# Objectives

The project is mainly based on following objectives:

* To create a project using C-programming and its features.
* To implement features like control statement, structures and file handlings.
* To be familiar with resource reusability by making user defined function.
* To make the program easy while running it.
* To concise the memory of program as far as possible.
* To get an idea about making a simple project using C.
* To be able to solve problems by Compiling and Debugging.

# Introduction

The mini project ‘Library Management’ is the sample project for managing library. The project aims at developing library management system using the C language that enables an organization to maintain its library.

The project demonstrates the creation of a user interface of a system, without the use of C Graphics library.

The application uses basic C function to generate menus, show message boxes and print text on the screen.

To display customized text with colors and fonts according to application requirements, functions have been created in the application, which fetch the exact video memory addresses of a target location, to write text at a particular location.

The application also implements the concept of structures to define the library items. It also effectively applies the various C concepts such as file operations, looping and branching constructs and string manipulation functions.

# Introduction to c

**C** was evolved from **ALGOL**, **BCPL** and **B** by Dennis Ritchie at the bell laboratories in 1972. **C** uses many concepts from these languages and added the concept of data types and other powerful features.

**C** is a powerful, flexible, portable and elegantly structured programming language. The increasing popularity of c is probably due to its many desirable quantities. It is a robust language whose rich set of built in functions and operators can be used to write any complex of a high level language and therefore it is well suited for writing both system software and business packages.

**C** is highly portable. This means that c programs written for one computer can be run on another with little or no modification. Portability is important if we plan to use a new computer with a different operating system.

**C** language is well suited for structured programming, thus requiring the user to think of a problem in terms of function modules or blocks. A proper collection of these modules would make a complete program. This modular structure makes program debugging, testing and maintenance easier.

Another important feature of c is its ability to extend itself. A c program is basically a collection of functions that are supported but the c library. We can continuously ad our own functions to **C** library. With the availability of a large number of functions the programming task becomes simple.

To assure that the **C** language remains standard, in 1983, American national standards institute (**ANSI**) appointed a technical committee to define a standard for **C**. the committee approved but the international standards organization (**IDO**) in 1990 which is now known as **ANSI C** .it was then approved but the international standards organization (**ISO**) in 1990. This version of c is also referred to as **C89**.

## Control statement

C language possesses such decision making capabilities by supporting the follow statements:

* If statement
* Switch statement
* Conditional operator statement

These statements are popularly known as decision-making statements. Since these statements ‘control’ the flow execution, they are known as control statements.

1. **control statement**

The if-statement is a powerful decision making statement and is used to control the floexecution of statements. It is basically a two-way decision statement and is used in conjunction with an expression. It takes the following form:

If(test expression)

It allows the computer to evaluate the expression first and then, depending on whether the value expression (relation or condition) is ‘true’ (or non-zero) of ‘false’ (zero),it transfers the control to a particular statement. This point of program has two paths to follow, one for the true condition and the other for the false condition as shown below:

Entry

Test

Expression

?

false

true

The if statement may be implemented in different forms depending on the complexity of conditions to be tested.the different forms are:

* Simple if statement
* If……else statement
* Nested if………else statement
* Else if ladder.

**2**. Switch statement:-

C has a built-in multiway decision statement known as a switch. The switch statement tests the value of a given variable (or expression) against a list of case values and when a match is found, a block of statements associated with that case is executed. The general form of the switch statement is as shown below:

switch (expression)

{

case value-1:

block-1

break;

case value-2:

block-2

break;

………..

……….

default:

default-block

break;

}

statement-x;

The expression is an integer expression or characters.value-1,value-2…..are constants or constant expressions (evaluable to an integral constant) and are known as case labels. Each of these values should be unique within a switch statement. block-1,block-2…..are

Statement lists and may contain zero or more statements. There is no need to put braces around these blocks. Note that case labels end with a colon(:).

**3.** Conditional operator statement**:-**

The c language has an unusual operator, useful for making two-way decisions. This operator is a combination of? And takes three operands. This operator is popularly known as the conditional operators. The general form of use of the conditional operator is as follows:

Conditional expression? expression1:expression2

The conditional expression is evaluated first. if the result is nonzero,expression1 is evaluated and is returned as the value of the conditional expression.Otherwise,expression2 is evaluated and its value is returned.

## Function

Most languages allow you to create functions of some sort. Functions are used to break up large programs into named sections. You have already been using a function which is the main function. Functions are often used when the same piece of code has to run multiple times.

In this case you can put this piece of code in a function and give that function a name. When the piece of code is required you just have to call the function by its name. (So you only have to type the piece of code once).

In the example below we declare a function with the name MyPrint. The only thing that this function does is to print the sentence: Printing from a function. If we want to use the function we just have to call MyPrint() and the printf statement will be executed. (Don’t forget to put the round brackets behind the function name when you call it or declare it).

Take a look at the example:

#include<stdio.h>

void MyPrint()

{

printf("Printing from a function.\n");

}

int main()

{

MyPrint();

return 0;

}

**Parameters and return**

Functions can accept parameters and can return a result. (C functions can accept an unlimited number ofparameters).

here the functions are declared in your program does not matter, as long as a functions name is known to thecompiler before it is called. In other words: when there are two functions, i.e. functions A and B, and B mst call A, than A has to be declared in front of B.

Let’s take a look at an example where a result is returned:

#include<stdio.h>

int Add(int output1,int output2 )

{

printf("%d", output1);

printf("%d", output2);

return output1 + output2;

}

int main()

{

int answer, input1, input2;

scanf("%d", &input1);

scanf("%d", &input2);

anwser = Add(input1,input2);

printf(" answer = %d\n", answer);

return 0;

}

The main() function starts with the declaration of three integers. Then the user can input two whole numbers. These numbers are used as input of function Add(). Input1 is stored in output1 and output2 is stored in Second. The function Add() prints the two numbers onto the screen and will return the result of output1 + output2. The return value is stored in the integer answer. The number stored in answer is then printed onto the screen.

**Void**

If you don’t want to return a result from a function, you can use void return type instead of the int.  
So let’s take a look at an example of a function that will not return an integer:

void our\_site()

{

printf("www");

printf(".NextDawn");

printf(".nl");

}

**Note:** As you can see there is not an int before our\_site() and there is not a return 0; in the function.

The function can be called by the following statement: our\_site();

**Global and local variables**

A local variable is a variable that is declared inside a function. A global variable is a variable that is declared outside **all** functions. A local variable can only be used in the function where it is declared. A global variable can be used in all functions.

See the following example:

#include<stdio.h>

// Global variables

int A;

int B;

int Add()

{

return A + B;

}

int main()

{

int answer; // Local variable

A = 5;

B = 7;

answer = Add();

printf("%d\n",answer);

return 0;

}

As you can see two global variables are declared, A and B. These variables can be used in main() and Add().  
The local variable answer can only be used in main().

That’s all for this tutorial

More about function

In this C programming language tutorial we will talk some more about functions. We will take a look at command-line parameters and function prototypes.

**Command-line parameters**

In some cases you want to give a parameter at the start of a program.  
For example:

# myprogram -i

The program myprogram will start and something extra will be done because of the command-line parameter -i (What it will do is up to you, this is just an example).

Now let’s make a program that will print the words that are typed behind the program at execution time. (Compile the program with the name myprogram).  
Here is the example:

#include<stdio.h>

int main(int argc, char \*argv[])

{ int x;

printf("%d\n",argc);

for (x=0; x < argc; x++)

printf("%s\n",argv[x]);

return 0;

}

After compiling the program “myprogram” start it as follows:

# myprogram aa bb cc dd

In this code, the main program accepts two parameters, argc and argv. The argv parameter is an array of pointers to a string that contains the parameters entered when the program was invoked at the UNIX command line. (Pointers will be explained in a later tutorial, for now it is enough to know that it points to an address in memory where the parameters are stored). The argc integer contains a count of the number of parameters. (In this case four).

First the program will print the number of parameters given when the program was invoked (stored in argc). This number will be used in a “for loop”. In this case it will print four times.

The second printf statement will print the parameters given when the program was invoked, one by one. **Try it!** Command-line parameters can be used for many things.

**Function prototypes**

A function prototype declares the function name, its parameters, and its return type to the rest of the program. This is done before a function is declared. (In most cases at the beginning of a program).  
To understand why function prototypes are useful, try the following program:

#include<stdio.h>

void main()

{

printf("%d\n",Add(3)); /\* <- There is the error ! \*/

}

int Add(int a, int b)

{

return a + b;

}

**Note:** just one parameter when the function is called. (Not a typo).

The example above will be compiled on many of the compilers. They just give some warning, because the function Add() needs two parameters as input, not one. The result of this program cannot be foreseen. It works because many C compilers do not check for parameters matching either in type or count. The result is that you will spend a lot of time debugging programs, because you made a mistake of passing on to many or too few parameters to a function.

To solve this problem you can make use of function prototypes. If you use prototypes, C checks the types and count of the parameter list.

Try the next example:

#include<stdio.h>

int Add (int,int); /\* function prototype for Add \*/

void main()

{

printf("%d\n",add(3)); /\* <- There is the error ! \*/

}

int Add(int a, int b)

{

return a + b;

}

When you try to compile this program, the compiler will flag an error on the printf statement.

So in the future use function prototypes for every function you declare. It will save you a lot of time!

## Arrays and pointers

### Array

An array can be defined as a group of homogeneous elements sharing a common name.

In many of the programming situations, we may require the processing of data items that have common characteristics.now in such case, it would be easier if we place these data items into one single variable called array which is capable of storting number of data, sharing common name.

The individual data items can be characters, integers, and floating-point numbers and so on they must all, however be of the same type and the same storage class. The individual data items in an array are called array elements.

Each array element is referred to by specifying the array name followed by one or more subscript enclosed in square brackets. Each subscript or index must be expressed as non-negative integer.

Thus, we represent array containing elements as:

x[n]

Where, x is array name

N is subscript &

x[n] has its array element as x[0],x[1],x[2],………x[n-1].

The value of each subscript can be expressed as an integer constant,integer variable or a more complex integer expression. The number of subscript determines the dimensionality

of the array.

### Pointers

A pointer is a derived data type in C. It is built from one of the fundamental data types available in C. Pointers contain memory addresses as their values. Since these memory addresses are the locations in the computer memory where program instructions and data are stored, pointers can be used to access and manipulate data stored in the memory.

Pointers are undoubtedly one of the most distinct and exciting features of c language. It

has added power and flexibility to the language. Although they appear little confusing and difficult to understand for a beginner, they are a powerful tool and handy to use once they are mastered.

**Pointer Declaration:-**

Pointer variable like other variables must be declared before they may be used in a C Program. The interpretation of pointer declaration differs, however from the interpretation of other variable declarations. When a pointer variable is declared, the variable name must be preceded by an asterisk (\*). This identifies the fact that the variable is a pointer.

A pointer declaration may be written in general terms as

*Data-type \*ptvar;*

Where ptvar is the name of the pointer, and data-type refers to the data type of the pointer’s object. Remember that an asterisk must precede ptvar.

**Passing pointer to a function:-**

Pointer are often passed to a function as arguments. This allows data items within the calling portion of the program to be accessed by the program in altered form. We refer to this use of pointers as passing arguments by reference, in contrast to passing arguments by value.

**Arrays of pointers:-**

One important use of pointers is in handling of a table of strings. Consider the following array of strings:

char name [3][25];

This says that the name is a table containing three names, each with a maximum length of 25 characters (including null character). The total storage requirements for the name table are 75 bytes.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

We know that rarely the individual strings will be of equal lengths. Therefore, instead of making each row a fixed number of characters, we can make it a pointer to a string of varying length. For example,

char \*name[3]={“New Zealand”,”Australia”,”Nepal”};

declares name to be an array of three pointers to characters, each pointer pointing to a particular name as:

name[0] New Zealand

name[1] Australia

name[2] Nepal

This declaration allocates only 28 bytes, sufficient to hold all the characters as shown

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **N** | **e** | **w** |  | **Z** | **e** | **a** | **l** | **a** | **n** | **d** | **\0** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **u** | **s** | **t** | **r** | **a** | **l** | **i** | **a** | **\0** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **N** | **e** | **p** | **a** | **l** | **\0** |

The following statement would print out all the three names:

for ( i=0; i<=2; i++)

printf(“%s\n”, name[i] );

to access he jth  character in the ith name, we may write as

\*(name[i]+j);

The character arrays with the rows of varying length are called ‘ragged arrays’ and are better handled by pointers.

# **Structure**

C is provided with a constructed data type known as structures, a mechanism for packing data of different types. A structure is a convenient tool for handling a group of logically related data items. For example, it can be used to represent a set of attributes, such as student \_ name, roll\_ number and marks. The concept of a structure is analogous to that of a ‘record’ in many other languages.

Structures help to organize complex data in a more meaningful way. It is a powerful concept that we may often need to use in our program design. This chapter is devoted to the study of structures and their applications in program development.

In general, the syntax for structure definition is:

struct struct\_name

{

data \_ type mem1;

data \_ type mem2;

…………………………

………………………...

data \_type memn;

};

The structure definition starts with keyword struct followed by an identifier or tag name. The tag name is structure name and can e used for instantiating structure variable. In above definition, struct \_name is referred to as structure name or structure tag name; and mem1, mem2, memn are known as structure members or elements or fields. These members are enclosed within braces and terminated by semicolon.

After the structure has been specified, the structure variable can be declared as standard data type:

struct struct\_name var1, var2,………,varn;

The structure definition serves as a template for user defined data type. It does not reserve memory unless a variable of structure data type is declared.

**Accessing Member of structures**

In C programming language, the members of a structure are processed individually as separate entities. We make use of period or dot “.” Operator to access the individual members of structure.The syntax for accessing member of a structure variable is follows:

struct \_variable. member

Where struct \_ variable refers to the name of a structure variable, and member refers to the name of member within the structure. The dot (.) is an operator that separates the variable name from the member name. We noticed that the dit operator must have precedence among all operators and has left to right associatively. Consider the following statement:

struct employee e1;

Now, each member of the structure variable e1 can be accessed using the dot operator as follows:

e1.emp\_id

The employee’s employee IDnumer is accessed;

e1.name

The employee’s name is accessed. e1.salaryThe employee’s salary is accessed.

# File Handling

The console function like printf() and scanf() have been used for input/output .This scheme is adequate if the volume of data involved in not so large or it is not necessary to store the information for further use. However, many applications may require a large amount of data to be read, processed, and also saved for later use. Such information is stored on the auxiliary memory device in the form of data file.And a file is a collection of bytes that is given a name. In most computer systems, files are used as a unit of storage primarily on floppy-disk or fixed-disk data storage system (or they can be CDs or other storage device). Thus data files allow us to store information permanently, and to access and alter that information whenever necessary.

The file handling function available in standard library in order to implement I/O midel is classified as follows:

a) File access

b) Operation input/output

c) Formatted input/output

d) Character input/output

e) Direct input/output

f) File positioning

g) Error handling

The file access included the function like fopen() to open a file, fclose() to close a file , fflush () to flush out the buffer associated with a file, and freopen() to change the file associated with a stream. Also setvbuf() and setbuf() functions are use to allow the users explicitly control the file buffering strategy.

The operation on file includes like remove() to remove a file, remname() to rename a file ,tempfile() to create a temporary binary file and tmpnam() to generate a unique filename.

Formatted input/output group includes the n functions fscanf(), scanf() and sscanf() to read formatted data. Similarly fprintf() ,printf(), sprint(), vfprintf(), vprintf() and vsprintf() to write formatted data.

The character input/output group includes the functions fgetc() ,getc() and getchar() to read a character from an input stream and functions ungetc() to push back a character to an input stream. The functions fgets() and gets() are to read strings and the output functions fputc(),putc(), putchar(), fputs() and puts() are also included in this group.

The direct input/output group includes functions fread() to read and fwrite() to write a certain number of data items specified size.

File positioning group includes functions fread() to read and fwrite() to write a certain number if data items specified size.

File positioning group includes functions to set the file positon to some specified value to allow access to a specific portion of the seek(),interrogate the current file position ftell(),and reset the file position to the beginning of the file rewind().

Error handling group include functions to test whether EOF returned by a function indicates an end-of-file or an error (feof and ferror), clear end-of-file and indicators clearer, and map the error number errno to an error message perror.

**File Accessing:-**

To work with file using any file handling library functions, C requires a pointer to structure FILE defined in <stdio.h>.The declaration of file pointer is done as follows:

FILE \*fptr;

This declaration creates a variable fptr, which is pointed to the FILE structure. The pointer to the FILE structure is also called file pointer. The FILE structure describes the current state of a file such as file status flag, file descriptor, file buffer etc.

There are also predefined file pointers such stdin, stdout, and stderr which refers to standard input (normally keyboard), standard output (normally monitor) and standard error (connected to screen for error handling).

The file access functions provide the facilities to open and close a file, flush out the file buffer, change the file content and control the file buffering strategy.

Before working with a file it should be opened first. To open a file and associate it with a stream, we use fopen().Its prototype is shown here:

FILE \*fopen(char \*fname,char \*mode);

The fopen() function, like all the file-system functions ,uses the header <stdio.h> .The name of the file including the path is specified by fname and the mode specifies how file is accessed. Both of these parameters are string. The string specifying the mode is shown in following table.

|  |  |  |
| --- | --- | --- |
| **S.N** | **Mode** | **Meaning** |
| 1 | "r" | Open a text file for reading |
| 2 | "w" | Create a text file for writing |
| 3 | "a" | Append to a text file |
| 4 | "rb" | Open a binary file for reading |
| 5 | "wb" | Open a binary file for writing |
| 6 | "ab" | Append to binary file |
| 7 | "r+" | Open a text file for read/write |
| 8 | "w+" | Create a text file for read/write |
| 9 | "a+" | Append or creat a text file for read/write |
| 10 | "r+b" | Open a binary file for read/write |
| 11 | "w+b" | Create a binary file for read/write |
| 12 | "a+b" | Append a binary file for read/write |

# Algorithm

**Main function**

**steps**

1. Start with welcome screen
2. Get password from user
3. Check password. Is password correct

Yes:-goto step 4

No:-print wrong message and goto step 2

1. Display mainmenu as below

* 1.add books
* 2.delete books
* 3.search books
* 4.issue books
* 5.view book list
* 6.edit book’s record
* 7.close application

1. Get choice from user

Choice:-1 call function addbook

Choice:-2 call function deletebook

Choice:-3 call function searchbook

Choice:-4 call function Issuebooks

Choice:-5 call function viewbook

Choice:-6 call function editbooks

Choice:-7 goto step 6

6.stop

**Addbook function**

**Steps**

1.declare file pointer ‘fp’

2.display categories of book to be added

3.get option from user.Is user want to add books

Yes:-goto step 3

No:-go back to main menu

4. Open file ‘fp’ to write

5. Assign the pointer to the end of the file to write

6. get data from user

7. write input data on a file

8. close file

9.print option to add another books

Yes:-goto step 1

No:-go back to main menu

**Deletebook function**

**Steps**

1.get book ID from user to delete

2. Declare file pointer ‘ft’

3.open file ‘fp’ to read

4. Assign the pointer to the beginning of the file to be read

5.loop until ‘End of file’ is not encountered read data from file.

6.Is user input book id= book Id on a file

Yes:- a. open ‘ft’ file and copy all data of ‘fp’ file in ‘ft’file except that data which we want to delete

b. delete ‘fp’ file and rename ft file by ‘fp’ file name and goto step 7

No:-print error message and close file and go back to main menu

7.close file

8.print option to delete another book

Yes:-goto step1

No:-go back to main menu

**Searchbook function**

**Steps**

1. Display option for search

* 1. Search by id
* 2.Search by book name

1. Open the fie ‘fp’ to read
2. Assign the pointer to the beginning of the file to be read
3. If option is search by id

* Get book id from user
* Loop until ‘End of file’ is not encountered read data from file
* Is user input book id =book id on a file

Yes:- Display all information about that book id and goto step 6

No:- Print sorry message and goto step 6

1. If option is search by book name

* Get book name from user
* Loop until ‘End of file’ is not encountered read data from file
* Is user input book name=book name on a file

Yes:-display all information about that book and goto step 6

No:-print sorry message and goto step 7

1. Close file
2. Display option to search another book

Yes:-goto step 1

No:- go back to main menu

**Issuebook function**

**Steps**

1. Display option as below

* 1.Issue book
* 2.View issued book
* 3.Search issued book
* 4.Remove issued book

1. Open ‘fs’ file
2. Assign the pointer to the beginning of the file to be read
3. Get choice from user

Choice -1

1. Get book id from user
2. Loop until ‘End of file’ is not encountered read data from file
3. Is input book id=book id on a file

Yes:-get student name whom book has to issue

No:-print sorry message and goto step e

1. Write information of student and issued data on a file
2. Display option to issue another book

Yes:-goto step a

No:-goto step 5

Choice-2

1. Loop until ‘End of file’ is not encountered read data from file
2. Display list of issued book with student name, issued date and return date
3. Goto step 5

Choice-3

1. Get book id from user
2. Loop until ‘End of file’ is not encountered read data from file

c. Is user input book id =book id on a file

Yes:- Display student name whom book was issued

No:- Print sorry message and goto step d

1. Display option to search another issued book

Yes:- goto step a

No:-goto step 5

Choice-4

a.declare the file pointer ‘fg’

b. get book Id from user to delete from issued list

c. Loop until ‘End of file’ is not encountered read data from file

d. Is user input book id= book Id on a file

Yes:- 1. open another ‘fg’ file and copy all data of ‘fs’ file in ‘fg’ file except that data which we want to delete

2. delete ‘fs’ file and rename ‘fg’ file by ‘fs’ file name and goto step e

No:-print error message and goto step e

e.print option to search another book

Yes:-goto step b

No:-goto step 6

5. Close file

6. Go back to main menu

**Viewbook function**

**Steps**

1. Open ‘fs’ file
2. Assign the pointer to the beginning of the file to be read
3. Loop until ‘End of file’ is not encountered read data from file
4. Display list of all book with complete information
5. Show total number of books in library
6. Close file
7. Go back to main menu

**Editbook function**

**Steps**

1. Get book id to be edited from user
2. Open file ‘fs’
3. Assign the pointer to the beginning of the file to be read
4. Loop until ‘End of file ‘ is not encountered read data from file
5. Call function checkid. Is checkid=1

Yes:-goto step 6

No:-display sorry message and goto step 7

1. a.get new data from user of that book which to be edited

b.Assign the pointer to the current position

c.Overwrite the new data on old data of that book and goto step 7

No:- Display sorry message and goto step 7

1. Close file
2. Display option to edit another book

Yes:-goto step 1

No:-goto step 8

1. Go back to main menu

**Checkid function**

**Steps**

1. Assign the pointer to the beginning of the file to be read
2. Loop until ‘End of file ‘is not encountered read data from file
3. If user input id=id on a file

Yes:-return value 1

No:-return value 0

This function check whether user input id already exit in a file or not.

**Getdata function**

**Steps**

1. Print”Enter information below”
2. Get categories from user
3. Get book id
4. Call function checkid. Is checkid=0

Yes:-goto step6

No:-goto step 5

1. Get book name,book author name,quantity,price and rack no. where that book located from user
2. Go back to main menu

This function get data from the user for all information of book.

**Password function**

**Steps**

1. Declare two string ‘pass’ and ‘ password=pokhara’
2. Print “Enter the password”.
3. Until user entered the enter key get character for user
4. Store input character in string ‘pass’
5. Compare two string ‘pass’ and ‘password’ .Is both equal

Yes:-goto step 6

No:-print error message and goto step 2

6. Print password match message

7. Goto main menu

# Source code

//list of header files

#include<stdio.h> //contains printf,scanf etc

#include<conio.h> //contains delay(),getch(),gotoxy(),etc.

#include <stdlib.h> // contains exit();

#include<string.h> //contains strcmp(),strcpy(),strlen(),etc

#include<ctype.h> //contains toupper(), tolower(),etc

#include<dos.h> //contains \_dos\_getdate

#include<time.h>

#include<bios.h> //contains show\_mouse();

#define RETURNTIME 15

//list of function prototype

void returnfunc(void);

void mainmenu(void);

void addbooks(void);

void deletebooks(void);

void editbooks(void);

void searchbooks(void);

void issuebooks(void);

void viewbooks(void);

void closeapplication(void);

int getdata();

int checkid(int);

int t(void);

void show\_mouse(void);

void Password();

void issuerecord();

//list of global files that can be acceed form anywhere in program

FILE \*fp,\*ft,\*fs;

//list of global variable

Char catagories[][15]={"Computer","Electronics","Electrical","Civil","Mechnnical","Architecture"};

int s;

char findbook;

char password[10]={"pokhara"};

struct meroDate

{

int mm,dd,yy;

};

struct books

{

int id;

char stname[20];

char name[20];

char Author[20];

int quantity;

float Price;

int count;

int rackno;

char \*cat;

struct meroDate issued;

struct meroDate duedate;

};

struct books a;

int main()

{

Password();

getch();

return 0;

}

void mainmenu()

{

clrscr();

// textbackground(13);

int i;

gotoxy(20,3);

printf("\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2 MAIN MENU \xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2");

show\_mouse();

gotoxy(20,5);

printf("\xDB\xDB\xDB\xDB\xB2 1. Add Books ");

gotoxy(20,7);

printf("\xDB\xDB\xDB\xDB\xB2 2. Delete books");

gotoxy(20,9);

printf("\xDB\xDB\xDB\xDB\xB2 3. Search Books");

gotoxy(20,11);

printf("\xDB\xDB\xDB\xDB\xB2 4. Issue Books");

gotoxy(20,13);

printf("\xDB\xDB\xDB\xDB\xB2 5. View Book list");

gotoxy(20,15);

printf("\xDB\xDB\xDB\xDB\xB2 6. Edit Book's Record");

gotoxy(20,17);

printf("\xDB\xDB\xDB\xDB\xB2 7. Close Application");

gotoxy(20,19);

printf("\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2");

gotoxy(20,20);

t();

gotoxy(20,21);

printf("Enter your choice:");

switch(getch())

{

case '1':

addbooks();

break;

case '2':

deletebooks();

break;

case '3':

searchbooks();

break;

case '4':

issuebooks();

break;

case '5':

viewbooks();

break;

case '6':

editbooks();

break;

case '7':

{

clrscr();

gotoxy(16,3);

printf("Programmers....");

gotoxy(16,4);

printf("1. Bibek Subedi (066/BCT/506)");

gotoxy(16,5);

printf(" Mobile:9846311430 E-mail:subedi\_bibek@yahoo.co.in");

gotoxy(16,7);

printf("2. Dinesh Subedi (066/BCT/512)");

gotoxy(16,8);

printf(" Mobile:9841569394 E-mail:dinesh\_subedi92@yahoo.com");

gotoxy(16,10);

printf("3. Sijan Bhandari (066/BCT/537)");

gotoxy(16,11);

printf(" Mobile:9849516774 E-mail:sijan\_nasa@yahoo.com");

gotoxy(16,13);

printf("With the Unexplainable Help of Mr.Ashok Basnet");

gotoxy(10,17);

printf("Exiting in 3 second...........>");

flushall();

delay(3000);

exit(0);

}

default:

{

gotoxy(10,23);

printf("\aWrong Entry!!Please re-entered correct option");

if(getch())

mainmenu();

}

}

}

void addbooks(void) //funtion that add books

{

clrscr();

int i;

gotoxy(20,5);

printf("\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2SELECT CATEGOIES\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2");

gotoxy(20,7);

printf("\xDB\xDB\xDB\xDB\xB2 1. Computer");

gotoxy(20,9);

printf("\xDB\xDB\xDB\xDB\xB2 2. Electronics");

gotoxy(20,11);

printf("\xDB\xDB\xDB\xDB\xB2 3. Electrical");

gotoxy(20,13);

printf("\xDB\xDB\xDB\xDB\xB2 4. Civil");

gotoxy(20,15);

printf("\xDB\xDB\xDB\xDB\xB2 5. Mechanical");

gotoxy(20,17);

printf("\xDB\xDB\xDB\xDB\xB2 6. Architecture");

gotoxy(20,19);

printf("\xDB\xDB\xDB\xDB\xB2 7. Back to main menu");

gotoxy(20,21);

printf("\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2");

gotoxy(20,22);

printf("Enter your choice:");

scanf("%d",&s);

if(s==7)

mainmenu() ;

clrscr();

fp=fopen("Bibek.dat","ab+");

if(getdata()==1)

{

a.cat=catagories[s-1];

fseek(fp,0,SEEK\_END);

fwrite(&a,sizeof(a),1,fp);

fclose(fp);

gotoxy(21,14);

printf("The record is sucessfully saved");

gotoxy(21,15);

printf("Save any more?(Y / N):");

if(getch()=='n')

mainmenu();

else

clrscr();

addbooks();

}

}

void deletebooks() //function that delete items from file fp

{

clrscr();

int d;

char another='y';

while(another=='y') //infinite loop

{

clrscr();

gotoxy(10,5);

printf("Enter the Book ID to delete:");

scanf("%d",&d);

fp=fopen("Bibek.dat","rb+");

rewind(fp);

while(fread(&a,sizeof(a),1,fp)==1)

{

if(a.id==d)

{

gotoxy(10,7);

printf("The book record is available");

gotoxy(10,8);

printf("Book name is %s",a.name);

gotoxy(10,9);

printf("Rack No. is %d",a.rackno);

findbook='t';

}

}

if(findbook!='t')

{

gotoxy(10,10);

printf("No record is found modify the search");

if(getch())

mainmenu();

}

if(findbook=='t' )

{

gotoxy(10,9);

printf("Do you want to delete it?(Y/N):");

if(getch()=='y')

{

ft=fopen("test.dat","wb+"); //temporary file for delete

rewind(fp);

while(fread(&a,sizeof(a),1,fp)==1)

{

if(a.id!=d)

{

fseek(ft,0,SEEK\_CUR);

fwrite(&a,sizeof(a),1,ft); //write all in tempory file except that

} //we want to delete

}

fclose(ft);

fclose(fp);

remove("Bibek.dat");

rename("test.dat","Bibek.dat"); //copy all item from temporary file to fp except that

fp=fopen("Bibek.dat","rb+"); //we want to delete

if(findbook=='t')

{

gotoxy(10,10);

printf("The record is sucessfully deleted");

gotoxy(10,11);

printf("Delete another record?(Y/N)");

}

}

else

mainmenu();

fflush(stdin);

another=getch();

}

}

gotoxy(10,15);

mainmenu();

}

void searchbooks()

{

clrscr();

int d;

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Search Books\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

gotoxy(20,10);

printf("\xDB\xDB\xDB\xB2 1. Search By ID");

gotoxy(20,14);

printf("\xDB\xDB\xDB\xB2 2. Search By Name");

gotoxy( 15,20);

printf("Enter Your Choice");

fp=fopen("Bibek.dat","rb+"); //open file for reading propose

rewind(fp); //move pointer at the begining of file

switch(getch())

{

case '1':

{

clrscr();

gotoxy(25,4);

printf("\*\*\*\*Search Books By Id\*\*\*\*");

gotoxy(20,5);

printf("Enter the book id:");

scanf("%d",&d);

while(fread(&a,sizeof(a),1,fp)==1)

{

if(a.id==d)

{

gotoxy(20,7);

printf("The Book is available");

gotoxy(20,8);

printf("\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2");

gotoxy(20,9);

printf("\xB2 ID:%d",a.id);gotoxy(47,9);printf("\xB2");

gotoxy(20,10);

printf("\xB2 Name:%s",a.name);gotoxy(47,10);printf("\xB2");

gotoxy(20,11);

printf("\xB2 Author:%s ",a.Author);gotoxy(47,11);printf("\xB2");

gotoxy(20,12);

printf("\xB2 Qantity:%d ",a.quantity);gotoxy(47,12);printf("\xB2"); gotoxy(47,11);printf("\xB2");

gotoxy(20,13);

printf("\xB2 Price:Rs.%.2f",a.Price);gotoxy(47,13);printf("\xB2");

gotoxy(20,14);

printf("\xB2 Rack No:%d ",a.rackno);gotoxy(47,14);printf("\xB2");

gotoxy(20,15);

printf("\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2");

findbook='t';

}

}

if(findbook!='t') //checks whether conditiion enters inside loop or not

{

gotoxy(20,8);

printf("\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2");

gotoxy(20,9);printf("\xB2"); gotoxy(38,9);printf("\xB2");

gotoxy(20,10);

printf("\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2");

gotoxy(22,9);printf("\aNo Record Found");

}

gotoxy(20,17);

printf("Try another search?(Y/N)");

if(getch()=='y')

searchbooks();

else

mainmenu();

break;

}

case '2':

{

char s[15];

clrscr();

gotoxy(25,4);

printf("\*\*\*\*Search Books By Name\*\*\*\*");

gotoxy(20,5);

printf("Enter Book Name:");

scanf("%s",s);

int d=0;

while(fread(&a,sizeof(a),1,fp)==1)

{

if(strcmp(a.name,(s))==0) //checks whether a.name is equal to s or not

{

gotoxy(20,7);

printf("The Book is available");

gotoxy(20,8);

printf("\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2");

gotoxy(20,9);

printf("\xB2 ID:%d",a.id);gotoxy(47,9);printf("\xB2");

gotoxy(20,10);

printf("\xB2 Name:%s",a.name);gotoxy(47,10);printf("\xB2");

gotoxy(20,11);

printf("\xB2 Author:%s",a.Author);gotoxy(47,11);printf("\xB2");

gotoxy(20,12);

printf("\xB2 Qantity:%d",a.quantity);gotoxy(47,12);printf("\xB2");

gotoxy(20,13);

printf("\xB2 Price:Rs.%.2f",a.Price);gotoxy(47,13);printf("\xB2");

gotoxy(20,14);

printf("\xB2 Rack No:%d ",a.rackno);gotoxy(47,14);printf("\xB2");

gotoxy(20,15);

printf("\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2");

d++;

}

}

if(d==0)

{

gotoxy(20,8);

printf("\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2");

gotoxy(20,9);printf("\xB2"); gotoxy(38,9);printf("\xB2");

gotoxy(20,10);

printf("\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2");

gotoxy(22,9);printf("\aNo Record Found");

}

gotoxy(20,17);

printf("Try another search?(Y/N)");

if(getch()=='y')

searchbooks();

else

mainmenu();

break;

}

default :

getch();

searchbooks();

}

fclose(fp);

}

void issuebooks(void) //function that issue books from library

{

int t;

clrscr();

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ISSUE SECTION\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

gotoxy(10,5);

printf("\xDB\xDB\xDB\xDb\xB2 1. Issue a Book");

gotoxy(10,7);

printf("\xDB\xDB\xDB\xDb\xB2 2. View Issued Book");

gotoxy(10,9);

printf("\xDB\xDB\xDB\xDb\xB2 3. Search Issued Book");

gotoxy(10,11);

printf("\xDB\xDB\xDB\xDb\xB2 4. Remove Issued Book");

gotoxy(10,14);

printf("Enter a Choice:");

switch(getch())

{

case '1': //issue book

{

clrscr();

int c=0;

char another='y';

while(another=='y')

{

clrscr();

gotoxy(15,4);

printf("\*\*\*Issue Book section\*\*\*");

gotoxy(10,6);

printf("Enter the Book Id:");

scanf("%d",&t);

fp=fopen("Bibek.dat","rb");

fs=fopen("Issue.dat","ab+");

if(checkid(t)==0) //issues those which are present in library

{

gotoxy(10,8);

printf("The book record is available");

gotoxy(10,9);

printf("There are %d unissued books in library ",a.quantity);

gotoxy(10,10);

printf("The name of book is %s",a.name);

gotoxy(10,11);

printf("Enter student name:");

scanf("%s",a.stname);

struct dosdate\_t d; //for current date

\_dos\_getdate(&d);

a.issued.dd=d.day;

a.issued.mm=d.month;

a.issued.yy=d.year;

gotoxy(10,12);

printf("Issued date=%d-%d-%d",a.issued.dd,a.issued.mm,a.issued.yy);

gotoxy(10,13);

printf("The BOOK of ID %d is issued",a.id);

a.duedate.dd=a.issued.dd+RETURNTIME; //for return date

a.duedate.mm=a.issued.mm;

a.duedate.yy=a.issued.yy;

if(a.duedate.dd>30)

{

a.duedate.mm+=a.duedate.dd/30;

a.duedate.dd-=30;

}

if(a.duedate.mm>12)

{

a.duedate.yy+=a.duedate.mm/12;

a.duedate.mm-=12;

}

gotoxy(10,14);

printf("To be return:%d-%d-%d",a.duedate.dd,a.duedate.mm,a.duedate.yy);

fseek(fs,sizeof(a),SEEK\_END);

fwrite(&a,sizeof(a),1,fs);

fclose(fs);

c=1;

}

if(c==0)

{

gotoxy(10,11);

printf("No record found");

}

gotoxy(10,15);

printf("Issue any more(Y/N):");

fflush(stdin);

another=getche();

fclose(fp);

}

break;

}

case '2': //show issued book list

{

clrscr();

int j=4;

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Issued book list\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

gotoxy(2,2);

printf("STUDENT NAME CATEGORY ID BOOK NAME ISSUED DATE RETURN DATE");

fs=fopen("Issue.dat","rb");

while(fread(&a,sizeof(a),1,fs)==1)

{

gotoxy(2,j);

printf("%s",a.stname);

gotoxy(18,j);

printf("%s",a.cat);

gotoxy(30,j);

printf("%d",a.id);

gotoxy(36,j);

printf("%s",a.name);

gotoxy(51,j);

printf("%d-%d-%d",a.issued.dd,a.issued.mm,a.issued.yy );

gotoxy(65,j);

printf("%d-%d-%d",a.duedate.dd,a.duedate.mm,a.duedate.yy);

struct dosdate\_t d;

\_dos\_getdate(&d);

gotoxy(50,25);

printf("Current date=%d-%d-%d",d.day,d.month,d.year);

j++;

}

fclose(fs);

gotoxy(1,25);

returnfunc();

}

break;

case '3': //search issued books by id

{

clrscr();

int p,c=0;

char another='y';

while(another=='y')

{

gotoxy(10,6);

printf("Enter Book ID:");

scanf("%d",&p);

fs=fopen("Issue.dat","rb");

while(fread(&a,sizeof(a),1,fs)==1)

{

if(a.id==p)

{

issuerecord();

gotoxy(10,12);

printf("Press any key.......");

getch();

issuerecord();

c=1;

}

}

fflush(stdin);

fclose(fs);

if(c==0)

{

gotoxy(10,8);

printf("No Record Found");

}

gotoxy(10,13);

printf("Try Another Search?(Y/N)");

another=getch();

}

}

break;

case '4': //remove issued books from list

{

clrscr();

int b;

FILE \*fg; //declaration of temporary file for delete

char another='y';

while(another=='y')

{

gotoxy(10,5);

printf("Enter book id to remove:");

scanf("%d",&b);

fs=fopen("Issue.dat","rb+");

while(fread(&a,sizeof(a),1,fs)==1)

{

if(a.id==b)

{

issuerecord();

findbook='t';

}

if(findbook=='t')

{

gotoxy(10,12);

printf("Do You Want to Remove it?(Y/N)");

if(getch()=='y')

{

fg=fopen("record.dat","wb+");

rewind(fs);

while(fread(&a,sizeof(a),1,fs)==1)

{

if(a.id!=b)

{

fseek(fs,0,SEEK\_CUR);

fwrite(&a,sizeof(a),1,fg);

}

}

fclose(fs);

fclose(fg);

remove("Issue.dat");

rename("record.dat","Issue.dat");

gotoxy(10,14);

printf("The issued book is removed from list");

}

}

if(findbook!='t')

{

gotoxy(10,15);

printf("No Record Found");

}

}

gotoxy(10,16);

printf("Delete any more?(Y/N)");

another=getch();

}

}

default:

// gotoxy(10,18);

// printf("\aWrong Entry!!");

//getch();

//issuebooks();

break;

}

gotoxy(1,30);

returnfunc();

}

void viewbooks(void) //show the list of book persists in library

{

int i=0,j;

clrscr();

gotoxy(1,1);

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Book List\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

gotoxy(2,2);

printf(" CATEGORY ID BOOK NAME AUTHOR QTY PRICE RackNo ");

j=4;

fp=fopen("Bibek.dat","rb");

while(fread(&a,sizeof(a),1,fp)==1)

{

gotoxy(3,j);

printf("%s",a.cat);

gotoxy(16,j);

printf("%d",a.id);

gotoxy(22,j);

printf("%s",a.name);

gotoxy(36,j);

printf("%s",a.Author);

gotoxy(50,j);

printf("%d",a.quantity);

gotoxy(57,j);

printf("%.2f",a.Price);

gotoxy(69,j);

printf("%d",a.rackno);

printf("\n\n");

j++;

i=i+a.quantity;

}

gotoxy(3,25);

printf("Total Books =%d",i);

fclose(fp);

gotoxy(35,25);

returnfunc();

}

void editbooks(void) //edit information about book

{

clrscr();

int c=0;

int d,e;

gotoxy(20,4);

printf("\*\*\*\*Edit Books Section\*\*\*\*");

char another='y';

while(another=='y')

{

clrscr();

gotoxy(15,6);

printf("Enter Book Id to be edited:");

scanf("%d",&d);

fp=fopen("Bibek.dat","rb+");

while(fread(&a,sizeof(a),1,fp)==1)

{

if(checkid(d)==0)

{

gotoxy(15,7);

printf("The book is availble");

gotoxy(15,8);

printf("The Book ID:%d",a.id);

gotoxy(15,9);

printf("Enter new name:");scanf("%s",a.name);

gotoxy(15,10);

printf("Enter new Author:");scanf("%s",a.Author);

gotoxy(15,11);

printf("Enter new quantity:");scanf("%d",&a.quantity);

gotoxy(15,12);

printf("Enter new price:");scanf("%f",&a.Price);

gotoxy(15,13);

printf("Enter new rackno:");scanf("%d",&a.rackno);

gotoxy(15,14);

printf("The record is modified");

fseek(fp,ftell(fp)-sizeof(a),0);

fwrite(&a,sizeof(a),1,fp);

fclose(fp);

c=1;

}

if(c==0)

{

gotoxy(15,9);

printf("No record found");

}

}

gotoxy(15,16);

printf("Modify another Record?(Y/N)");

fflush(stdin);

another=getch() ;

}

returnfunc();

}

void returnfunc(void)

{

{

printf(" Press ENTER to return to main menu");

}

a:

if(getch()==13) //allow only use of enter

mainmenu();

else

goto a;

}

int getdata()

{

int t;

gotoxy(20,3);printf("Enter the Information Below");

gotoxy(20,4);printf("\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2");

gotoxy(20,5);

printf("\xB2");gotoxy(46,5);printf("\xB2");

gotoxy(20,6);

printf("\xB2");gotoxy(46,6);printf("\xB2");

gotoxy(20,7);

printf("\xB2");gotoxy(46,7);printf("\xB2");

gotoxy(20,8);

printf("\xB2");gotoxy(46,8);printf("\xB2");

gotoxy(20,9);

printf("\xB2");gotoxy(46,9);printf("\xB2");

gotoxy(20,10);

printf("\xB2");gotoxy(46,10);printf("\xB2");

gotoxy(20,11);

printf("\xB2");gotoxy(46,11);printf("\xB2");

gotoxy(20,12);

printf("\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2");

gotoxy(21,5);

printf("Category:");

gotoxy(31,5);

printf("%s",catagories[s-1]);

gotoxy(21,6);

printf("Book ID:\t");

gotoxy(30,6);

scanf("%d",&t);

if(checkid(t) == 0)

{

gotoxy(21,13);

printf("\aThe book id already exists\a");

getch();

mainmenu();

return 0;

}

a.id=t;

gotoxy(21,7);

printf("Book Name:");gotoxy(33,7);

scanf("%s",a.name);

gotoxy(21,8);

printf("Author:");gotoxy(30,8);

scanf("%s",a.Author);

gotoxy(21,9);

printf("Quantity:");gotoxy(31,9);

scanf("%d",&a.quantity);

gotoxy(21,10);

printf("Price:");gotoxy(28,10);

scanf("%f",&a.Price);

gotoxy(21,11);

printf("Rack No:");gotoxy(30,11);

scanf("%d",&a.rackno);

return 1;

}

int checkid(int t) //check whether the book is exist in library or not

{

rewind(fp);

while(fread(&a,sizeof(a),1,fp)==1)

if(a.id==t)

return 0; //returns 0 if book exits

return 1; //return 1 if it not

}

int t(void) //for time

{

time\_t t;

time(&t);

printf("Date and time:%s\n",ctime(&t));

return 0 ;

}

void show\_mouse(void) //show inactive mouse pointer in programming

{

union REGS in,out;

in.x.ax = 0x1;

int86(0x33,&in,&out);

}

void Password(void) //for password option

{

clrscr();

char ch,pass[10];

int i=0;

textbackground(WHITE);

textcolor(RED);

gotoxy(15,5);

printf("\*\*\*Password Protected\*\*\*");

gotoxy(15,7);

printf("Enter Password:");

while(ch!=13)

{

ch=getch();

putch('\*');

if(ch!=13){

pass[i] = ch;

i++;

}

}

pass[i] = '\0';

if(strcmp(pass,password)==0)

{

gotoxy(15,9);

//textcolor(BLINK);

printf("Password match");

gotoxy(17,10);

printf("Press any key to countinue.....");

getch();

mainmenu();

}

else

{

gotoxy(15,16);

printf("\aWarning!! Incorrect Password");

getch();

Password();

}

}

void issuerecord() //display issued book's information

{

clrscr();

gotoxy(10,8);

printf("The Book has taken by Mr. %s",a.stname);

gotoxy(10,9);

printf("Issued Date:%d-%d-%d",a.issued.dd,a.issued.mm,a.issued.yy);

gotoxy(10,10);

printf("Returning Date:%d-%d-%d",a.duedate.dd,a.duedate.mm,a.duedate.yy);

}

//End of program

# Program Output

The screen below, asks the user for pass word and if password matched then allow the user to go inside the program.

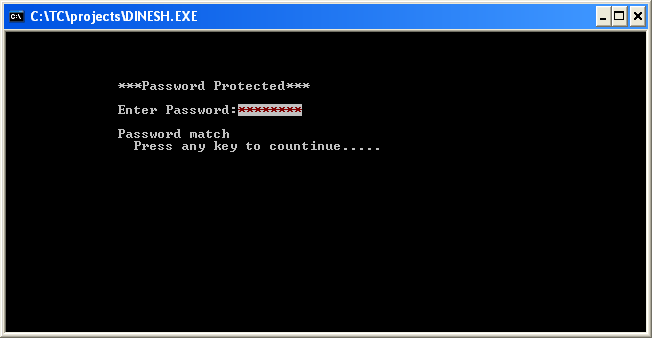


Fig:-1

The screen shows the style of mainmenu.

 Fig:-2

Screen showing Add books.

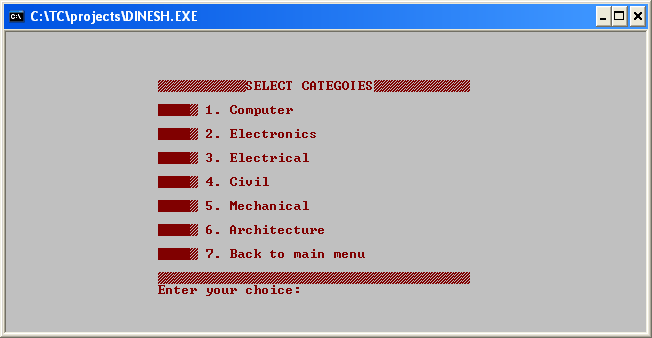


Fig:-3

Screen showing saving books process.

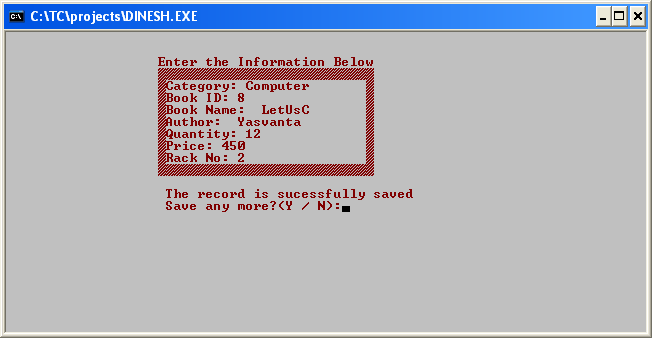
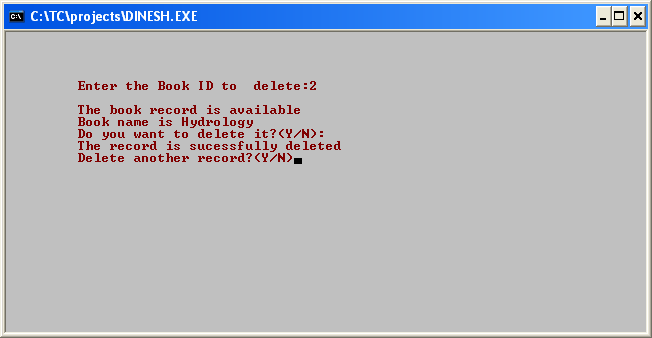


Fig:-4

Screen showing Process of Deleting.

 Fig:-5

Screen for search books by id and by name:-

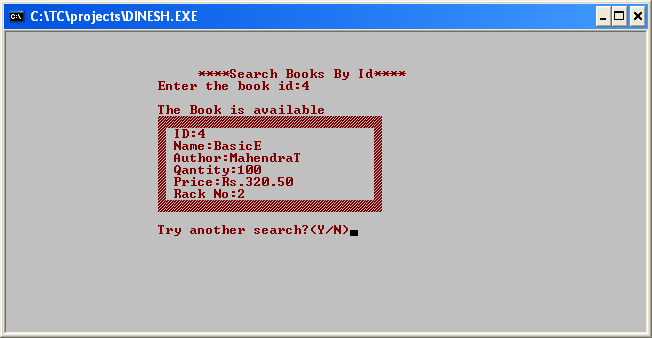
****

Fig:-6.1

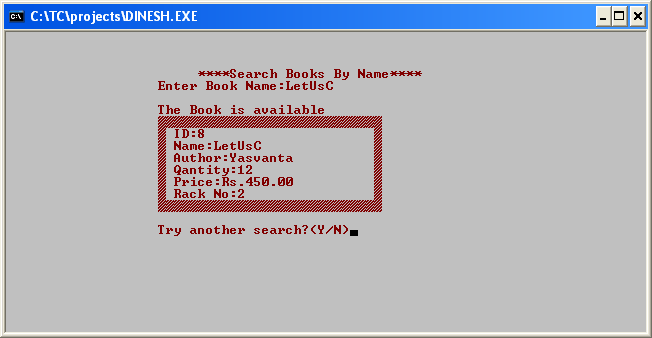


Fig:-6.2

Screen showing Books Record in library:

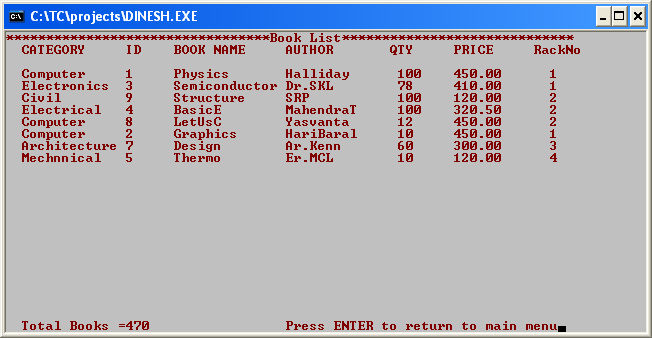


Fig:7

Screen showing Issue process:- 1. Process of issue:

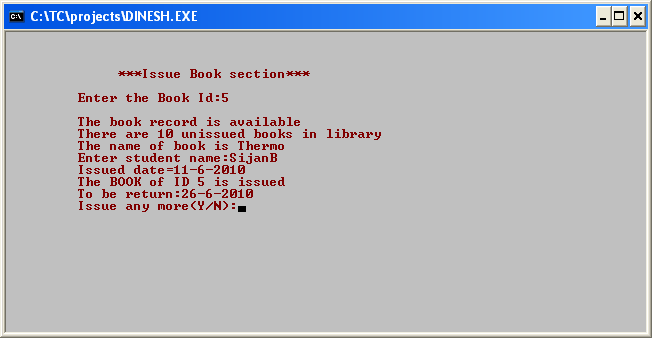


Fig:-8.1

2. View Issued books:-

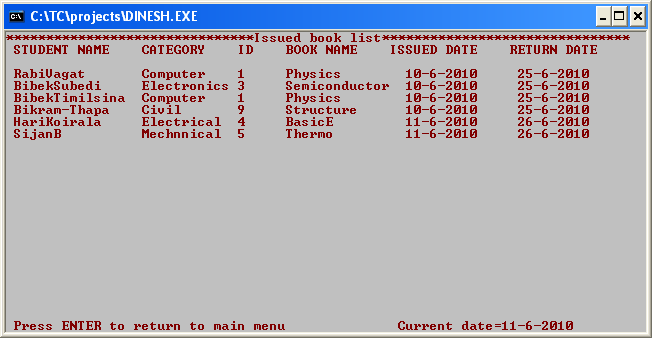


Fig:-8.20

3. Search issued Books:-

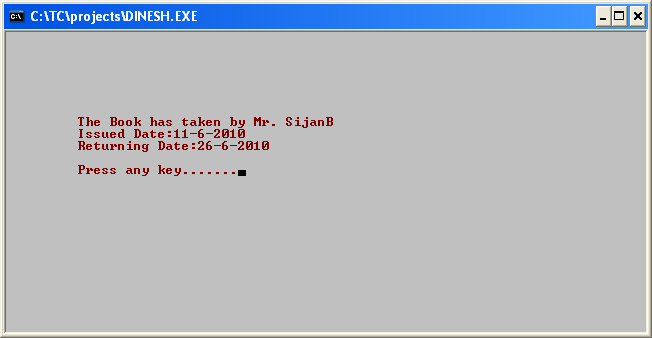


Fig:-8.3

Screen showing process of Editing:-

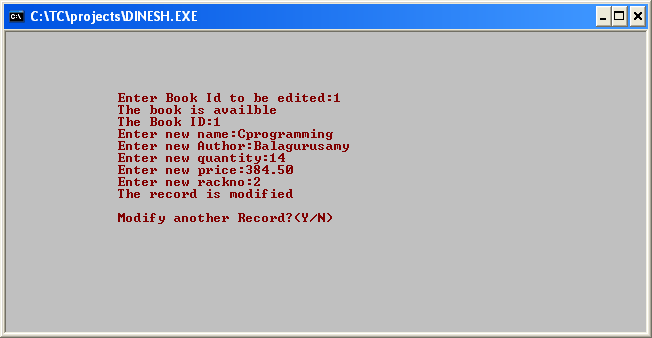
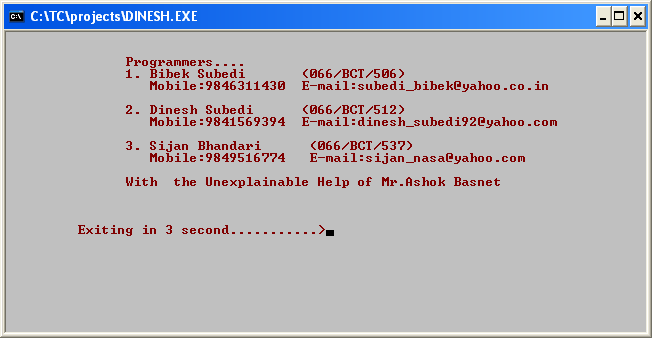


Fig:-9

Screen showing Closing application:-

 Fig:-10

Screen showing Warning message:-

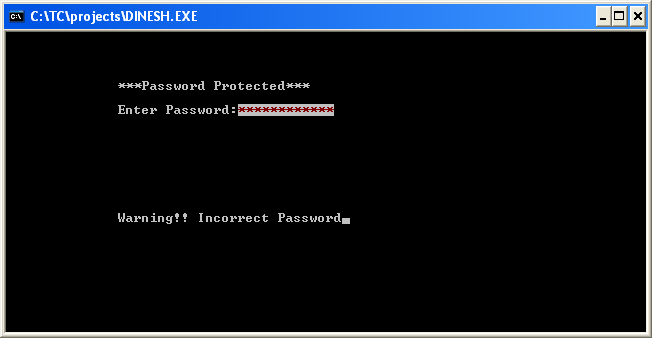


Fig:-11

# Discussion and Program Limitations

After a continuous and devoted attempt, we have finally completed our program ‘Library Management’. The program basically gives the idea about library management, how to manage books, how to issue them to students etc. The main propose of the program is to support library to manage it, and its contents easily and permanently. During the program execution we encountered tremendous problems which we are going to explain here.

We have divided our program into SIX division i.e. Add books, Delete Books, Search Books, View Booklist, Issue Books and Edit Books with sub divisions. We try to handle our program by password option. But there is no option to modify the password because we fixed the permanent password in our program i.e. we didn’t create any file to save password instead we simply compare the string of fixed permanent password with entered one. This is one of the limitations of our program. In order to change the password we have to change it in program source code.

In add books we have given a user the choice that in which Category he wants to save the books. When he entered one of the options the category will automatically save in the book record. Category, Book id, Book name, Author, quantity, Price and Rack number should be given in order to save the information about the book. Another limitation in ‘add books’ is that the all books given in ‘quantity’ have same id. Different types of book are given different ID but same types of book have same ID no matter how many books are there in this type. The books are stored in Rack No. as entered by the user. This will help the user to find the book easily and manage it. But there is no limitation of Rack No. the user can enter any number as freely as he wants. So user itself divides the rack and enters the rack number carefully. Also the Book ID should be entered very carefully because there is no option to modify the book id in edit option. In case one entered wrong ID he/she should delete this book and should again add it in library.

In delete books option the user should entered the Id of books which he wants to delete. So user must know the Id of books which he wants to delete. If the program finds the books, according to the Id entered by user, in library, it displays the information about the books and asks the user “Do you want to delete it(Y/N)?” if he enters Y, the book will be deleted from record. Otherwise it shows a message “No record Found” that is, the program didn’t find any books belonging to ID as entered by user in library record. In (Y/N) option throughout our program if one entered Y then the program do as he want but any key except Y will do the same task as done by N option. To delete books, we create a temporary file and copy the entire item from record file to temporary file except one which we want to delete. After this we remove the record file and rename the temporary file as record file.

In search books option we give two option to user 1.Search by Id and 2. Search by Name. But either id or name of the books the user must know in order to search the books. In search by id

the user should enter the Id of the book. The program search the Book’s information that belongs to the id in record list. If it finds the Books it displays the information otherwise it Displays the message “No Record Found”. On the other hand in search by name user must enter the name of the book to search the books. But one important care should be while doing this, that is, user must enter the name of the book in the same case as it is. In case user entered the name in different case the program shows the message “No Record Found”.

We have divided the issue books section in four main subdivisions *viz.,* ‘Issue book’, ‘view issued books’, ’Search issued books’ and ‘remove issued books’. In issue books section one the record of students and book’s name is saved along with Issued date and Return date. Student should return books after 15 days from the date of issue. The current date is abstract from \_dos\_getdate contain in <dos.h>. One of the major limitation in the issue book section is that one book can be issued more than its quantity simultaneously. i.e. even the books are not returned we can issue the books several times greater than the number of books present in library record.

We tried several times to avoid this but due to time limit and other factor we were unable to do this. In ‘view issued book’ section one can sees the books which are issued with student name, book id, book name, issued date and return date. To make the user easy to do his/her task we have give the current date in the screen. But here is sum limitation. In C programming, one of the disadvantages is that output screen does not contain any option for scrolling. So when books list exceeds the size more than screen size, we cannot see the earlier books record. So is the case in our program. So the screen can hold only the limited amount of records. In ‘search issued books’ section we can find the books record containing the name of student taking the books, issued date and return date. In remove issued section, user can delete the record of books after students returns the books.

In view books list, the record of books is displayed. If books are removed then books record is removed from the books list. Scrolling is here also persists as problem.

Last option “Edit Book’s record” edits the books record, if user accidently saves the wrong record for given id of the books. Here except book id everything can be modified. So, one should careful while inserting book id at the time of adding books.

We write our program by the combination of many user defined function, calling them frequently when necessary. There may not be the proper arrangement of function and codes in the program. But tried our best to avoid above discussed limitation but still we think we are unable to do this. And we commit we will possibly avoid them in out coming projects.

# Summary

Hence after the completion of the project we got familiar with the C programming and its features.

A complete and useful library management can only be developed with lot of intensive effort and time. Due to lack of time and we are beginners in programming program that we expected can’t be developed by us. Our library management may not be must useful for library in our college but it will be the most useful for study and programming practice using C.

As a whole, the project has been a good learning experience for us. We have gained knowledge about the various aspects of C programming. At the same time, we have developed a deep understanding about the file handling in C.

We still want to emphasize that the program is not complete by itself. There is still a lot of room for improvement. Graphics may be added to program to make it more attractive. The mouse cursor may be initialized in order to make the program even more interactive.

# References

*Balagurusamy .E., Programming In ANSI C, the Tata McGraw-Hill Companies, 8th Edition, 2008.*

*Kanetkar Yashavant, Let Us C, BPB Publication, 9th Edition, 2009.*

*Gottfriend, Baryon S, Schaum’s outlines Programming With C, the Tata McGraw-Hill, 2007.*

[*www.google.com*](http://www.google.com)

[*www.sourcecodesworld.com*](http://www.sourcecodesworld.com)

[*www.cprogramming.com*](http://www.cprogramming.com)