

DATA STRUCTURES IN C
A PROJECT REPORT ON
BOOKSTORE MANAGEMENT SYSTEM

SUBMITTED BY

Name of the students	PRN
Parth Nikam	20070123120
Jude Praneet Maria	20070123105
Pratham Yogesh Talekar	20070123113
Satvik Sasidhar	20070123089
Sattenapalli	



SYMBIOSIS INSTITUTE OF TECHNOLOGY
A CONSTITUENT OF SYMBIOSIS INTERNATIONAL (DEEMED
UNIVERSITY)

Pune - 412115

2021-22

Aim:

The aim of this project is to emulate a system comprising of the functions of a working bookstore management set-up in a console application environment.

Program Objectives :

- Create a basic yet navigable user interface for the system.
- Implement “add items to inventory/stock.”
- Implement “View list of items in inventory/stock.”
- Implement “Highest Rated Book in the inventory.”
- Start-up and Exit Procedures.

Program Details :

- Number of User Defined Functions : Nine
- Constructs Used : Switch Case and If Constructs.
- Libraries Used : stdio and string
- Data Structures Used : 2 (Including Read and Unread)
- Predefined Functions Used : fprintf, scanf, fopen, fclose, fread, fwrite, strcpy, strcmp.

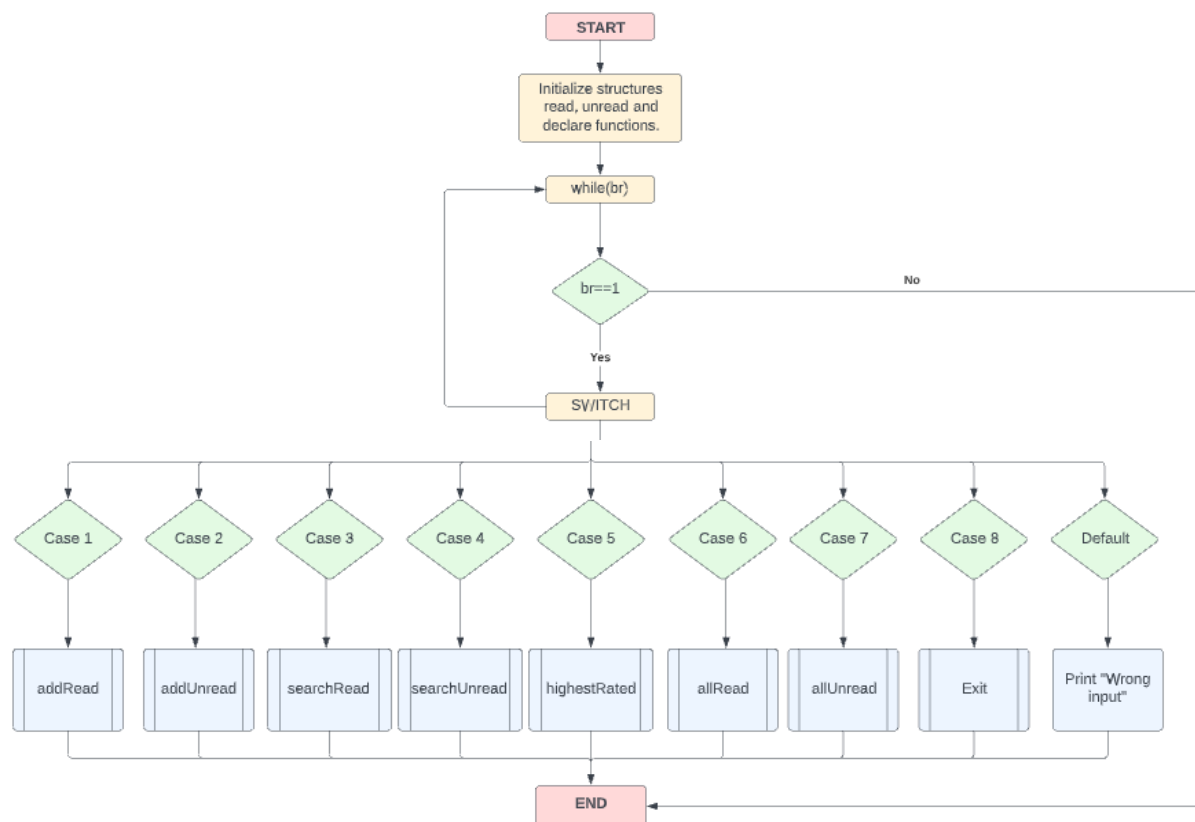
Introduction

The mini-project “Book Store management system project in C” is a console application using the C programming language. *The aim of this project is to simulate the working i.e., all the functions of a real world bookstore – hence the name ‘Bookstore Management System’ through the mode of a console application.* The code for this program is designed to perform operations such as Inserting a new book name/author/rating to a read/unread list and perform the delete operation for the same, in addition to this, the program also facilitates the viewing of all of the items in the list of read/unread books.

This particular application of Data Structures (in C) was specifically chosen by our group considering current trend data and other related info. Post the severity of the Coronavirus Pandemic, several old retail practices have suffered a considerable pitfall in terms of sales figures and even to the point of being closed. In an article by VOX media, the percentage of imminent threat to closing of *independent bookstores* in terms of being shut down was estimated to an *aggregate of twenty*. This is despite the functioning of several renowned charitable foundations.^[1] Hence, we decided to build a console application system wherein, bookstores can add the data pertaining to the stock of books in-store or in their respective inventories.

Therefore, the system proposed in the project is that of the same functionality. In this program, we utilize techniques such as file handling to store data in .txt files. This stored data is further extracted from the same .txt files by the use of file pointers, file based command (like fread and fwrite) and the accessed using structs. In this way, the code of this program functions. On the front end side of things, a switch case set-up has been designed for ease of operation wherein, each case pertains to each distinct functionality of this system and there are ten cases. In each case methods for the dedicated task has been called (with/without inputs) thus fulfilling the motive of that particular case. In this manner, the whole program depicts the functionalities of a working set-up thus being one itself.

Flowchart



Algorithm

Start

Declare and define functions

Define struct read and unread

Set variables br to 1 and choice = 0

while (br)

case 1 :

call addRead()

break

case 2 :

call addUnread()

break

case 3 :

call search()

break

case 4 :

call searchUnread()

break

case 5 :

call highestRated()

break

case 6 :

call allRead()

break

case 7:

call allUnread()

break

default

END

Algorithms for Functions

search()

```

Start
Define char sk[30]
Declare file pointer
Declare structure to store the data
Open file in read mode
    If file does not exists, print error
While loop to read till end of the file
if input given to search within the file
    print searching and print
    name,author,genre,rating
else
    print no book found
Close file
End
  
```

addUnread()

```

Start
Open file in append mode
    If outfile is null
        print error opening file
Declare structure
print enter the details of book you want to read
    print name
    print author
    print genre
Declare structure to write
print contents to file written succesfully
else
print error
Close file
End
  
```

search()

```

Start
Define char sk[30]
Declare file pointer
Declare structure to store the data
Open file in read mode
    If file does not exists, print error
While loop to read till end of the file
if input given to search within the
file
    print searching and print
    name,author and genre
else
    print no book found
Close file
  
```

Highest()

```

Start
Set highest to 0
set max character to 30
Declare structure
Open file in read mode
If inf = null
    print error opening file
While loop to read till end of the file
If input stars is greater than highest, input
stars = highest stars
set bookname = inp.name
Close file
print highest rated book
print rating
End
  
```

Allread()

Start

Declare file pointer

Open file (read.txt) in read mode

Declare structure to read data (from read)

While loop to read till end of file

print name, author, rating

End

Allunread()

Start

Declare file pointer

Define char filename [100]

Open file (unread.txt) in read mode

Declare structure to read data (from unread)

While loop to read till end of file

Print name,author and genre.

End

Code

```
#include <stdio.h>
#include <string.h>

/*
    Functions:

    add to read (done)
    add to unread (done)
    seearch using name (done)
    find highest rated (done)
    add form unread to read
    delete from read (done)
    delete form unread (done)
    view all read
    view all unread

    */

//all functions declared here
void addRead();
void addUnread();
void search(char sk[30]);
void searchUnread(char sk[30]);
void deleteUnread();
int highestRated();
void deleteRead(char sk[30]);
//void addUnrRea();
void allRead();
void allUnread();
void addUnrRea(char sk[30]);

//struct for books that have been read
//includes name, author, rating, and genre
struct read{

    char name[30];
    char author[30];
    char genre[20];
    int stars;

};
```

```
//struct for books that have not been read
//includes name, author and genre

struct unread{

    char name[30];
    char author[30];
    char genre[20];

};

//main function, ie, the menu of the program
//the user can choose from the following options

int main(){

    int br=1, choice=0;

    while(br){

        printf("\n=====MENU=====\\n\\n");
        printf("(1) Add to READ \\n(2) Add to UNREAD \\n(3) Search READ\\n(4)
Search UNREAD\\n(5) Find highest rated \\n(6) Add from unread to read \\n(7)
View all read\\n(8) View All unread\\n(9) EXIT\\n\\nEnter your choice: ");

        scanf("%d",&choice);

        switch (choice)
        {
            case 1:
                addRead();
                break;

            case 2:
                addUnread();
                break;

            case 3:
                char key[30];
                printf("\\nEnter the name of the book you want to find: ");
                scanf("%s",key);
                search(key);
                break;
```


case 4:

```
char key0[30];
printf("\nEnter the name of the book you want to find1: ");
scanf("%s",key);
searchUnread(key);
break;
```

case 5:

```
int top=0;
top=highestRated();
printf("%d",top);
break;
```

case 6:

```
char key4[30];
printf("Mark book as read\n");
printf("Enter name of book: ");
scanf("%s",key4);
addUnrRea(key4);
break;
```

case 7:

```
printf("\nAll Read\n");
allRead();
break;
```

case 8:

```
printf("All Unread\n");
allUnread();
break;
```

case 9:

```
printf("\n***** G O O D B Y E *****\n\n");
br=0;
break;
```

default:

```
printf("Invalid choice. Please try again");
break;
```

```
}
```

```
}
```

```
}
```

```
void searchUnread(char sk[30]){

    //file pointer
    FILE *inf;

    //struct to store the data (inp being input)
    struct read inp;

    //open file, read.txt in read mode
    inf = fopen ("unread.txt", "r");

    //if function for if file exists or not
    if (inf == NULL) {
        fprintf(stderr, "\nError to open the file\n");
        //exit (1);
    }

    //while loop to read the file
    //fread if file read, input is stored in inp, size of read, file name

    //pointer to a block of memory, size in bytes of each element to be read
    //number of elements, each one with a size of size bytes
    //pointer to a FILE object that specifies an input stream
    while(fread(&inp, sizeof(struct unread), 1, inf)){

        //if statement to compare strings, ie, input given and to see if it exists in the file
        if(!strcmp(sk,inp.name)){
            printf("\nSearching...\n");
            printf("NAME: %s\nAUTHOR: %s\nGENRE: %s\n",inp.name,inp.author,inp.genre);
            break;
            exit(1);
        }
        else{
            printf("\nSearching...\n");
            printf("No book found\n");
            break;
            exit(1);
        }
    }
}
```

```
//close file
fclose (inf);

}

//function for to display all books that have been read
void allRead(){

    //file pointer
    FILE *inf;

    //open file read.txt in read mode
    inf=fopen("read.txt","r");

    //read data from unread in the format of read struct and print it till end of file
    (eof)
    struct read r;

    while(fread(&r,sizeof(struct read),1,inf)){
        printf("\nNAME: %s\nAUTHOR: %s\nGENRE: %s\nRATING:
%d\n",r.name,r.author,r.genre,r.stars);
    }
}

//function for to display all books that have not been read
void allUnread(){

    //file pointer
    FILE *inf;
    //char filename[100], c;
    //open file unread.txt in read mode
    inf=fopen("unread.txt","r");

    //read data from unread in the format of unread struct and print it till end of
    file (eof)
    struct unread r;

    while(fread(&r,sizeof(struct unread),1,inf)){
        printf("\nNAME: %s\nAUTHOR: %s\nGENRE:
%s\n\n",r.name,r.author,r.genre);
    }
}
```

```
//function to search for a book that has been read
void search(char sk[30]){

    //file pointer
    FILE *inf;

    //struct to store the data (inp being input)
    struct read inp;

    //open file, read.txt in read mode
    inf = fopen ("read.txt", "r");

    //if function for if file exists or not
    if (inf == NULL) {
        fprintf(stderr, "\nError to open the file\n");
        //exit (1);
    }

    //while loop to read the file
    //fread if file read, input is stored in inp, size of read, file name

    //pointer to a block of memory, size in bytes of each element to be read
    //number of elements, each one with a size of size bytes
    //pointer to a FILE object that specifies an input stream
    while(fread(&inp, sizeof(struct read), 1, inf)){

        //if statement to compare strings, ie, input given and to see if it exists in the file
        if(!strcmp(sk,inp.name)){
            printf("\nSearching...\n");
            printf("NAME: %s\nAUTHOR: %s\nGENRE: %s\nRATING:
%d\n",inp.name,inp.author,inp.genre,inp.stars);
            break;
            exit(1);
        }
        else{
            printf("\nSearching...\n");
            printf("No book found\n");
            break;
            exit(1);
        }
    }
}
```

```
//close file
fclose (inf);

}

void deleteUnread(){
    FILE *inp;
    FILE *inp1;
    struct unread s;

    int j,name,found=0;

    inp = fopen("unread.txt","r");
    inp1 = fopen("unread1.txt","w");
    printf("enter name of book to delete: ");
    scanf("%s",name);

    while(fread(&s,sizeof(struct unread),1,inp)){
        if(s.name==name){
            found=1;
        }
        else{
            fwrite(&s,sizeof(struct unread),1,inp1);
        }
    }
    fclose(inp);
    fclose(inp1);
    if(found){
        inp1 = fopen("unread1.txt","r");
        inp = fopen("unread.txt","w");

        while(fread(&s,sizeof(struct unread),1,inp1)){
            fwrite(&s,sizeof(struct unread),1,inp);
        }
        fclose(inp);
        fclose(inp1);
    }
    return;
}

void addRead(){
```

```
FILE *outfile;

// open file for writing
outfile = fopen ("read.txt", "a");
if (outfile == NULL)
{
    fprintf(stderr, "\nError opening file\n");
    //exit (1);
}

struct read a;

printf("\nEnter the details of the book you just read in the format below\n");
printf("\nName: "); scanf("%s",&a.name);
printf("\nAuthor: "); scanf("%s",&a.author);
printf("\nGenre: "); scanf("%s", &a.genre);
printf("\nRating (Out of 5): "); scanf("%d",&a.stars);

// write struct to file
fwrite(&a, sizeof(struct read) ,1 ,outfile);

if(fwrite != 0){
    printf("\nContents to file written successfully!\n");
}
else{
    printf("Error writing to file!\n");
}

// close file
fclose (outfile);

return;

}

void addUnread(){

    FILE *outfile;

    // open file for writing
    outfile = fopen ("unread.txt", "a");
    if (outfile == NULL)
    {
```

```
    fprintf(stderr, "\nError opening file\n");
    //exit (1);
}
```

```
struct unread a;
```

```
printf("\nEnter the details of the book you want to read\n");
printf("\nName:"); scanf("%s",&a.name);
printf("\nAuthor:"); scanf("%s",&a.author);
printf("\nGenre: "); scanf("%s", &a.genre);
```

```
// write struct to file
fwrite(&a, sizeof(struct unread) ,1 ,outfile);
```

```
if(fwrite != 0){
    printf("\nContents to file written successfully!\n");
}
else{
    printf("Error writing file!\n");
}
```

```
// close file
fclose (outfile);
```

```
return;
```

```
}
```

```
int highestRated(){
```

```
    FILE *inf;
    struct read inp;
```

```
    int highest=0;
    char bookname[30];
```

```
    inf = fopen ("read.txt", "r");
    if (inf == NULL) {
        fprintf(stderr, "\nError to open the file\n");
        //exit (1);
    }
```

```
while(fread(&inp, sizeof(struct read), 1, inf)){
    //printf ("stars = %d\n", inp.stars);

    if(inp.stars>highest){

        highest=inp.stars;

        strcpy(bookname,inp.name);

    }
}
fclose (inf);

printf("\nHighest Rated Book: %s\n",bookname);
printf("Rating: ");
return highest;

}

void deleteRead(char sk[30]){
    FILE *inf;
    struct read inp;
    struct read readbooks[100];
    int counter=0;

    inf = fopen ("read.txt", "r");
    if (inf == NULL) {
        fprintf(stderr, "\nError to open the file\n");
        //exit (1);
    }
    while(fread(&inp, sizeof(struct read), 1, inf)){

        strcpy(readbooks[counter].name,inp.name);
        strcpy(readbooks[counter].author,inp.author);
        strcpy(readbooks[counter].genre,inp.genre);
        readbooks[counter].stars=inp.stars;

        counter++;
    }

    fclose (inf);

    for(int i=0;i<counter;i++){
```



```

    printf("Deleting the following data:\n");
    printf("NAME: %s\nAUTHOR: %s\nGENRE: %s\nRATING:%d\n",
readbooks[i].name, readbooks[i].author, readbooks[i].genre,readbooks[i].stars);
    exit(1);
}

FILE *outfile;

// open file for writing
outfile = fopen ("read.txt", "w");
if (outfile == NULL)
{
    fprintf(stderr, "\nError opening file\n");
    //exit (1);
}

for(int i=0;i<counter;i++){
    // write struct to file

    if(strcmp(sk,readbooks[i].name)){
        fwrite(&readbooks[i], sizeof(struct read) ,1 ,outfile);
    }
    // close file
}

fclose (outfile);
return;
}

void addUnrRea(char sk[30]){

    struct read a;

    FILE *inf;
    struct unread inp;

    inf = fopen ("unread.txt", "r");
    if (inf == NULL) {
        fprintf(stderr, "\nError to open the file\n");
        //exit (1);
    }
    while(fread(&inp, sizeof(struct unread), 1, inf)){
        printf ("name = %s\n", inp.name);

```

```
    if(!strcmp(sk,inp.name)){

        strcpy(a.name,inp.name);
        strcpy(a.author,inp.author);
        strcpy(a.genre,inp.genre);
        printf("Rating (out of 5)\n"); scanf("%d",&a.stars);

        break;
        exit(1);
    }
    fclose (inf);
}
```

```
FILE *outfile;
```

```
// open file for writing
outfile = fopen ("read.txt", "a");
if (outfile == NULL)
{
    fprintf(stderr, "\nError opening file\n");
    //exit (1);
}
```

```
// write struct to file
fwrite(&a, sizeof(struct read) ,1 ,outfile);
```

```
if(fwrite != 0){
    printf("Book marked as read!\n");
    exit(1);
}
else{
    printf("error writing file !\n");
}
```

```
// close file
fclose (outfile);
```

```
deleteRead(a.name);
return;
}
```

CODE

```

1 #include <stdio.h>
2 #include <string.h>
3
4 /*
5  Functions:
6
7  add to read (done)
8  add to unread (done)
9  search using name (done)
10 find highest rated (done)
11 add from unread to read
12 delete from read (done)
13 delete from unread (done)
14 view all read
15 view all unread
16
17 */
18
19 //all functions declared here
20 void addRead();
21 void addUnread();
22 void search(char sk[30]);
23 void searchUnread(char sk[30]);
24 void deleteUnread();
25 int highestRated();
26 void deleteRead(char sk[30]);
27 void addUnread();
28 void allRead();
29 void allUnread();
30 void addUnread(char sk[30]);
31
32 //struct for books that have been read
33 //includes name, author, rating, and genre
34 struct read{
35     char name[30];
36     char author[30];
37     char genre[20];
38     int stars;
39 };
40
41 //struct for books that have not been read
42 //includes name, author and genre

```

```

49
50 struct unread{
51     char name[30];
52     char author[30];
53     char genre[20];
54 };
55
56 //main function, is the menu of the program
57 //the user can choose from the following options
58
59 int main(){
60     int br=1, choice=0;
61     while(br){
62         printf("\n=====MENU=====\\n\\n");
63         printf("(1) Add to READ \\n(2) Add to UNREAD \\n(3) Search READ\\n(4) Search UNREAD\\n(5) Find highest rated \\n(6) Add from unread to read \\n(7) View all read\\n(8) View All unread\\n(9) EXIT\\n\\nEnter your choice: ");
64         scanf("%d",&choice);
65         switch (choice)
66         {
67             case 1:
68                 addRead();
69                 break;
70             case 2:
71                 addUnread();
72                 break;
73             case 3:
74                 char key[30];
75                 printf("\\nEnter the name of the book you want to find: ");
76                 scanf("%s",key);
77                 search(key);
78                 break;
79             case 4:
80                 char key0[30];
81                 printf("\\nEnter the name of the book you want to find: ");
82                 scanf("%s",key0);
83                 searchUnread(key0);
84                 break;
85         }
86     }
87 }

```

```

108     case 5:
109         int top=0;
110         top=highestRated();
111         printf("%d",top);
112         break;
113
114     case 6:
115         char key4[30];
116         printf("Mark book as read\n");
117         printf("Enter name of book: ");
118         scanf("%s",key4);
119         addUnRead(key4);
120         break;
121
122     case 7:
123         printf("\nAll Read\n");
124         allRead();
125         break;
126
127     case 8:
128         printf("All Unread\n");
129         allUnread();
130         break;
131
132     case 9:
133         printf("\n*** G O O D B Y E ***\n");
134         br=0;
135         break;
136
137     default:
138         printf("Invalid choice. Please try again");
139         break;
140 }
141 }
142
143 void searchUnread(char sk[30]){
144
145     //file pointer
146     FILE *inf;
147
148     //struct to store the data (inp being input)
149     struct read inp;
150
151     //open file, read.txt in read mode
152     inf = fopen ("unread.txt", "r");
153
154     //if function for if file exists or not

```

```

146     if (inf == NULL) {
147         fprintf(stderr, "\nError to open the file\n");
148         //exit (1);
149     }
150
151     //while loop to read the file
152     //fread if file read, input is stored in inp, size of read, file name
153
154     //pointer to a block of memory, size in bytes of each element to be read
155     //number of elements, each one with a size of size bytes
156     //pointer to a FILE object that specifies an input stream
157     while(fread(&inp, sizeof(struct unread), 1, inf)){
158
159         //if statement to compare strings, ie, input given and to see if it exists in the file
160         if(!strcmp(sk,inp.name)){
161             printf("\nSearching...\n");
162             printf("NAME: %s\nAUTHOR: %s\nGENRE: %s\n",inp.name,inp.author,inp.genre);
163             break;
164             exit(1);
165         }
166         else{
167             printf("\nSearching...\n");
168             printf("No book found\n");
169             break;
170             exit(1);
171         }
172     }
173
174     //close file
175     fclose (inf);
176 }
177
178 //function for to display all books that have been read
179 void allRead(){
180
181     //file pointer
182     FILE *inf;
183
184     //open file read.txt in read mode
185     inf=fopen("read.txt","r");
186
187     //read data from unread in the format of read struct and print it till end of file (eof)
188     struct read r;
189
190     while(fread(&r,sizeof(struct read),1,inf)){
191         printf("\nNAME: %s\nAUTHOR: %s\nGENRE: %s\nRATING: %d\n",r.name,r.author,r.genre,r.stars);
192     }
193 }

```

```

main.c main()
196
197 //function for to display all books that have not been read
198 void allUnread(){
199
200     //file pointer
201     FILE *inf;
202     //char filename[100], c;
203     //open file unread.txt in read mode
204     inf=fopen("unread.txt","r");
205
206     //read data from unread in the format of unread struct and print it till end of file (eof)
207     struct unread r;
208
209     while(fread(&r,sizeof(struct unread),1,inf)){
210         printf("\nNAME: %s\nAUTHOR: %s\nGENRE: %s\n\n",r.name,r.author,r.genre);
211     }
212 }
213
214
215 //function to search for a book that has been read
216 void search(char sk[30]){
217
218     //file pointer
219     FILE *inf;
220
221     //struct to store the data (inp being input)
222     struct read inp;
223
224     //open file, read.txt in read mode
225     inf = fopen ("read.txt", "r");
226
227     //if function for if file exists or not
228     if (inf == NULL) {
229         fprintf(stderr, "\nError to open the file\n");
230         //exit (1);
231     }
232
233     //while loop to read the file
234     //fread if file read, input is stored in inp, size of read, file name
235
236     //pointer to a block of memory, size in bytes of each element to be read
237     //number of elements, each one with a size of size bytes
238     //pointer to a FILE object that specifies an input stream
239     while(fread(&inp, sizeof(struct read), 1, inf)){
240
241         //if statement to compare strings, ie, input given and to see if it exists in the file
242         if(!strcmp(sk,inp.name)){
243             printf("\nSearching...\n");

```

```

main.c main()
245         break;
246         exit(1);
247     }
248     else{
249         printf("\nSearching...\n");
250         printf("No book found\n");
251         break;
252         exit(1);
253     }
254 }
255
256 //close file
257 fclose (inf);
258 }
259
260
261 void deleteUnread(){
262     FILE *inp;
263     FILE *inp1;
264     struct unread s;
265
266     int j,name,found=0;
267
268     inp = fopen("unread.txt","r");
269     inp1 = fopen("unread1.txt","w");
270     printf("enter name of book to delete: ");
271     scanf("%s",name);
272
273     while(fread(&s,sizeof(struct unread),1,inp)){
274         if(s.name==name){
275             found=1;
276         }
277         else{
278             fwrite(&s,sizeof(struct unread),1,inp1);
279         }
280     }
281     fclose(inp);
282     fclose(inp1);
283     if(found){
284         inp1 = fopen("unread1.txt","r");
285         inp = fopen("unread.txt","w");
286
287         while(fread(&s,sizeof(struct unread),1,inp1)){
288             fwrite(&s,sizeof(struct unread),1,inp);
289         }
290     }
291     fclose(inp);
292     fclose(inp1);

```

```
main.c main()
297
298 void addRead(){
299
300     FILE *outfile;
301
302     // open file for writing
303     outfile = fopen ("read.txt", "a");
304     if (outfile == NULL)
305     {
306         fprintf(stderr, "\nError opening file\n");
307         //exit (1);
308     }
309
310     struct read a;
311
312     printf("\nEnter the details of the book you just read in the format below\n");
313     printf("\nName: "); scanf("%s",&a.name);
314     printf("\nAuthor: "); scanf("%s",&a.author);
315     printf("\nGenre: "); scanf("%s", &a.genre);
316     printf("\nRating (Out of 5): "); scanf("%d",&a.stars);
317
318     // write struct to file
319     fwrite(&a, sizeof(struct read) ,1 ,outfile);
320
321     if(fwrite != 0){
322         printf("\nContents to file written successfully!\n");
323     }
324     else{
325         printf("Error writing to file!\n");
326     }
327
328     // close file
329     fclose (outfile);
330
331     return;
332 }
333
334 void addUnread(){
335
336     FILE *outfile;
337
338     // open file for writing
339     outfile = fopen ("unread.txt", "a");
340     if (outfile == NULL)
341     {
342         fprintf(stderr, "\nError opening file\n");
343         //exit (1);
344     }
```

```
main.c main()
350
351     printf("\nEnter the details of the book you want to read\n");
352     printf("\nName: "); scanf("%s",&a.name);
353     printf("\nAuthor: "); scanf("%s",&a.author);
354     printf("\nGenre: "); scanf("%s", &a.genre);
355
356     // write struct to file
357     fwrite(&a, sizeof(struct unread) ,1 ,outfile);
358
359     if(fwrite != 0){
360         printf("\nContents to file written successfully!\n");
361     }
362     else{
363         printf("Error writing file!\n");
364     }
365
366     // close file
367     fclose (outfile);
368
369     return;
370 }
371
372 int highestRated(){
373
374     FILE *inf;
375     struct read inp;
376
377     int highest=0;
378     char bookname[30];
379
380     inf = fopen ("read.txt", "r");
381     if (inf == NULL) {
382         fprintf(stderr, "\nError to open the file\n");
383         //exit (1);
384     }
385
386     while(fread(&inp, sizeof(struct read), 1, inf)){
387         //printf ("stars = %d\n", inp.stars);
388
389         if(inp.stars>highest){
390             highest=inp.stars;
391             strcpy(bookname,inp.name);
392         }
393     }
394 }
395
396 }
```

```

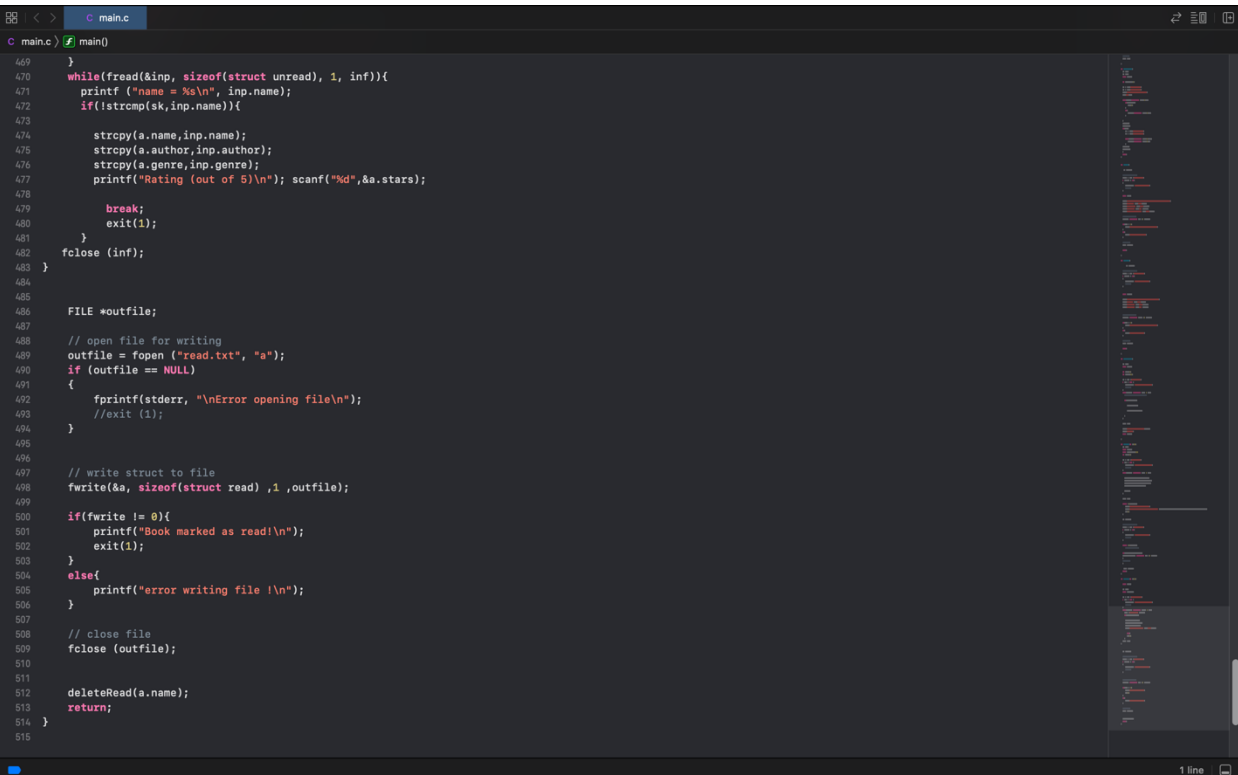
399
400 printf("\nHighest Rated Book: %s\n",bookname);
401 printf("Rating: ");
402 return highest;
403 }
404
405 void deleteRead(char sk[30]){
406     FILE *inf;
407     struct read inp;
408     struct read readbooks[100];
409     int counter=0;
410
411     inf = fopen ("read.txt", "r");
412     if (inf == NULL) {
413         fprintf(stderr, "\nError to open the file\n");
414         //exit (1);
415     }
416     while(fread(&inp, sizeof(struct read), 1, inf)){
417
418         strcpy(readbooks[counter].name,inp.name);
419         strcpy(readbooks[counter].author,inp.author);
420         strcpy(readbooks[counter].genre,inp.genre);
421         readbooks[counter].stars=inp.stars;
422
423         counter++;
424     }
425     fclose (inf);
426
427     for(int i=0;i<counter;i++){
428         printf("Deleting the following data:\n");
429         printf("NAME: %s\nAUTHOR: %s\nGENRE: %s\nRATING:%d\n", readbooks[i].name, readbooks[i].author, readbooks[i].genre,readbooks[i].stars);
430         exit(1);
431     }
432
433     FILE *outfile;
434
435     // open file for writing
436     outfile = fopen ("read.txt", "w");
437     if (outfile == NULL)
438     {
439         fprintf(stderr, "\nError opening file\n");
440         //exit (1);
441     }
442
443     for(int i=0;i<counter;i++){
444         // write struct to file

```

```

445
446     if(strcmp(sk,readbooks[i].name)){
447         fwrite(&readbooks[i], sizeof(struct read) ,1 ,outfile);
448     }
449     // close file
450 }
451
452     fclose (outfile);
453     return;
454 }
455
456 void addUnrRea(char sk[30]){
457
458     struct read a;
459
460     FILE *inf;
461     struct unread inp;
462
463     inf = fopen ("unread.txt", "r");
464     if (inf == NULL) {
465         fprintf(stderr, "\nError to open the file\n");
466         //exit (1);
467     }
468     while(fread(&inp, sizeof(struct unread), 1, inf)){
469
470         printf ("name = %s\n", inp.name);
471         if(!strcmp(sk,inp.name)){
472
473             strcpy(a.name,inp.name);
474             strcpy(a.author,inp.author);
475             strcpy(a.genre,inp.genre);
476             printf("Rating (out of 5)\n"); scanf("%d",&a.stars);
477
478             break;
479             exit(1);
480         }
481     }
482     fclose (inf);
483
484     FILE *outfile;
485
486     // open file for writing
487     outfile = fopen ("read.txt", "a");
488     if (outfile == NULL)
489     {
490         fprintf(stderr, "\nError opening file\n");
491         //exit (1);
492     }
493
494 }

```



```
469 }
470 while(fread(&inp, sizeof(struct unread), 1, inf)){
471     printf("name = %s\n", inp.name);
472     if(!strcmp(sk, inp.name)){
473
474         strcpy(a.name, inp.name);
475         strcpy(a.author, inp.author);
476         strcpy(a.genre, inp.genre);
477         printf("Rating (out of 5)\n"); scanf("%d", &a.stars);
478
479         break;
480         exit(1);
481     }
482     fclose (inf);
483 }
484
485 FILE *outfile;
486
487 // open file for writing
488 outfile = fopen ("read.txt", "a");
489 if (outfile == NULL)
490 {
491     fprintf(stderr, "\nError opening file\n");
492     //exit (1);
493 }
494
495 // write struct to file
496 fwrite(&a, sizeof(struct read), 1, outfile);
497
498 if(fwrite != 0){
499     printf("Book marked as read!\n");
500     exit(1);
501 }
502 else{
503     printf("error writing file !\n");
504 }
505
506 // close file
507 fclose (outfile);
508
509 deleteRead(a.name);
510 return;
511 }
512
513 }
```


Output

```
=====MENU=====
```

- (1) Add to READ
- (2) Add to UNREAD
- (3) Search READ
- (4) Search UNREAD
- (5) Find highest rated
- (6) Add from unread to read
- (7) View all read
- (8) View All unread
- (9) EXIT

Project Learnings

- How to write a console application
- How to manage data in .txt files and extract data from them using file pointers
- How to work with file-based commands like fread, fwrite.
- Working with constructs like switch, if and several loops
- Managing several functions and their execution in one single project.

Conclusion

This project has been a profound for me and my team-mates in every manner. From learning how to build a console application to using it to code our formulated idea, we have learnt several new lessons in C. I would like to take this opportunity to thank our professor, Dr Prabhat Thakur for giving us this opportunity to code such a project and put ourselves to this test.

References

1. How bookstores are weathering the Pandemic – VOX Media