**Step-1:**Start.

**Step-2:**Include necessary header files: stdlib.h, string.h, and stdio.h.

**Step-3:**Define constants and macros: MAX\_BOOKS for the maximum number of books and FILENAME for the binary file name.

**Step-4:**Declare an enumeration enum Category to represent book categories: Fictional, Physics, and History.

**Step-5:**Define a structure struct Book to store book details: book number, title, author, number of pages, category, and cost.

**Step-6:**Declare the following function prototypes :

void saveBookDetails(struct Book \*books, int count);

void displayCategoryWiseList(struct Book \*books, int count);

void displayCategoryWiseTotalAndAverage(struct Book \*books, int count);

void displayBookDetails(struct Book \*books, int count, int bookNumber);

void saveBookDetailsToFile(struct Book \*books, int count);

int loadBookDetailsFromFile(struct Book \*books);

void deleteBook(struct Book \*books, int \*count, int bookNumber);

void clearAllBooks(struct Book \*books, int \*count);

**Step-7:**Dynamically allocate memory for an array of struct Book to store book details (books).

**Step-8:**Declare bookCount , choice and initialize to bookCount=0.

**Step-9:**Display a welcome message and the main menu.

**Step-10:**Assign bookCount = loadBookDetailsFromFile(books), call the function and pass parameters, to check number of books

**Step-11:**Input choice.

**Step-12:**If user choice =1,Goto step (a).Else goto step 13.

**(a):**Call the function --- saveBookDetails(struct Book \*books, int count) and pass the parameters books, bookCount.

**(b):**If count<MAX\_BOOKS, Goto step (c), else goto step (l).

**(c):**Input book number.Goto step (d).

**(d):**Validate input for book details.Goto step (e).

**(e):**declare and initiate i to zero.

**(f):**Set a for loop: If i<count, goto step (g). Else goto step (j).

**(g):**If books[i].bookNumber == bookNumber.Goto step (h).Else goto step (i).

**(h):**Display book already exists. Goto step (c).

**(i):**Increment i. Goto step (f).

**(j):**Set books[count].bookNumber = bookNumber.

**(k):**Input book title, author, pages, category ,cost. Goto step (m).

**(l):**Display maximum limit reached and break out.

**(m):**Call the function --- saveBookDetailsToFile and pass the parameters: books, bookCount.

**(n):**Declare a FILE pointer file and open the file in "rb+" mode.

**(o):**If file != NULL,goto step (p),else goto step (r).

**(p):**Write data to file using fwrite. [fwrite(books, sizeof(struct Book), count, file)]

**(q):**Close the file.

**(r):**Display error opening file.

**(s):**Increment bookCount.Goto step 11.

**Step-13:**If user choice =2,Goto step (a).Else goto step 14.

**(a):**Call the function --- displayCategoryWiseList and pass the parameter: books, bookCount.

**(b):**Display heading to display Fictional category.

**(c):**Declare and set i=0.

**(d):**If i<count, goto step (e), else goto step (h).

**(e):**If books[i].category == 1, goto step (f), else goto step (g).

**(f):**Print details of book[i].

**(g):**Increment i.

**(h):**Repeat the process for physics and history.

**(i):**Goto step 11.

**Step-14:**If user choice =3,Goto step (a).Else goto step 15.

**(a):**Call the function --- displayCategoryWiseTotalAndAverage and pass the parameters : books, bookCount)

**(b):**Declare categoryChoice.

**(c):**Input categoryChoice.

**(d):**Check the validity of input number i.e is number b/w 1-3.

**(e):**Declare totalCost, averageCost,bookCount, i and set to 0.

**(f):**If i<count, goto step (g), else goto step (j).

**(g):**If books[i].category == categoryChoice, goto step (h), else goto (i).

**(h):**totalCost += books[i].bookCost and increment bookCount. Goto step (i).

**(i):**Increment i.

**(j):**If bookCount >0, goto step (k), else goto step (m).

**(k):**Calculate averageCost = totalCost/bookCount.

**(l):**Print totalCost and averageCost.

**(m):**Display no books found in selected category.

**(n):**Goto step 9.

**Step-15:**If choice=4,Goto step (a), else goto step 16.

**(a):**Declare bookNumber.

**(b):**Input bookNumber

**(c):**Check the validity of input.

**(d):**Call the function --- displayBookDetails and pass the parameters: books, bookCount, bookNumber

**(e):**Declare i and index, set index=-1,i=0.

**(f):**If i<count, goto step (g), else goto step (k).

**(g):**If books[i].bookNumber == bookNumber, goto step (h), else goto step (i).

**(h):**index=i.Goto step (k).

**(i):**Increment i.

**(k):**If index!=-1, goto step (l), else goto step (m).

**(l):**Display book[i] details.

**(m):**Display Book with the specified number not found.

**(n):**Goto step 9.

**Step-16:**If choice=5,Goto step (a), else goto step 17.

**(a):**Declare bookNumber.

**(b):**Input bookNumber.

**(c):**Call the function deleteBook and pass the parameters: books, &bookCount, bookNumber.

**(d):**Declare i,index.Set i=0, index=-1.

**(e):**If i<\*count,goto step (f), else goto step (i).

**(f):**If books[i].bookNumber == bookNumber, goto step (g), else goto step (h).

**(g):**index=i. Goto step (i).

**(h):**Increment i.Goto step (e).

**(i):**If index!=-1, goto step (j), else goto step (o).

**(j):**If i<\*count, goto step (k), else goto step (m).

**(k):**books[i] = books[i + 1].

**(l):**Increment i.Goto step (j).

**(m):**\*count--.

**(n):**Display book has been deleted.Goto step (p).

**(o):**Display Book number not found.Goto step (p).

**(p):**Call the saveBookDetailsToFile function and pass the parameters: books, bookCount.

**(q):**Goto step 9.

**Step-17:** If choice=6, goto step (a), else goto step 18.

**(a):**Call the function clearAllBooks and pass the parameters :books, &bookCount.

**(b):**Set \*count=0.

**(c):**Display all books cleared.

**(d):**Call the function -- saveBookDetailsToFile and pass the parameters :books, bookCount.

**(e):**Perform the function as told from step 12(m) to 12(r).

**(f):**Goto step 9.

**Step-18:** If choice=7, Display Exiting the program, goto step 20;else goto step 19.

**Step-19:** Display Enter proper Input.Goto step 20.

**Step-20:**Stop.