

Full name: Lin Cai

University: Loughborough University London

Major: Digital Application Development

Report’s Title: Book Management Application

**Word count: 1532 words**

TABLE OF CONTENTS

[1. Introduction 2](#_Toc88817059)

[2. Program Structure, functionality and application development 2](#_Toc88817060)

[2.1. Main application Functionalities 2](#_Toc88817061)

[2.2. Program Structure 2](#_Toc88817062)

[2.3. Application structure 3](#_Toc88817063)

[2.3.1. Main screen 6](#_Toc88817064)

[2.3.2. List of Books screen 8](#_Toc88817065)

[2.3.3. Insert New Book Screen 11](#_Toc88817066)

[2.3.4. Delete Book screen 17](#_Toc88817067)

[2.3.5. Search book screen 22](#_Toc88817068)

[3. Conclusion and Further Work 27](#_Toc88817069)

# Introduction

The book management system which includes library items such as books, CDs, photographs and so on are growing its importance to any libraries, universities, training centers and book stores. It is clear that a well-structured bibliographic system would enable users to search for an item in a blink of an eye and therefore facilitating a smooth and time-saving system. Therefore, the ultimate purpose of this report is to devise a well-organized program which addresses arising need of book management system. The detailed report describes my ideas on how to devise the architecture of the program, which forms the basis for the code development followed by this report.

# Program Structure, functionality and application development

In order to describe the program structure, functionality and application development, this section would first describe generally on “main application functionality” and “Program structure” in section 2.1 and 2.2. Section 2.3 would go further into describe how to transform the idea of functionality by block diagrams to actual functions of the application via C code.

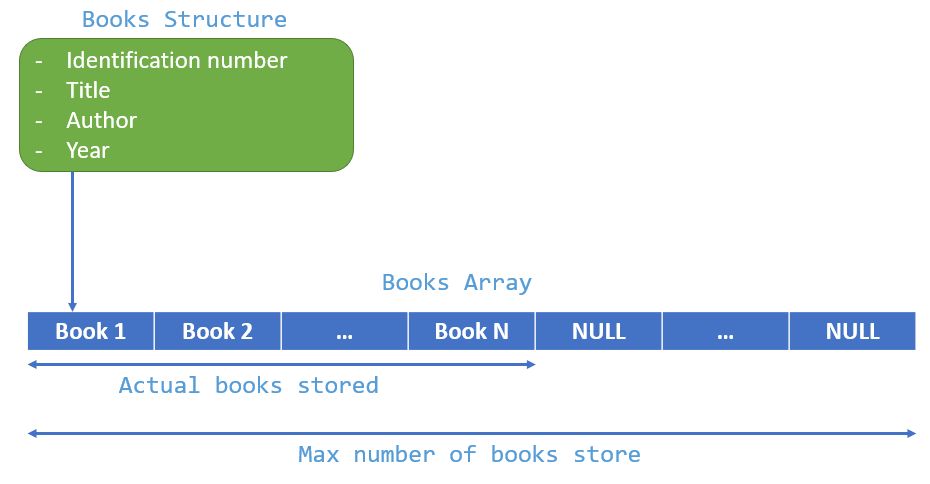
## Main application Functionalities

The application I develop has 4 main functionalities: (i) To list all of books in the system, (ii) To insert new books, (iii) To delete existing books and (iv) Searching for books.

## Program Structure

It is necessary for devise the structure of the application prior to producing the code. The application is designed to store basic information of a book, including Identification Number, Title, Author and Year. Books inserted will be stored in a global array.

Figure 1: Book Structure



The following code is used to generate the program structure:

|  |
| --- |
| #define MAX\_NUMBER\_OF\_BOOKS\_STORED 1000  //Struct for Book  //it contains 4 members: idNo, title, author and year.  //idNo must be unique, and shall be validated when user insert new book.  typedef struct  {      char idNo[20];      char title[100];      char author[100];      int year;  } Book;  Book books[MAX\_NUMBER\_OF\_BOOKS\_STORED]; //A global array variable to store inserted books.  int bookCount = 0;//A global integer variable to count the number of inserted books. |

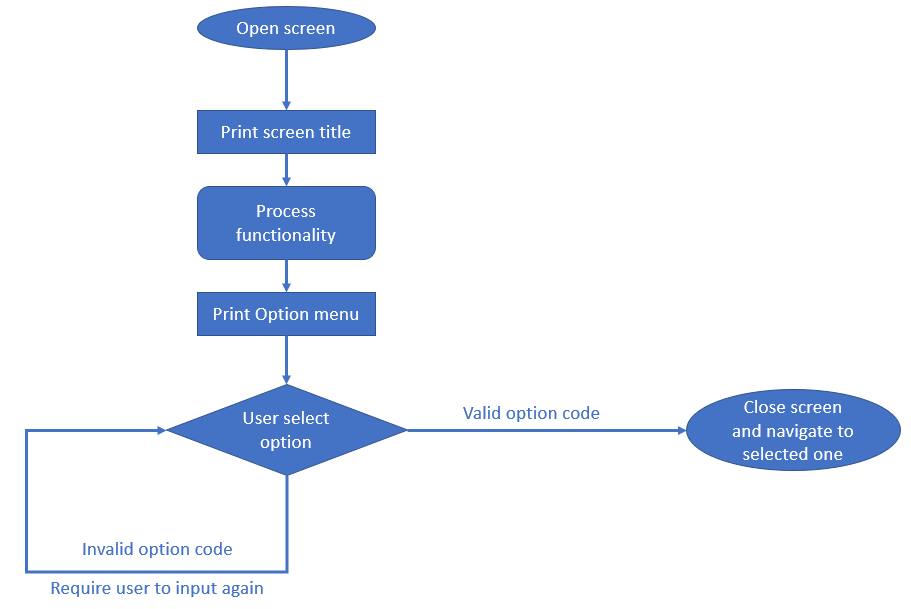
## Application structure

In terms of application functionalities, the application will have 5 corresponding screens:

* Main screen.
* “Book List” screen
* “Book Insert” screen
* “Book Removal” screen
* “Book Search” screen

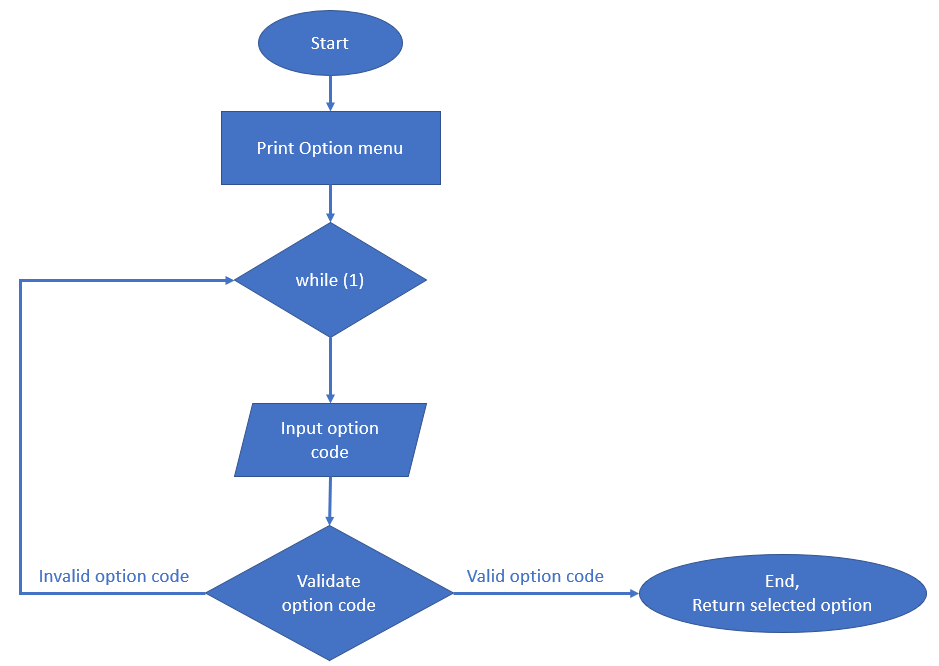
Each screen has its own functionalities can navigate to another screen from a specific screen. Generally, each screen also has same points in operation. They show screen title, content with their functionality, and option menu.

Figure 2:



All screens would have “Print option menu” and “User Select Opition” menu. In case an user inputs/keys in an invalid value, the program makes an infinity loop to deal with re-inputting option code as below.

Figure 3: Infinite loop to deal with invalid input values



The following code is to generate the above-described function:

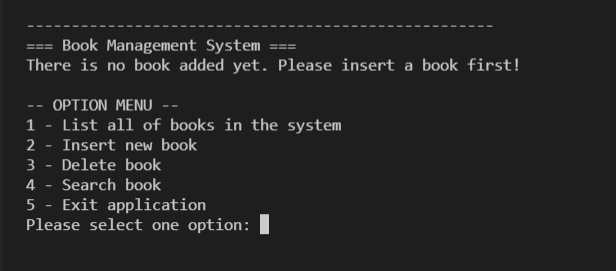
|  |
| --- |
| //This function is for generating Option menu,  //receive user's choice, validate and return selection option code.  char displayOptionSelection(char \*options[], int size)  {      printf("\n-- OPTION MENU --\n");      for (int i = 0; i < size; i++)      {          printf("%s\n", options[i]);      }        char selectedOptionCode;      //infinity loop for user to re-input option code in case of invalid code inputted      while (1)      {          printf("Please select one option: ");          scanf(" %c", &selectedOptionCode);          int validOptionCode = 0; //It's just a flag to check if any option  match to input value          char optionCode;          for (int i = 0; i < size; i++)          {              optionCode = options[i][0];              if (selectedOptionCode == optionCode) //If it matchs to an existing option, toggle  value of the flag 'validOptionCode' and exit loop.              {                  validOptionCode = 1;                  break;              }          }          if (!validOptionCode) //If inputted option doesnot match to any available option code, then prompt message and let user try again.          {             printf("You just enterred invalid option code, please try again\n");          }          else          {              return selectedOptionCode; //exit infintity loop and return selected option code          }      }  } |

### Main screen

* ***Main Description of the screen***

The main screen is the first one to show to an user when the user starts the application. If the system has yet to have any book added, the main screen shows “There is no book added yet. Please insert a book” message and as well as the option menu.

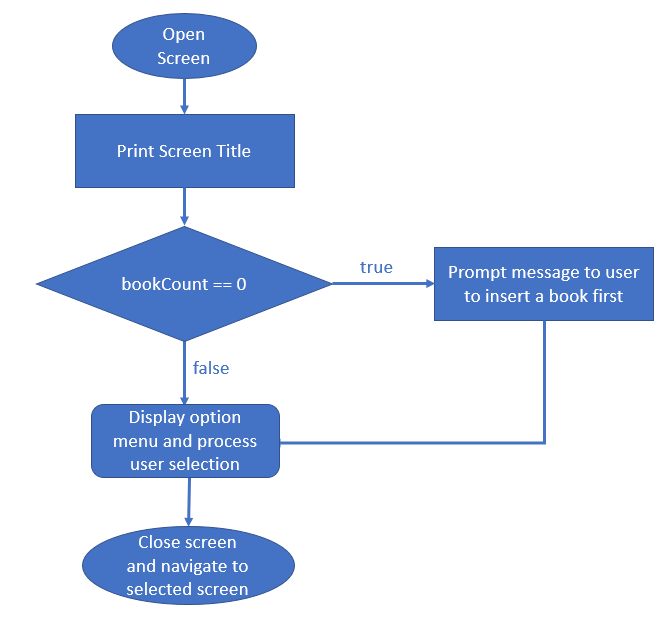
Figure 4: The main screen appearance



As described above, the application has 4 main functions “List all of the books in the system, insert new book, delete book, search book” which are listed in the fist 4 options. The final option is to “exit application”. Apart from the last option, the user can switch to a particular screen after entering the first 4 options.

* ***The function notifying users on no book existing in the system***

Figure 5: The block diagram on the loop necessary for requiring the user to insert a new book if there is no book on the system



The code for the above diagram is as follows:

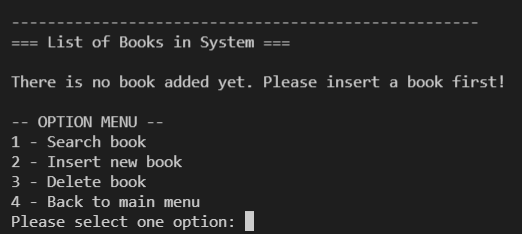
|  |
| --- |
| //This function for generating and operating Main screen  void displayMainScreen()  {      printf(SEPARATOR);      printf("=== Book Management System ===\n");      if(bookCount == 0) {          printf("There is no book added yet. Please insert a book first!\n");      }      //These are options supported in Main screen.      char \*options[5] = {          "1 - List all of books in the system",          "2 - Insert new book",          "3 - Delete book",          "4 - Search book",          "5 - Exit application"};      char optionCode = displayOptionSelection(options, 5);      //This section for screen navigation      //Option 1 selected -> display List of Books screen      if (optionCode == options[0][0])      {          displayListOfBooksScreen();      }      //Option 2 selected -> display Insert Book screen      else if (optionCode == options[1][0])      {          displayBookInsertionScreen();      }      //Option 3 selected -> display Delete Book screen      else if (optionCode == options[2][0])      {          displayBookDeletionScreen();      }      //Option 4 selected -> display Search Book screen      else if (optionCode == options[3][0])      {          displayBookSearchScreen();      }      //If user select 5, function will reach to the end and finish, no need to handle by code.      //Other invalid option codes were handled in the method 'displayOptionSelection'  //above already.  } |

### List of Books screen

* ***Screen Description***

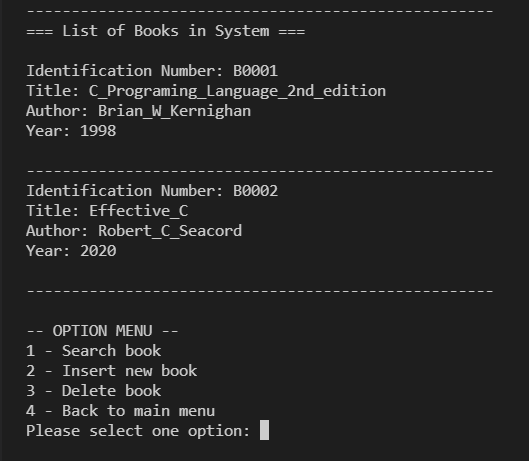
This screen function is to list all books stored in this application. In case the application has yet to have any books added yet, it prompts a message to user to insert a book first.

Figure 6: The List of Books screen in case of no books



If the system has stored books, the screen will show the details of books added as follows.

Figure 7: The "List of Books" Screen in case books are already added



This screen has 4 available options

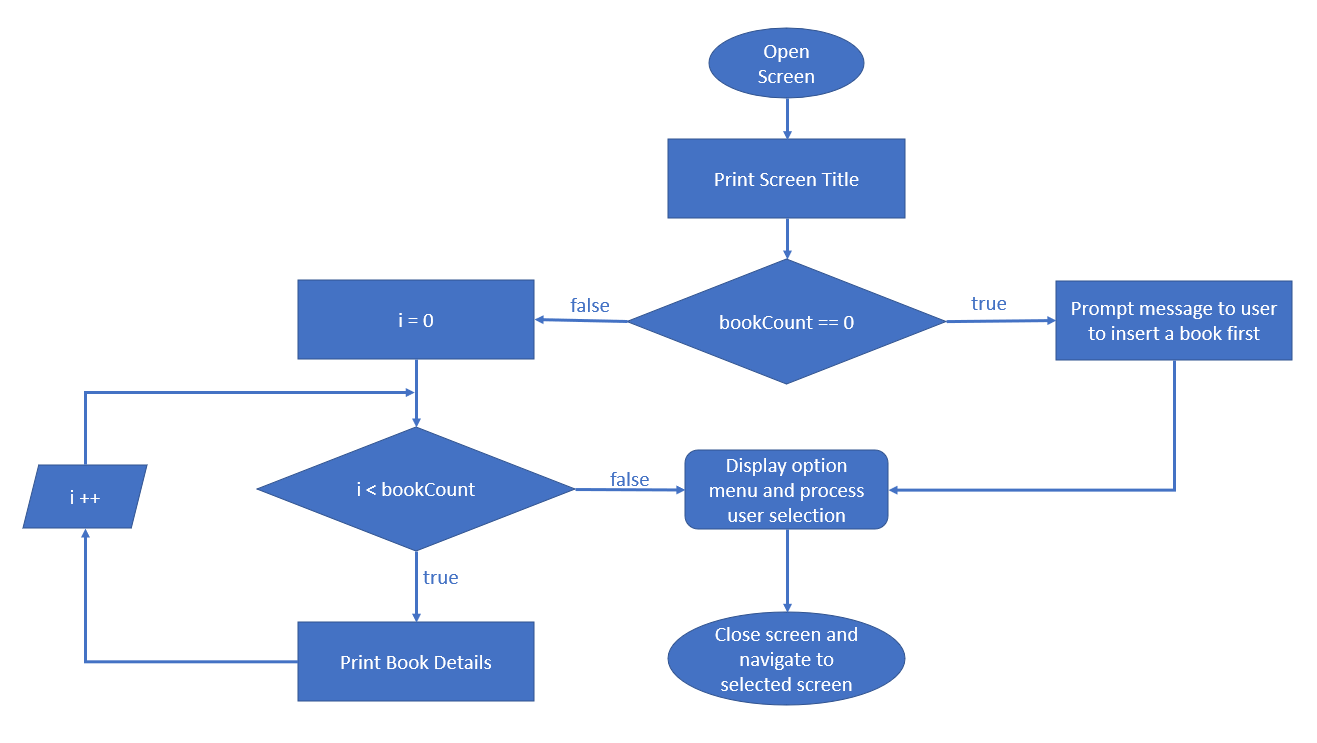
1 - Search book

2 - Insert new book

3 - Delete book

4 - Back to main menu

Figure 8: Block diagram describes the option flow



The code to generate the above options:

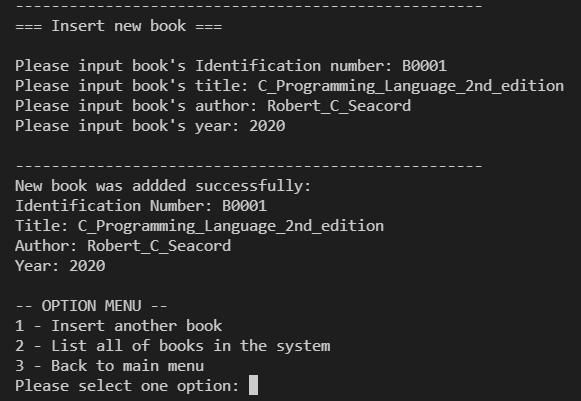
|  |
| --- |
| void displayListOfBooksScreen()  {      printf(SEPARATOR);      printf("=== List of Books in System ===\n\n");      //In case of no book added, prompt message to add book first.      if (bookCount == 0)      {          printf("There is no book added yet. Please insert a book first!\n");      }      else      {          //Make a loop to traverse throught books array          //in order to print individual book detail          for (int i = 0; i < bookCount; i++)          {              printBookDetail(books[i]);              printf(SEPARATOR);          }      }      //These are options supported in List of Books screen.      char \*options[4] = {          "1 - Search book",          "2 - Insert new book",          "3 - Delete book",          "4 - Back to main menu"};      char optionCode = displayOptionSelection(options, 4);      //This section for screen navigation      //Option 1 selected -> display Search Book screen      if (optionCode == options[0][0])      {          displayBookSearchScreen();      }      //Option 2 selected -> display Insert Book screen      else if (optionCode == options[1][0])      {          displayBookInsertionScreen();      }      //Option 3 selected -> display Delete Book screen      else if (optionCode == options[2][0])      {          displayBookDeletionScreen();      }      //Option 4 selected -> display Main screen      else if (optionCode == options[3][0])      {          displayMainScreen();      }  } |

### Insert New Book Screen

* ***Screen Description***

This screen will allow an user to insert new book into system.

Figure 9: "Insert New Book" Screen



In general, the book has three main menus:

1 - Insert another book

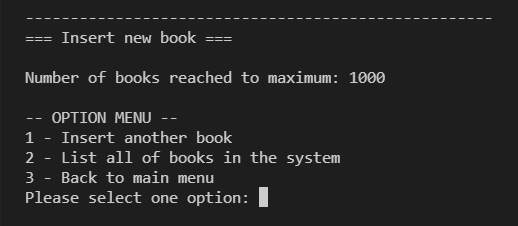
2 - List all of books in the system

3 - Back to main menu

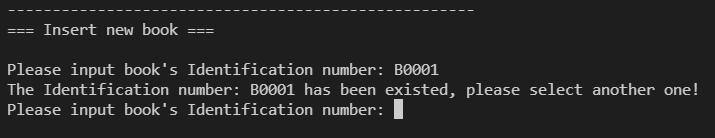
* ***Validation of the Book Identification Number***

I have designed the maximum number of books is 1000. In case the number of book reaches this threshold, the following message is shown on the screen to inform the user.

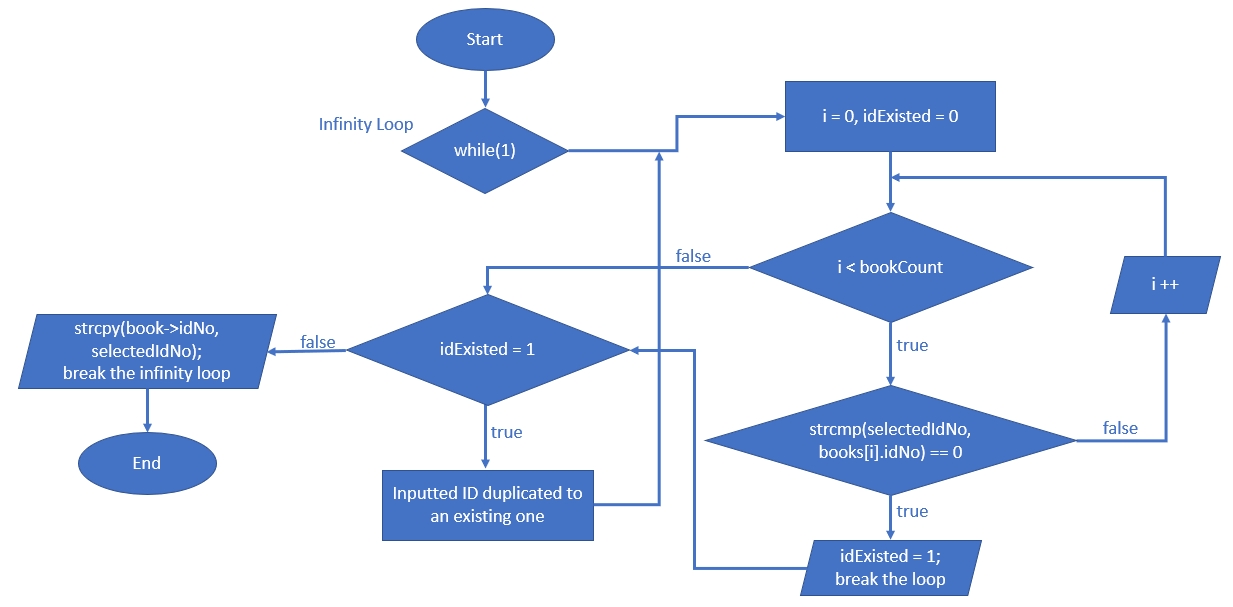
Figure 10: "Insert New Book" Screen if maximum number of books is reached.



As I mentioned above, book’s identification number is unique. Each time , the user keys in an identification number, the application needs a validation to validate inputted value. If this value is not duplicated with any existing book’s identification number in the system, the application allows the user to input other information (title, author and year). Otherwise, it will prompt a message to inform the user on existing book identification number.



The below block diagram describes the validation process, in order to let the user re-input again after each validation failed. In other words, an infinity loop is used and the loop only exists once the user keys in a valid value.

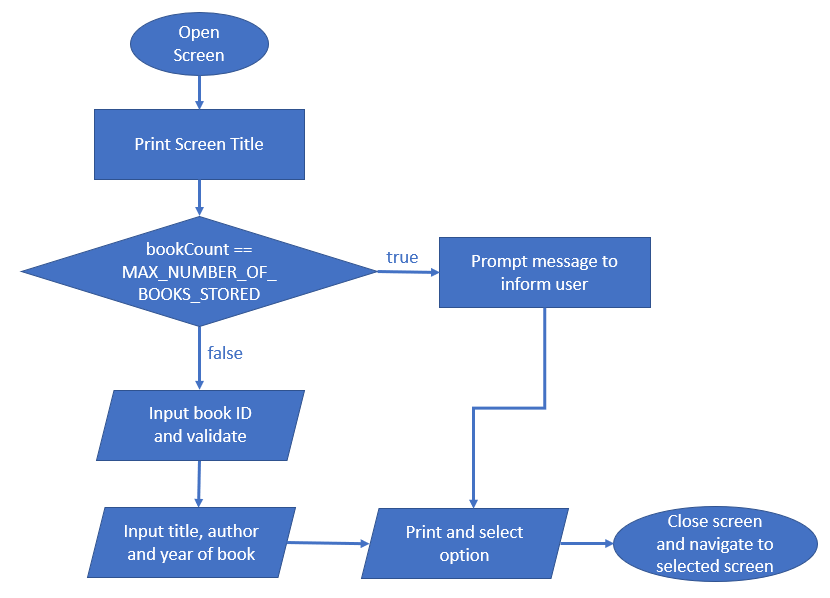
Figure 11: Infinite loop to validate identification number

The following code is used to generate the validation above:

|  |
| --- |
| //This function is for inputting book's Identification number  //and validate with existing IDs after  void inputBookIdAndValidate(Book \*book)  {      char idNo[20];      //infinity loop for user to re-input book's ID number in case of inputted ID number existed.      while (1)      {          printf("Please input book's Identification number: ");          char selectedIdNo[20];          scanf("%s", selectedIdNo);          int idExisted = 0; //This's a flag to check whether inputted ID is existed in the  system or not.          for (int i = 0; i < bookCount; i++)          {              if (strcmp(selectedIdNo, books[i].idNo) == 0) //if inputted ID equals to an  existing ID in the system, then update flag 'idExisted' and exit loop.              {                  idExisted = 1;                  break;              }          }          if (idExisted) //if flag 'idExisted' = 1 means it's existed already,  prompt message to user to input again.          {              printf("The Identification number: %s has been existed, please  select another one!\n", selectedIdNo);          }          else          {  //if inputted ID is not existed in the system, copy value to selected book and exit loop.              strcpy(book->idNo, selectedIdNo);              break;          }      }  } |

* ***Codes to generate options for the screen***

Figure 12: Block Diagram to generate different options for this screen



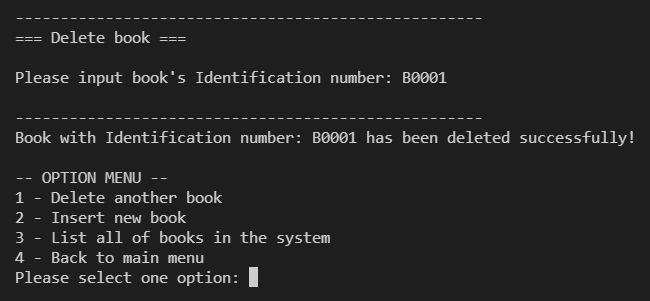
The following code is to used to generate the options for this screen

|  |
| --- |
| //This function for generating and operating Insert Book screen  void displayBookInsertionScreen()  {      printf(SEPARATOR);      printf("=== Insert new book ===\n\n");      //If the number of books reached to the maximum, prompt message to inform user.      if (bookCount == MAX\_NUMBER\_OF\_BOOKS\_STORED)      {          printf("Number of books reached to maximum: %d\n", MAX\_NUMBER\_OF\_BOOKS\_STORED);      }      else      {          //This method for inputting and validating book's Identification number.          inputBookIdAndValidate(&books[bookCount]);          printf("Please input book's title: ");          scanf(" %s", books[bookCount].title);          printf("Please input book's author: ");          scanf(" %s", books[bookCount].author);          printf("Please input book's year: ");          scanf(" %d", &books[bookCount].year);          //print book's detail after adding          printf(SEPARATOR);          printf("New book was addded successfully:\n");          printBookDetail(books[bookCount]);          bookCount++; //increase bookCount by 1 after each insertion      }      //These are options supported in Insert Book screen.      char \*options[3] = {          "1 - Insert another book",          "2 - List all of books in the system",          "3 - Back to main menu"};      char optionCode = displayOptionSelection(options, 3);      //This section for screen navigation      //Option 1 selected -> display Insert Book screen      if (optionCode == options[0][0])      {          displayBookInsertionScreen();      }      //Option 2 selected -> display List of Books screen      else if (optionCode == options[1][0])      {          displayListOfBooksScreen();      }      //Option 3 selected -> display Main screen      else if (optionCode == options[2][0])      {          displayMainScreen();      }  } |

### Delete Book screen

* ***Screen Description***

This screen is purported to delete an existing book by entering its identification number. While other values such as title, authors and years allow duplication, identification number is unique. Therefore, identification is chosen as the key-in value to delete a book since it is always able to specify a book with a given ID.



Generally, there are 4 functions on this screen

1 - Delete another book

2 - Insert new book

3 - List all of books in the system

4 - Back to main menu

If no book is found, the following message is shown to the user.

Figure 13: "Delete Book" Screen if no books are found

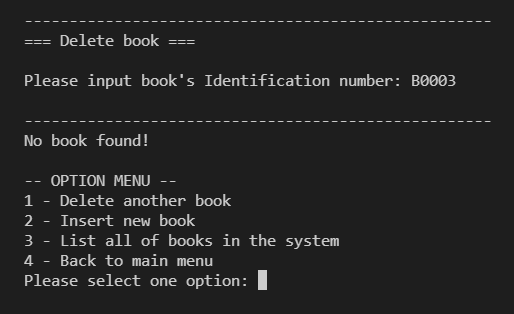
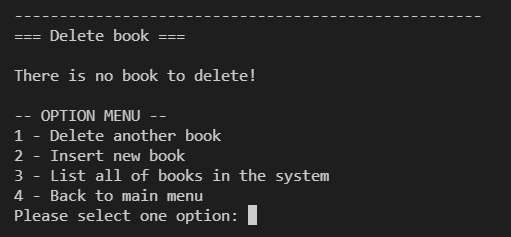


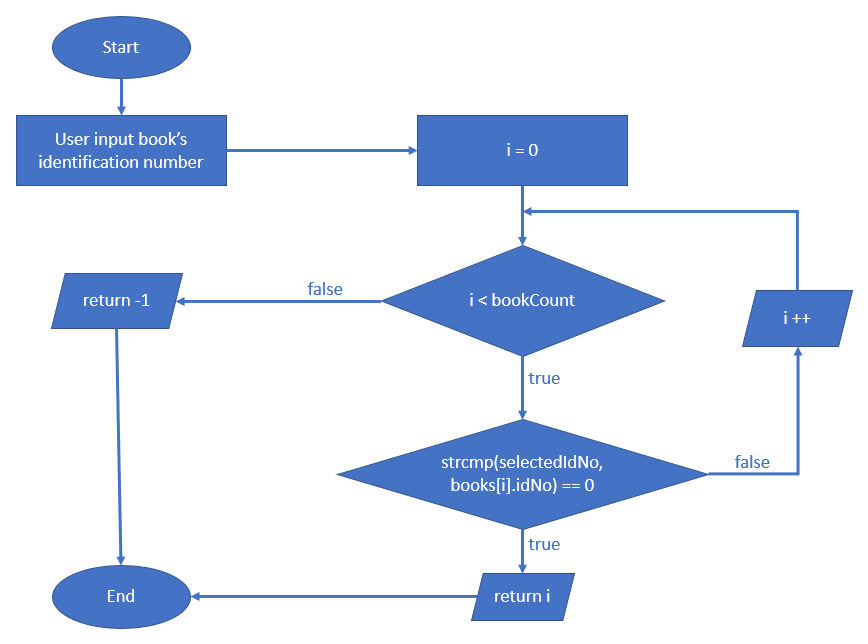
Figure 14: "Delete book" Screen if no books exist in the system



In order to specify the book with given ID, search function is used. The function will receive book’s identification number from user, traverse through all books stored in the system by a loop. If any book’s ID match to the given value, break the loop and return the index of book in the book array (return -1 if no book found).

* ***How deleting function is designed and works***

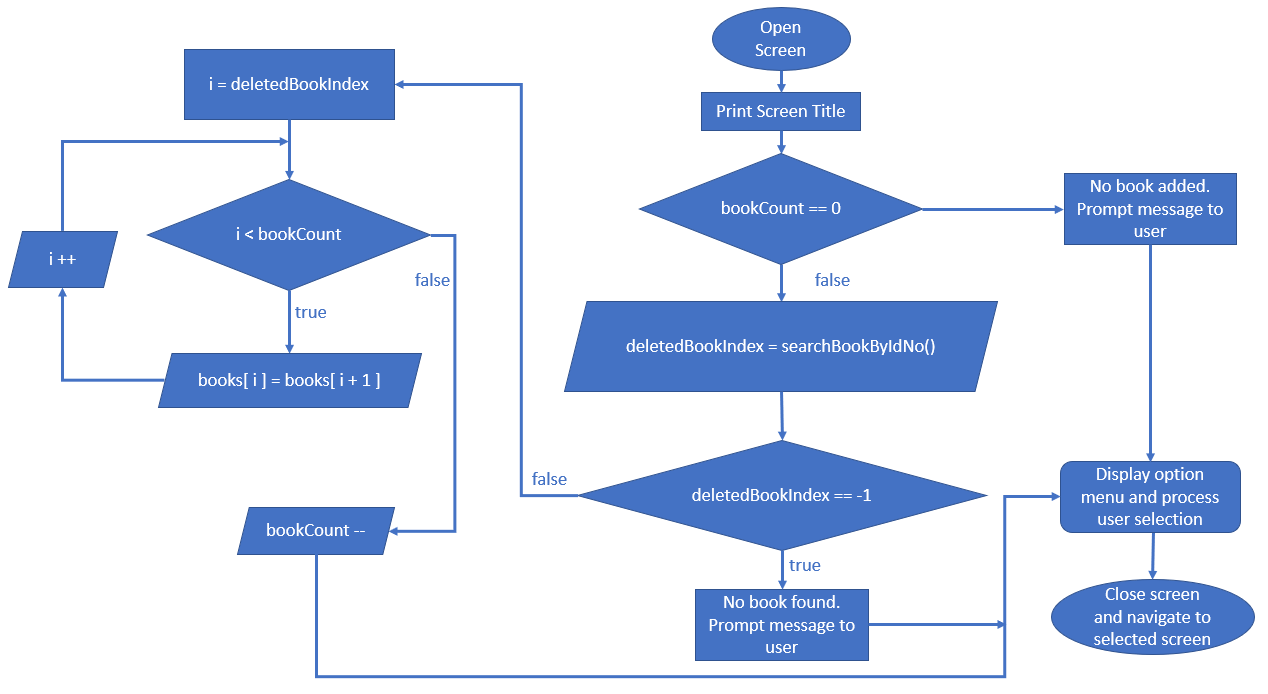
Figure 15: Block Diagram which describes the search function



The following code is to generate the loop function

|  |
| --- |
| //This function is for searching Book by Identification number from user input  //and return index of book in books array (return -1 if no book found)  int searchBookByIdNo()  {      printf("Please input book's Identification number: ");      char selectedIdNo[20];      scanf("%s", selectedIdNo);      for (int i = 0; i < bookCount; i++)      {          if (strcmp(selectedIdNo, books[i].idNo) == 0)          {              return i;          }      }      return -1; //No book found  } |

Should the value from the search function returns -1, it implies that no book is found and the application is designed to push notification to the user. Otherwise, the application start deleting books as inputted by the user. The algorithm I applied here is shifting all books on the right side from deleted book index to left by 1. Shifting is implemented by coping value of right element to the left element. Finally, bookCount (the global variable for counting books stored) will be decreased by 1.

Figure 16: The block diagram describing how books are deleted

The following code is used for the above diagram

|  |
| --- |
| /This function for generating and operating Delete Book screen  void displayBookDeletionScreen()  {      printf(SEPARATOR);      printf("=== Delete book ===\n\n");      //If there is no book added yet, prompt message to user.      if (bookCount == 0)      {          printf("There is no book to delete!\n");      }      else      {          int deletedBookIndex = searchBookByIdNo();          //searchBookByIdNo function return -1 means no book found.          if (deletedBookIndex == -1)          {              printf(SEPARATOR);              printf("No book found!\n");          }          else          {              char idNo[20];              strcpy(idNo, books[deletedBookIndex].idNo);                //start deleting book              for (int i = deletedBookIndex; i < bookCount; i++)              {                  //shift all books on the right side from deletedBookIndex to left by 1                  books[i] = books[i + 1];              }              bookCount--; //decrease bookCount by 1 after deletion              printf(SEPARATOR);              printf("Book with Identification number: %s has been deleted successfully!\n", idNo);          }      } |

### Search book screen

* ***Screen Description***

This screen is for searching book by a given book’s identification number and display book’s details to an user. It also uses ‘searchBookByIdNo’ function to proceed searching. Generally, 5 options are available for this screen:

1 - Search another book

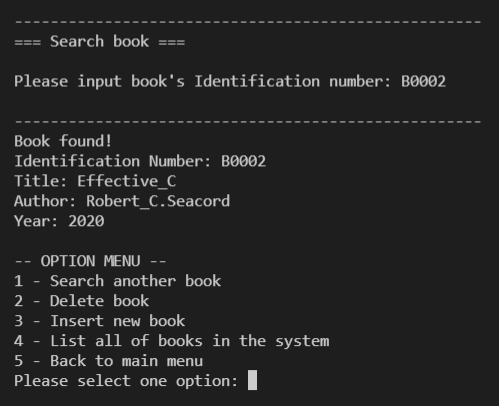
2 - Delete book

3 - Insert new book

4 - List all of books in the system

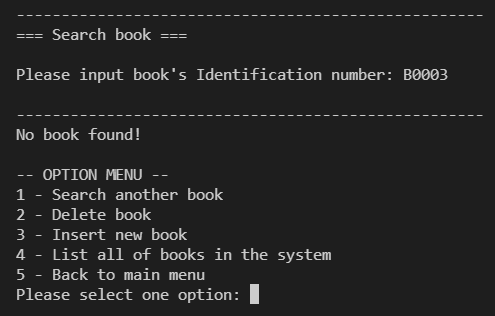
5 - Back to main menu

Figure 17: "Search book" Screen



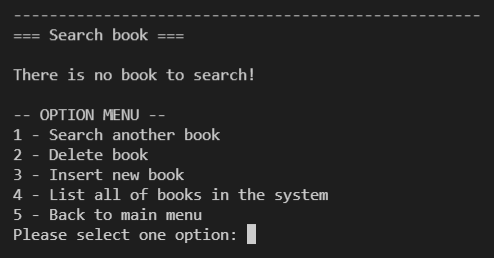
If no book is found, the following message is pushed to the user.

Figure 18: The Search Screen in case no book is found



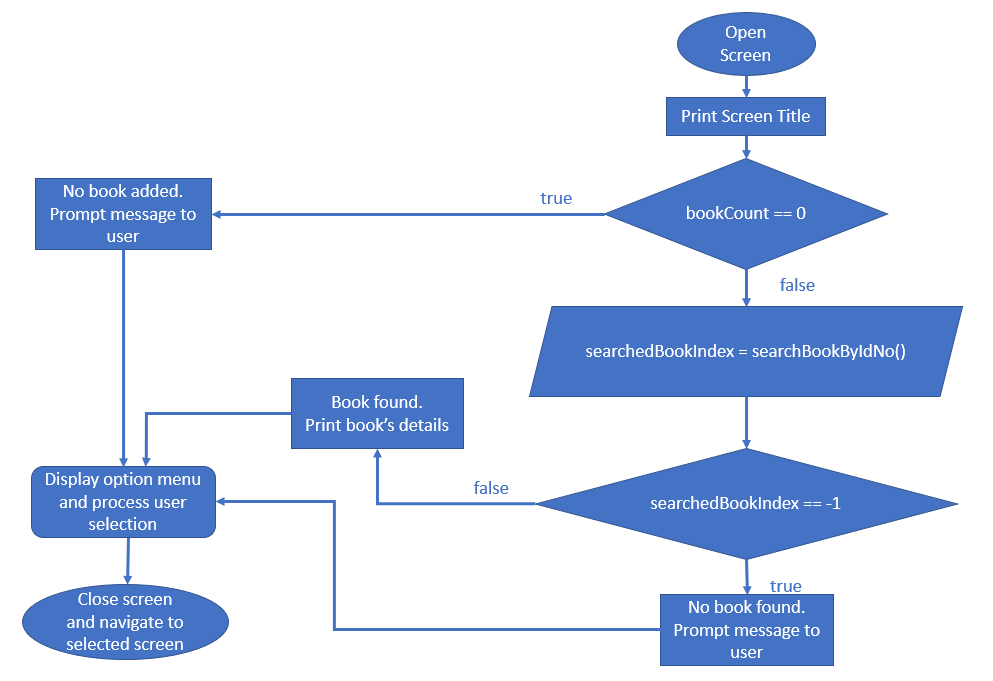
In case of no book exists, the application will prompt message to inform user.

Figure 19: The "Search Screen" in case of no book exists



* ***How options of Search Screen is designed and work***

Figure 20: Block diagram describing the option flow



The following code is used to generate the above functions

|  |
| --- |
| //This function for generating and operating Search Book screen  void displayBookSearchScreen()  {      printf(SEPARATOR);      printf("=== Search book ===\n\n");      //In case of no book added yet, prompt message to user.      if (bookCount == 0)      {          printf("There is no book to search!\n");      }      else      {          int searchedBookIndex = searchBookByIdNo();          //searchBookByIdNo method return -1 means no book found.          if (searchedBookIndex == -1)          {              printf(SEPARATOR);              printf("No book found!\n");          }          else          {              //If there is book found with inputted ID number              //then print inform user and print book's details              printf(SEPARATOR);              printf("Book found!\n");              printBookDetail(books[searchedBookIndex]);          }      }      //These are options supported in Search Book screen.      char \*options[5] = {          "1 - Search another book",          "2 - Delete book",          "3 - Insert new book",          "4 - List all of books in the system",          "5 - Back to main menu"};      char optionCode = displayOptionSelection(options, 5);      //This section for screen navigation      //Option 1 selected -> display Search Book screen      if (optionCode == options[0][0])      {          displayBookSearchScreen();      }      //Option 2 selected -> display Delete Book screen      else if (optionCode == options[1][0])      {          displayBookDeletionScreen();      }      //Option 3 selected -> display Insert Book screen      else if (optionCode == options[2][0])      {          displayBookInsertionScreen();      }      //Option 4 selected -> display List of Books screen      else if (optionCode == options[3][0])      {          displayListOfBooksScreen();      }      //Option 5 selected -> display Main screen      else if (optionCode == options[4][0])      {          displayMainScreen();      }  } |

# Conclusion and Further Work

The application I have developed did work well during the process of implementation. However, there are a few issues which I believe needs further development work in the future. First of all, when entering the title of the book, the application does not allow blanks so I have to use “-“ to separate different words of a book name. Secondly, the “List of Books Screen” is in its basic form rather than table form which contains columns and rows for management. In order to improve the program, possible options is to modify the display of the “List of Books” Screen and the export function can also be added to enable users to have some reporting function.