in a database. Whether it's adding new books, removing old ones, or searching for something specific, this program will handle it all.

The goal is to make managing a small library database straightforward and efficient while learning more about how C programming works in real-world scenarios. This project is also a great way to practice working with files and handling data in a structured way. By the end of this, we'll have a simple yet powerful program that's practical, easy to use, and a testament to the power of coding!

Structure:

We use the Structure to store the details of the book under struct named book. And we used typedef to simplify the use of the structure. When we use typedef we need to give a name for our structure so whenever we want to store it in there, we only need to put the structure's name. you can see it in (Figure 1).

Algorithm Steps:

1. **Initialize Program:**
    - Allocate memory for storing book data.
    - Load existing book data from the CSV file.
2. **Main Menu Loop:**
    - Display options to the user:
        1. Add Book
        2. Remove Book

3. Display Books
4. Search Book
5. Check Availability
6. Exit

- Take user input for menu choice.

3. **Handle User Choices:**
   - **Choice 1: Add Book**
     - Check if the library is full.
     - Prompt for book details: ID, author, title, genre, quantity.
     - Store the new book in the array.
     - Save updated data to the CSV file. o **Choice 2: Remove Book**
     - Prompt for the Book ID to remove.
     - Search for the book.
     - If found, remove the book by shifting the array.
     - Save updated data to the CSV file. o **Choice 3: Display Books**
     - If no books exist, inform the user.
     - Print each book's details in tabular format.
   - **Choice 4: Search Book**
     - Prompt for the Book ID.
     - Search and display the book's details if found.
   - **Choice 5: Check Availability**
     - Prompt for the Book ID.
     - If found, check if the quantity is greater than zero.
     - Display availability status. o **Choice 6: Exit**

- Free allocated memory.
- Exit the program.

4. **File Handling Functions:**
    - **saveToCSV:** Writes the current list of books to the CSV file.
    - **loadFromCSV:** Loads book data from the CSV file.

Functions:

Here are some functions that we are going to use in our code: (Figure 1)

1. Add Books: Quickly add new books to the database.
2. Remove Books: Delete books using their unique ID.
3. Display Books: See everything in the database neatly displayed.
4. Search for books: Find any book instantly using its ID.
5. Save changes: Save all your work into a CSV file so nothing gets lost.
6. Load data: Open your saved CSV file and continue from where you left off.
7. Checking Availability: It will check the quantity and the availability of the book by the ID of the book.

Main Function:

The Main Function is going to be the main part of the program since we are going to use dynamic memory to handle the book records quickly and make it flexible enough to manage the large amount of data. When we start the program, it will load everything from CSV so users can pick up right where they left. The program will give a list of things such as adding, removing, displaying, searching and checking the availability and it will ask the user to choose one of the options to do some operation. At the end before closing the program will save everything inside the CSV file to keep things organized and running perfectly.

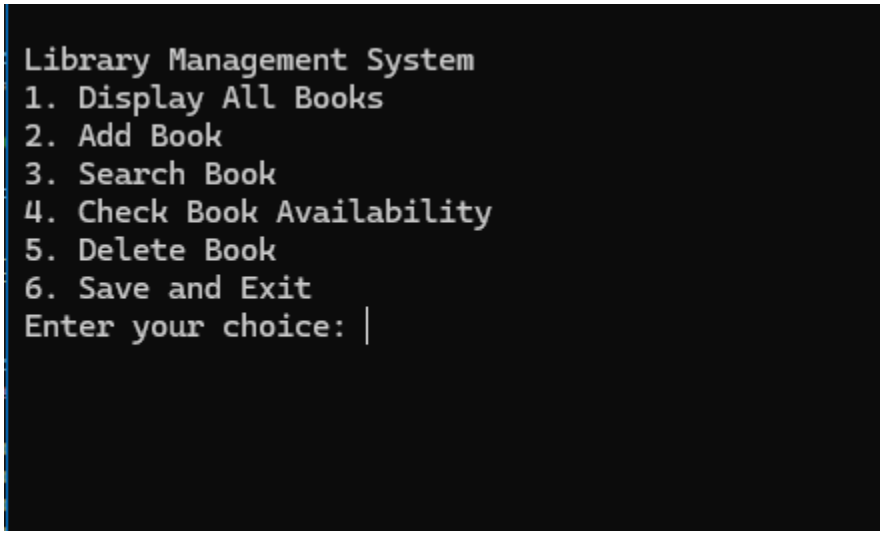
```c
int main()
{
    Book books[100];  // Array to store up to 100 books
    int count = 0;    // Current number of books
    FILE* file;

    // Load books from the CSV file
    file = fopen("library.csv", "r");
    if (file != NULL)
    {
        loadBooks(file, books, &count);
        fclose(file);
    }

    int choice = 0;
    while (choice != 6)
    {
        printf("\nLibrary Management System\n");
        printf("1. Display All Books\n");
        printf("2. Add Book\n");
        printf("3. Search Book\n");
        printf("4. Check Book Availability\n");
        printf("5. Delete Book\n");
        printf("6. Save and Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice)
        {
        case 1:
            displayBooks(books, count);
            break;
        case 2:
            addBook(books, &count);
            break;
        case 3:
            searchBook(books, count);
            break;
        case 4:
            checkAvailability(books, count);
            break;
        case 5:
            deleteBook(books, &count);
            break;
        case 6:
            // Save books to the CSV file
            file = fopen("library.csv", "w");
            if (file != NULL)
            {
                saveBooks(file, books, count);
                fclose(file);
            }
            printf("Data saved. Exiting...\n");
            break;
        default:
            printf("Invalid choice. Please try again.\n");
        }
    }

    return 0;
}
```



Add Function:

The addBook function is a very important part of the Library Management System, making it easy to add new books to the collection. It guides the user to input details like the book's ID, author, title, genre, and quantity. Before adding, it checks if the library has reached its maximum capacity (MAX_BOOKS) and lets the user know if no more books can be added. The function uses simple methods to handle multi-word inputs and ensures the details are formatted correctly. Once the book is added, the total count is updated, and the user gets confirmation. To keep the records safe, it immediately saves the updated data to a CSV file, so nothing is lost. This function keeps the process simple, reliable, and efficient, helping users grow the library with ease.

```c
// Add a new book
void addBook(Book books[], int* count)
{
    printf("Enter Book ID: ");
    scanf("%d", &books[*count].bookID);
    printf("Enter Title: ");
    scanf(" %[^\n]s", books[*count].title);
    printf("Enter Author: ");
    scanf(" %[^\n]s", books[*count].author);
    printf("Enter Quantity: ");
    scanf("%d", &books[*count].quantity);
    (*count)++;
    printf("Book added successfully.\n");
}
```

```
Library Management System
1. Display All Books
2. Add Book
3. Search Book
4. Check Book Availability
5. Delete Book
6. Save and Exit
Enter your choice: 2
Enter Book ID: 1
Enter Title: Atomic Habits
Enter Author: James Clear
Enter Quantity: 3
Book added successfully.

Library Management System
1. Display All Books
2. Add Book
3. Search Book
4. Check Book Availability
5. Delete Book
6. Save and Exit
Enter your choice: 6
Data saved. Exiting...
```

	A	B	C	D
1	Book ID	Book Name	Author Name	Quantity
2	2	wf	bgr	3
3	3	efw	wr	4
4	1	Atomic Habits	James Clear	3
5				

Delete Function:

The removeBook function is a crucial part of the Library Management System, allowing users to efficiently remove books from the collection. It prompts the user to enter the ID of the book they wish to delete and then searches the library's records for a matching entry. If the book is found, it removes the entry by shifting subsequent records up to fill the gap, ensuring no empty slots remain. The total book count is updated, and the user receives confirmation that the book was successfully removed. If the book isn't found, an appropriate message informs the user. To maintain data integrity, the function automatically saves the updated collection to a CSV file, ensuring that all changes are securely stored. This streamlined process ensures accurate recordkeeping and keeps the library database up-to-date.

```c
// Delete a book
void deleteBook(Book books[], int* count)
{
    int id;
    printf("Enter Book ID to delete: ");
    scanf("%d", &id);
    for (int i = 0; i < *count; i++)
    {
        if (books[i].bookID == id)
        {
            for (int j = i; j < *count - 1; j++)
            {
                books[j] = books[j + 1];
            }
            (*count)--;
            printf("Book deleted successfully.\n");
            return;
        }
    }
    printf("Book not found.\n");
}
```

```
2. Add Book
3. Search Book
4. Check Book Availability
5. Delete Book
6. Save and Exit
Enter your choice: 5
Enter Book ID to delete: 2
Book deleted successfully.

Library Management System
1. Display All Books
2. Add Book
3. Search Book
4. Check Book Availability
5. Delete Book
6. Save and Exit
Enter your choice: 1

Book ID    Title                          Author              Quantity
3          efw                            wr                  4
1          Atomic Habits                  James Clear         3
```

	A	B	C	D
1	Book ID	Book Name	Author Name	Quantity
2	1	Atomic Habits	James Clear	3
3	2	stranger	albert	4
4	3	lolita	vladmir	6
5	4	same as ever	morgan housel	10
6	5	the mokn	jay shetty	45
7	6	rich dad poor dad	kiosaki	5
8				

Display Book Function:

```c
// Display all books
void displayBooks(Book books[], int count)
{
    printf("\n%-10s %-30s %-20s %-10s\n", "Book ID", "Title", "Author", "Quantity");
    for (int i = 0; i < count; i++)
    {
        printf("%-10d %-30s %-20s %-10d\n",
            books[i].bookID,
            books[i].title,
            books[i].author,
            books[i].quantity);
    }
}
```

The displayBooks function makes it easy to view all the books in the Library Management System. It first checks if there are any books in the collection. If the library is empty, it informs the user. If there are books, it shows their details, including the ID, author, title, genre, and quantity, in a clear and organized table format. This helps users quickly see what books are available and their information. The function keeps everything simple and user-friendly, making it easy to manage and review the library's collection.

```
Library Management System
1. Display All Books
2. Add Book
3. Search Book
4. Check Book Availability
5. Delete Book
6. Save and Exit
Enter your choice: 1

Book ID    Title                          Author              Quantity
1          Atomic Habits                  James Clear         3
2          stranger                       albert              4
3          lolita                         vladmir             6
4          same as ever                   morgan housel       10
5          the mokn                       jay shetty          45
6          rich dad poor dad              kiosaki             5
```

Search Function:

The searchBook function allows users to search for a book by its ID. When the user enters the book ID, the function goes through the library collection to find a match. If the book is found, it displays the book's details, including its ID, author, title, genre, and quantity. If no book matches the ID entered, it informs the user that the book was not found. This function helps users quickly locate specific books in the library's collection.

```c
// Search for a book
void searchBook(Book books[], int count)
{
    int searchType;
    printf("Search by:\n");
    printf("1. Book ID\n");
    printf("2. Title\n");
    printf("3. Author\n");
    printf("Enter your choice: ");
    scanf("%d", &searchType);

    if (searchType == 1) // Search by Book ID
    {
        int bookID;
        printf("Enter Book ID to search: ");
        scanf("%d", &bookID);
        printf("\n%-10s %-30s %-20s %-10s\n", "Book ID", "Title", "Author", "Quantity");
        for (int i = 0; i < count; i++)
        {
            if (books[i].bookID == bookID)
            {
                printf("%-10d %-30s %-20s %-10d\n",
                    books[i].bookID,
                    books[i].title,
                    books[i].author,
                    books[i].quantity);
                return; // Exit once the book is found
            }
        }
        printf("Book not found.\n");
    }
    else if (searchType == 2 || searchType == 3) // Search by Title or Author
    {
        char query[100];
        printf("Enter search query: ");
        scanf(" %[^\n]s", query);

        printf("\n%-10s %-30s %-20s %-10s\n", "Book ID", "Title", "Author", "Quantity");
        int found = 0;
        for (int i = 0; i < count; i++)
        {
            if ((searchType == 2 && strstr(books[i].title, query)) || // Search by Title
                (searchType == 3 && strstr(books[i].author, query)))   // Search by Author
            {
                printf("%-10d %-30s %-20s %-10d\n",
                    books[i].bookID,
                    books[i].title,
                    books[i].author,
                    books[i].quantity);
                found = 1;
            }
        }
        if (!found)
        {
            printf("No books found matching the query.\n");
        }
    }
    else
    {
        printf("Invalid search type.\n");
    }
}
```

```
Library Management System
1. Display All Books
2. Add Book
3. Search Book
4. Check Book Availability
5. Delete Book
6. Save and Exit
Enter your choice: 3
Search by:
1. Book ID
2. Title
3. Author
Enter your choice: 1
Enter Book ID to search: 3

Book ID    Title                          Author               Quantity
3          lolita                         vladmir              6
```

Check Availability:

The checkAvailability function helps users see if a specific book is available in the library. It asks for the book's ID and then searches through the collection to find it. If the book is found, it checks the quantity: if there are copies available, it tells the user how many are in stock. If the quantity is zero, it informs the user that the book is currently out of stock. If the book isn't found at all, the function lets the user know. This feature ensures users can quickly and easily find out if a book is ready for borrowing.

```c
// Check book availability
void checkAvailability(Book books[], int count)
{
    int id;
    printf("Enter Book ID to check availability: ");
    scanf("%d", &id);
    for (int i = 0; i < count; i++)
    {
        if (books[i].bookID == id)
        {
            printf("Book: %s, Quantity: %d\n", books[i].title, books[i].quantity);
            return;
        }
    }
    printf("Book not found.\n");
}
```

```
Library Management System
1. Display All Books
2. Add Book
3. Search Book
4. Check Book Availability
5. Delete Book
6. Save and Exit
Enter your choice: 4
Enter Book ID to check availability: 3
Book: lolita, Quantity: 6
```

SAVE AND LOAD FUNCTION:

The saveToCSV and loadFromCSV functions work together to ensure the library's book data is properly stored and retrieved. The loadFromCSV function reads the existing book records from a CSV file when the program starts, loading them into the program's memory. This ensures that any data added or removed in previous sessions is available when the program restarts. On the other hand, the saveToCSV function updates the CSV file by saving any changes made to the library, such as adding or removing books. Together, these functions maintain a consistent and up-to-date record of the library's collection, ensuring data is not lost and is always accessible.

```c
// Load Books
void loadBooks(FILE* file, Book books[], int* count)
{
    char line[256];
    int isFirstLine = 1; // Flag to skip the header row

    while (fgets(line, sizeof(line), file))
    {
        if (isFirstLine) // Skip the header line
        {
            isFirstLine = 0;
            continue;
        }

        // Parse the line into the book fields
        if (sscanf(line, "%d,%99[^,],%99[^,],%d",
            &books[*count].bookID,
            books[*count].title,
            books[*count].author,
            &books[*count].quantity) == 4) // Ensure all fields are read
        {
            (*count)++;
        }
        else
        {
            printf("Error reading line: %s\n", line);
        }
    }
}

// Save books to the CSV file
void saveBooks(FILE* file, Book books[], int count)
{
    fprintf(file, "Book ID,Book Name,Author Name,Quantity\n");
    for (int i = 0; i < count; i++)
    {
        fprintf(file, "%d,%s,%s,%d\n",
            books[i].bookID,
            books[i].title,
            books[i].author,
            books[i].quantity);
    }
}
```