LIBRARY BOOKS

Presentation by Group-1 CSE

PPS | 1-1

KMCE

PROJECT - 3

TEAM MEMBERS

23P81A0528	K RUSHI	CSE
23P81A0502	N AKHILA	CSE
23P81A0523	J BABITHA	CSE
23P81A0552	S ISHANTH REDDY	CSE
23P81A0599	K HEMASHASHEENDRA	CSE

GUIDE: MR. UMESH GOGTE

SOFTWARE USED

- DEV C++
- C COMPILER
- WINDOWS OS

DOCUMENTATION

PROBLEM STATEMENT:

Save details of at least 10 books in a library in a binary file

REQUIREMENTS:

- Category Wise list.
- Category Wise total and average book cost
- For a given book number, give its details.

PROCEDURAL PROGRAMMING

- void saveBookDetails();
- void displayCategoryWiseList();
- void displayCategoryWiseTotalAndAverage();
- void displayBookDetails();
- void saveBookDetailsToFile();
- int loadBookDetailsFromFile();
- void deleteBook();
- void clearAllBooks()
- int main().

ALGORITHM

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.

ALGORITHM

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);
```

ALGORITHM

```
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.
```

ALGORITHM

```
(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 9.
```

ALGORITHM

```
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.
```

ALGORITHM

```
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.
```

ALGORITHM

```
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.
```

ALGORITHM

```
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.
```

ALGORITHM

```
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.
```

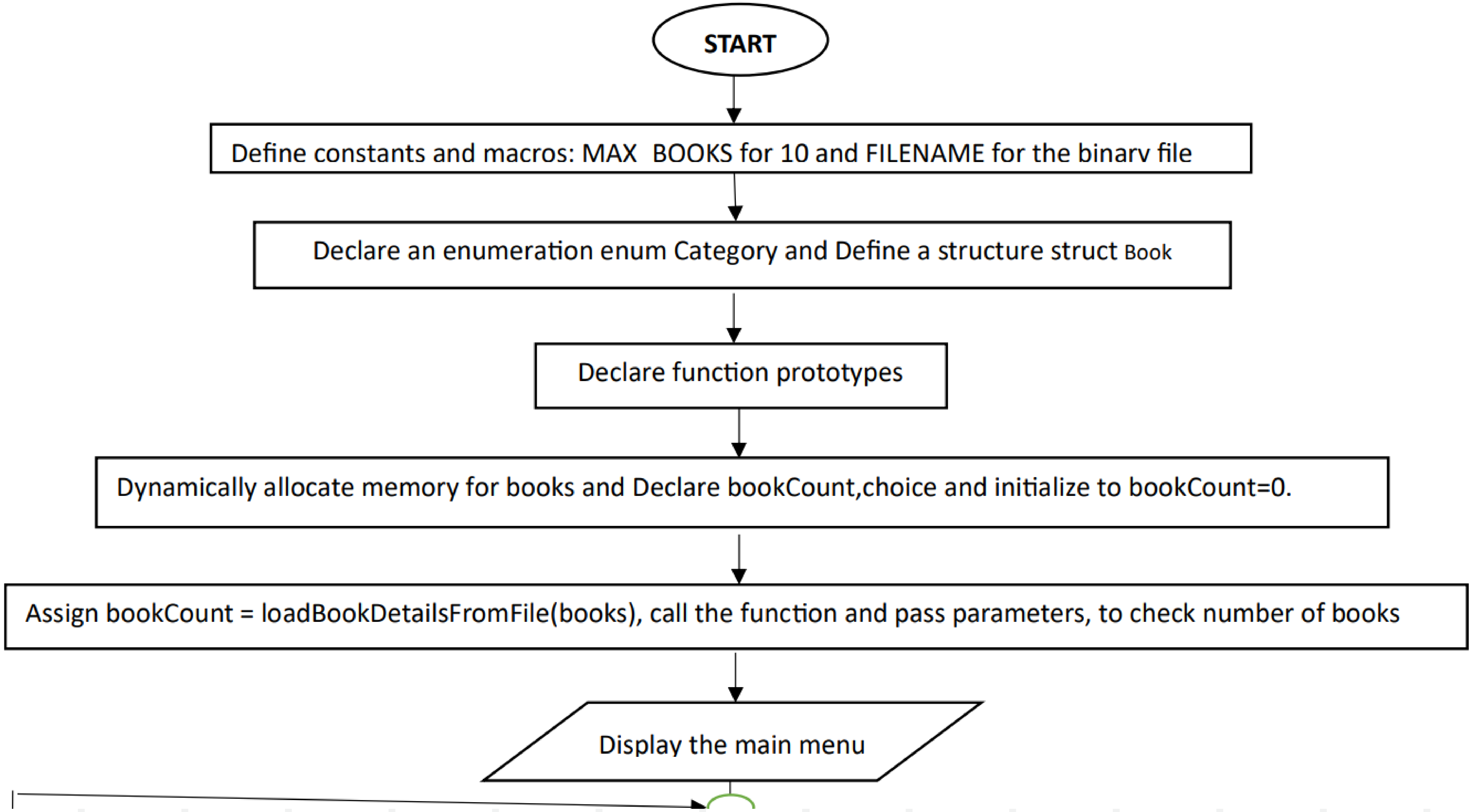
ALGORITHM

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

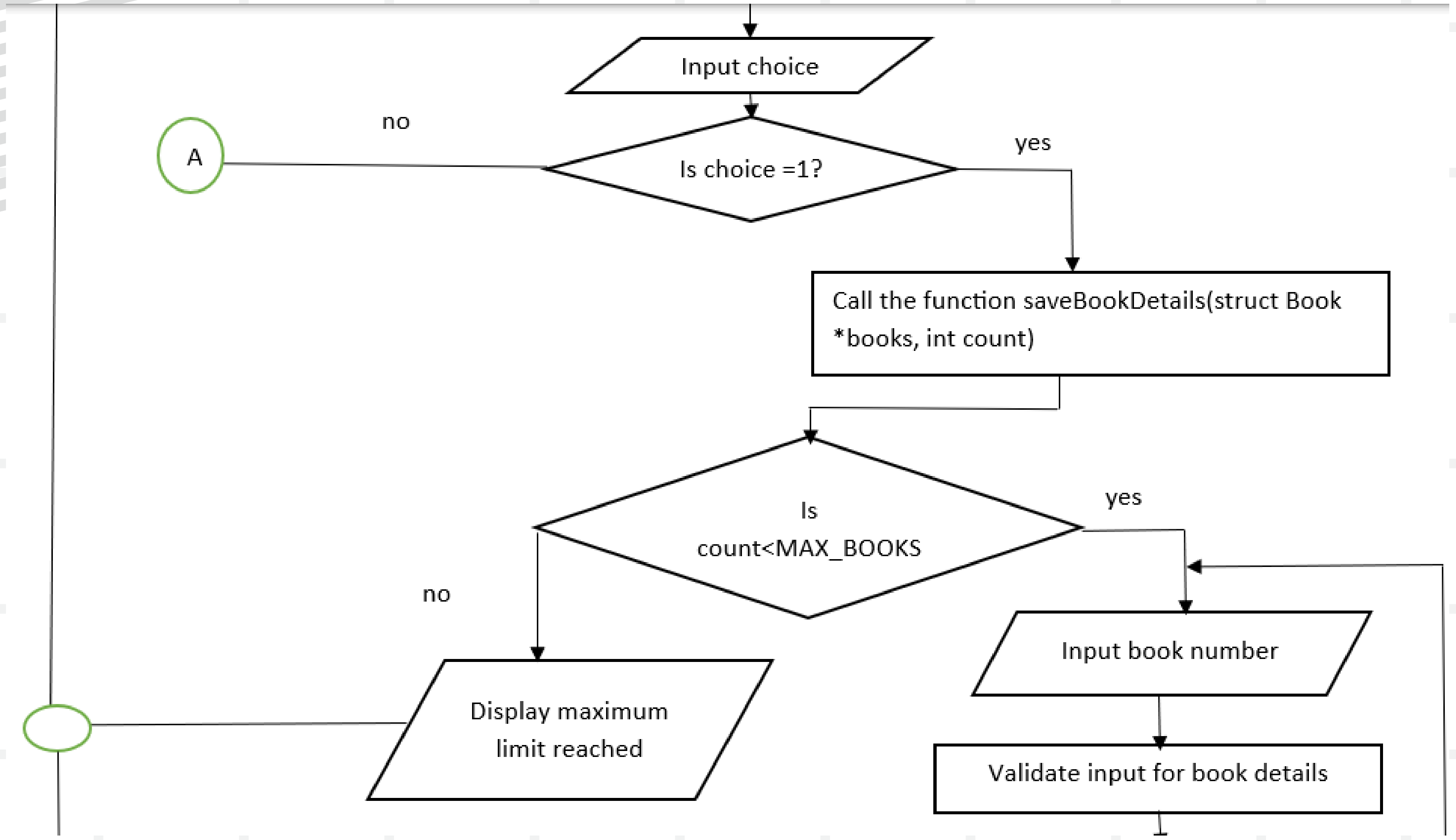
Step-19: Display Enter proper Input.Goto step 20.

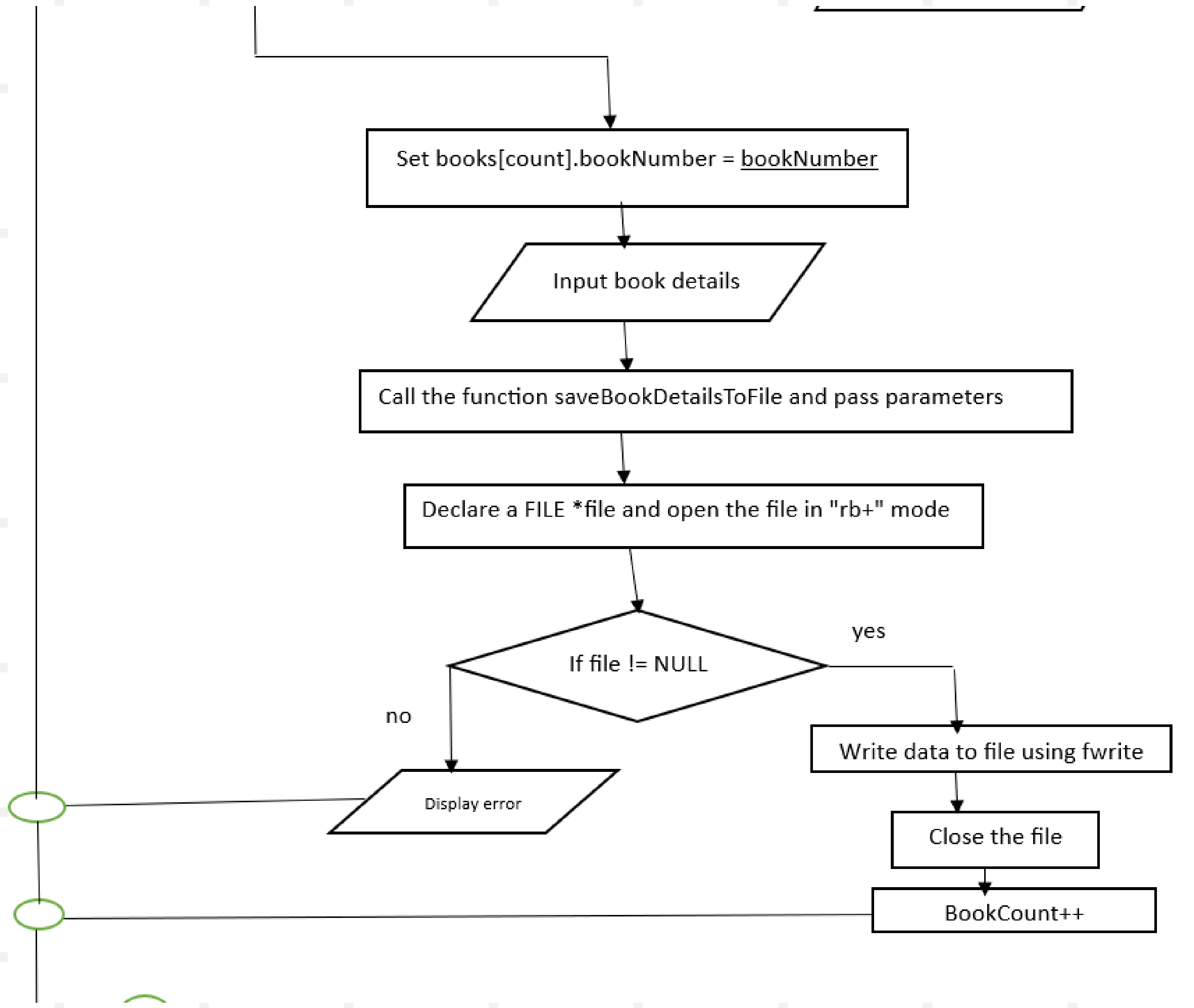
Step-20: Stop.

FLOWCHART

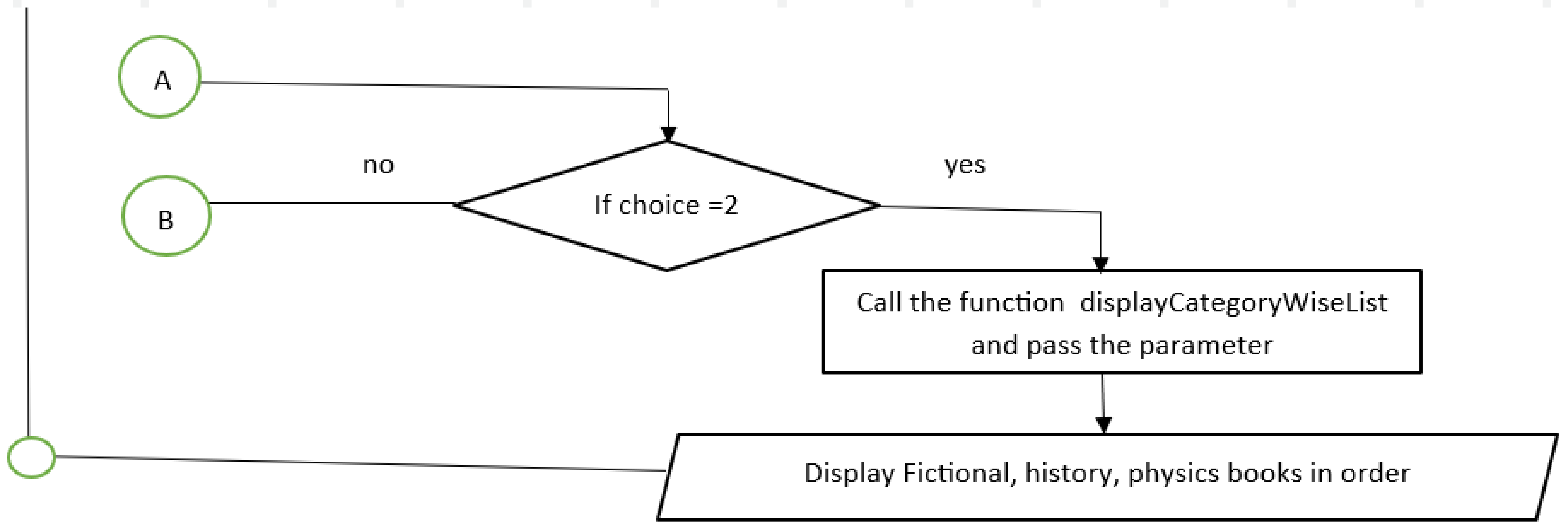

START
Define constants and macros: MAX BOOKS for 10 and FILENAME for the binarv file
Declare an enumeration enum Category and Define a structure struct Book
Declare function prototypes
Dynamically allocate memory for books and Declare bookCount,choice and initialize to bookCount=0.
Assign bookCount = loadBookDetailsFromFile(books), call the function and pass parameters, to check number of books
Display the main menu

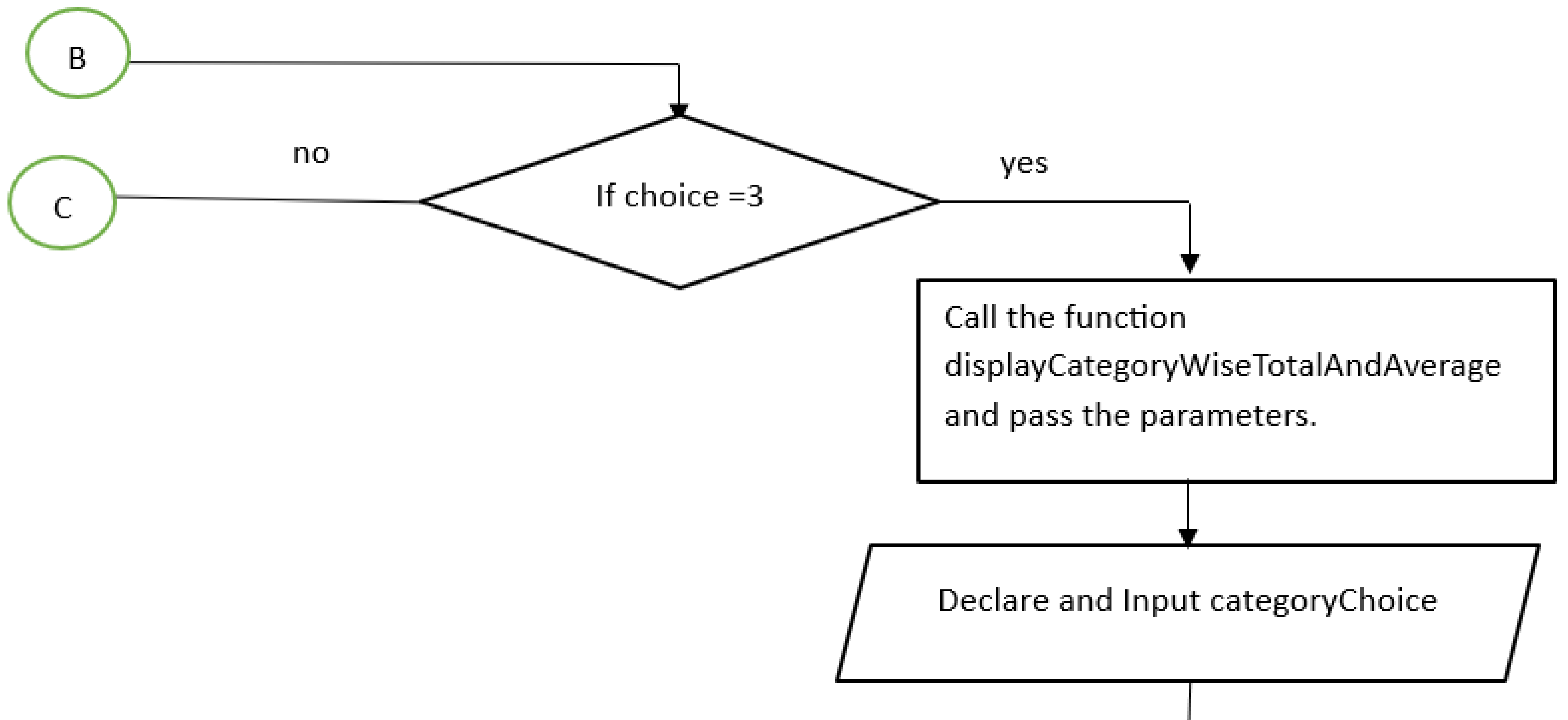
FLOWCHART


Input choice
no
A
Is choice =1?
yes
Call the function saveBookDetails(struct Book *books, int count)
Is count<MAX_BOOKS
yes
no
Input book number
Display maximum limit reached
Validate input for book details

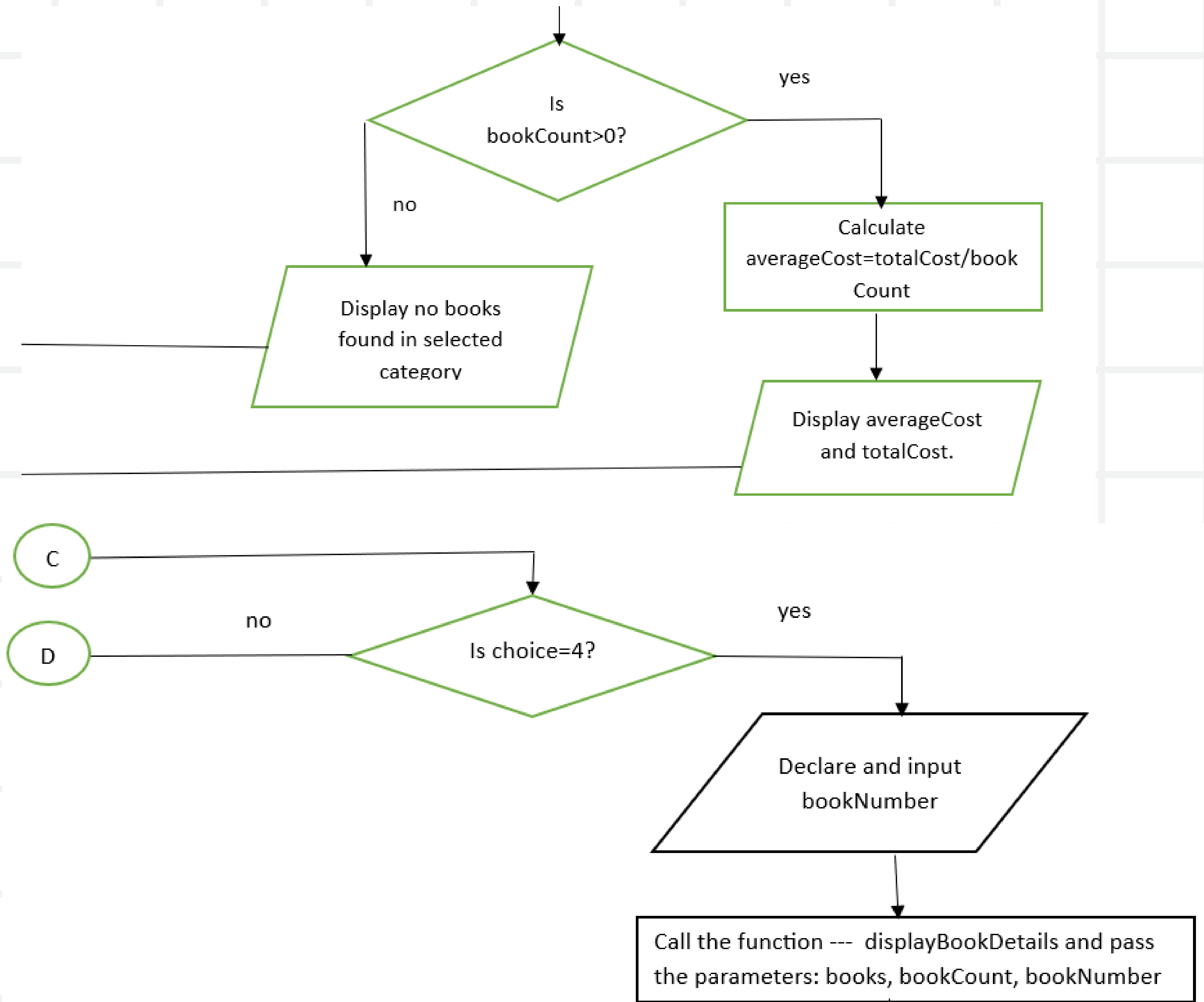

Set books[count].bookNumber = bookNumber
Input book details
Call the function saveBookDetailsToFile and pass parameters
Declare a FILE *file and open the file in "rb+" mode
If file != NULL
yes
no
Write data to file using fwrite
Display error
Close the file
BookCount++

FLOWCHART

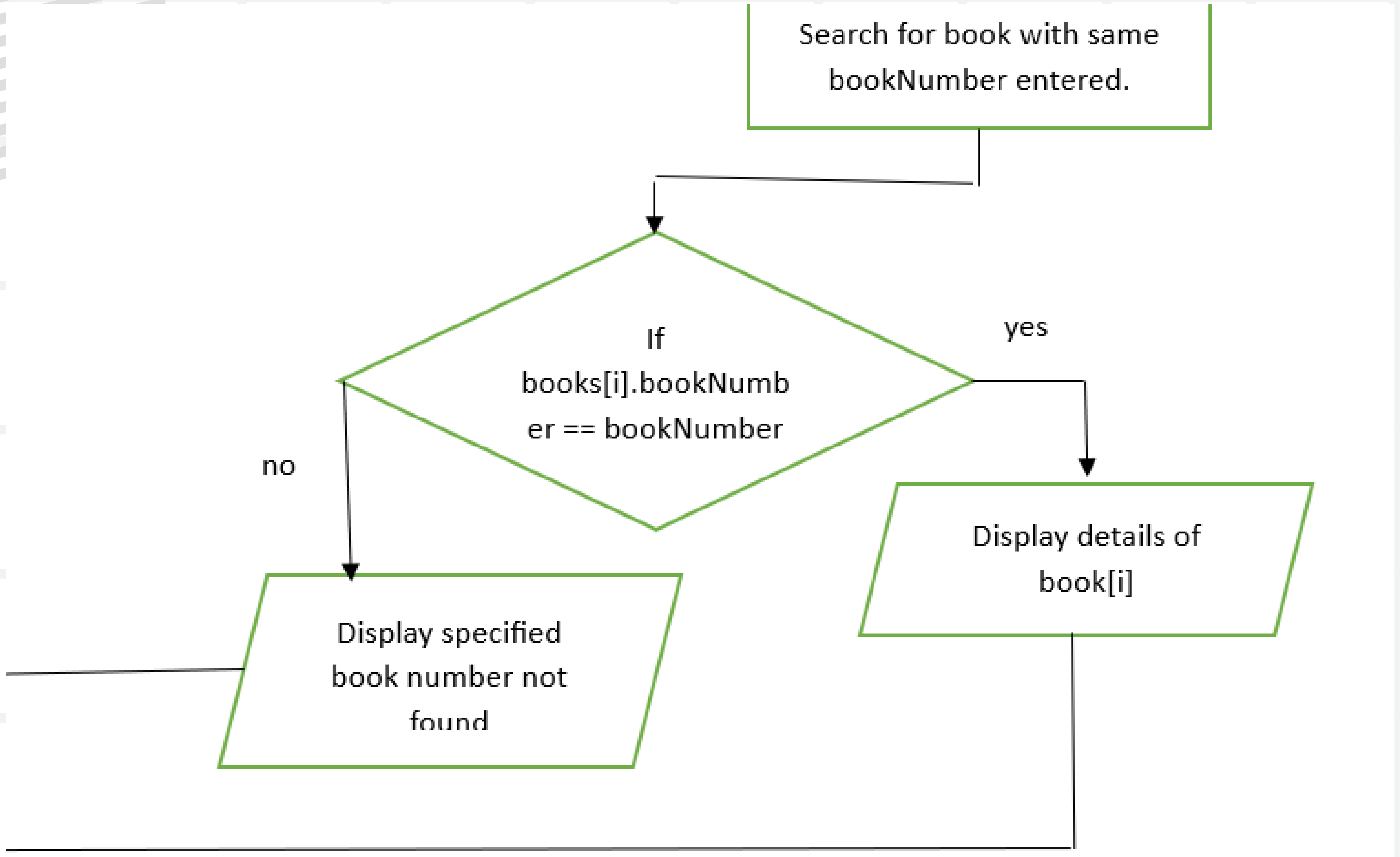

A
B
no
If choice =2
yes
Call the function displayCategoryWiseList and pass the parameter
Display Fictional, history, physics books in order


B
C
no
If choice =3
yes
Call the function displayCategoryWiseTotalAndAverage and pass the parameters.
Declare and Input categoryChoice

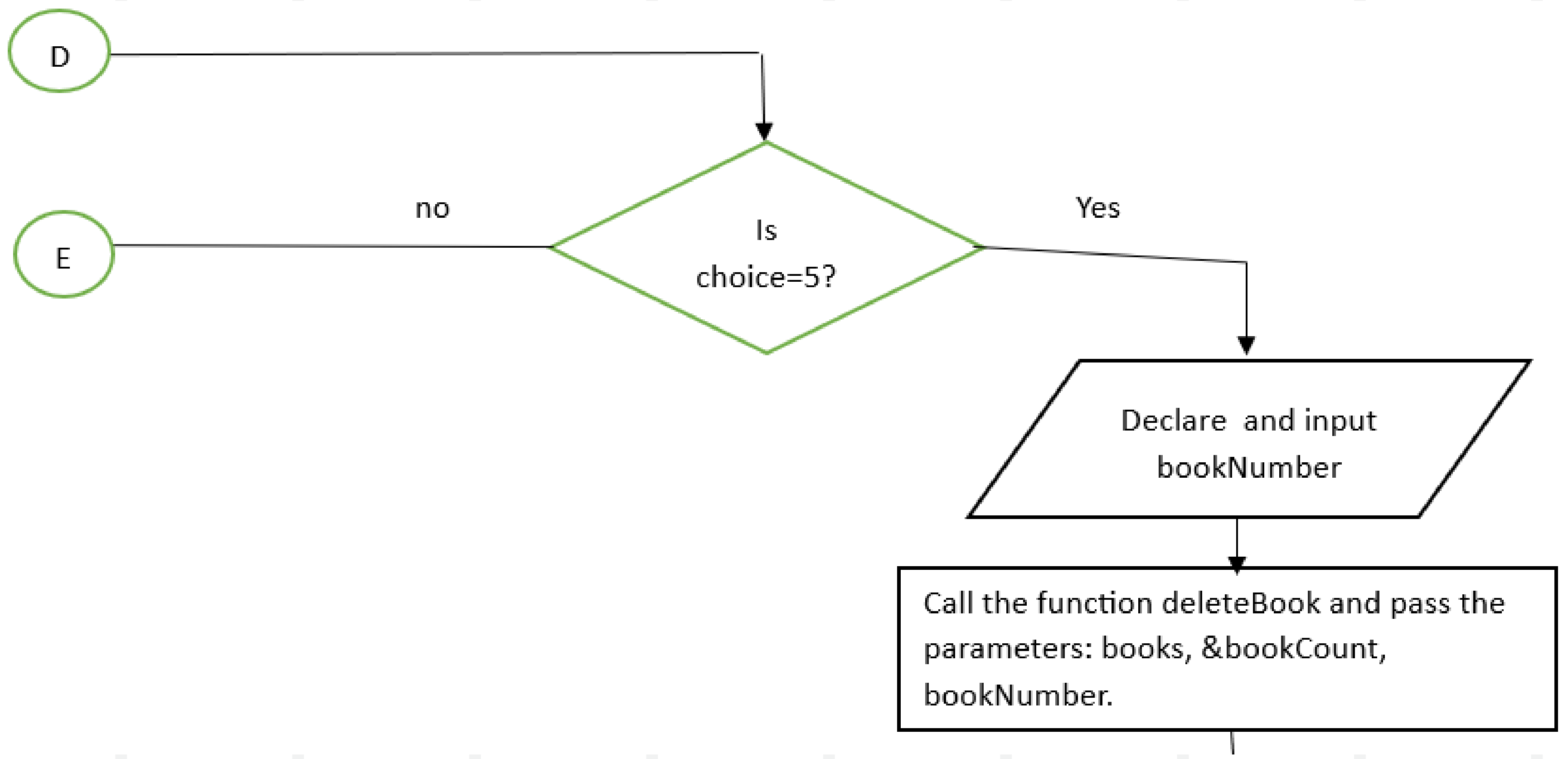
FLOWCHART


Is bookCount>0?
yes
no
Calculate averageCost=totalCost/book Count
Display no books found in selected category
Display averageCost and totalCost.
C
D
no
Is choice=4?
yes
Declare and input bookNumber
Call the function --- displayBookDetails and pass the parameters: books, bookCount, bookNumber

FLOWCHART

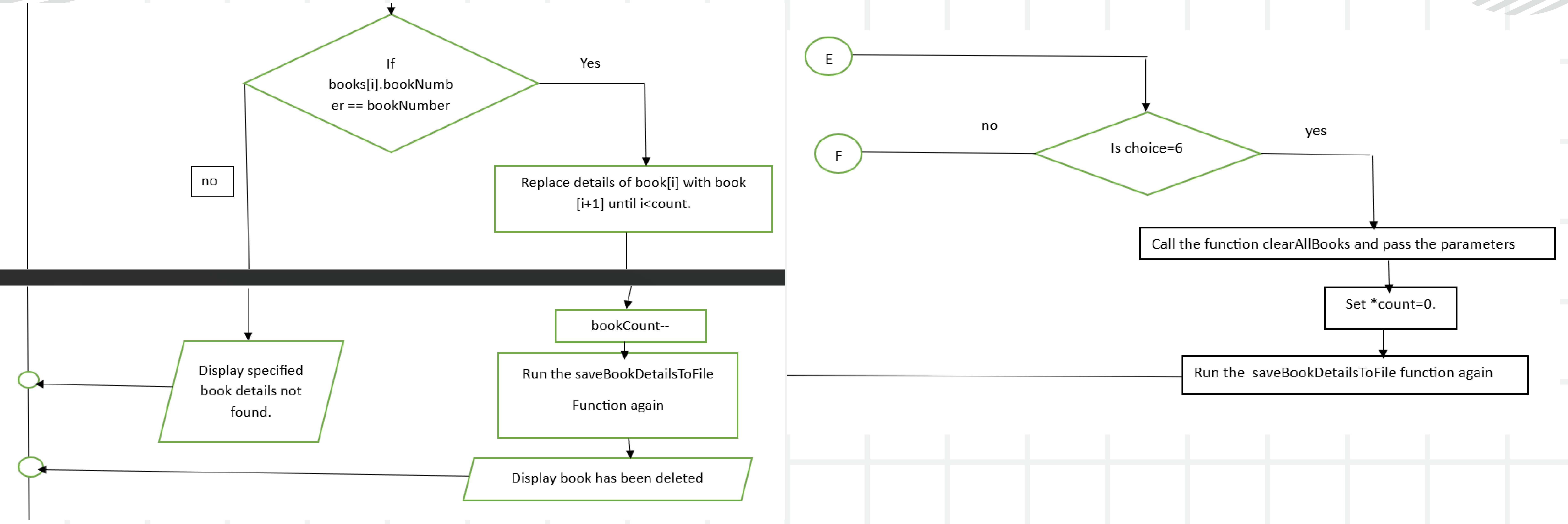


Search for book with same bookNumber entered.
If books[i].bookNumber == bookNumber
yes
no
Display details of book[i]
Display specified book number not found

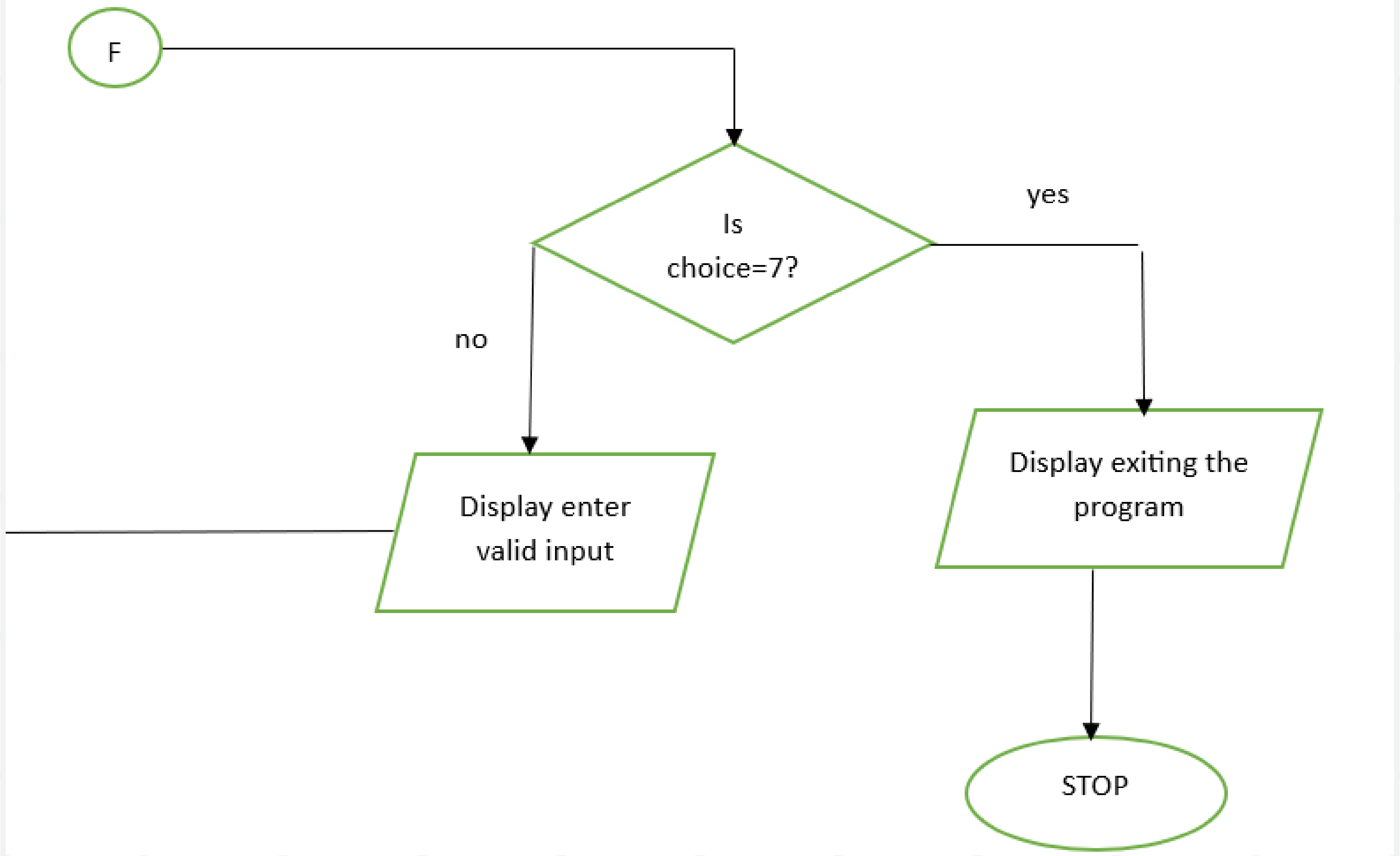


D
E
no
Is choice=5?
Yes
Declare and input bookNumber
Call the function deleteBook and pass the parameters: books, &bookCount, bookNumber.

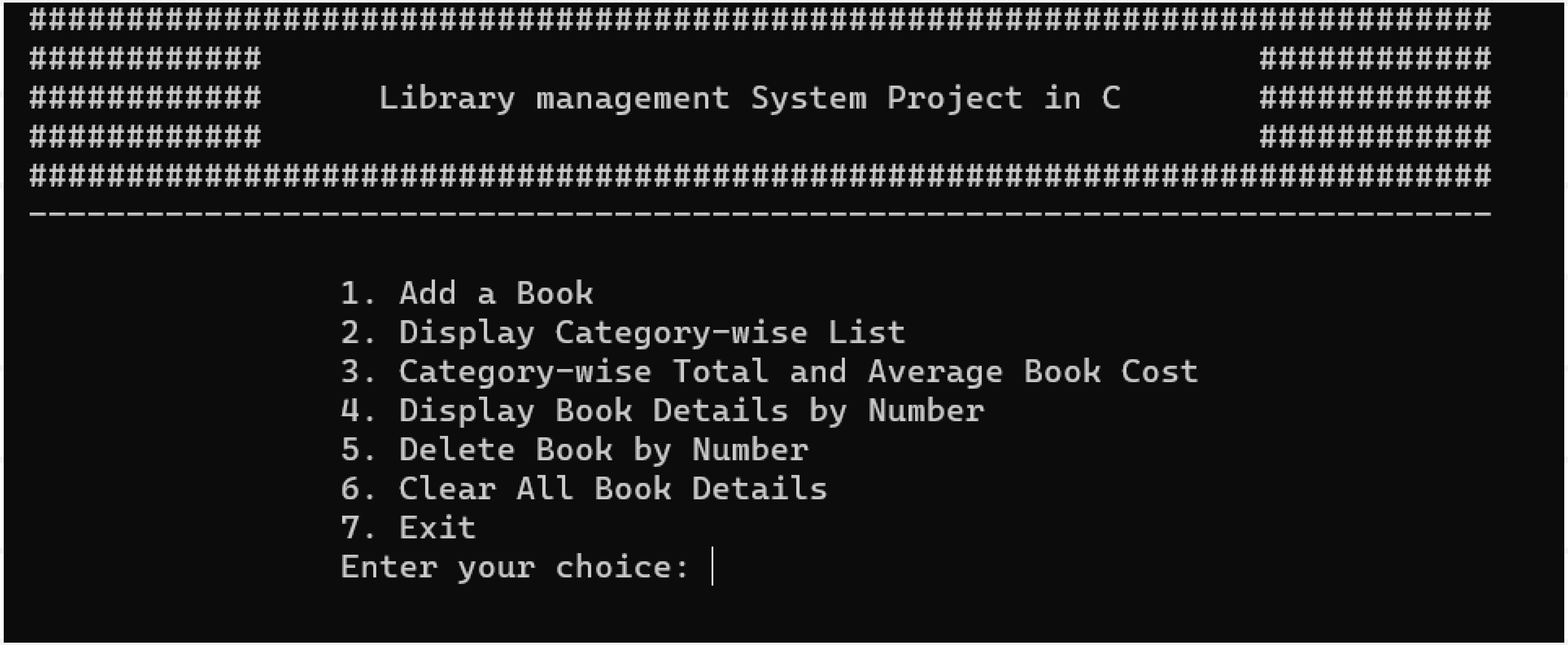
FLOWCHART


If books[i].bookNumber == bookNumber
Yes
no
Replace details of book[i] with book [i+1] until i<count.
bookCount--
Run the saveBookDetailsToFile Function again
Display specified book details not found.
Display book has been deleted
E
F
no
Is choice=6
yes
Call the function clearAllBooks and pass the parameters
Set *count=0.
Run the saveBookDetailsToFile function again

FLOWCHART


F
Is choice=7?
yes
no
Display enter valid input
Display exiting the program
STOP

MAIN MENU

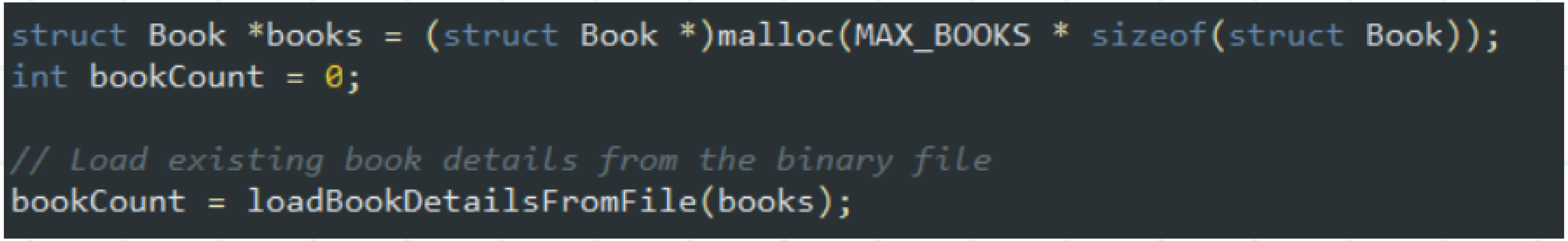


```
########################################################################
###########                                                ###########
###########     Library management System Project in C     ###########
###########                                                ###########
########################################################################
------------------------------------------------------------------------


               1. Add a Book
               2. Display Category-wise List
               3. Category-wise Total and Average Book Cost
               4. Display Book Details by Number
               5. Delete Book by Number
               6. Clear All Book Details
               7. Exit
               Enter your choice: 
```

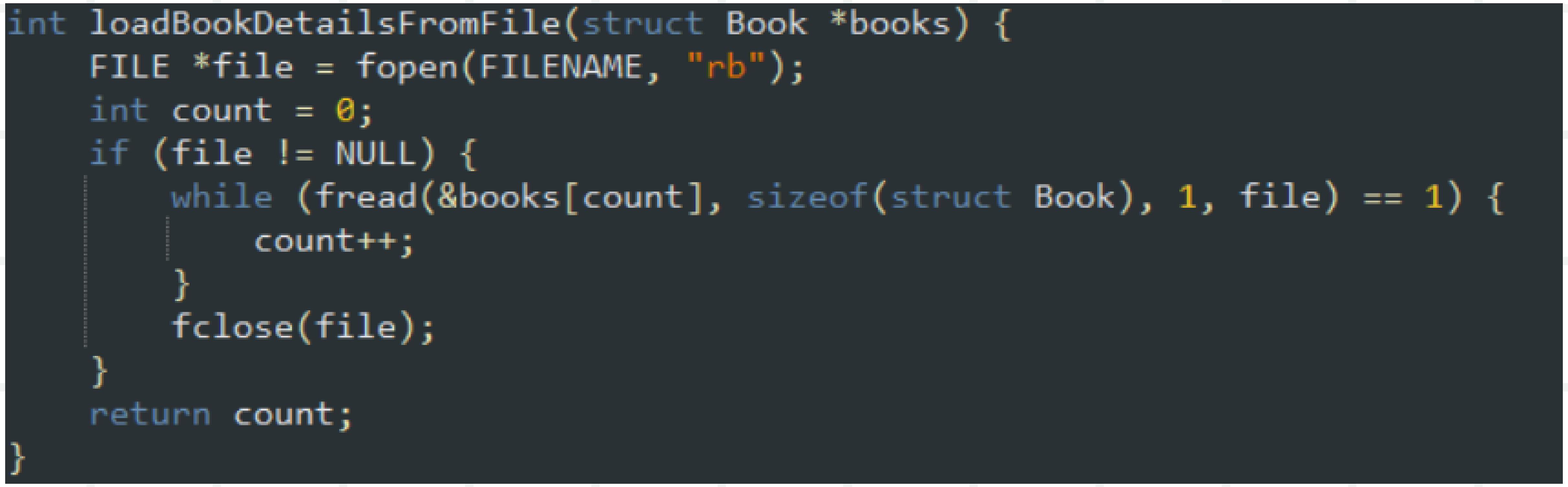
DEMONSTRATION

int main(). --- Rushi
int loadBookDetailsFromFile(); --- Rushi
void saveBookDetails(); --- Akhila
void saveBookDetailsToFile(); --- Akhila
void displayCategoryWiseList(); --- Shashi
void displayCategoryWiseTotalAndAverage();--- Shashi
void displayBookDetails(); --- Ishanth
void deleteBook(); --- Babitha
void clearAllBooks() --- Babitha

CODE

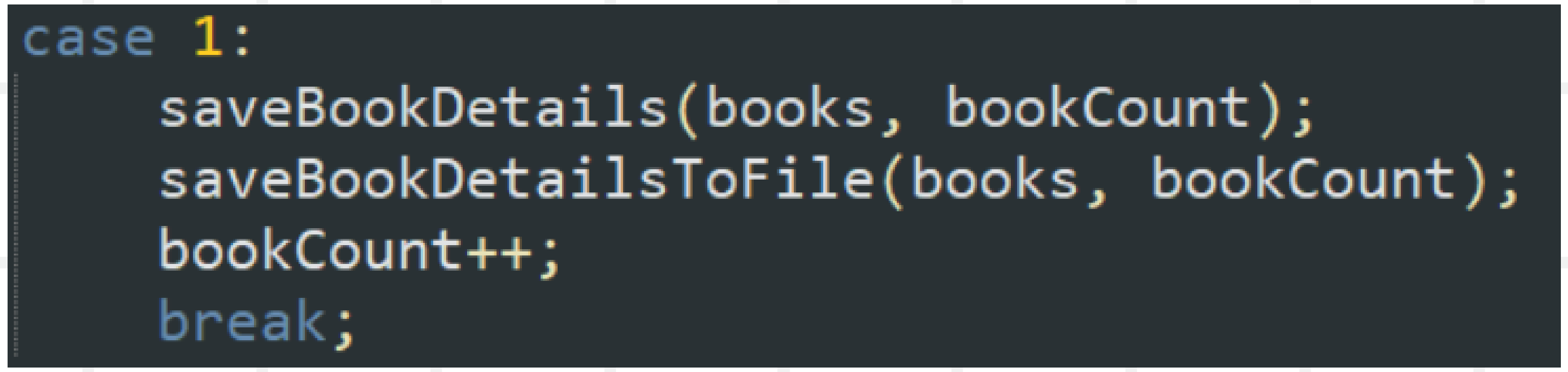


```c
struct Book *books = (struct Book *)malloc(MAX_BOOKS * sizeof(struct Book));
int bookCount = 0;

// Load existing book details from the binary file
bookCount = loadBookDetailsFromFile(books);
```

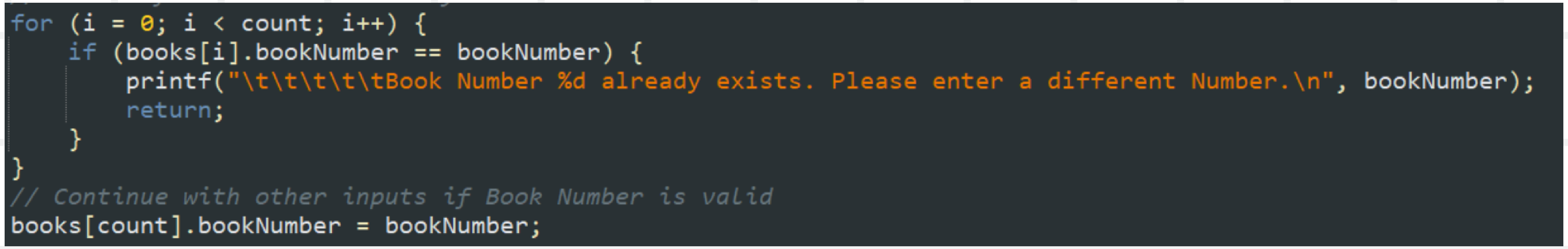


```c
int loadBookDetailsFromFile(struct Book *books) {
    FILE *file = fopen(FILENAME, "rb");
    int count = 0;
    if (file != NULL) {
        while (fread(&books[count], sizeof(struct Book), 1, file) == 1) {
            count++;
        }
        fclose(file);
    }
    return count;
}
```

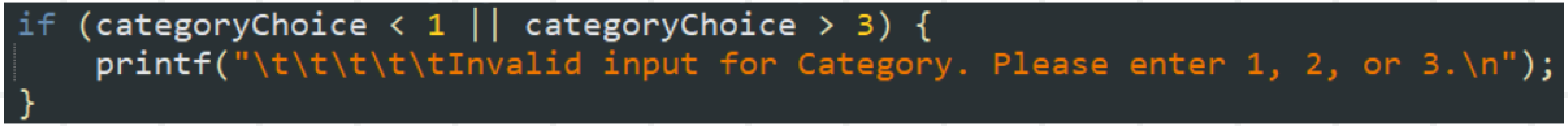
CODE



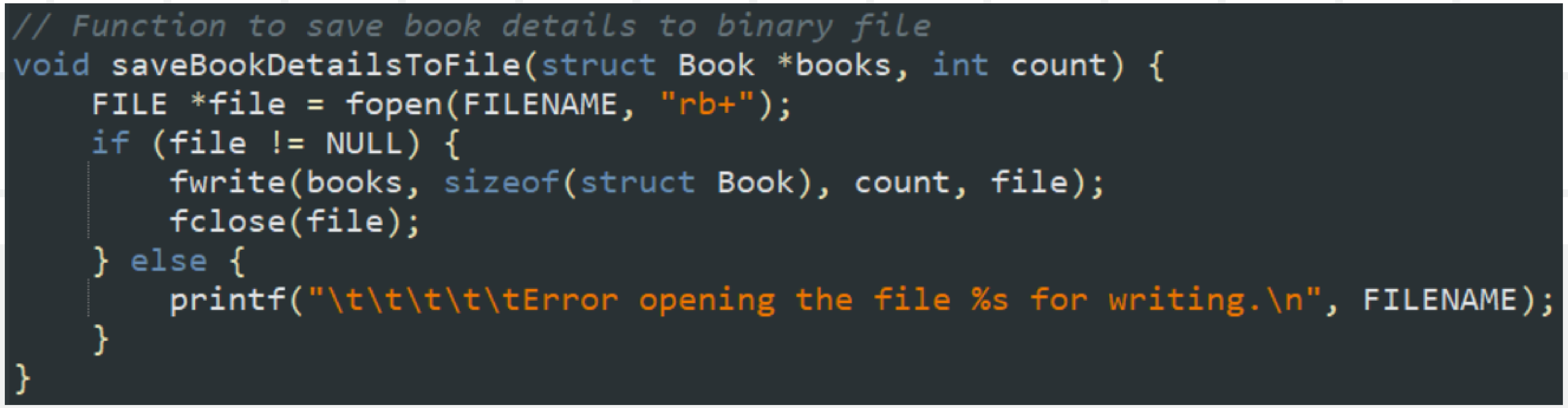
```c
case 1:
    saveBookDetails(books, bookCount);
    saveBookDetailsToFile(books, bookCount);
    bookCount++;
    break;
```



```c
for (i = 0; i < count; i++) {
    if (books[i].bookNumber == bookNumber) {
        printf("\t\t\t\t\tBook Number %d already exists. Please enter a different Number.\n", bookNumber);
        return;
    }
}
// Continue with other inputs if Book Number is valid
books[count].bookNumber = bookNumber;
```

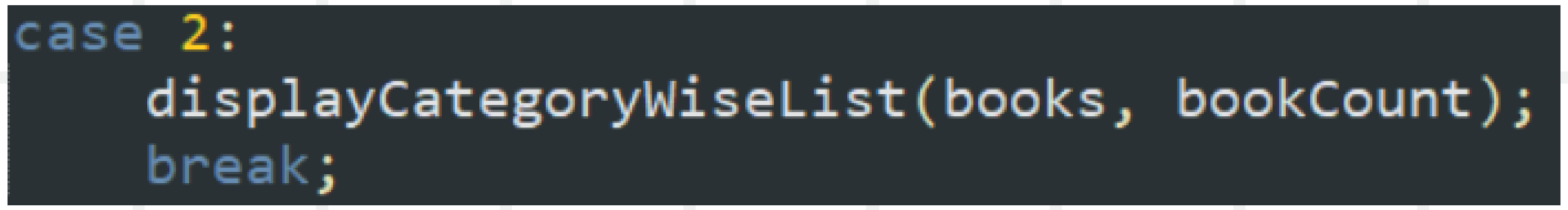


```c
if (categoryChoice < 1 || categoryChoice > 3) {
    printf("\t\t\t\t\tInvalid input for Category. Please enter 1, 2, or 3.\n");
}
```

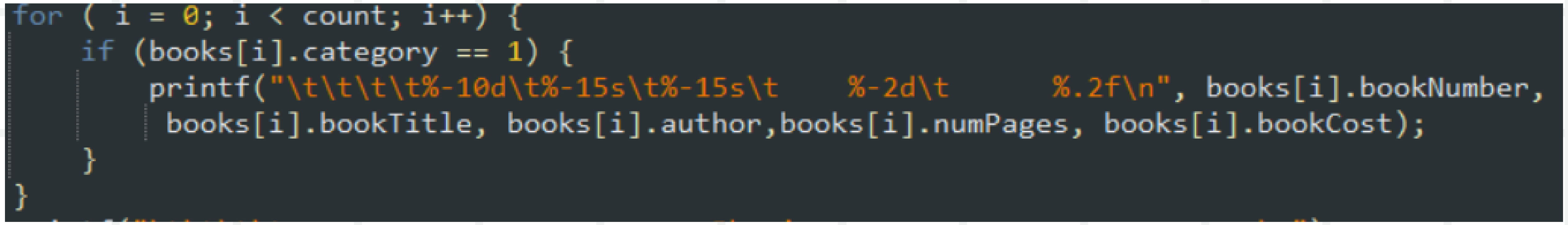


```c
// Function to save book details to binary file
void saveBookDetailsToFile(struct Book *books, int count) {
    FILE *file = fopen(FILENAME, "rb+");
    if (file != NULL) {
        fwrite(books, sizeof(struct Book), count, file);
        fclose(file);
    } else {
        printf("\t\t\t\t\tError opening the file %s for writing.\n", FILENAME);
    }
}
```

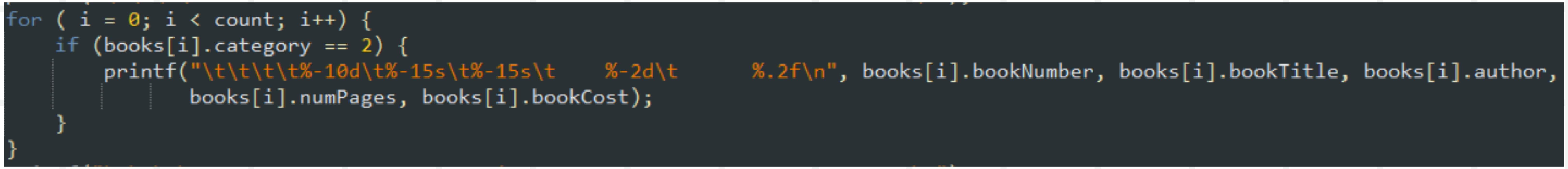
CODE



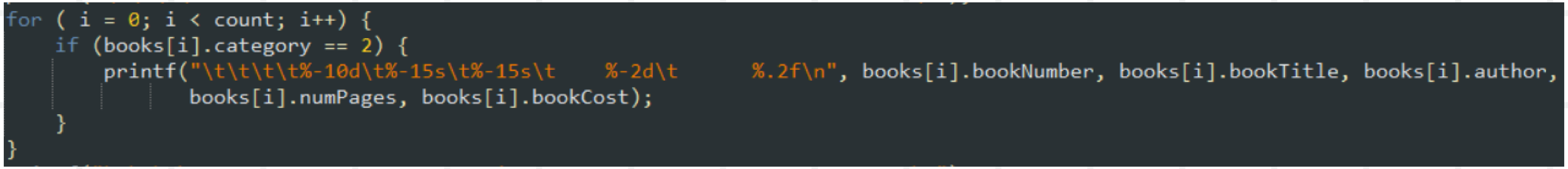
```c
case 2:
    displayCategoryWiseList(books, bookCount);
    break;
```



```c
for ( i = 0; i < count; i++) {
    if (books[i].category == 1) {
        printf("\t\t\t\t%-10d\t%-15s\t%-15s\t    %-2d\t      %.2f\n", books[i].bookNumber,
         books[i].bookTitle, books[i].author,books[i].numPages, books[i].bookCost);
    }
}
```

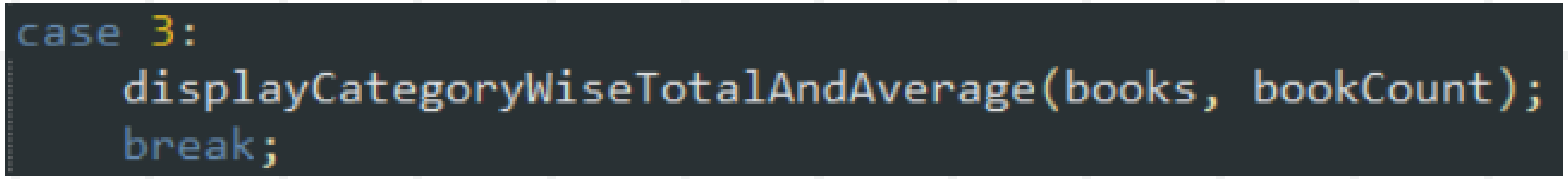


```c
for ( i = 0; i < count; i++) {
    if (books[i].category == 2) {
        printf("\t\t\t\t%-10d\t%-15s\t%-15s\t    %-2d\t      %.2f\n", books[i].bookNumber, books[i].bookTitle, books[i].author,
                books[i].numPages, books[i].bookCost);
    }
}
```

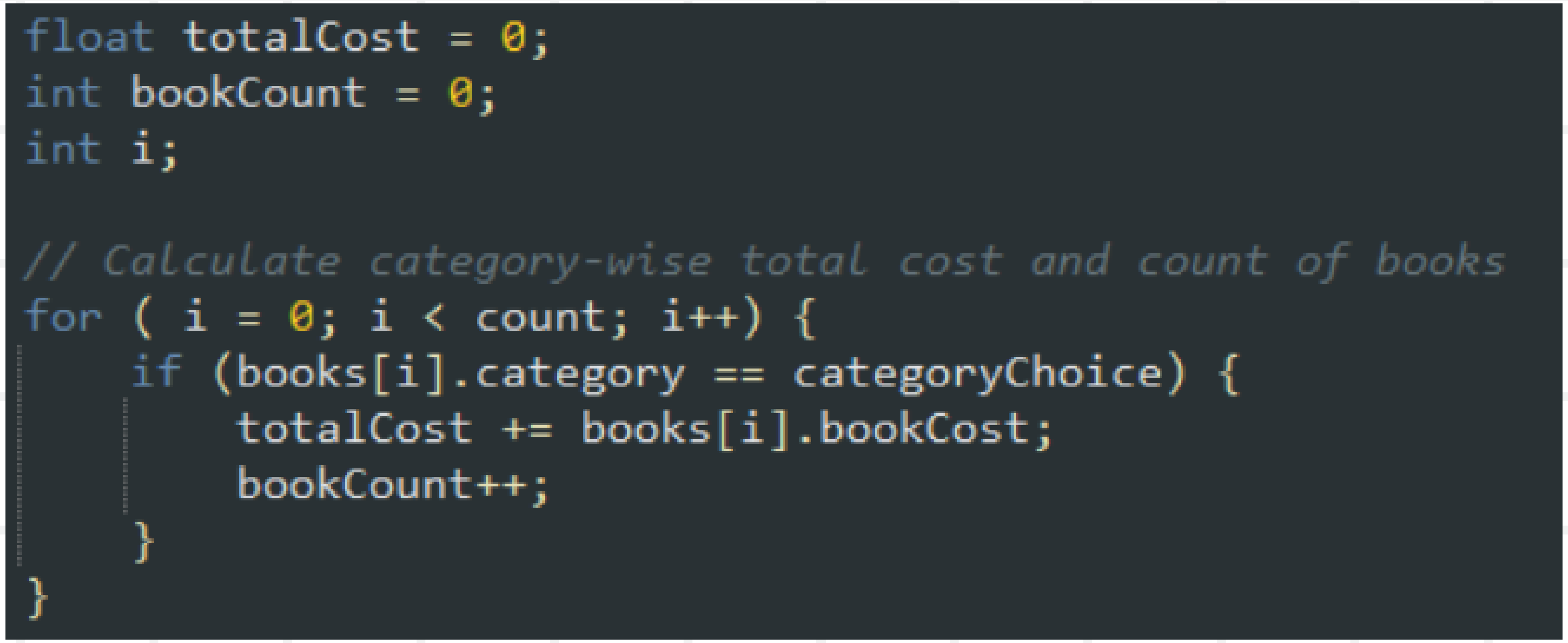


```c
for ( i = 0; i < count; i++) {
    if (books[i].category == 2) {
        printf("\t\t\t\t%-10d\t%-15s\t%-15s\t    %-2d\t      %.2f\n", books[i].bookNumber, books[i].bookTitle, books[i].author,
                books[i].numPages, books[i].bookCost);
    }
}
```

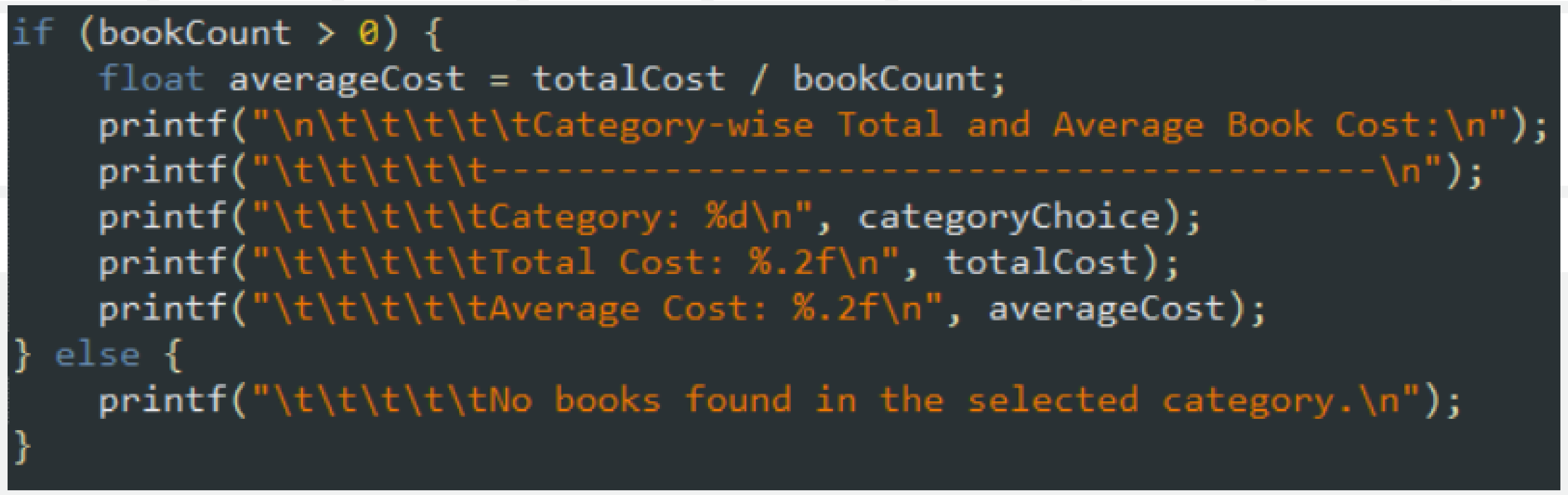
CODE



```c
case 3:
    displayCategoryWiseTotalAndAverage(books, bookCount);
    break;
```

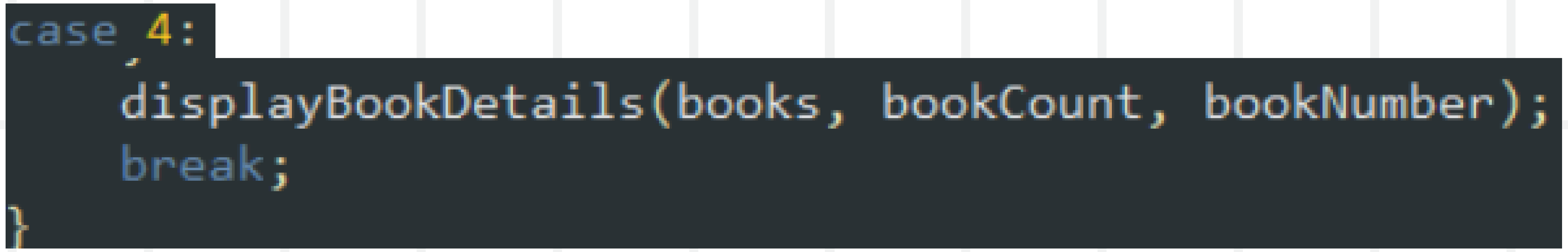


```c
float totalCost = 0;
int bookCount = 0;
int i;

// Calculate category-wise total cost and count of books
for ( i = 0; i < count; i++) {
    if (books[i].category == categoryChoice) {
        totalCost += books[i].bookCost;
        bookCount++;
    }
}
```

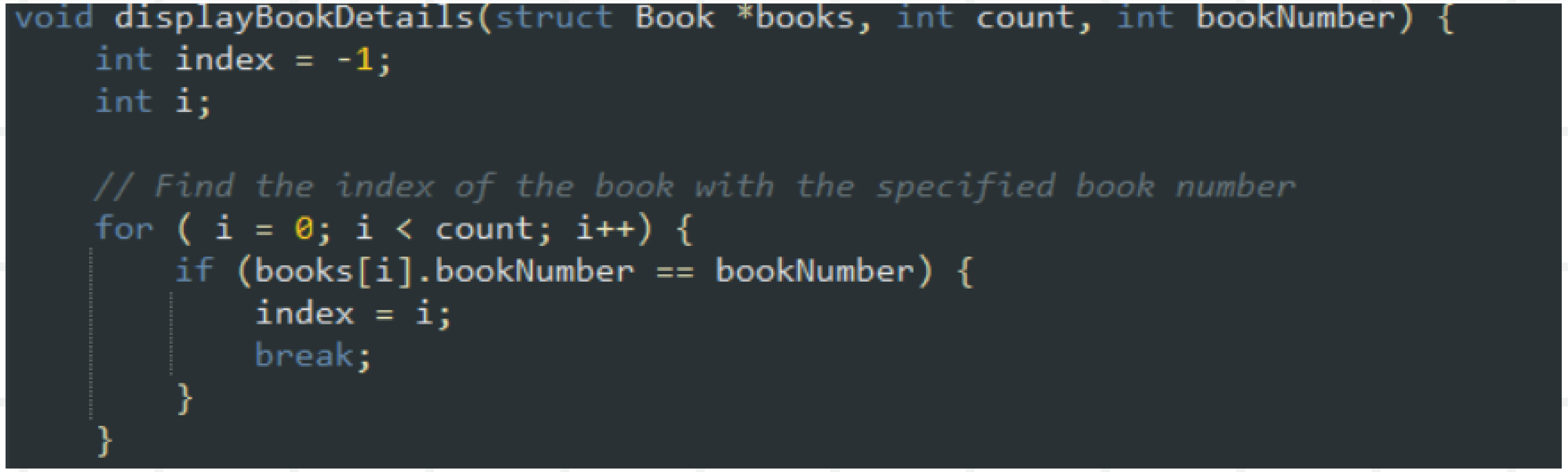


```c
if (bookCount > 0) {
    float averageCost = totalCost / bookCount;
    printf("\n\t\t\t\t\tCategory-wise Total and Average Book Cost:\n");
    printf("\t\t\t\t\t-----------------------------------------\n");
    printf("\t\t\t\t\tCategory: %d\n", categoryChoice);
    printf("\t\t\t\t\tTotal Cost: %.2f\n", totalCost);
    printf("\t\t\t\t\tAverage Cost: %.2f\n", averageCost);
} else {
    printf("\t\t\t\t\tNo books found in the selected category.\n");
}
```

CODE

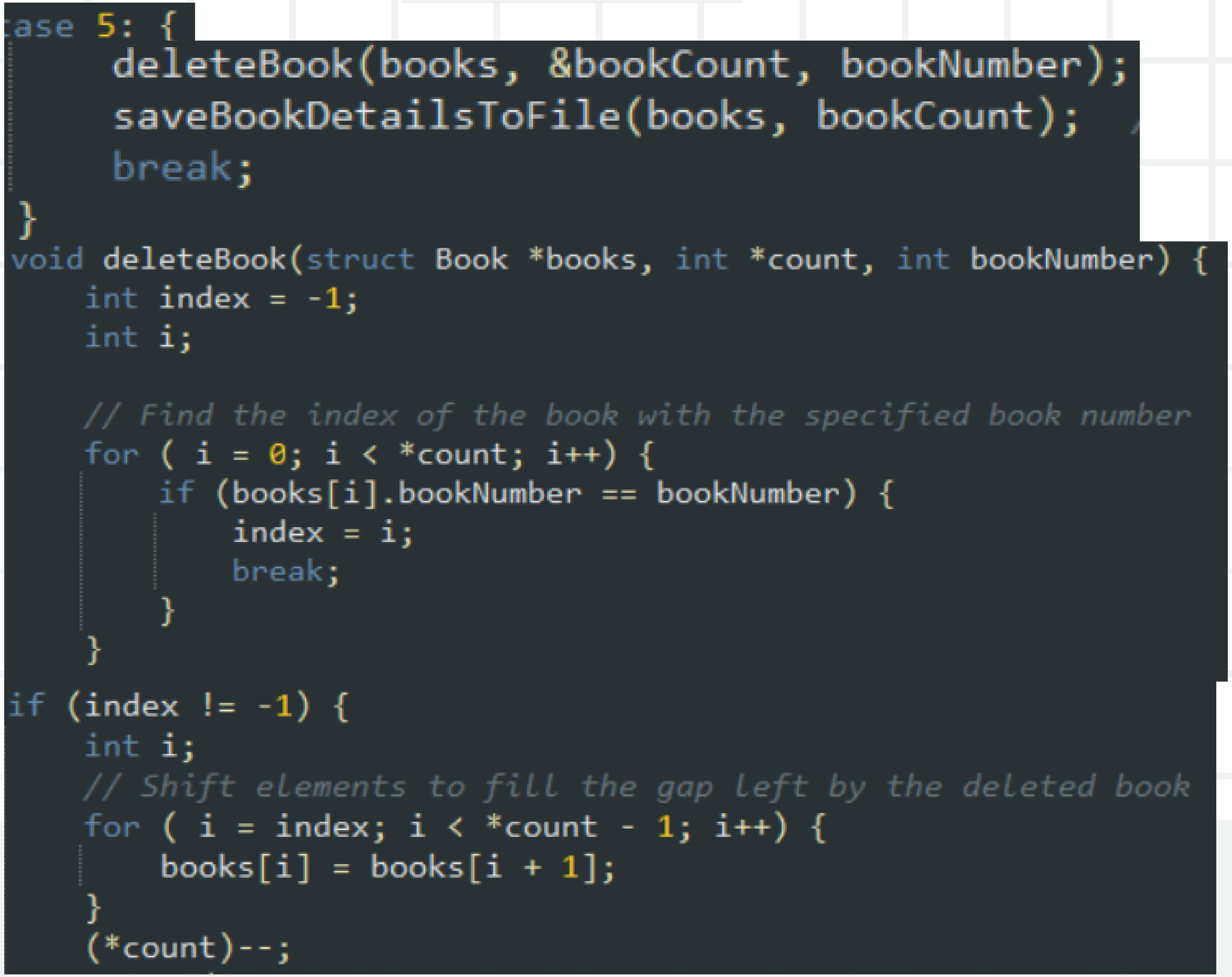


```
case 4:
    displayBookDetails(books, bookCount, bookNumber);
    break;
}
```



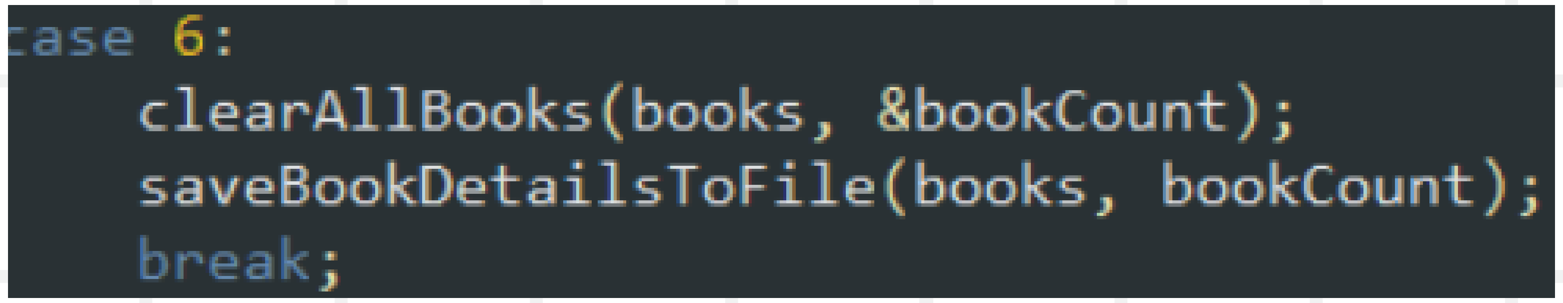
```
void displayBookDetails(struct Book *books, int count, int bookNumber) {
    int index = -1;
    int i;

    // Find the index of the book with the specified book number
    for ( i = 0; i < count; i++) {
        if (books[i].bookNumber == bookNumber) {
            index = i;
            break;
        }
    }
```

CODE

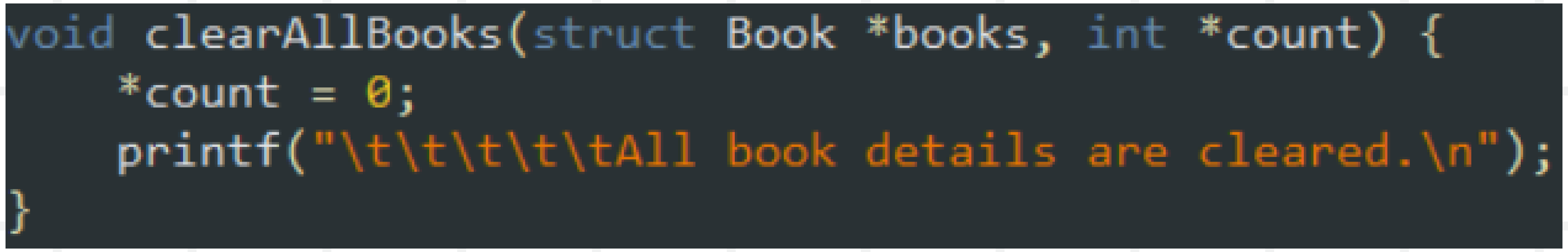


```
ase 5: {
        deleteBook(books, &bookCount, bookNumber);
        saveBookDetailsToFile(books, bookCount);  /
        break;
}
void deleteBook(struct Book *books, int *count, int bookNumber) {
    int index = -1;
    int i;

    // Find the index of the book with the specified book number
    for ( i = 0; i < *count; i++) {
        if (books[i].bookNumber == bookNumber) {
            index = i;
            break;
        }
    }
if (index != -1) {
    int i;
    // Shift elements to fill the gap left by the deleted book
    for ( i = index; i < *count - 1; i++) {
        books[i] = books[i + 1];
    }
    (*count)--;
```

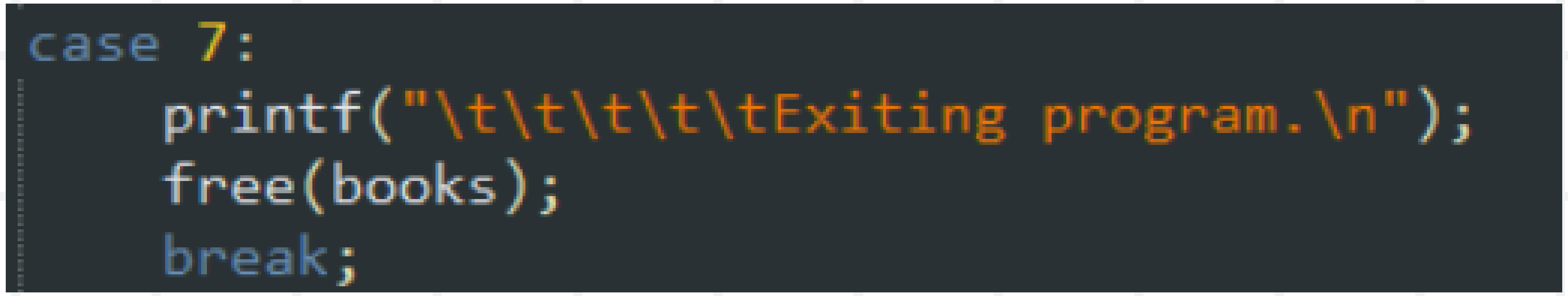
CODE



```
ase 6:
    clearAllBooks(books, &bookCount);
    saveBookDetailsToFile(books, bookCount);
    break;
```

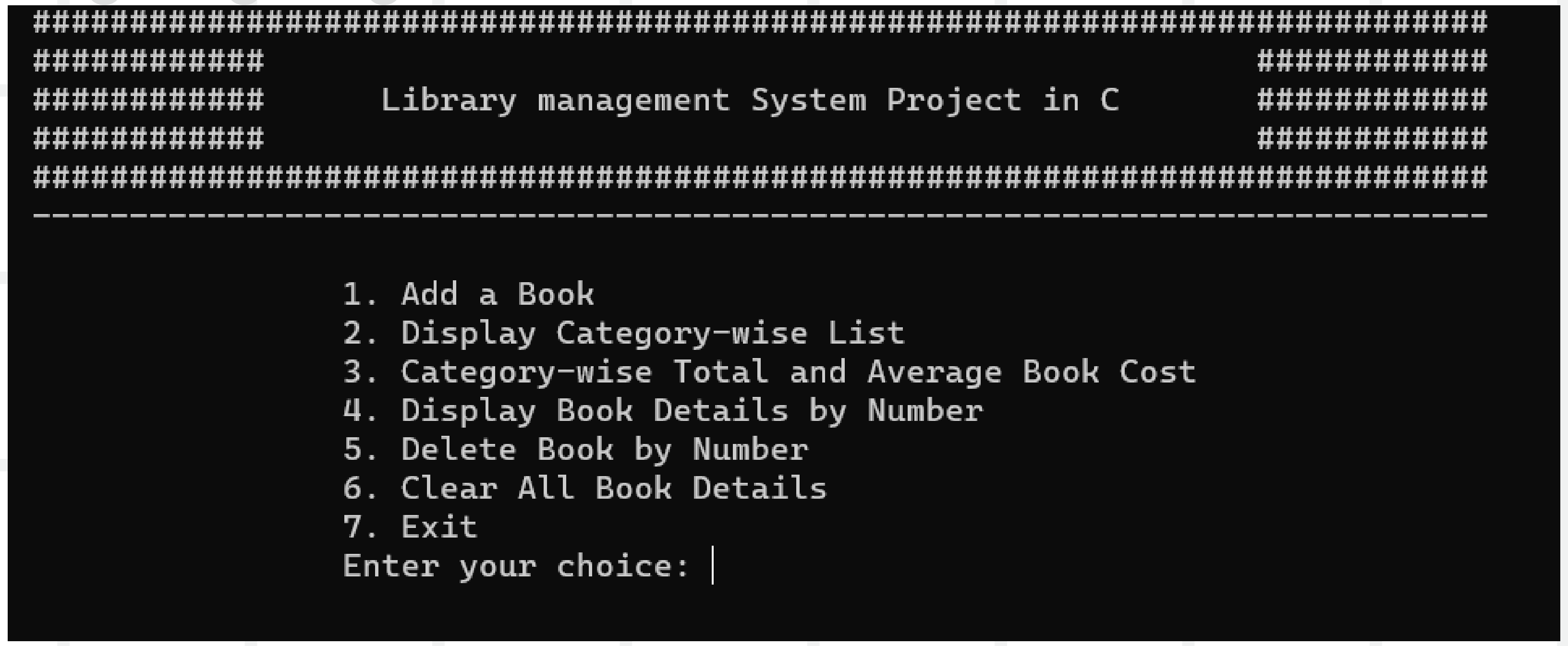


```
void clearAllBooks(struct Book *books, int *count) {
    *count = 0;
    printf("\t\t\t\t\tAll book details are cleared.\n");
}
```



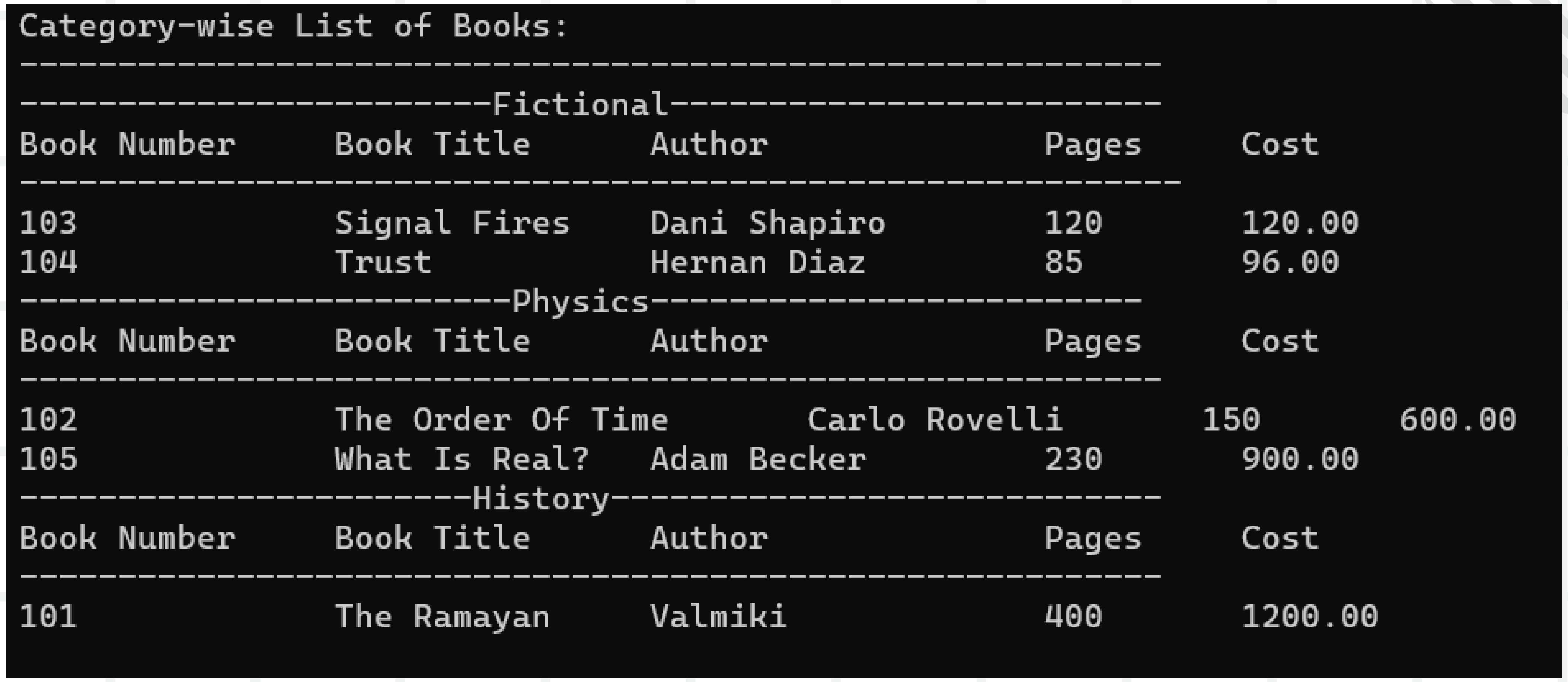
```
case 7:
    printf("\t\t\t\t\tExiting program.\n");
    free(books);
    break;
```

INPUT AND OUTPUT



```
############################################################################
############                                                    ############
############      Library management System Project in C        ############
############                                                    ############
############################################################################
----------------------------------------------------------------------------

                1. Add a Book
                2. Display Category-wise List
                3. Category-wise Total and Average Book Cost
                4. Display Book Details by Number
                5. Delete Book by Number
                6. Clear All Book Details
                7. Exit
                Enter your choice: 
```

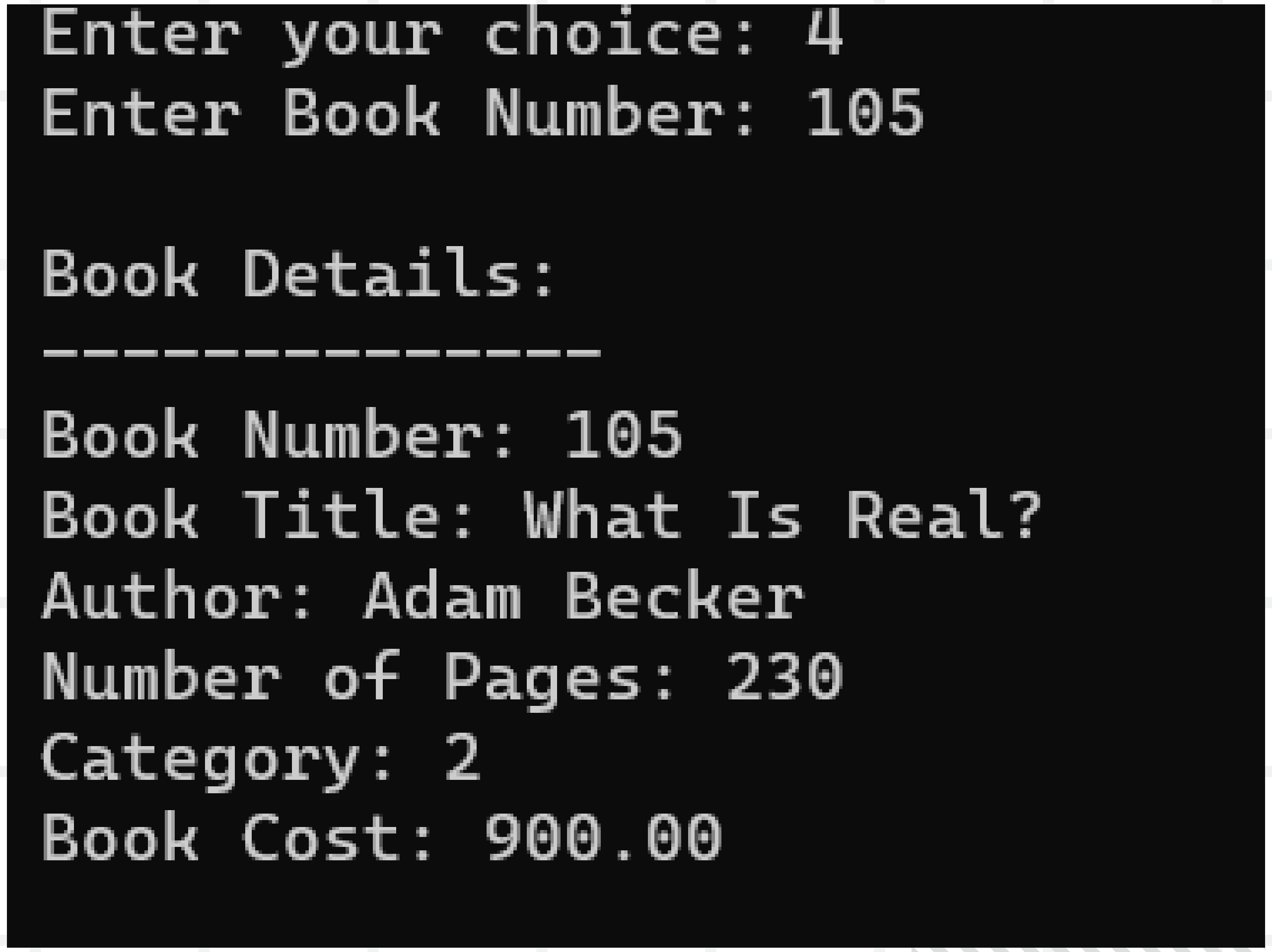
displayCategoryWiseList(books, bookCount)



```
Category-wise List of Books:
----------------------------------------------------------
------------------------Fictional-------------------------
Book Number     Book Title      Author              Pages     Cost
-----------------------------------------------------------

103             Signal Fires    Dani Shapiro        120       120.00
104             Trust           Hernan Diaz         85        96.00
-------------------------Physics-------------------------
Book Number     Book Title      Author              Pages     Cost
----------------------------------------------------------

102             The Order Of Time       Carlo Rovelli       150       600.00
105             What Is Real?   Adam Becker         230       900.00
-----------------------History----------------------------
Book Number     Book Title      Author              Pages     Cost
----------------------------------------------------------

101             The Ramayan     Valmiki             400       1200.00
```

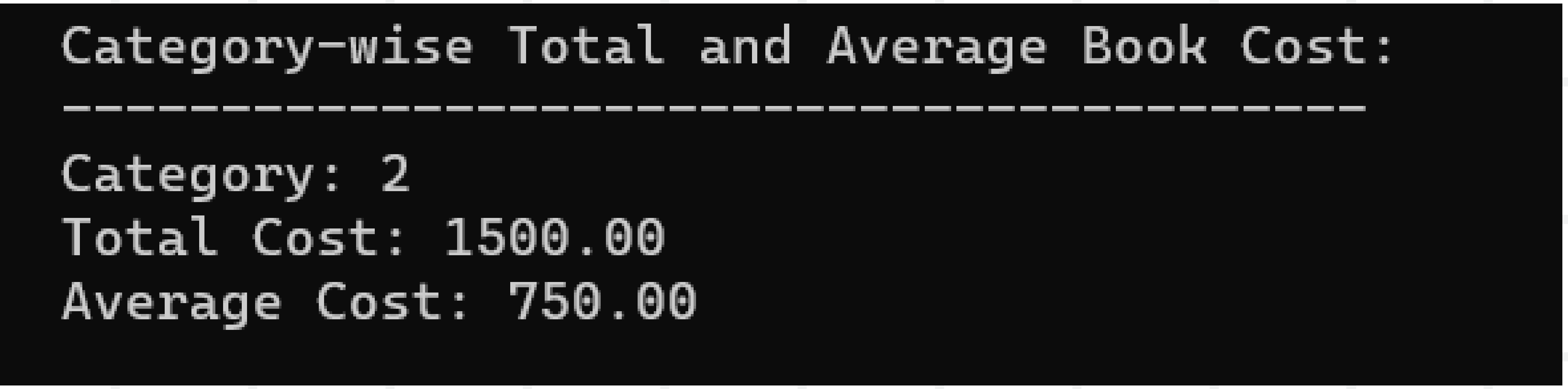
INPUT AND OUTPUT

displayBookDetails(books, bookCount, bookNumber)



```
Enter your choice: 4
Enter Book Number: 105

Book Details:
--------------
Book Number: 105
Book Title: What Is Real?
Author: Adam Becker
Number of Pages: 230
Category: 2
Book Cost: 900.00
```

displayCategoryWiseTotalAndAverage(books, bookCount);



```
Category-wise Total and Average Book Cost:
-----------------------------------------
Category: 2
Total Cost: 1500.00
Average Cost: 750.00
```

TOPICS COVERED

Arrays
Structures
Pointers
Loops
Binary Files
User Defined Functions
Menu Driven Interface
Macros
Dynamic Memory Allocation

ANY
QUESTiONS?

THANK YOU