|  |
| --- |
| photo-1542831371-29b0f74f9713.jpg |
| Introduction to Computing Project Report |
| ‘Library Management System’ |
| **[SLO – 11.6]**  By: |
| Wasal Rasool [20L-1403] |
| Daniyal Ahmed [20L-1507]  **Lab Instructor:**  Zeeshan Asif  EL-116 |
| **December, 2020** |

Department of Electrical Engineering

National University of Computing & Emerging Sciences

Foundation for Advancement in Science & Technology

Lahore Campus

* **Contents:**
* **Introduction**  --------------------------------------------------- Page 03 - 03
* **Modern IT Tools –** PLO 05 ------------------------------ Page 04 - 06
* **Codes** – SLO 5.1 ----------------------------------------------- Page 07 - 37
* **Flowchart of Functions** – SLO 11.1, 11.2, 11.5 & 9.1 --- Page 38 - 45
* **Explanation & Design** – SLO 5.3 ------------------------- Page 46 - 51
* **Comparison** – SLO 5.2 --------------------------------------- Page 52 - 55
* **Inputs & Solutions** – SLO 5.4, 11.4 ------------------------ Page 56 - 56
* **Issues & Limitations** – SLO 5.5 --------------------------- Page 57 - 57
* **Conclusion** --------------------------------------------------- Page 58 - 58
* **Introduction**

A ‘library management system’ is a method of organization used by the keeper to maintain and control the information about a library. It stores the books, their location, their information and their records. A library management system such as this one can be used for a small library that requires a single keeper.

Library management systems can be manual such as written on some form of a book or computerized like this one and there are many advantages of using a computerized system over a manual one for example a manual system would require a lot of space while a computerized system requires a single storage device, a manual system would take hours to review and edit while a computerized one takes minutes and much more.

For this project a ‘Library management system’ was to be created with the standard features which include a main menu for the user to interact with the program, a way to store the ID of a student, to add new books, to delete from the available books, to edit the details of a book, to be able to issue or return a book from a student, to be able to search for a book and to be able to check the books that have already been issued.

This program does NOT include a graphical user interface and the entire program executes on the console through which the user can interact with it. The following guidelines were given: -

* Read data from file into corresponding arrays.
* Print the content of all arrays on console in the same format as shown in the input file.
* Your program should display the updated books list on the console after the execution of add, delete and edit operation.
* Your program should keep displaying the menu after each function call unless user wants to exit.
* Data type ‘String’ is NOT allowed.
* **Modern IT Tools –** PLO 05

This program is written in the C++ programming language and created in Microsoft Visual Studio.

C++ is general purpose programming language and a newer version of the ‘C’ programming language as it contains many modern features like generic and object-oriented programming as well as low-level memory manipulation. C++ is a compiled language, which means that it directly translates to the machine, rather than an interpreted language that requires an interpreter. This makes C++ a perfect choice for systems programming, embedded systems design and high performance applications such as video games, desktop applications and servers. C++ is a great choice and has many applications for any computer-related application in electrical engineering.

Microsoft Visual Studio is an integrated development environment (IDE). It is used to develop computer programs as well as websites, web apps, web services and mobile apps. Visual Studio includes a code editor with IntelliSense (similar to auto complete in mobile text apps but for code) as well as an integrated debugger which can be used as both a source-level or machine-level debugger. It also includes other features such as a GUI designer, a web designer and a class designer. It also includes add-ons for many software like GIT (one of the most popular version control software). Furthermore, it supports 36 different programming languages including, but not limited to, C++, C, C#, Visual Basic, Python, Java, JavaScript, TypeScript, HTML, XML, CSS, Ruby and much more.

There are several different IDEs and programming languages that could have been used to create a program identical to this one. Below are some examples of modern IT Tools that can be used as an alternative.

* **IDEs & Code Editors:-**
* **Microsoft Visual Studio Code: -**

Microsoft Visual Studio Code is a source-code editor that may even be more popular than visual studio in a professional setting. It includes all features from debugging, highlighting, intelligent code and completion. The reason visual studio was used over visual studio code is that VS Code requires the programmer to download all of its components such as different debuggers and compilers separately while visual studio already comes equip with them.

* **JetBrains IDEs: -**

JetBrains is a software development company that has dozens of different IDEs including, but not limited to, CLion, PyCharm, RubyMine, PhpStorm, ReSharper C++ and much more. All of these IDEs are also used abundantly in a professional setting however they all require a subscription and have less online resources available than visual studio hence making them unfit for this particular project.

* **XCode & Android Studio: -**

XCode & Android Studio are both very popular IDEs that are used to create android and iOS apps as well as apps for other apple products, however Android Studio is cannot be used to make PC applications and XCode was not used because this project was created on a Windows PC.

* **Dev-C++: -**

Dev-C++ is a fully features integrated development environment (IDE) made specifically for the purpose of making C and C++ applications and includes its own complier however it cannot be used for other languages and lacks online resources hence visual studio was used instead.

* **Programming Languages: -**
* **Java: -**

Java is a class based object oriented programming language that is designed to have as few implementation dependencies as possible. It is a general purpose programming language and is a compiled and interpreted language. Java code is first compiled into bytecode that can only be read by the java virtual machine which means that a code written in java can run on any computing device that has the java virtual machine while for many other languages codes have to be rewritten for different machines. Java is the most used programming language in the world however its popularity is rapidly declining in favor of more easy languages such as python The reason C++ was used (other than being compulsory) is because java is a class based programming language and this program is written in generic code and also java is not suitable for any embedded systems application so it would be less useful to a electrical engineer than C++.

* **Python: -**

Python is an interpreted and a very high level language. It is a general purpose programming language that focuses on readability. It is mostly used for object oriented programming and can be used to create large and small scale projects. Python the fastest growing programming language in the world due to its readability as well as many modern features and libraries such as pandas, numby and matplotlib which allows you to use many features resent in matlab inside python, however since it is such a high level language it takes a lot of time to execute making it unfit for high performance software as well as any hardware application so C++ is still more useful for electrical engineers.

* **Codes –** SLO 5.1

Following are three separate codes for the ‘Library Management System.’ Please note that three codes were created as prototypes and **Code-1** has been used and submitted in the actual project.

* **Code-1: -**

// LIBRARY MANAGEMENT SYSTEM

//R# 20L-1403

//R# 20L-1507

//SECTION : 1A1

//DEPARTMENT : EE

#include <iostream>

#include <fstream>

#include <iomanip>

using namespace std;

ifstream in;

ofstream out;

void main\_menu();

void available (int,int , char[][45] , char[][45], char [][45], char[][45] , char[][45] , int);

void addbooks (int ,char [][15],int&  ,  int, char[][45] , char[][45], char [][45], char[][45] , char[][45] );

void delete\_books (int,int,int, int , int& , char , char [][45] , char[][45] , char[][45], char [][45], char[][45] , char[][45]);

void studentid (char[][7] );

void returnfunc();

void search (int ,int , int ,int , char , char [][45] , char[][45] , char[][45], char [][45], char[][45] , char[][45]  );

void edit (int,int , char[][15],char [][15] ,char[][7],char ,int, int ,int ,char [][45] , char[][45] , char[][45], char [][45], char[][45] , char[][45]);

void change\_student\_id(char [][7]);

void issue\_record(int,char [][15] , int,int,int ,char[][45] , char[][45], char [][45], char[][45] , char[][45] );

int main()

{

    cout <<endl<<endl;

    cout << setw(120)<<"WELCOME TO 'THE LIBRARY MANAGEMENT SYSTEM'" <<endl<<endl;

    char avail[1][15]= {'A','v','a','i','l','a','b','l','e'};

    char unavail[1][15] = {'U','n','a','v','a','i','l','a','b','l','e'};

    char id[1][7];                                  // to input user id

    studentid( id );

    char ans2 = '11';

    do

    {

    main\_menu();                                    //Displying main menu

    cin >> ans2;

    int t=0;

    int j = 0;                                      // used as an index for controlling rows or column in many functions

    char booknum [7][45];                           // used as a temporary array in whole program

    int lines = 0;                                  // used to count lines in file so used in many funcion such as books that in which line user will enter next book

    int x =1 ;                                      // used as temporary variable in specially in (IF THEN ELSE)

    char ans = 'u';                                 // to take answer from user that what task he want to perform in switches in functions

    const int rows = 15;                            // number of rows

    char bookname [rows][45] ={};

    char booknumber[rows][45] = {};

    char bookdepartment [rows][45] ={};

    char issue [rows][45] = {};

    int index =0;                                   // widely used allmost in every function

    char availability [rows][45] = {};

    int r =0;                                       // used temporarily as rows

    int c=0;                                        // used temporarily as columns

    switch ( ans2 )                                 // initialy it has 11 value ( as loop is do while so it wil; run at least one time

    {

        case '1' :

                available ( c,index , booknumber , bookname , issue , availability , bookdepartment , lines);

                break;

        case '2' :

                break;

        case '3' :

                addbooks (c,avail,lines , index,booknumber , bookname , issue , availability , bookdepartment);

                break;

        case '4' :

                delete\_books (j,x,c, index ,lines, ans , booknum , booknumber , bookname , issue , availability , bookdepartment);

                break;

        case '5' :

                edit (j,c,unavail,avail ,id,ans ,t,lines , index , booknum , booknumber , bookname , issue , availability , bookdepartment);

                break;

        case '6' :

                search (j,x,lines,index,ans ,  booknum , booknumber , bookname , issue , availability , bookdepartment );

                break;

        case '7' :

                 issue\_record (j,unavail,x,index,lines,booknumber , bookname , issue , availability , bookdepartment );

                break;

        case '8' :

            change\_student\_id(id);

            break;

        default :

            return 0;                               // if user anter any wrong condition the progra will end

            break;

    }

}

    while ( ans2 != 'e' );                          // loop will stop executing when user presses 'e'

    return 0;

}

void addbooks ( int c,char avail[][15] ,int& lines , int index,char booknumber[][45], char bookname[][45] , char issue[][45], char availability[][45] , char bookdepartment[][45])

{

    in.open("Books.txt");

    lines = 0;

    while ( !in.eof() )

    {

        in >> bookname[index] >> booknumber[index] >> bookdepartment[index] >> availability[index] >> issue[index];

        index++;

        lines++;

    }

    index = 0;

    lines = lines-1;

    if ( lines < 11 )

    {

        cout <<endl;

    cout << "Enter the book name you want to add : " ;

    scanf(" %[^\n]" , bookname[lines]);

    cout <<endl;

    cout << " Enter booknumber : "  ;

    scanf(" %[^\n]" , booknumber[lines]);

    cout <<endl;

    cout << " Enter Bookdepartment : " ;

    scanf(" %[^\n]" , bookdepartment[lines]);

    cout <<endl<<endl;

        for ( index = 0 ; index < 10 ; index ++)

        {

            availability[lines][index] = avail[0][index];

        }

        for (index = 0 ; index < 6 ; index++)

        {

        issue[lines][index] = '-';

        }

    index=0;

    while ( !in.eof() )

    {

        in >> bookname[index] >> booknumber[index] >> bookdepartment[index] >> availability[index] >> issue[index];

        index++;

    }

    in.close();

    out.open("Books.txt");

    cout <<"                                    Book added successfully in the Library"<<endl<<endl<<endl;

    for ( index=0 ; index <=lines ; index++)

    {

        cout <<left<<setw(50)<< bookname[index] <<setw(20)<< booknumber[index] <<setw(35)<< bookdepartment[index] <<setw(20)<< availability[index] <<setw(5)<< issue[index]<<endl;

            for(c=0 ; c <40 ; c++)

            {

                if(bookname[index][c] == ' ')

                    bookname[index][c]='\_';

                if(bookdepartment[index][c] == ' ')

                    bookdepartment[index][c]='\_';

            }

        out <<left<<setw(50)<< bookname[index] <<setw(20)<< booknumber[index] <<setw(35)<< bookdepartment[index] <<setw(20)<< availability[index] <<setw(5)<< issue[index]<<endl;

    }

    cout <<endl<<endl;

    out.close();

    }

    else

    {

        cout <<endl<<endl;

        cout << " Max Limit of storing books have reached" <<endl;

        cout << " Delete any book to add new book " << endl<<endl;

    }

}

void available (int c ,int index, char booknumber[][45], char bookname[][45] , char issue[][45], char availability[][45] , char bookdepartment[][45] , int lines )

{

    ifstream in;

    in.open("Books.txt");

    cout << "                               Available Books in Library " << endl<<endl<<endl;

    while ( !in.eof ())

    {

        in >> bookname[index] >> booknumber[index] >> bookdepartment[index] >> availability[index] >> issue[index];

        index++;

        lines++;

    }

    for(index = 0 ; index <= lines ; index++)

    {

        for( c=0 ; c <40 ; c++)

            {

                if(bookname[index][c] == '\_')

                    bookname[index][c]=' ';

                if(bookdepartment[index][c] == '\_')

                    bookdepartment[index][c]=' ';

            }

    cout <<left<<setw(50)<< bookname[index] <<setw(20)<< booknumber[index] <<setw(35)<< bookdepartment[index] <<setw(20)<< availability[index] <<setw(5)<< issue[index]<<endl;

    }

    cout <<endl<<endl;

    in.close();

}

void studentid(char id[][7])

{

    cout << "Enter the Student's ID Number." << endl;

    cout << "#";

    scanf(" %[^\n]" , id[0]);

    cout << endl;

    cout << "Thank you; ID number is verified" <<endl<<endl;

}

void delete\_books (int j,int x,int c, int index ,int& lines , char ans , char booknum[][45] , char booknumber[][45], char bookname[][45] , char issue[][45], char availability[][45] , char bookdepartment[][45])

{

    int a;

    j=0;

    x=0;

    ifstream in;

    ofstream out;

    in.open("Books.txt");

    a=1;

        cout <<"                                    DELETE MODE ENABLED"<<endl<<endl;

        cout << " Enter book number " << endl;

        scanf(" %[^\n]" , booknum[0]);

    while ( !in.eof ())

    {

        in >> bookname[index] >> booknumber[index] >> bookdepartment[index] >> availability[index] >> issue[index];

        index++;

        lines++;

    }

    in.close();

    for ( index = 0 ; index <= lines-1  ; index++ )

    {

        x=0;

        for (  j = 0 ; j < strlen(booknumber[index]) ; j++)

        {

            if  (booknumber[index][j] == booknum[0][j]  )

            {

                x= x+1;

                if ( (strlen(booknum[0])+200) == x+200 )

                {

                    cout <<endl;

                    cout << "Found Successfully !" <<endl<<endl;

                    cout << " Name        :         "<< bookname[index] <<endl;

                    cout << " Book Number  :        "<< booknumber[index]<<endl;

                    cout << " Department   :        "<< bookdepartment[index]<<endl;

                    cout << " Status          :     "<< availability[index]<<endl;

                    cout << " Issue Details:        "<< issue[index]<<endl;

                    cout << endl<<endl;

                    cout << " Is this the book that you want to delete ? (Enter  'y' for yes & 'n' for no) " <<endl;

                    cin >> ans;

                    switch (ans)

                    {

                    case 'y':

                        a=2;

                        for ( int i = 0; i < 40 ; i++)                                  //200 is for columns; every column have value it can be empty, but we have to make sure even if the name of the book is large program will work fine

                        {

                        booknumber[index][i] = booknumber[lines-2][i];

                        bookname[index][i] = bookname[lines-2][i];

                        availability[index][i] = availability[lines-2][i];

                         bookdepartment[index][i] =  bookdepartment[lines-2][i];

                         issue[index][i] = issue[lines-2][i];

                        }

                        cout <<endl;

                         cout <<"Book Deleted successfully !" <<endl<<endl;

                         cout << "                          Updated Books in Library " <<endl<<endl<<endl;

                    continue;                               //to control loop so that i chech again for booknumber

                    break;                                      //switch break

                    case 'n':

                        cout << "                           Books in Library " <<endl<<endl<<endl;

                        break;

                   }

                }

            }

        }

    }

    out.open("Books.txt");

    for(int i = 0 ; i < (lines-a) ; i++)

    {

        out <<left<<setw(50)<< bookname[i] <<setw(20)<< booknumber[i] <<setw(35)<< bookdepartment[i] <<setw(20)<< availability[i] <<setw(5)<< issue[i]<<endl;

        for(c=0 ; c <40 ; c++)

            {

                if(bookname[i][c] == '\_')

                    bookname[i][c]=' ';

                if(bookdepartment[i][c] == '\_')

                    bookdepartment[i][c]=' ';

            }

    cout <<left<<setw(50)<< bookname[i] <<setw(20)<< booknumber[i] <<setw(35)<< bookdepartment[i] <<setw(20)<< availability[i] <<setw(5)<< issue[i]<<endl;

    }

    cout <<endl;

    out.close();

        cout <<"                                    DELETE MODE DISABLED"<<endl<<endl<<endl;

}

void search (int j ,int x, int lines , int index,char ans , char booknum[][45] , char booknumber[][45], char bookname[][45] , char issue[][45], char availability[][45] , char bookdepartment[][45] )

{

    ifstream in;

    in.open("books.txt");

    cout << "                                                           SEARCH  ENABLED " << endl<<endl;

        cout << " Enter book number " << endl;

        scanf(" %[^\n]" , booknum[0]);              //temporary Array to store book number entered by user

    lines=0;

    while ( !in.eof ())

    {

        in >> bookname[index] >> booknumber[index] >> bookdepartment[index] >> availability[index] >> issue[index];

        index++;

        lines++;

    }

    in.close();

    for ( index = 0 ; index <+ lines-1  ; index++ )

    {

        x=0;

        for (  j = 0 ; j < strlen(booknumber[index]) ; j++)

        {

            if  (booknumber[index][j] == booknum[0][j]  )

            {

                x= x+1;

                if ( (strlen(booknum[0])+200) == x+200 )

                {

                    cout << "Book found !" <<endl<<endl;

                    cout << " Name        :         "<< bookname[index] <<endl;

                    cout << " Book Number  :        "<< booknumber[index]<<endl;

                    cout << " Department   :        "<< bookdepartment[index]<<endl;

                    cout << " Status          :     "<< availability[index]<<endl;

                    cout << " Issue Details:        "<< issue[index]<<endl;

                    cout << endl<<endl;

                    ans = 'x';                                                          //used as check that it found book or not

                }

            }

        }

    }

    if ( ans != 'x')

    {

        cout << " Failed to found " <<endl;

        cout << " Make sure entered 'BOOK NUMBER' is correct " <<endl<<endl;

    }

    cout <<"                                                            SEARCH  DISABLED"<<endl<<endl<<endl;

}

void edit (int j ,int c,char unavail[][15] , char avail[][15] ,char id [][7] ,char ans ,int t,int lines ,int index, char temporary[][45] , char booknumber[][45], char bookname[][45] , char issue[][45], char availability[][45] , char bookdepartment[][45])

{

    t=0;

    ifstream in;

    ofstream out;

    in.open("Books.txt");

    cout << "                                       Edit Book enabled " << endl<<endl;

        cout << " Enter book number " << endl;

        scanf(" %[^\n]" , temporary[0]);

        cout <<endl<<endl;

    while ( !in.eof ())

    {

        in >> bookname[index] >> booknumber[index] >> bookdepartment[index] >> availability[index] >> issue[index];

        index++;

        lines++;

    }

    in.close();

    for ( index = 0 ; index <= lines-1  ; index++ )

    {

        t=0;

        for (  j = 0 ; j < strlen(booknumber[index]) ; j++)

        {

            if  (booknumber[index][j] == temporary[0][j]    )

            {

                t= t+1;

                if ( (strlen(temporary[0])+200) == t+200 )

                {

                    cout << "Found Successfully !" <<endl<<endl;

                    cout << " Name        :         "<< bookname[index] <<endl;

                    cout << " Book Number  :        "<< booknumber[index]<<endl;

                    cout << " Department   :        "<< bookdepartment[index]<<endl;

                    cout << " Status          :     "<< availability[index]<<endl;

                    cout << " Issue Details:        "<< issue[index]<<endl;

                    cout << endl<<endl;

                    ans = 'u';

                    cout << "Enter     'e'    to edit Book Details"<<endl;

                    cout << "Enter     'i'    to issue/unissue book"<<endl;

                    cin >> ans;

                    switch (ans)

                    {

                    case 'e':

                    cout << "Now you can edit data for this Book " <<endl;

                    cout << " Name        :         ";

                    scanf(" %[^\n]" , temporary[1]);

                    cout << " Book Number  :        ";

                    scanf(" %[^\n]" , temporary[2]);

                    cout << " Department   :        ";

                    scanf(" %[^\n]" , temporary[3]);

                    for ( int i = 0; i < 40 ; i++)                                  //200 is for columns; every column have value it can be empty, but we have to make sure even if the name of the book is large program will work fine

                        {

                        booknumber[index][i] =temporary[2][i];

                        bookname[index][i] = temporary[1][i];

                        bookdepartment[index][i] = temporary[3][i];

                        }

                    cout << "                                                           Updated Books in library after editing " <<endl<<endl<<endl;

                    break;

                    case 'i':

                    cout << endl<<endl;

                        ans ='a';

                        cout << "Do you want to issue or unissue the book? ( Press 'i' for ISSUE  &  'u' for UNISSUE )" <<endl;

                        cin>>ans;

                        switch (ans)

                        {

                        case 'i':

                            for ( int i = 0; i < 40 ; i++)                                  //200 is for columns; every column have value it can be empty, but we have to make sure even if the name of the book is large program will work fine

                        {

                        availability[index][i]=unavail[0][i];

                        issue[index][i] =id[0][i];

                        }

                            cout <<"                                                            Status of Book is updated to Unavailable"<<endl<<endl;

                            break;

                        case 'u':

                            for ( int i = 0; i < 40 ; i++)                                  //200 is for columns; every column have value it can be empty, but we have to make sure even if the name of the book is large program will work fine

                        {

                        availability[index][i]=avail[0][i];

                        }

                            for (int i = 0 ; i < 6 ; i++)

                            {

                                issue[index][i] = '-';

                            }

                            cout << "                                                           Status of Book is updated to Available"<<endl<<endl;

                            break;

                        }

                    continue;

                    }

                }

            }

        }

    }

    t=0;

    out.open("Books.txt");

    for(int i = 0 ; i < (lines-1) ; i++)

    {

        for(c=0 ; c <40 ; c++)

            {

                if(bookname[i][c] == '\_')

                    bookname[i][c]=' ';

                if(bookdepartment[i][c] == '\_')

                    bookdepartment[i][c]=' ';

            }

        cout <<left<<setw(50)<< bookname[i] <<setw(20)<< booknumber[i] <<setw(35)<< bookdepartment[i] <<setw(20)<< availability[i] <<setw(5)<< issue[i]<<endl;

        for(c=0 ; c <40 ; c++)

            {

                if(bookname[i][c] == ' ')

                    bookname[i][c]='\_';

                if(bookdepartment[i][c] == ' ')

                    bookdepartment[i][c]='\_';

            }

    out <<left<<setw(50)<< bookname[i] <<setw(20)<< booknumber[i] <<setw(35)<< bookdepartment[i] <<setw(20)<< availability[i] <<setw(5)<< issue[i]<<endl;

    }

    cout <<endl<<endl;                                          //one book deleted means one line deleted and we hav changed this in whole program as lines are passed by reference and take changes in whole program

    out.close();

        /\*lines=0;

        available(index , temporary , bookname , issue , availability ,bookdepartment ,lines );\*/

}

void main\_menu()

{

    cout << "   Press 1 : To see all available/unavailable books.   " << endl;

    cout << "   Press 2 : To return to Main Menu.   " << endl;

    cout << "   Press 3 : To add new books. " << endl;

    cout << "   Press 4 : To delete books from availabe books.  " << endl;

    cout << "   Press 5 : To edit the details of a book.    " << endl;

    cout << "   Press 6 : To search a book from Library Management System.  " << endl;

    cout << "   Press 7 : To check the book issue record.   " << endl;

    cout << "   Press 8 : To change student id. "<<endl;

    cout << "   Press E : To exit from LIBRARY MANAGMENT SYSTEM " <<endl<<endl;

    cout << "   Select any option " <<endl;

}

void change\_student\_id(char id[][7])

{

    studentid(id);

}

void issue\_record(int j ,char unavail [][15],int x, int index,int lines, char booknumber[][45], char bookname[][45] , char issue[][45], char availability[][45] , char bookdepartment[][45]  )

{

    ifstream in;

    in.open("books.txt");

    cout << "                                   ISSUED  BOOKS " << endl<<endl;

    lines=0;

    while ( !in.eof ())

    {

        in >> bookname[index] >> booknumber[index] >> bookdepartment[index] >> availability[index] >> issue[index];

        lines++;

    }

    in.close();

    int a;

    a=1;

    for ( index = 0 ; index <= lines-1  ; index++ )

    {

        x=0;

        for (  j = 0 ; j < strlen( unavail[0]) ; j++)

        {

            if  ( availability[index][j] == unavail[0][j]   )

            {

                x= x+1;

                if ( (strlen(unavail[0])+200) == x+200 )

                {

                    if (a==1)

                    {

                    cout <<left<<setw(50)<<" Book Name :"<<setw(20)<< "Book Number :"<<setw(35)<<" Department :"<<setw(20)<<"Status:"<<setw(40)<< "Issue Details: "<<endl<<endl;

                    }

                    a=0;

                    cout <<left<<setw(50)<< bookname[index] <<setw(20)<< booknumber[index] <<setw(35)<< bookdepartment[index] <<setw(20)<< availability[index] <<setw(40)<< issue[index]<<endl;

                }

            }

        }

    }

    cout << endl<<endl;

}

void returnfunc()

{

    main\_menu();

}

* **Code-2: -**
* #include <iostream>
* #include <fstream>
* #include <iomanip>
* using namespace std;
* ifstream fin;
* ofstream fout;
* void studentid(int&);
* void mainmenu();
* void returnfunc();
* void availablebooks(int i, char bn[15][45], int num[], char bd[15][45], char a[15][45], int it[]);
* void addbooks(int i, char bn[15][45], int num[], char bd[15][45], char a[15][45], int it[]);
* void searchbooks(int i, char bn[15][45], int num[], char bd[15][45], char a[15][45], int it[]);
* void deletebooks(int i, char bn[15][45], int num[], char bd[15][45], char a[15][45], int it[]);
* int main()
* {
* int id = 0;
* int option;
* const int loopconstant = 0;
* int index = 0;
* char bookname[15][45] = {};
* int booknumber[500] = {};
* char bookdepartment[15][45] = {};
* char availability[15][45] = {};
* int issueto[500] = {};
* fin.close();
* fout.close();
* cout << endl << setw(60) << "WELCOME TO 'THE LIBRARY MANAGEMENT SYSTEM'" << endl << endl;
* studentid(id);
* mainmenu();
* cout << "Enter the option number you want to choose:-" << endl;
* cin >> option;
* while (loopconstant == 0)
* {
* switch (option)
* {
* case 1:
* {
* availablebooks(index, bookname, booknumber, bookdepartment, availability, issueto);
* break;
* }
* case 2:
* {
* returnfunc();
* break;
* }
* case 3:
* {
* addbooks(index, bookname, booknumber, bookdepartment, availability, issueto);
* availablebooks(index, bookname, booknumber, bookdepartment, availability, issueto);
* break;
* }
* case 4:
* {
* deletebooks(index, bookname, booknumber, bookdepartment, availability, issueto);
* availablebooks(index, bookname, booknumber, bookdepartment, availability, issueto);
* break;
* }
* case 5:
* {
* deletebooks(index, bookname, booknumber, bookdepartment, availability, issueto);
* addbooks(index, bookname, booknumber, bookdepartment, availability, issueto);
* availablebooks(index, bookname, booknumber, bookdepartment, availability, issueto);
* break;
* }
* case 6:
* {
* searchbooks(index, bookname, booknumber, bookdepartment, availability, issueto);
* break;
* }
* case 7:
* {
* availablebooks(index, bookname, booknumber, bookdepartment, availability, issueto);
* break;
* }
* default:
* {
* return 0;
* }
* }
* if (option != 2)
* {
* returnfunc();
* }
* cout << "Select another option:-" << endl;
* cin >> option;
* }
* return 0;
* }
* void studentid(int& num1)
* {
* cout << "Enter the Student's ID Number." << endl;
* cout << "#";
* cin >> num1;
* cout << endl;
* }
* void mainmenu()
* {
* cout << "Press 1:- To see all available & unavailable books." << endl;
* cout << "Press 2:- To return to the main menu." << endl;
* cout << "Press 3:- To add new books." << endl;
* cout << "Press 4:- To delete books from available books." << endl;
* cout << "Press 5:- To edit the details of a book." << endl;
* cout << "Press 6:- To search a book from library management system." << endl;
* cout << "Press 7:- To check the issue record." << endl;
* cout << "Press E:- To Exit." << endl;
* cout << endl;
* }
* void returnfunc()
* {
* mainmenu();
* }
* void availablebooks(int i, char bn[15][45], int num[], char bd[15][45], char a[15][45], int it[])
* {
* fin.open("testbooks.txt");
* int space = 0;
* while (!fin.eof())
* {
* fin >> bn[i] >> num[i] >> bd[i] >> a[i] >> it[i];
* if (space == 0)
* {
* cout << endl;
* cout << left << setw(20) << " Book Name:" << setw(10) << "Number:" << left << setw(23) << " Department:" << left << setw(16) << "Availability:" << setw(10) << "Issued To:" << endl;
* cout << endl;
* }
* for (int uts = 0; uts < sizeof(bn[i]); uts++)
* {
* if (bn[i][uts] == '\_' || bn[i][uts] == '~')
* {
* bn[i][uts] = ' ';
* }
* if (bd[i][uts] == '\_' || bd[i][uts] == '~')
* {
* bd[i][uts] = ' ';
* }
* }
* cout << left << setw(20) << bn[i] << setw(10) << num[i] << left << setw(23) << bd[i] << left << setw(16) << a[i] << setw(10) << it[i] << endl;
* i++;
* space++;
* }
* fin.close();
* cout << endl << endl;
* }
* void addbooks(int i, char bn[15][45], int num[], char bd[15][45], char a[15][45], int it[])
* {
* fin.open("testbooks.txt");
* int ic = 0;
* while (!fin.eof())
* {
* fin >> bn[i] >> num[i] >> bd[i] >> a[i] >> it[i];
* i++;
* ic++;
* }
* fout.open("testbooks.txt");
* for (int oi = 0; oi < ic; oi++)
* {
* fout << bn[oi] << setw(10) << num[oi] << setw(19) << bd[oi] << setw(20) << a[oi] << setw(15) << it[oi] << endl;
* }
* char ibname[50];
* int ibnum;
* char ibd[50];
* int x;
* cout << endl;
* cout << "PUT A FULLSTOP (.) AT THE END OF THE DEPARTMENT NAME BEFORE PRESSING ENTER!" << endl;
* cout << "Book Name: ";
* for (int ri = 0; ri < sizeof(ibname); ri++)
* {
* cin.get(ibname[ri]);
* if (ibname[ri] == '\n')
* {
* ibname[ri] = '~';
* }
* if (ibname[ri] == ' ')
* {
* ibname[ri] = '\_';
* }
* if (ibname[ri] == '.')
* {
* break;
* }
* }
* cout << endl;
* cout << "Book Number: ";
* cin >> ibnum;
* cout << endl;
* cout << "PUT A FULLSTOP (.) AT THE END OF THE DEPARTMENT NAME BEFORE PRESSING ENTER!" << endl;
* cout << "Department: ";
* for (int di = 0; di < sizeof(ibd); di++)
* {
* cin.get(ibd[di]);
* if (ibd[di] == '\n')
* {
* ibd[di] = '~';
* }
* if (ibd[di] == ' ')
* {
* ibd[di] = '\_';
* }
* if (ibd[di] == '.')
* {
* break;
* }
* if (ibd[di] == '.')
* {
* break;
* }
* }
* for (int nmi = 0; nmi < sizeof(ibname); nmi++)
* {
* if (ibname[nmi] == '.')
* {
* break;
* }
* fout << ibname[nmi];
* }
* fout << setw(10) << ibnum << setw(10);
* for (int odi = 0; odi < sizeof(ibd); odi++)
* {
* if (ibd[odi] == '.')
* {
* break;
* }
* fout << ibd[odi];
* }
* fout << setw(20) << "Available" << setw(15) << "000000";
* fin.close();
* fout.close();
* cout << "Book Added!" << endl;
* cout << endl;
* }
* void searchbooks(int i, char bn[15][45], int num[], char bd[15][45], char a[15][45], int it[])
* {
* fin.open("testbooks.txt");
* while (!fin.eof())
* {
* fin >> bn[i] >> num[i] >> bd[i] >> a[i] >> it[i];
* i++;
* }
* int ibnum;
* int x;
* cout << endl;
* cout << "Enter the number of the book that you want to search for." << endl;
* cout << "Book Number: ";
* cin >> ibnum;
* cout << endl;
* int comp = 0;
* int st;
* for (st = 0; st < sizeof(num); st++)
* {
* if (ibnum == num[st])
* {
* comp = 1;
* break;
* }
* }
* cout << endl;
* if (comp == 1)
* {
* cout << left << setw(20) << bn[st] << setw(10) << num[st] << left << setw(23) << bd[st] << left << setw(16) << a[st] << setw(10) << it[st] << endl;
* }
* else
* {
* cout << "That Book is NOT Available in the Library." << endl;
* }
* fin.close();
* cout << endl << endl;
* }
* void deletebooks(int i, char bn[15][45], int num[], char bd[15][45], char a[15][45], int it[])
* {
* fin.open("testbooks.txt");
* int ic = 0;
* while (!fin.eof())
* {
* fin >> bn[i] >> num[i] >> bd[i] >> a[i] >> it[i];
* i++;
* ic++;
* }
* fout.open("testbooks.txt");
* char ibname[50];
* int ibnum;
* char ibd[50];
* int x;
* int delnum;
* char delchoice = 'n';
* cout << "Enter the number of the book that you want to delete." << endl;
* cin >> delnum;
* for (int dc = 0; dc < sizeof(num); dc++)
* {
* if (num[dc] == delnum)
* {
* cout << "Is this the book you want to delete? (Enter 'y' for 'yes' & 'n' for 'no'.)" << endl << endl;
* cout << left << setw(20) << bn[dc] << setw(10) << num[dc] << left << setw(23) << bd[dc] << left << setw(16) << a[dc] << setw(10) << it[dc] << endl;
* cout << endl;
* cin >> delchoice;
* cout << endl;
* if (delchoice == 'y')
* {
* for (int oi = 0; oi < ic; oi++)
* {
* if (oi == dc)
* {
* oi++;
* }
* fout << bn[oi] << setw(10) << num[oi] << setw(19) << bd[oi] << setw(20) << a[oi] << setw(15) << it[oi] << endl;
* }
* fin.close();
* fout.close();
* cout << "Book Deleted!" << endl;
* }
* else
* {
* for (int oi = 0; oi < ic; oi++)
* {
* fout << bn[oi] << setw(10) << num[oi] << setw(19) << bd[oi] << setw(20) << a[oi] << setw(15) << it[oi] << endl;
* }
* fin.close();
* fout.close();
* }
* }
* }
* cout << endl;
* }
* **Code-3: -**
* #include <iostream>
* #include <fstream>
* using namespace std;
* ifstream fin;
* ofstream fout;
* void studentid(int&);
* void mainmenu();
* void returnfunc();
* int index = 0;
* void availablebooks(int&, char[]);
* void addbooks(int&, char[]);
* int main()
* {
* int id = 0;
* int option;
* const int loopconstant = 0;
* char bookarray[500];
* studentid(id);
* mainmenu();
* cout << "Enter the option number you want to choose:-" << endl;
* cin >> option;
* while (loopconstant == 0)
* {
* switch (option)
* {
* case 1:
* {
* availablebooks(index, bookarray);
* break;
* }
* case 2:
* {
* returnfunc();
* break;
* }
* case 3:
* {
* addbooks(index, bookarray);
* break;
* }
* case 4:
* {
* break;
* }
* case 5:
* {
* break;
* }
* case 6:
* {
* break;
* }
* case 7:
* {
* break;
* }
* default:
* {
* return 0;
* }
* }
* if (option != 2)
* {
* returnfunc();
* }
* cout << "Select another option:-" << endl;
* cin >> option;
* }
* return 0;
* }
* void studentid(int& num1)
* {
* cout << "Enter the Student's ID Number." << endl;
* cout << "#";
* cin >> num1;
* cout << endl;
* }
* void mainmenu()
* {
* cout << "Press 1:- To see all available & unavailable books." << endl;
* cout << "Press 2:- To return to the main menu." << endl;
* cout << "Press 3:- To add new books." << endl;
* cout << "Press 4:- To delete books from available books." << endl;
* cout << "Press 5:- To edit the details of a book." << endl;
* cout << "Press 6:- To search a book from library management system." << endl;
* cout << "Press 7:- To check the issue record." << endl;
* cout << endl;
* }
* void returnfunc()
* {
* mainmenu();
* }
* void availablebooks(int& i, char arr[])
* {
* fin.open("books.txt");
* while (!fin.eof())
* {
* fin.get(arr[i]);
* cout << arr[i];
* i++;
* }
* fin.close();
* cout << endl << endl;
* }
* void addbooks(int& i, char arr[])
* {
* int temp = i;
* i = 0;
* int in = 0;
* char nbn[50];
* char nbnum[3];
* char nbd[10];
* fout.open("books.txt");
* while (i < temp)
* {
* fout << arr[i];
* i++;
* }
* fout << endl;
* cout << "Book Name: ";
* for (int in = 0; in < 50; in++)
* {
* cin >> nbn[in];
* if (nbn[in] == '\n')
* {
* break;
* }
* }
* temp = 0;
* while (temp < in)
* {
* fout << nbn[temp];
* temp++;
* }
* fout << "     " << "101" << "     " << "Electrical" << "     " << "Available" << "       " << "------" << "     " << "-------";
* fout.close();
* cout << endl << endl;
* }

Above are three different methods of creating the same ‘Library Management System’. Code-1 uses 2-D Arrays and an excess of functions build into C++, also all variable are passed by reference in this code. Code-2 uses 2-D arrays as well however it contains much fewer build-in functions and there is an excess of local variables declared inside functions. Code-3 uses 1-D Arrays and a few build-in functions. After thoroughly testing all three of the programs we collectively came to the conclusion that Code-1 works the best as it was able to handle a lot of exceptional and unpredictable cases. All the Codes work to a certain extent however there are a few cases in which codes 2 and 3 fail (explored in the ‘comparison’ section) hence Code-1 was chosen.

The Design of the used program (Code-1) will be discussed in the ‘Explanation & Design’ section, where all the functions will be thoroughly explained.

All three codes will be compared function by function in the ‘Comparison’ section to justify the decision of using Code-1 as well as exploring alternate methods for creating this project.

* **Flowchart of Functions –** SLO 11.1, 11.2, 11.5 & 9.1

Please note that since all three codes were created in parallel, all their functions were created simultaneously as well so they share a similar flowchart.

Basic Layout of **main()**,creation of the switch statement from which all the functions will be called and the **studentid()** function as well as declaration of the first few variables and arrays for switch.

**[December 18, 2020]**

Creation of the first few functions including the **mainmenu()** function, **returnfunc()** function and **available()** function as well as creation of the **“Books.txt”** file from which all data is read and to which all data is stored. Declaration of more variables and arrays requires for the functions.

**[December 19, 2020]**

Creation of the **addbooks()** function and declaration of variables and arrays required for it.

**[December 20 - 23, 2020]**

Passing all local variables and arrays by reference, refining the code, deleting unused variables and beautifying the code.

**[December 30, 2020]**

Creation of the **issue\_record()** and **change\_student\_id()** functions and final few variables and arrays to be used.

**[December 29, 2020]**

Creation of **search()**, **delete\_books()** and **edit()** functions as well as declaring more variables and arrays to be used.

**[December 24 – 28, 2020]**

**Arguments & Return Types of Functions: -**

This section goes through the return types of and arguments used in functions as well as their prototypes; it does not go into details about the working of the functions for those please see the **‘Explanation & Design’** section.

**Main Menu: -**

void main\_menu();

The function has no input arguments does not return a value, it displays the main menu by outputting to console.

**Available Books: -**

void available (int, int, char[][45], char[][45], char[][45], char[][45], char[][45], int);

1. int c;
2. int index;
3. int lines;
4. char booknumber[][45];
5. char bookname[][45];
6. char issue[][45];
7. char availability[][45];
8. char bookdepartment[][45];

The function has 8 input arguments; 3 integers (for controlling rows, columns and indexes of arrays) and 5 character arrays, each corresponding to a book detail i.e. Book name, Book number, Department, Availability & Issued ID Number. This function does not return a value instead it performs the desired function and outputs directly to the file or console.

**Add Books: -**

void addbooks(int, char[][15], int&, int, char[][45], char[][45], char[][45], char[][45], char[][45]);

1. int c;
2. int index;
3. int& lines;
4. char booknumber[][45];
5. char bookname[][45];
6. char issue[][45];
7. char availability[][45];
8. char bookdepartment[][45];
9. char avail[][15];

The function has 9 arguments; 3 integers (for controlling rows, columns and indexes of arrays) and 6 character arrays, each corresponding to a book detail i.e. Book name, Book number, Department, Availability & Issued ID Number and a temporary availability array. This function does not return a value instead it performs the desired function and outputs directly to the file or console.

**Delete Books: -**

void delete\_books(int, int, int, int, int&, char, char[][45], char[][45], char[][45], char[][45], char[][45], char[][45]);

1. int c;
2. int index;
3. int& lines;
4. int j;
5. int x;
6. char ans;
7. char booknumber[][45];
8. char bookname[][45];
9. char issue[][45];
10. char availability[][45];
11. char bookdepartment[][45];
12. char booknum[][45];

This function has 12 arguments; 5 integers (for controlling rows, columns and indexes of arrays as well as storing temporary values and loop controlling integers), 1 character (To store the options entered by the user) and 6 character arrays, each corresponding to a book detail i.e. Book name, Book number, Department, Availability & Issued ID Number and a temporary number array. This function does not return a value instead it performs the desired function and outputs directly to the file or console.

**Student ID:-**

void studentid(char[][7]);

1. char id[][7];

This function has 1 argument; a character array that takes the student id from console. It does not return a value.

**Return to Main Menu: -**

void returnfunc();

This function does not have any arguments nor does it return a value, it is used to return back to the main menu.

**Search: -**

void search (int, int, int, int, char, char[][45], char[][45], char[][45], char[][45], char[][45], char[][45]);

1. int index;
2. int lines;
3. int j;
4. int x;
5. char ans;
6. char booknumber[][45];
7. char bookname[][45];
8. char issue[][45];
9. char availability[][45];
10. char bookdepartment[][45];
11. char booknum[][45];

This function has 11 arguments; 4 integers (for controlling rows, columns and indexes of arrays as well as storing loop controlling integers), 1 character (To store the options entered by the user) and 6 character arrays, each corresponding to a book detail i.e. Book name, Book number, Department, Availability & Issued ID Number and a temporary number array. This function does not return a value instead it performs the desired function and outputs directly to the file or console.

**Edit Books: -**

void edit (int, int, char[][15], char[][15], char[][7], char, int, int, int, char[][45], char[][45], char[][45], char[][45], char[][45], char[][45]);

1. int c;
2. int index;
3. int lines;
4. int j;
5. int t;
6. char ans;
7. char unavail[][15];
8. char avail[][15];
9. char id[][7];
10. char booknumber[][45];
11. char bookname[][45];
12. char issue[][45];
13. char availability[][45];
14. char bookdepartment[][45];
15. char temporary[][45];

This function has 15 arguments; 5 integers (for controlling rows, columns and indexes of arrays as well as storing temporary values and loop controlling integers), 1 character (To store the options entered by the user) and 9 character arrays, each corresponding to a book detail i.e. Book name, Book number, Department, Availability & Issued ID Number and a temporary number array and temporary arrays for availability and unavailability. This function does not return a value instead it performs the desired function and outputs directly to the file or console.

**Changing Student ID: -**

void change\_student\_id(char[][7]);

1. char id[][7];

This function has 1 argument; a character array that takes the student id from console to replace the previous one. It does not return a value.

**Issue Record: -**

void issue\_record(int, char[][15], int, int, int, char[][45], char[][45], char[][45], char[][45], char[][45]);

1. int index;
2. int lines;
3. int j;
4. int x;
5. char booknumber[][45];
6. char bookname[][45];
7. char issue[][45];
8. char availability[][45];
9. char bookdepartment[][45];
10. char unavail[][15];

This function has 10 arguments; 4 integers (for controlling rows, columns and indexes of arrays as well as storing loop controlling integers) and 6 character arrays, each corresponding to a book detail i.e. Book name, Book number, Department, Availability & Issued ID Number and a temporary availability array. This function does not return a value instead it performs the desired function and outputs directly to the file or console.

**Function Implementation & Schedule: -**

Since all three codes were made in parallel, so they share the same timeline.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Day**  **Week** | **Monday** | **Tuesday** | **Wednesday** | **Thursday** | **Friday** | **Saturday** | **Sunday** |
| **Week 1** | Layout of **main()** and creation of **studentid()** functions + Problem analysis. | Creation of **mainmenu(), returnfunc()** and **available()** functions as well as the “**Books.txt**” file. | **\_\_\_** | Started to explore the **addbooks()** function, without using arrays. | Created the first few character arrays and stored them in the file. | Took inputs in character arrays from user and stored them in file. Finished creating **addbooks()** function. | Started to explore the **search(),** **deleteooks()** and **edit()** functions. Finished creating the **search()** function. |
| **Week 2** | Attempted to create **deletebooks()** function, deleted non-array characters from file; took and stored relevant inputs from user in array | Completed the **deletebooks()** function, to delete multiple strings from 2D character arrays. | Attempted to create **edit()** function, by trying to merge **deletebooks()** and **addbooks()** functions | Finalized the **edit()** function by reforming it. Implemented conditions into **edit()** function for editing details or issuing. | Created the final two functions of **issue\_record()** and **change\_student\_id()**. | Finalized the project, passed all local variables by reference and beautified the code. Started to create project report. | Wrote the majority of project report and completed the project report. |

**Group Member Participation: -**

Even though both participants worked on all the functions on all three projects, a majority of the **delete\_books()**, **edit()** & **issue\_record()** functions were created by Wasal Rasool (20L-1403) and a majority of the **available()**, **addbooks()** & **search()** functions were made by Daniyal Ahmed (20L-1507). All function were made on google meet due to the covid pandemic and input was taken from both the members, all opinions were considered.

* **Explanation & Design –** SLO 5.3

In this section we will go over the complete working and design of the used code (Code-1) function by function. Please note that codes will not be compared here for that see the ‘Comparison’ section. In this section only the used code (Code-1) will be analyzed.

The code is a ‘Library management system’ and has 10 different features. It can display a main menu, return to that main menu, take student’s id, change that id, show all available books, search for a particular book, add a new book, delete a existing book, edit the details of a book, issue or un issue a book and display a list of the issued books.

It uses 3 libraries shown below: -

* **include <iostream>**
* **include <fstream>**
* **include <iomanip>**

The **<iostream>** library includes the basic input and output features build into C++ such as cin and cout as well as ‘>>’ and ‘<<’ operators and also functions like cin.get(), cin.getline(), cin.ignore(), scanf() and many more.

The **<fstream>** library includes the functions required to read out of files as well as write into files. It includes functions like ifstream and ofstream which are used to declare reserved words which are stream elements and are used in other functions to manipulate files.

The **<iomanip>** library includes the functions for manipulating text that will be displayed to either the console or written into files. It includes the setw() function that reserves a certain amount of space for that string to be written in, it also includes the setfill() function that is used to fill the reserved spaces by any character or symbol, and the setprecision() function that sets how many numbers can appear in an integer both before and after the decimal. There are also the left and right functions that shift the text to either the left or right of the spaces reserved by the setw() function.

Following are each of the functions of the program explained in detail: -

* **Main Function: -**

The main function includes all the declared variables as well as arrays that store the book information and temporary variables and arrays that are displayed directly (The prototypes of and variables in all functions are mentioned in the ‘Flowchart of Functions’ section.).

The main function also includes the structure of how the code is executed, it includes a switch statement nested inside a do-while loop that will continue to execute until ‘e’ is pressed. If the user enters ‘E’ character, the switch statement enters the default case where a value of 0 is returned to the function and the code stops executing from within the switch statement.

The switch statement has 9 cases (including the default case) corresponding to each key strike from keys 1 – 8. Each key pressed enters a new case where the required function is called. If multiple functions are needed, then multiple functions are called.

On pressing 1, the switch enters case 1 where the available books function is called and then the loop breaks. For 2, the main menu function is called. For 3, the add books function is called before breaking. For 4, the delete function is called, for 5, the edit function is called, for 6, the search function is called, for 7, issue record function is called and for 8 the change student id function is called. Finally, for ‘E’, the default function is called and the code ends.

After performing each case the do-while loop asks the user to enter a new number for a function to be called in switch, this keeps happening until the user exits the code from within switch by pressing ‘E’ and entering the default case.

* **Add Books Function: -**

The add books function starts by the ifstream variable opening “Books.txt” file and then storing each element in its corresponding array separated by spaces to differentiate between the elements. While storing the books into their corresponding arrays, two variables are calculated, one to determine the rows of the 2D array and the other to determine the row in which the new book is to be added.

If the maximum capacity has not been reached, the function then uses the scanf() function to take and store the input entered by the user into its corresponding array and row. The input is read from the user until the end line character (‘\n’) is encountered upon which the input for the next detail is taken until all inputs are stored in their correct positions.

Next two ‘for loops’ are used to make the newly added book automatically available and the assign ‘-’ to the ‘issue to’ field.

Now the file is re-opened by the ofstream variable and a ‘for loop’ outputs the contents of the file (including the newly added book) to the console, but a nested ‘for loop’ first checks if any empty spaces are present in the element, if so then they are replaced by ‘\_’ character before being stored into the file, this is done to prevent any error from the elements being read from the file.

Then the contents of the array are re-written into the file (this includes the newly added book) before closing the ofstream.

Finally, if the maximum capacity of books has been reached then the function does not take any input from the user and instead informs the user that the maximum capacity has been reached before exiting the add books function.

* **Available Books function: -**

This function performs similarly to the previous function. First the ifstream variable opens the file and writes the information into its corresponding array, at the same time the number of columns of 2D array and the location of the last line is calculated.

Then a ‘for loop’ outputs each element to the console, a nested ‘for loop’ checks to for any ‘\_’ and replaces them with empty spaces, in the correct order as were in the file.

* **Student ID function: -**

This function simply prompts the user to enter their ID, then stores it in a 2D array to be used later. No loops or conditional statements required.

* **Delete Books function: -**

This function first prompts the user to enter the book number of the book they want to delete and stores it in a 2D array.

It then copies each element of the file to its corresponding array while storing the number of columns.

Then the function enters a multitude of nested ‘for loops’ and conditional statements. First it tests whether the book with the book number entered by the user exists in the book number array. The outer loop searched the line that the number is stored in and the inner loop matches the number character by character to the numbers stored in the book number array.

If the book is found, then the function displays the details of the book and prompts the user to enter if they are sure that they want to delete that book, if the user confirms the delete then the function re-writes each element of the array to its current location but skips the row in which the book to be deleted is stored.

It then re-opens the file and writes the elements of the new array to the file, doing this the book to be deleted is not included in the file. It then replaces all ‘\_’ character with empty spaces before outputting the file (with the book deleted) to the console.

* **Search Books function: -**

This functions works partially similar to the delete books function. It prompts the user the user to enter the book number of the book they want to search for, then stores that book number.

It then writes everything from the file to its corresponding array and stores the number of columns.

Then nested ‘for loops’ search the column of the required book and each character of the book number array is matched to the number entered by the user, if the book is found its details are displayed to the user.

If the book is not found, then the function informs the user.

* **Edit Books function: -**

This function also has a similar layout to the delete function. It first prompts the user to enter the book number of the book they want to edit the details of, and then writes everything from the file to the arrays.

It then re-enters a multitude of nested ‘for loops’ and conditional statements to check the columns and each character in array for the book with the same number as that which was entered.

If the book is found, it then asks the user if they want to edit the details of the book or issue/return the book.

If the user chooses to edit the details of the book, it then prompts the user to enter a new book name, book number, and department then stores them in a temporary array. It then writes the elements of the temporary array to the corresponding book details arrays.

If the user chooses to issue/return the book, the program then prompts the user to enter if they want to issue or return it, depending upon what the user chooses, the availability status is updated and the student’s ID number is either removed from or written to the issue array.

The program then replaces any ‘\_’ character with empty spaces and outputs the entire file (with the book edited) to the console. It then replaces the empty spaces with ‘\_’ character and stores the new information to the file.

* **Main Menu function: -**

This function simply outputs, to the console, all the key strikes that the user can make to perform a certain function. No arrays, files, loops or conditional statements needed.

* **Change Student ID function: -**

This function simply calls the Student ID function.

* **Return Menu function: -**

This function simply calls the Main Menu function.

* **Issue Records function: -**

This function works very similarly to the available books function. It first reads everything from the file and stores it into its corresponding array, it then checks which columns have unavailable in their availability array and outputs only those functions to the console by using an if condition and comparing the length of the string currently present in the availability array.

It does NOT re-write the new information to the file.

* **Comparison –** SLO 5.2

In this section we will compare the three different solutions (Codes 1, 2, 3) from which this program could have been made function by function. Please note that details of each function will not be given here, for detailed explanation of each function see the ‘Explanation & Design’ Section.

|  |  |  |  |
| --- | --- | --- | --- |
| **Codes**  **Functions** | **Code - 1** | **Code - 2** | **Code - 3** |
| **Main Function** | This code works by trapping the switch statement in a while loop until ‘e’ is pressed so the code will end if either ‘e’ or ‘E’ is pressed. | This code works by trapping the switch statement in an infinite loop until the default case of the switch is reached, then a value of 0 is returned and the program ends. | This code works by trapping the switch statement in an infinite loop until the default case of the switch is reached, then a value of 0 is returned and the program ends. |
| **Main Menu** | Directly outputs all possible key strikes to the console for the user to choose. | Directly outputs all possible key strikes to the console for the user to choose. | Directly outputs all possible key strikes to the console for the user to choose. |
| **Available Books** | Reads from file and writes to array, replaces underscores with empty spaces and outputs to console. | Reads from file and writes to array, replaces underscores and tilde with empty spaces and outputs to console. | Reads the entire 1D array and outputs to the console directly. |
| **Add Books**  **Add Books (Continued.)** | The contents of the file are written to the main array. User is prompted to enter the details of the book, scanf() function is used instead of cin.get(), so the input is taken until the user enters the ‘\n’ function. The new data is then written directly to the main arrays and displayed on the console. Data from main arrays is written to the file with new book included. | The contents of the file are written to the main array. User is prompted to enter the details of the book, the first end line character is replaced by ‘~’ so cin.get() can be used, spaces are replaced with ‘\_’, then code breaks when ‘\n’ is reached, data is stored in temporary arrays. Data from temporary array is written to end of file, and then content of the new file is read into the main arrays. | User is prompted to input the details of the book and the details are added to end of the 1D array. |
| **Delete Books**  **Delete Books (Continued.)** | The contents of the file are read into the main array. User is prompted to enter the number of the book to be deleted. The entered number is compared to each number is book number array, if book exists then it is displayed to the used and they are asked if they want to delete the book, if they choose to then the details of the book to be deleted are overwritten by the details of the last book in the 2D array, and the last book is deleted. The new content is then displayed to the console and written to the file. | The contents of the file are written to main array. User is prompted to enter the number of the book they want to delete. The entered number is compared to each number is book number array, if book exists then it is displayed to the used and they are asked if they want to delete the book, if they choose to then the contents of the main array are written to the file and the column in which the book to be deleted is stored is not printed in the file by skipping its index number, then the available books function is called which writes the new file to the main array. | The Delete Books function could not be made. |
| **Student ID** | User is prompted to enter their ID, which is then stored in a 2D array. | User is prompted to enter their ID, which is then stored in a int variable. | User is prompted to enter their ID, which is then stored in a int variable. |
| **Return Main Menu** | Main Menu function is called. | Main Menu function is called. | Main Menu function is called. |
| **Search Books** | The user is prompted to enter the book number of the book they want to search for. The entered number is compared to all book numbers in book number array, if book number is found then the book details are displayed from all arrays to the console. | The user is prompted to enter the book number of the book they want to search for. The entered number is compared to all book numbers in book number array, if book number is found then the book details are displayed from all arrays to the console. | The Search Books function could not be made. |
| **Edit Books**  **Edit Books (Continued.)** | The user is prompted to enter the book number of the book they want to edit, then they are asked if they want to edit the details of the book or issue/return that book, if they choose issue/return then the availability status is changed and the student ID is added/removed to its corresponding array. If edit details function is chosen then the user is prompted to enter a new book name, number and department which then overwrites the previous ones. | Simply first the delete books function is called and then the add books function is called immediately after. | The Edit Books function could not be made. |
| **Change Student ID** | Student ID function is called. | Student ID function is called. | Student ID function is called. |
| **Issue Records** | The availability array for all books is checked, every book that is unavailable is output to the console. | The availability array for all books is checked, every book that is unavailable is output to the console. | The Issue Records function could not be made. |

* **Inputs & Solutions –** SLO 5.4 & 11.4

In this section we will verify our design for different inputs and suggest solutions that lead to an improved quality of the program.

* This code has been tested for many inputs that might have been thought to create an error; each error was analyzed and fixed (more detail in ‘Issues’ section)
* If an incorrect input is entered that might potentially cause the code to give an error, it will be detected and the user will be notified.
* There are several ways this program could have been improved. Using the full capacity of C++ including the use of the string library would greatly simplify this entire project and remove a lot of errors.
* Using pointers would also remove a lot of impractical loops and increase the readability of the code and make it much easier to edit and improve the code by another programmer.
* Using Object Oriented Programming would also make this code a lot easier to manipulate. It would have taken a lot less time and effort to build with classes and objects. It would also make it much easier to add a feature to it.
* Furthermore, creating a graphical user interface would make this code easier to be used by the user, instead of simply displaying on the console, it would be much easier for the user to interact with a GUI rather than a CLI.
* **Issues & Limitations –** SLO 5.5

In this section we will discuss the encountered issues, how they were solved and the limitations that still exist in the program.

Several issues were encountered during the creation of this project.

* To read and write data from and into a file using multiple 2D arrays of different data types, it was resolves by a lot of trial and error, learning how data was being read and stored using different inputs.
* To understand how the extraction operator interacted with empty spaces and how the data that included empty spaces was read and stored in files and arrays, it was resolved by using multiple if conditions to replace empty spaces with underscores before storing in file and removing the underscore before displaying to console.
* Terminating an input when the ‘\n’ character was encountered, it was resolved using a scanf() function that can identify the ‘\n’ which was entered by the user.
* To understand how 2D arrays were passed by reference, it was resolved by consulting instructors and online resources.
* Comparing 2D arrays with one another, it was resolved by using loops and comparing each element of a row for each column.

Even though most issues with the code were resolved, there still remain a few limitations within the code.

* The book number of two different books should never be the same as book numbers are readily used in other functions.
* If the first digit of the book number is entered for a function, then that function will e performed on all books which have that as the first digit of their book number.
* Removing any underscores from the file can cause the code to potentially not run properly, throw an exception or even crash.
* **Conclusion**

To conclude, this is a library management system that was built on everything that we have been taught. From the very basics of declaring an integer to everything from using conditional statements to loops to more complex data structures such as 1D and 2D arrays, using several libraries and functions build into C++ as well as creating many of our own. This project provides a satisfactory summary and assessment of what we have learnt throughout the entire course.