# Book Management Application

I - Requirement

Write a complete, well-structured program that begins by asking the user to enter bibliographic information of library items, such as books, CDs, photographs. The program will produce a database with the input values and display the stored ones to users.

II – General Architecture

0. Application Functionalities.

From requirements, the application will have 4 main functionalities:

- List all of books in the system

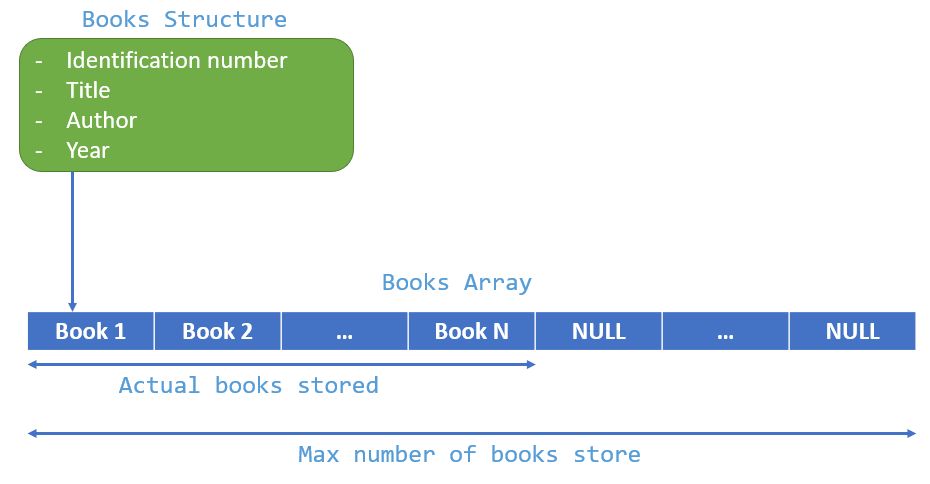
- Insert new book

- Delete existing book

- Search book

1. Data Structure

In order to store Book information, we need to design a structure for Book first. It should keep some important information of a book such as: Identification Number, Title, Author, Year. Books inserted will be stored in a global array.



Implementation:

Max number of books store

|  |
| --- |
| #define DEFAULT\_BOOK\_STORAGE\_CAPACITY 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;  int bookStorageCapacity = DEFAULT\_BOOK\_STORAGE\_CAPACITY;  Book \*books;       //A global pointer variable pointing to book storage.  int bookCount = 0; //A global integer variable to count the number of inserted books. |

Books array will be initialized and allocated to heap memory dynamically using “malloc” built-in function when the application starts and calls “main” function. Before exiting application, the books array memory will be deallocated by calling “free” function.

The code is implemented as below:

|  |
| --- |
| int main()  {      //init book storage array      books = (Book \*)malloc(bookStorageCapacity \* sizeof(Book));      //In startup, the application will show Main screen      displayMainScreen();      //free allocated memory of book storage array      free(books);      return 0;  } |

2. Application structure

As application functionalities, the application will have 6 corresponding screens:  
-Main screen.

-List of Books screen

-Insert Book screen

-Delete Book screen

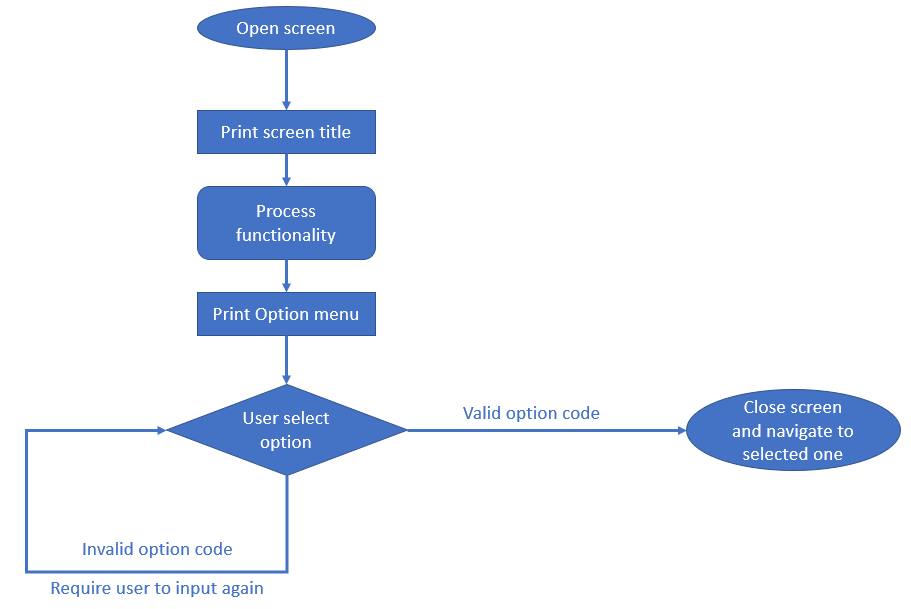
-Search Book screen

-Adjust book storage capacity screen

On each screen has its own functionalities, and can navigate to another screen from a specific screen.

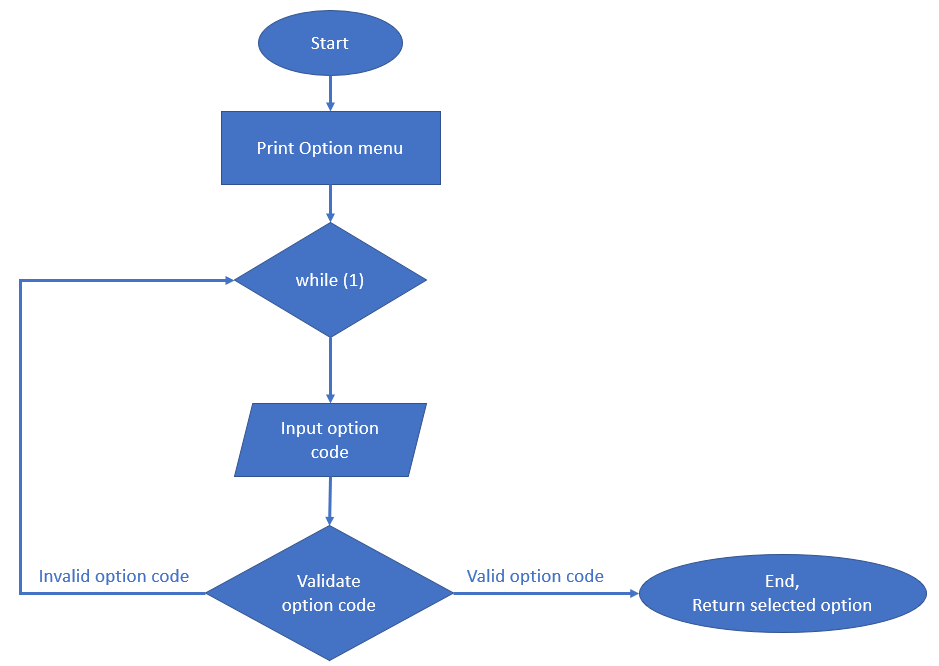
Generally, each screen also has same points in operation. They show screen title, content with their functionality, and option menu.

Block Diagram:



Printing option menu and letting user select option should be done in a common function, and can be used by all screens. In term of programming, I intend to make an infinity loop to deal with re-inputting option code when user keyed-in an invalid one before.

Block Diagram:

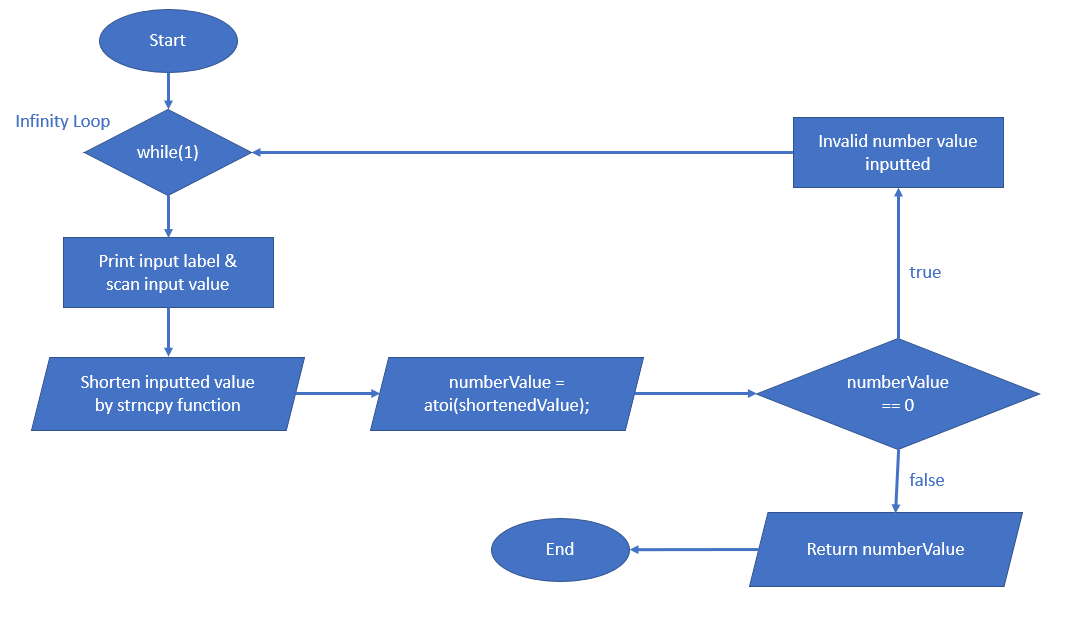


Implementation:

|  |
| --- |
| //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]);      }      int inputtedOptionNumber;      char selectedOptionCode;      //It's just a flag to check if any option match to input value      int validOptionCode = 0;      char optionCode;      while (1)      {          inputtedOptionNumber = inputNumberAsStringAndConvert("Please select one option: ", 1);          //since number options may be less than 10, so we can convert int to char by adding value by '0'.          selectedOptionCode = inputtedOptionNumber + '0';          for (int i = 0; i < size; i++)          {              //If it matches to an existing option, toggle value of the flag 'validOptionCode' and exit loop.              optionCode = options[i][0];              if (selectedOptionCode == optionCode)              {                  validOptionCode = 1;                  break;              }          }          //If inputted option doesn’t match to any available option code, then prompt message and let user try again.          if (!validOptionCode)          {              printf("You just entered invalid option code, please try again\n");          }          else          {              //exit infintity loop and return selected option code              return selectedOptionCode;          }      }  } |

As the code snippet above, there is a function “inputNumberAsStringAndConvert” used to read input value from user and convert to number. In the application, some places need this function such as input book’s year, or update book storage capacity.

Block Diagram:

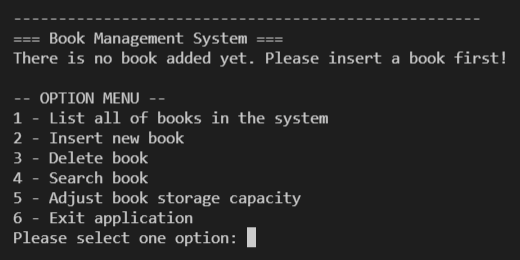


Implementation:

|  |
| --- |
| //This function is for user to input number value as string,  //then validate, return an corresponding integer value.  int inputNumberAsStringAndConvert(char \*inputLabel, int numberOfDigits)  {      //inputted number value is supposed to have 20 digits or less.      char inputValue[20];      char shortenedValue[numberOfDigits];      //Use infinity loop in case of invalid number string inputted.      while (1)      {          printf("%s", inputLabel);          scanf("%s%\*c", inputValue);          strncpy(shortenedValue, inputValue, numberOfDigits);          int numberValue = atoi(shortenedValue);          //the inputted value is invalid and cannot convert to an integer          if (numberValue == 0)          {              printf("Inputted value is invalid. Please try again!\n");          }          else          {              //return the converted number              return numberValue;          }      }  } |

2.1 Main screen

First of all, once use run the application, the main screen will be shown to user. If no book was added yet, the screen will notice user to insert book first.



There are 6 options for Main screen:

1- List all of books in the system

2 - Insert new book

3 - Delete book

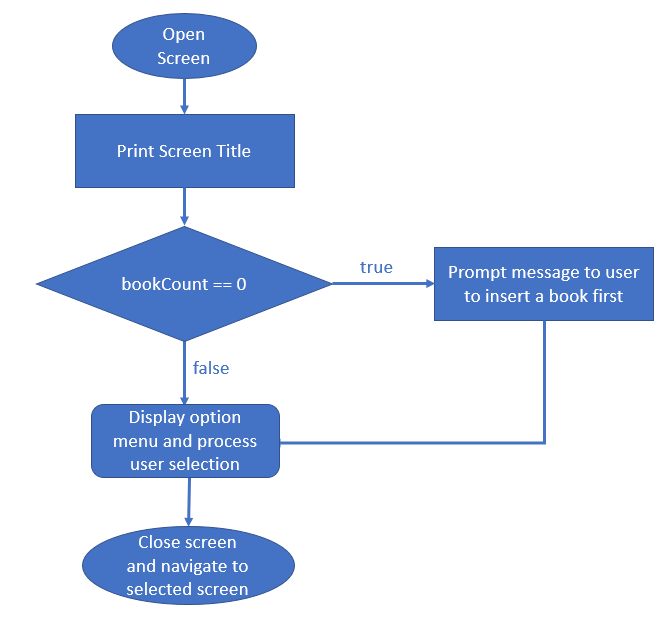
4 - Search book

5 – Adjust book storage capacity

6 - Exit application

Except of the last option, user will be navigated to the particular screen after he selects.

Block Diagram:

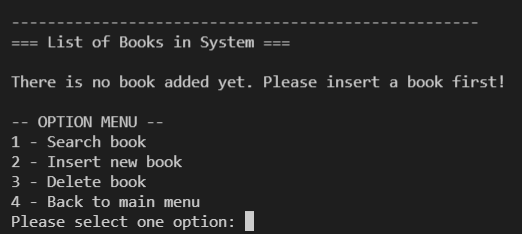


Implementation:

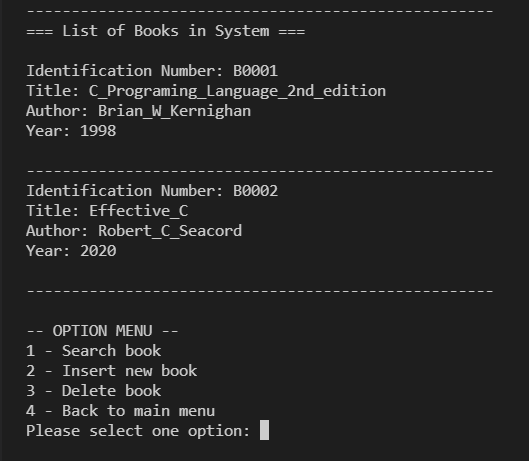
|  |
| --- |
| //This function for generating and operating Main screen  void displayMainScreen()  {      printf(SEPARATOR);      printf("=== Book Management System ===\n");      if(bookCount == 0) {          printf("There is no book added yet. Please insert a book first!\n");      }      //These are 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.  } |

2.2- List of Books screen

This screen will list all of books stored in the system. In case of no book added yet, it prompts a message to user to insert a book first.



Otherwise, it will show all books with details stored in the system.



This screen has 4 available options:

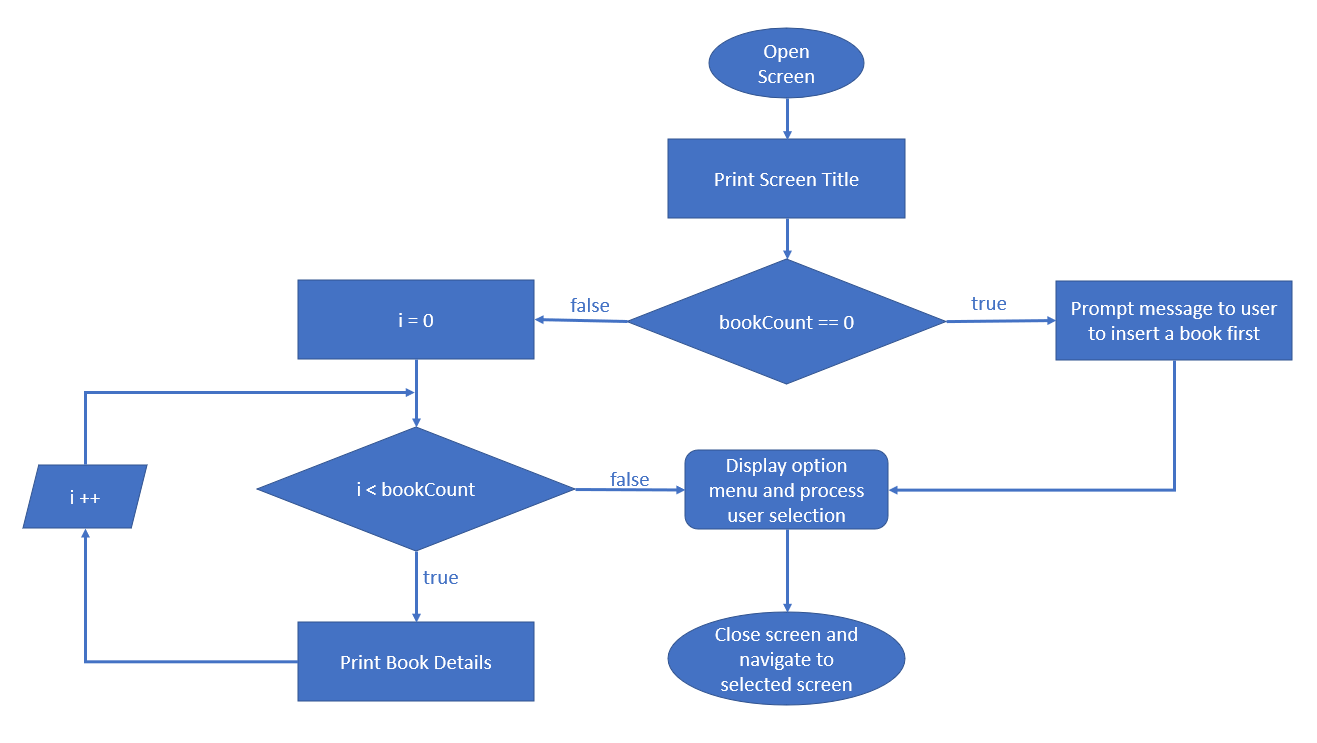
1 - Search book

2 - Insert new book

3 - Delete book

4 - Back to main menu

Block diagram

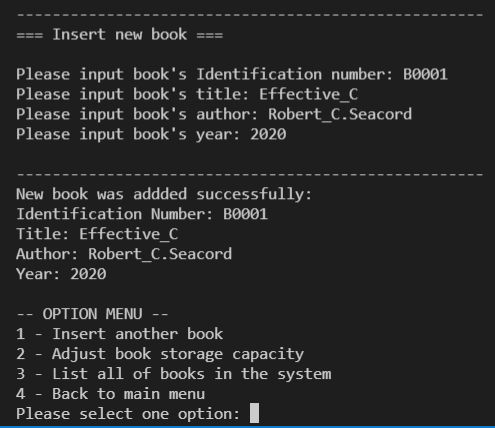


Implementation:

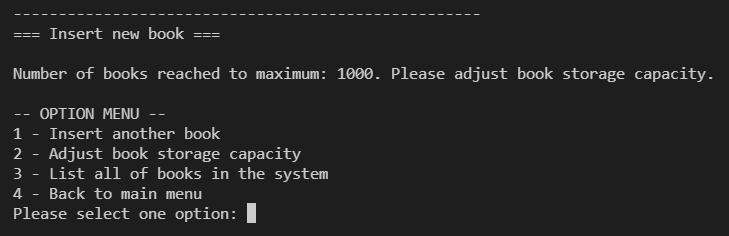
|  |
| --- |
| void displayListOfBooksScreen()  {      printf(SEPARATOR);      printf("=== List of Books in System ===\n\n");      //In case of no book added, prompt message to add book first.      if (bookCount == 0)      {          printf("There is no book added yet. Please insert a book first!\n");      }      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();      }  } |

2.3 – Insert Book screen

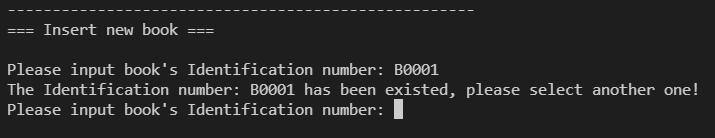
This screen will allow user to insert new book into system.



In the case that number of books reached to the limit of books stored, prompt message to inform user.

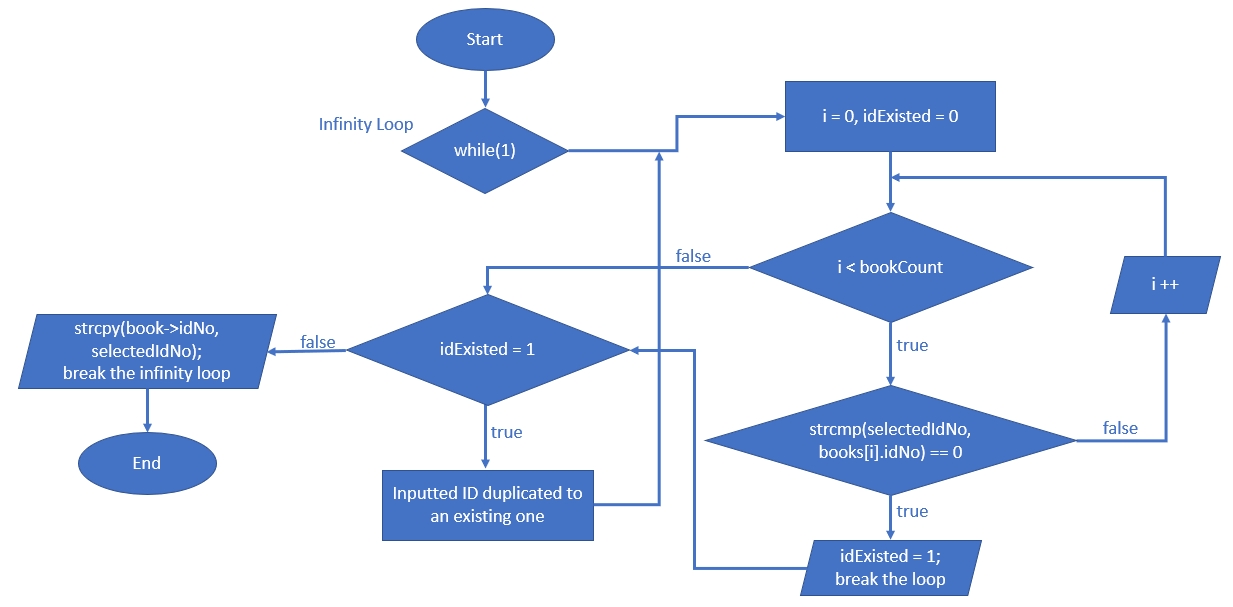


As I mentioned above, book’s identification number is unique. So, each time user key-in identification number, application will make a validation to validate inputted value. If this value is not duplicated with any existing book’s identification number in the system, application will allow user to keep inputting other book information (title, author and year). Otherwise, it will prompt a message to let user know the duplication.



Let’s take a look into the validation function, in order to let user re-input again after each validation failed, we use an infinity loop, and just only exit the loop once user key-in a value which is not duplicated.

Block Diagram:



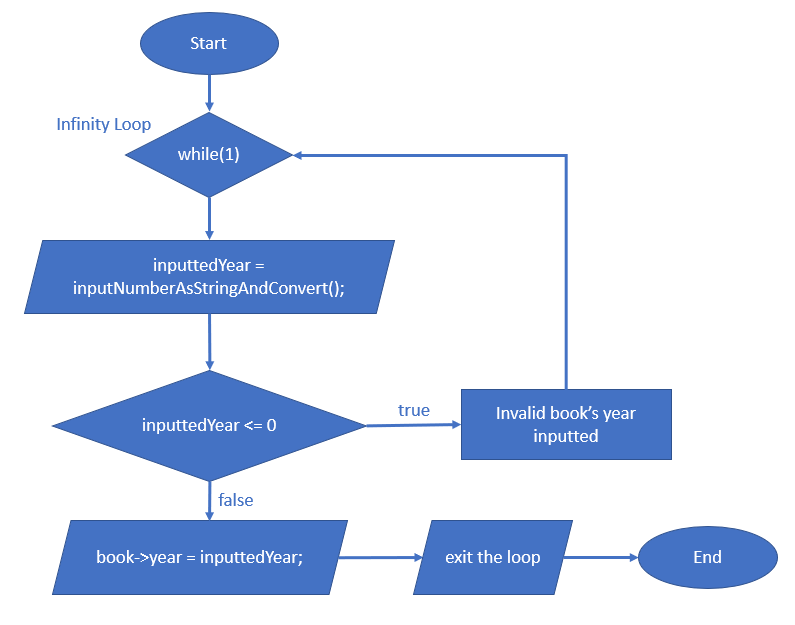
Implementation:

|  |
| --- |
| //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;          }      }  } |

There is also another function for inputting and validating book’s year. Actually, book’s year just need to be a positive integer number, and it should have 4 digits or less.

Remember that we have a function “inputNumberAsStringAndConvert” for inputting number value. This function can be reused here.

Block Diagram:



Implementation:

|  |
| --- |
| int inputBookYearAndValidate(Book \*book)  {      //Use infinity loop in case of book's year invalid.      while (1)      {          int inputtedYear = inputNumberAsStringAndConvert("Please input book's year: ", 4);          //In case of invalid number inputted.          if (inputtedYear <= 0)          {              printf("Invalid number inputted. Please try again!\n");              continue;          }          //Otherwise, update book's year value and exit the loop.          else          {              book->year = inputtedYear;              break;          }      }  } |

There are 4 options for this screen:

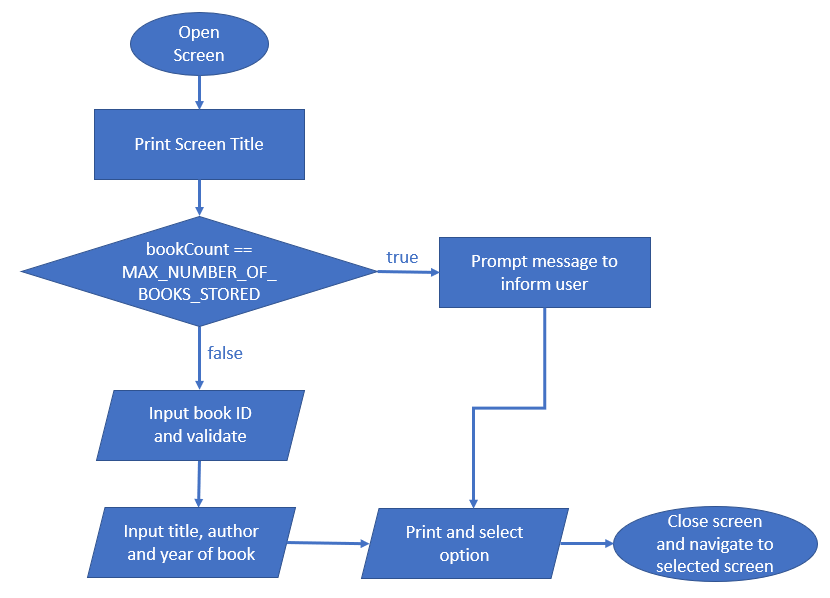
1 - Insert another book

2 – Adjust book storage capacity

3 - List all of books in the system

4 - Back to main menu

Block diagram of Insert book screen:

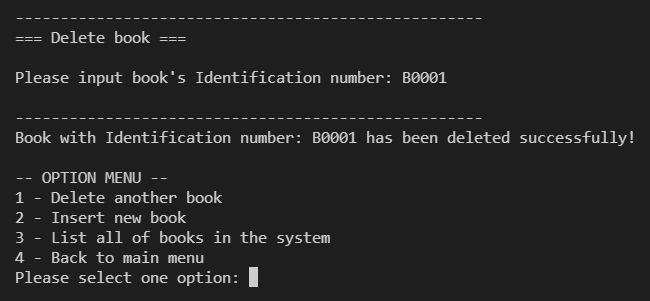


Implementation:

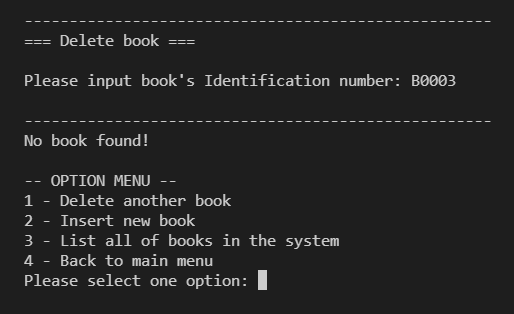
|  |
| --- |
| //This function for generating and operating Insert Book screen  void displayBookInsertionScreen()  {      printf(SEPARATOR);      printf("=== Insert new book ===\n\n");      //If the number of books reached to the maximum, prompt message to inform user.      if (bookCount == bookStorageCapacity)      {          printf("Number of books reached to maximum: %d. Please adjust book storage capacity.\n", bookCount);      }      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);          inputBookYearAndValidate(&books[bookCount]);          //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[4] = {          "1 - Insert another book",          "2 - Adjust book storage capacity",          "3 - List all of books in the system",          "4 - Back to main menu"};      char optionCode = displayOptionSelection(options, 4);      //This section for screen navigation      //Option 1 selected -> display Insert Book screen      if (optionCode == options[0][0])      {          displayBookInsertionScreen();      }      //Option 2 selected -> display Adjust book storage capacity screen      else if (optionCode == options[1][0])      {          displayAdjustBookStorageCapacityScreen();      }      //Option 3 selected -> display List of Books screen      else if (optionCode == options[2][0])      {          displayListOfBooksScreen();      }      //Option 4 selected -> display Main screen      else if (optionCode == options[3][0])      {          displayMainScreen();      }  } |

2.4 – Delete Book screen

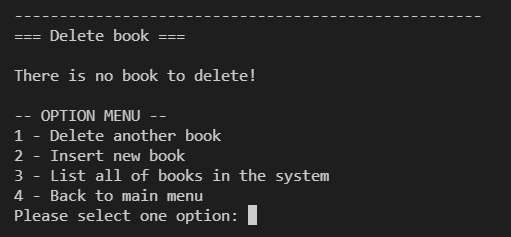
This screen is for deleting an existing book by its Identification number. Since identification number is unique, it is always able to specify a book with a given ID.



If no book found, prompt message to user.

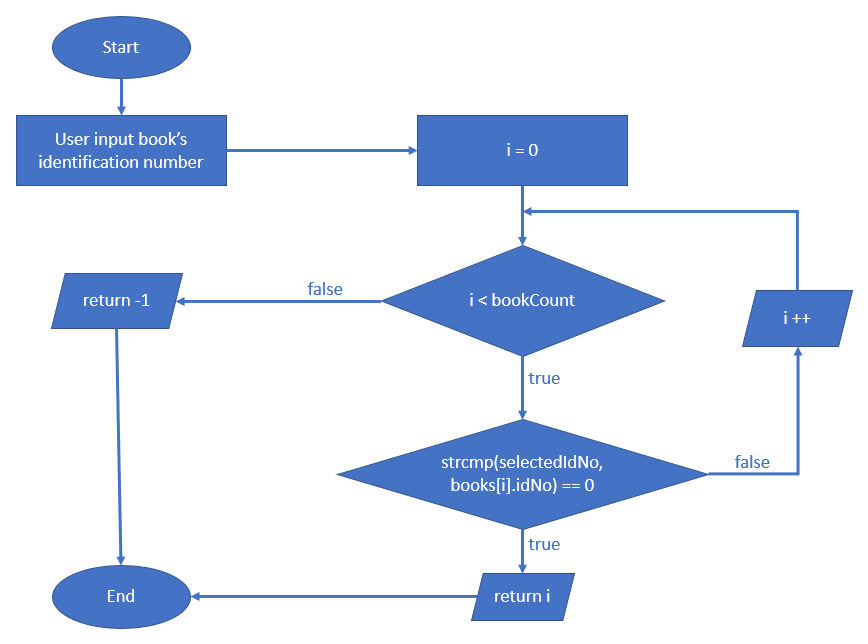


In the case of no book added yet, prompt message to user.



In order to specify the book with given ID, I have to write a search function to make it. 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).

Block diagram:

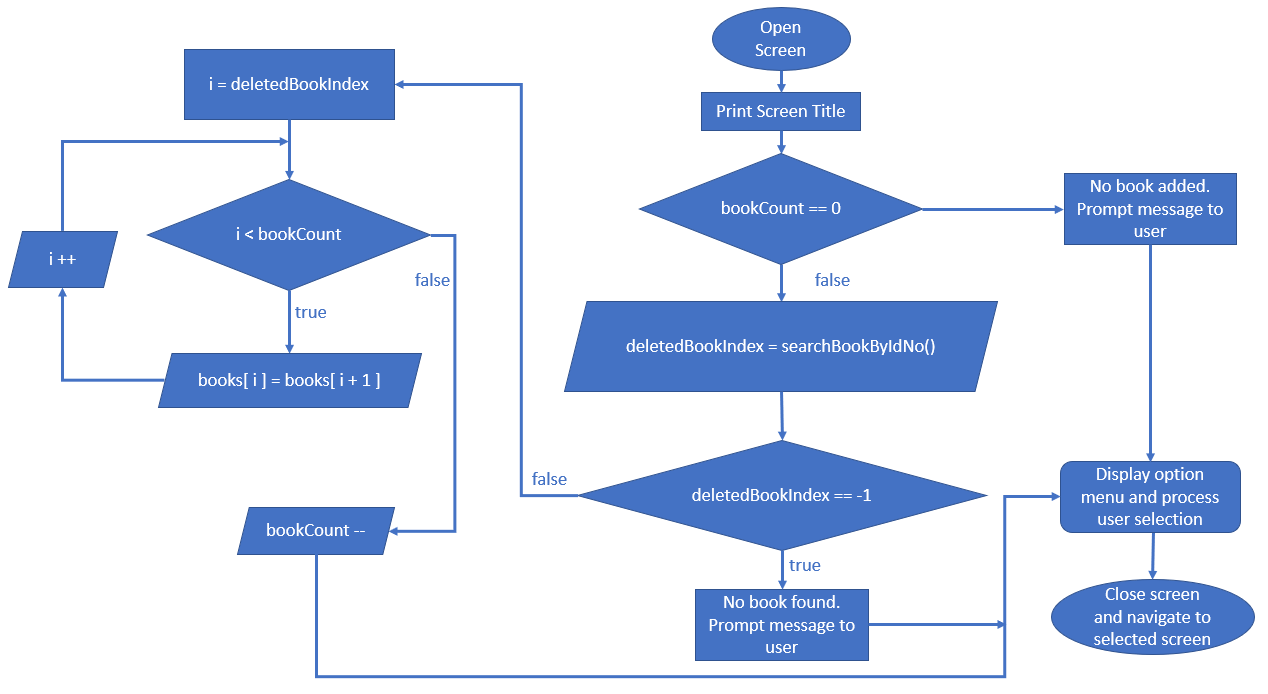


Implementation:

|  |
| --- |
| //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  } |

After result value from search function, if value = -1, it means no book found, application prompts message to inform user. Otherwise, application start deleting book. 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.

Block Diagram:



Implementation:

|  |
| --- |
| /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);          }      } |

There are 4 available options for this screen:

1 - Delete another book

2 - Insert new book

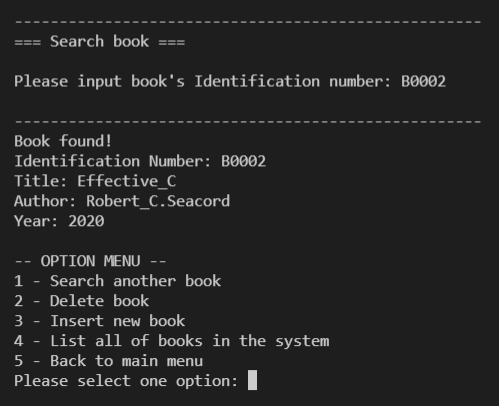
3 - List all of books in the system

4 - Back to main menu

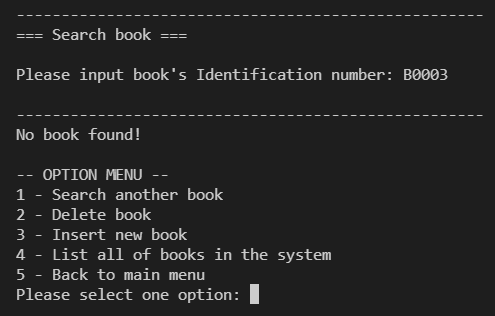
2.5 – Search book screen

This screen is for searching book by a given book’s identification number and display book’s details to user.

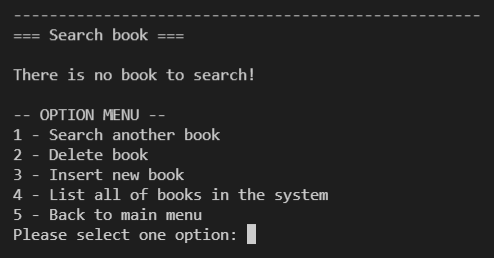
It also uses ‘searchBookByIdNo’ function to proceed searching.



If no book found, application will prompt message to inform user.



In case of no book added yet, application will prompt message to inform user.



There are 5 available option for this screen:

1 - Search another book

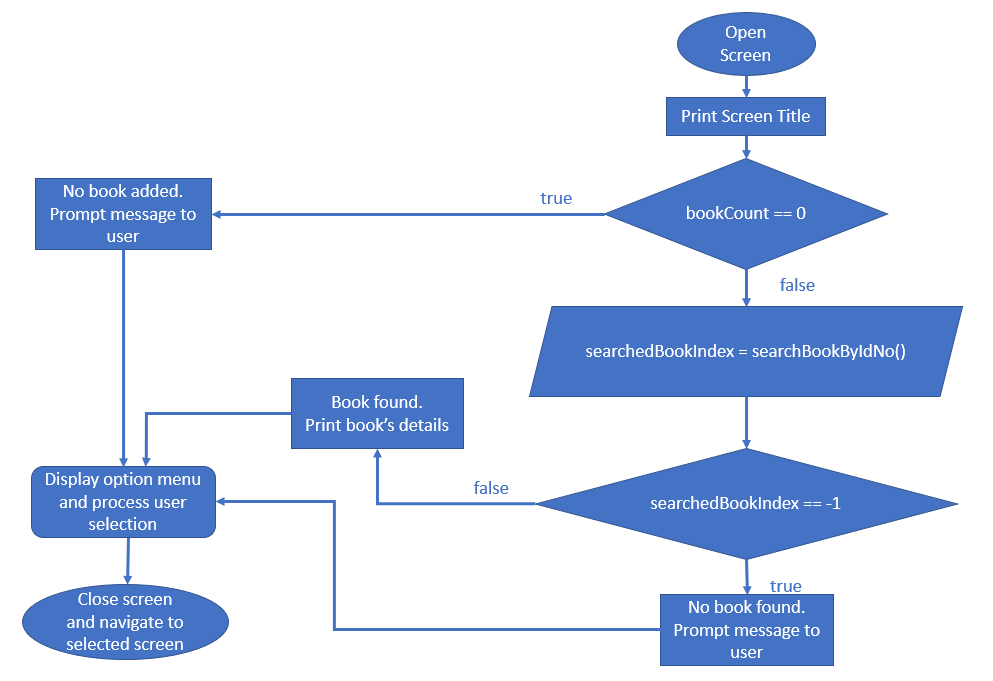
2 - Delete book

3 - Insert new book

4 - List all of books in the system

5 - Back to main menu

Block Diagram:

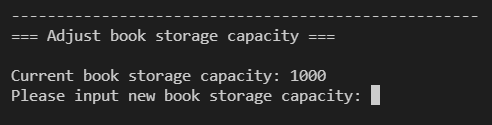


Implementation:

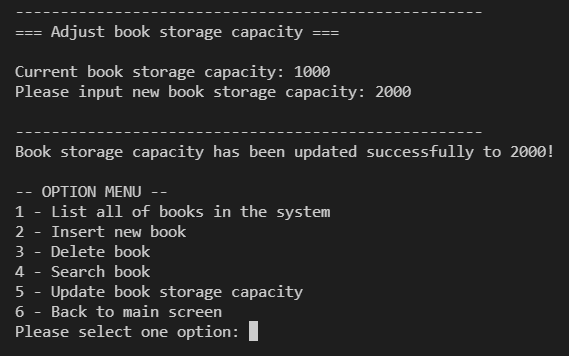
|  |
| --- |
| //This function for generating and operating Search Book screen  void displayBookSearchScreen()  {      printf(SEPARATOR);      printf("=== Search book ===\n\n");      //In case of no book added yet, prompt message to user.      if (bookCount == 0)      {          printf("There is no book 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();      }  } |

2.6 – Adjust book storage capacity screen.

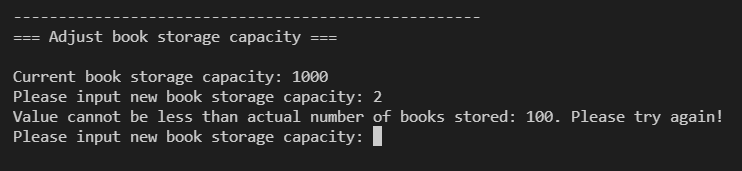
Once number of books in the storage reaches to the maximum, the application won’t allow user to add new book anymore, and notice user to update book storage capacity. Therefore, this screen lets user adjust book storage capacity.



In order to update new capacity successfully, user have to input a number greater than actual number of books stored.

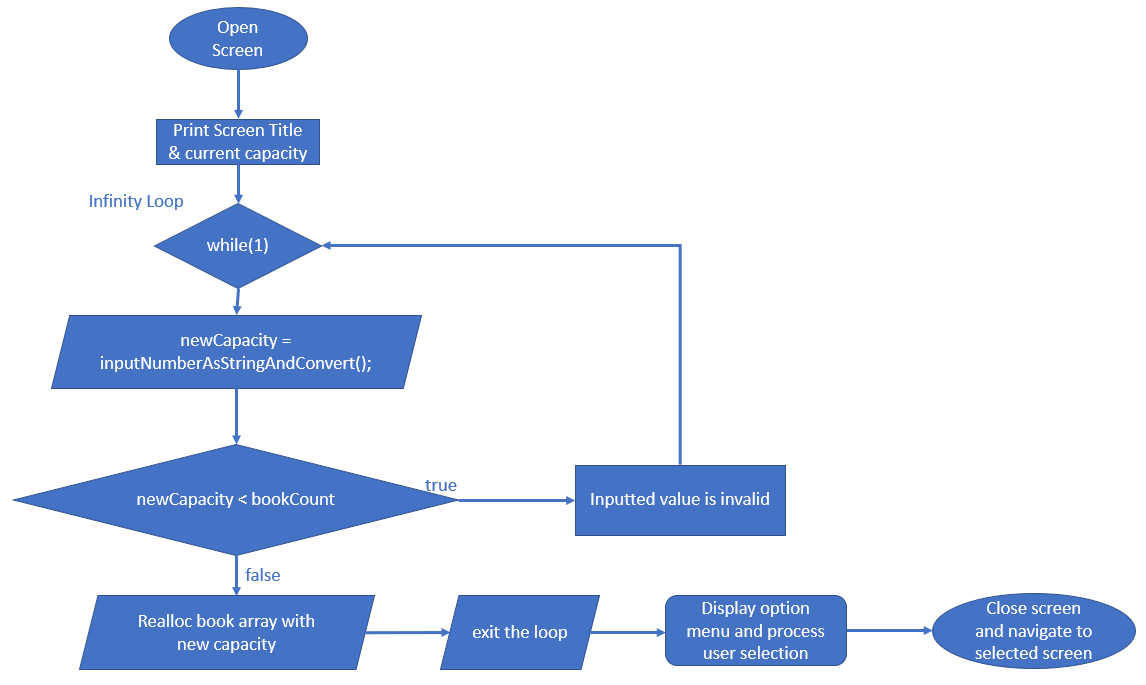


Otherwise, the message will be prompted to user.



This screen also use “inputNumberAsStringAndConvert” function to deal with inputting and validating new capacity value.

Block Diagram:



In term of C programming language, since books array is managed by a pointer and use dynamic memory allocation (using “malloc” function), in order to adjust size of book array, we shall use “realloc” function to deal with.

Implementation:

|  |
| --- |
| /This function for generating and operating Adjust book storage capacity screen  void displayAdjustBookStorageCapacityScreen()  {      printf(SEPARATOR);      printf("=== Adjust book storage capacity ===\n\n");      printf("Current book storage capacity: %d\n", bookStorageCapacity);      int newCapacity;      //Use an infinity loop in case of new book storage capacity invalid      while (1)      {          // printf("Please input new book storage capacity: ");          // scanf(" %d", &newCapacity);          newCapacity = inputNumberAsStringAndConvert("Please input new book storage capacity: ", 10); //newCapacity supposed has 10 digits or less.          //In case of inputted value less than number of books stored, require user to input again.          if (newCapacity < bookCount)          {              printf("Value cannot be less than actual number of books stored: %d. Please try again!\n", bookCount);              continue;          }          //Value is valid, update new capacity and break the infinity loop.          else          {              bookStorageCapacity = newCapacity;              books = (Book \*)realloc(books, bookStorageCapacity);              printf(SEPARATOR);              printf("Book storage capacity has been updated successfully to %d!\n", bookStorageCapacity);              break;          }      }      //These are options supported in Adjust book storage capacity screen.      char \*options[6] = {          "1 - List all of books in the system",          "2 - Insert new book",          "3 - Delete book",          "4 - Search book",          "5 - Update book storage capacity",          "6 - Back to main screen"};      char optionCode = displayOptionSelection(options, 6);      //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();      }      //Option 5 selected -> display Adjust book storage capacity screen      else if (optionCode == options[4][0])      {          displayAdjustBookStorageCapacityScreen();      }      //Option 6 selected -> display Main screen      else if (optionCode == options[5][0])      {          displayMainScreen();      }  } |

This screen has 6 available options:

1 - List all of books in the system

2 - Insert new book

3 - Delete book

4 - Search book

5 - Update book storage capacity

6 - Back to main screen