**ODD SEMESTER-2021**

**Book Inventory System**

**MTE PROJECT SYNOPSIS**

**Course Name : Object Oriented Programming**

**Course Code : CO-203**

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER ENGINEERING**

**Submitted by**

**Vivek Kumar Thakur - 2K20/CO/493**

**Yash - 2K20/CO/499**

**Under the supervision**

**of**

**Mrs. Indu Singh**

****

**DEPARTMENT OF COMPUTER ENGINEERING**

**DELHI TECHNOLOGICAL UNIVERSITY**

**(Formerly Delhi College of Engineering)**

**Bawana road, Delhi-110042**

**LIST OF CONTENTS**

1. ACKNOWLEDGEMENT
2. ABSTRACT
3. INTRODUCTION
4. OBJECTIVES
5. CONCEPTS USED
6. CODE
7. OUTPUT
8. MODULES
9. SYSTEM REQUIREMENTS
10. REFERENCES

ACKNOWLEDGEMENT

We would like to earnestly acknowledge the sincere efforts and valuable time given by our teacher **Ms. Indu Singh**. Her valuable guidance and feedback has helped us in completing our project based on Object-oriented programming, Book Inventory System.

We would also like to thank her for providing us with the required material and guidelines and for encouraging us to complete this project.

**Vivek Kumar Thakur (2K20/CO/493)**

**Yash (2K20/CO/493)**

ABSTRACT

A bookshop maintains the inventory of books that are being sold at the shop. The list includes details such as author, title, price, publisher and stock position. Whenever a customer wants a book, the sales person inputs the title and author and the system searches the list and displays whether it is available or not. If it is not, an appropriate message is displayed. If it is, then the system displays the book details and requests for the number of copies required. If the requested copies book details and requests for the number of copies required. If the requested copies are available, the total cost of the requested copies is displayed; otherwise the message “Required copies are not in stock” is displayed. We have designed a system with the help of classes, constructors and custom operators and implemented it in C++. We have tried to implement the concepts we have learnt in Object Oriented Programming.

INTRODUCTION

This project, as the name suggests, provides an interface to manage a book inventory. This project is designed to perform basic functionalities.

The main purpose for developing this project is to computerize all the activities performed in a book inventory, so that all of the operations become fast and easy, and to remove the possibility of any error. It stores all the information of the books currently present in the stock using the concept of file handling.

The user can either be a customer or an admin. To keep the program simple and intuitive, both the customer and the admin are provided similar kinds of services. The user can store the record of a book in inventory, update or modify the record, delete a record, search a book, purchase a book from the inventory or view the record of all the books currently in stock.

OBJECTIVES

1. To create a computerized book inventory system.
2. To reduce the possibilities of error while recording data, bill generation, etc.
3. To create a system where a user(admin) can efficiently make entries in the inventory and (customer) can easily purchase a book.
4. To synchronise the stock report with purchases being made.
5. To give an intuition of how an online book inventory would look like.

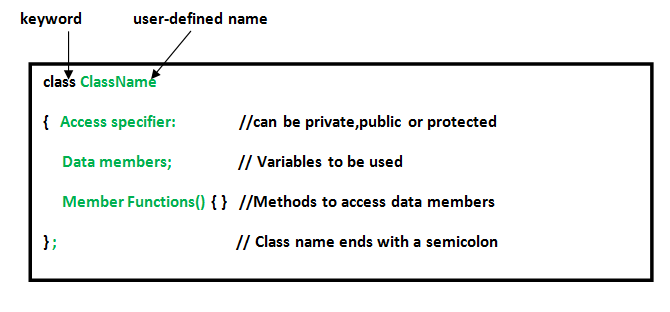
CONCEPTS USED

The following concepts of OOP has been used:

1. Classes and Objects

A class in C++ is the building block that leads to Object-Oriented programming. It is a user-defined data type, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class. A C++ class is like a blueprint for an object.

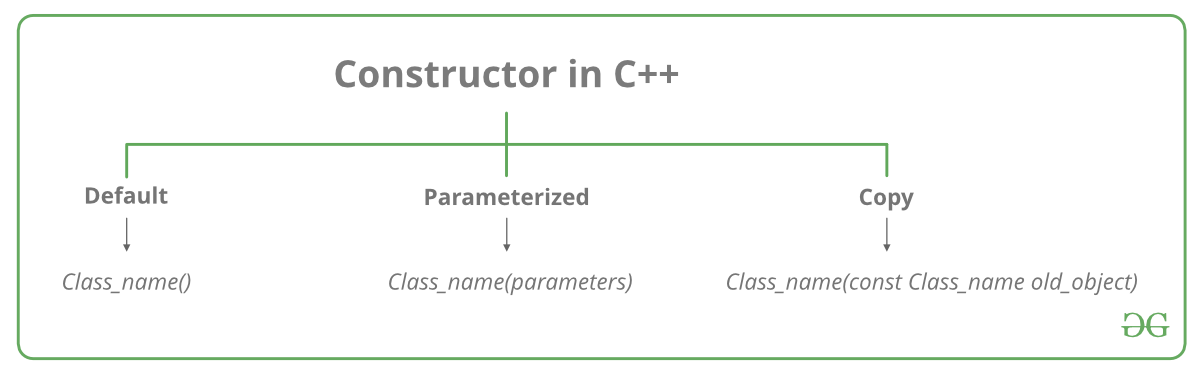
An Object is an instance of a Class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated.



1. Constructors

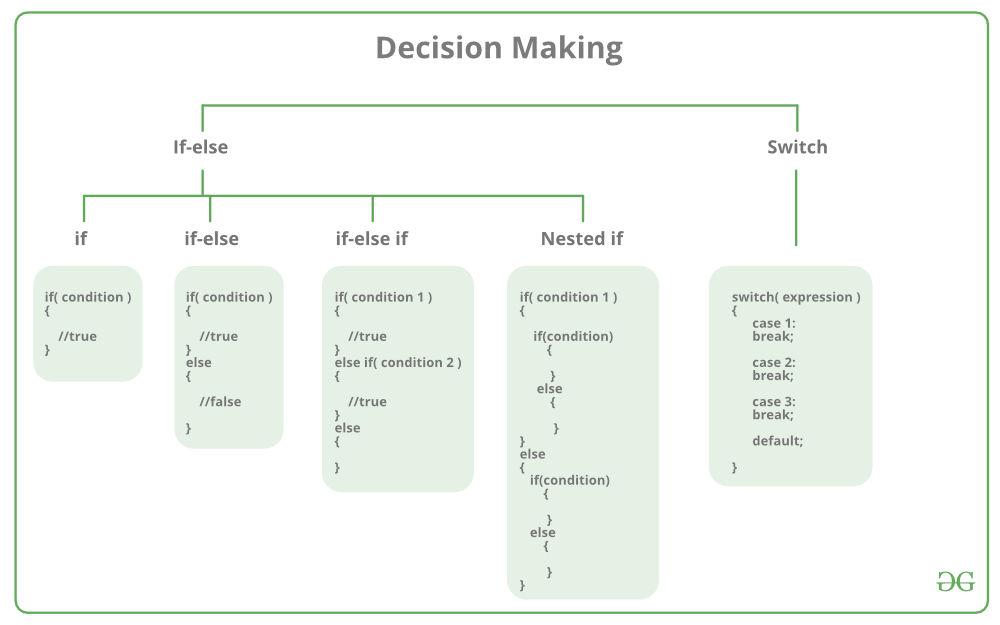
A constructor is a special type of member function of a class which initializes objects of a class. In C++, Constructor is automatically called when object(instance of class) create. It is special member function of the class because it does not have any return type.

* Constructor has same name as the class itself.
* Constructors don’t have return type
* A constructor is automatically called when an object is created.
* It must be placed in the public section of class.
* If we do not specify a constructor, C++ compiler generates a default constructor for object (expects no parameters and has an empty body).



1. Conditional Statements / Decision Making

There come situations in real life when we need to make some decisions and based on these decisions, we decide what should we do next. Similar situations arise in programming also where we need to make some decisions and based on these decisions we will execute the next block of code. For example, in C if x occurs then execute y else execute z. There can also be multiple conditions like in C if x occurs then execute p, else if condition y occurs execute q, else execute r. This condition of C else-if is one of the many ways of importing multiple conditions.



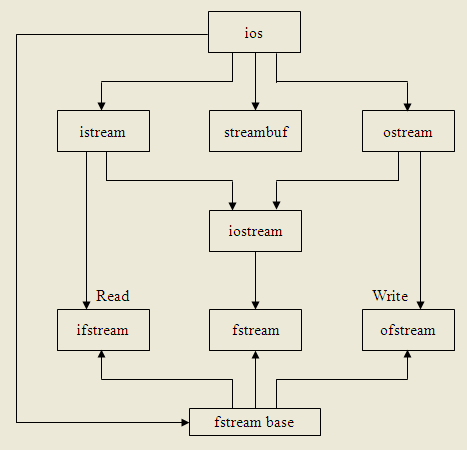
1. File handling

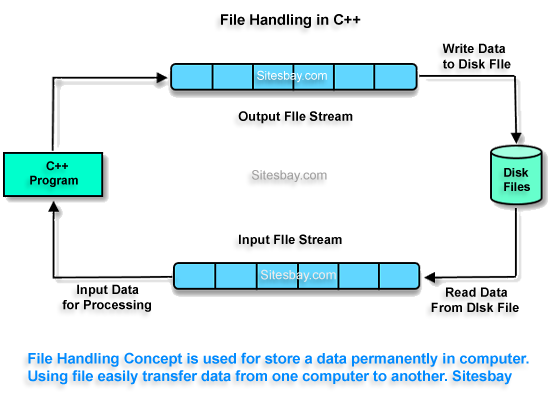
In C++, files are mainly dealt by using three classes fstream, ifstream, ofstream available in fstream header file.

ofstream: Stream class to write on files

ifstream: Stream class to read from files

fstream: Stream class to both read and write from/to files.





This topic of file handling is further divided into sub-topics:

* Create a file
* Open a file
* Read from a file
* Write to a file
* Close a file

C++ provides us with four different operations for file handling. They are:

* open() – This is used to create a file.
* read() – This is used to read the data from the file.
* write() – This is used to write new data to file.
* close() – This is used to close the file.

Modes in which a file can be opened :

* in - Open for reading
* out - Open for writing
* ate - Seek to end of file upon original open
* app - Append mode
* trunc - Truncate file if already exists
* nocreate - Open fails if file does not exists
* noreplace - Open fails if file already exists
* binary - Opens file as binary

1. File pointers

In C++, random access is achieved by manipulating seekg(), seekp(), tellg() and tellp() functions. The seekg() and tellg() functions allow you to set and examine the get\_pointer, and the seekp() and tellp() functions perform these operations on the put\_pointer.

The seekg() and tellg() functions are for input streams (ifstream) and seekp() and tellp() functions are for output streams (ofstream). However, if you use them with an fstream object then tellg() and tellp() return the same value. Also seekg() and seekp() work the same way in an fstream object.

1. *this* pointer

To understand ‘this’ pointer, it is important to know how objects look at functions and data members of a class.

Each object gets its own copy of the data member.

All-access the same function definition as present in the code segment.

Meaning each object gets its own copy of data members and all objects share a single copy of member functions.

Then now question is that if only one copy of each member function exists and is used by multiple objects, how are the proper data members are accessed and updated?

The compiler supplies an implicit pointer along with the names of the functions as ‘this’.

The ‘this’ pointer is passed as a hidden argument to all nonstatic member function calls and is available as a local variable within the body of all nonstatic functions. ‘this’ pointer is not available in static member functions as static member functions can be called without any object (with class name).

For a class X, the type of this pointer is ‘X\* ‘. Also, if a member function of X is declared as const, then the type of this pointer is ‘const X \*’

CODE :

#include <fstream>

#include <iostream>

#include <string.h>

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

/\*Book Inventory System made by Vivek Kumar Thakur(2K20/CO/493) and

Yash Munde(2K20/CO/499)\*/

using *namespace* std;

*class* book

{

*private:*

*char* author[20], title[20], publisher[20], bookID[13];

*float* price;

*int* stock;

*public:*

book() //Constructor

{

strcpy(author, "NULL");

strcpy(title, "NULL");

strcpy(publisher, "NULL");

strcpy(bookID, "NULL");

price = 0;

stock = 0;

}

*void* feeddata();

*void* feeddata(*char* \*, *char* \*, *char* \*, *int*, *int*);//overloaded function

*void* showdata();

*void* generatebookID();

*int* storebook();

*void* stock\_report();

*void* search\_title(*char* \*);

*void* search\_bookID(*char* \*);

*void* delete\_book(*char* \*);

*void* update\_data(*char* \*);

*void* update\_stocks(*int*);

*void* buy\_book(*int*, *char* \*);

};

*void* book::feeddata(*char* \**au*, *char* \**ti*, *char* \**pu*, *int* *pr*, *int* *st*)

{

strcpy(author, *au*);

strcpy(title, *ti*);

strcpy(publisher, *pu*);

generatebookID();

price = *pr*;

stock = *st*;

}

*void* book::update\_stocks(*int* *noc*)

{

stock = stock - *noc*;

}

*void* book::buy\_book(*int* *noc*, *char* \**ti*)

{

fstream file;

file.open("bookfile.txt", ios::in | ios::out | ios::ate | ios::binary);

file.seekg(0); //brings pointer from end to beginning

file.read((*char* \*)this, sizeof(\*this));

while (!file.eof())

{

/\*if (strcmp(ti, title) != 0)

{

cout << "\nBook not found";

break;

}

else\*/

if (strcmp(*ti*, title) == 0)

{

cout << "\nBOOK FOUND";

if (*noc* < stock)

{

cout << "\nSUFFICIENT COPIES ARE AVAILABLE";

/\*update\_stocks(noc);

int cost = noc \* price;

cout << "\n\nYou have to pay Rs. " << cost;

//int cost = noc \* price;

int count = (int)file.tellp() - sizeof(\*this); //+ sizeof(this->author) + sizeof(this->title) + sizeof(this->publisher) + sizeof(this->bookID); //check it

file.seekp(count);

file.write((char \*)this->stock, sizeof(this->stock)); //check it

//cout << "You have to pay Rs. " << cost;\*/

*char* au[20], ti[20], pu[20];

*int* pr, st;

strcpy(au, author);

strcpy(ti, title);

strcpy(pu, publisher);

pr = price;

st = stock - *noc*;

feeddata(au, ti, pu, pr, st); //this is called for collar object

*int* count = (*int*)file.tellp() - sizeof(\*this);

file.seekp(count); //when we read a record file

//pointer begins to point the next record, so we need to bring it back

//by one record

//file.tellp() tells where the file pointer is currently

file.write((*char* \*)this, sizeof(\*this));

cout << "\nRecord updated successfully";

*int* cost = *noc* \* price;

cout << "\n\nYOU HAVE TO PAY Rs. " << cost;

break;

}

}

file.read((*char* \*)this, sizeof(\*this));

}

file.close();

}

*void* book::update\_data(*char* \**ti*)

{

fstream file;

file.open("bookfile.txt", ios::in | ios::out | ios::ate | ios::binary);

file.seekg(0); // 0 bytes ahead of beginning

file.read((*char* \*)this, sizeof(\*this));

while (!file.eof())

{

if (strcmp(*ti*, title) == 0)

{

cout << "\nBook found";

feeddata(); //this is called for collar object

*int* count = (*int*)file.tellp() - sizeof(\*this);

file.seekp(count); //when we read a record file

//pointer begins to point the next record, so we need to bring it back

//by one record

//file.tellp() tells where the file pointer is currently

file.write((*char* \*)this, sizeof(\*this));

}

file.read((*char* \*)this, sizeof(\*this));

}

file.close();

}

*void* book::delete\_book(*char* \**ti*)

{

ifstream fin;

ofstream fout;

fin.open("bookfile.txt", ios::in | ios::binary);

if (!fin)

{

cout << "\nFile not found";

}

else

{

fout.open("newfile.txt", ios::app | ios::out | ios::binary);

fin.read((*char* \*)this, sizeof(\*this));

while (!fin.eof())

{

if (strcmp(*ti*, title) != 0)

{

fout.write((*char* \*)this, sizeof(\*this)); //writing to new file

}

fin.read((*char* \*)this, sizeof(\*this)); //pointing to next record in bookfile

}

fin.close();

fout.close();

remove("bookfile.txt");

rename("newfile.txt", "bookfile.txt");

}

}

*void* book::search\_title(*char* \**ti*)

{

*int* n = 0;

ifstream fin;

fin.open("bookfile.txt", ios::in | ios::binary);

if (!fin)

{

cout << "\nFile not found";

}

else

{

fin.read((*char* \*)this, sizeof(\*this));

while (!fin.eof())

{

if (strcmp(*ti*, title) == 0)

{

cout << "Book found!";

showdata();

n++;

break;

}

fin.read((*char* \*)this, sizeof(\*this)); //internally points to the next record automatically

}

if (n == 0)

{

cout << "Book not found";

}

fin.close();

}

}

*void* book::search\_bookID(*char* \**ID*)

{

*int* n = 0;

ifstream fin;

fin.open("bookfile.txt", ios::in | ios::binary);

if (!fin)

{

cout << "\nFile not found";

}

else

{

fin.read((*char* \*)this, sizeof(\*this));

while (!fin.eof())

{

if (strcmp(*ID*, bookID) == 0)

{

cout << "Book found!";

showdata();

n++;

break;

}

fin.read((*char* \*)this, sizeof(\*this)); //internally points to the next record automatically

}

if (n == 0)

{

cout << "Book not found";

}

fin.close();

}

}

*void* book::feeddata()

{

cin.ignore();

cout << "\nEnter Author Name: ";

cin.getline(author, 20); //getline assigns data to author; 20 indicates permitted size

cout << "Enter Title Name: ";

cin.getline(title, 20);

cout << "Enter Publisher Name: ";

cin.getline(publisher, 20);

generatebookID();

cout << "Enter Price: ";

cin >> price;

cout << "Enter Stock Position: ";

cin >> stock;

cout << "Book ID successfully generated : " << bookID;

}

*void* book::showdata()

{

cout << "\nAuthor Name: " << author;

cout << "\nTitle Name: " << title;

cout << "\nPublisher Name: " << publisher;

cout << "\nPrice: " << price;

cout << "\nStock Position: " << stock;

cout << "\nBook ID: " << bookID;

}

*void* book::generatebookID()

{

for (*int* i = 0; i < 4; i++)

{

\*(bookID + i) = \*(author + i);

}

for (*int* i = 4; i < 8; i++)

{

\*(bookID + i) = \*(title + i - 4);

}

for (*int* i = 8; i < 13; i++)

{

\*(bookID + i) = \*(publisher + i - 8);

}

}

*void* book::stock\_report()

{

ifstream fin;

fin.open("bookfile.txt", ios::in | ios::binary);

if (!fin)

{

cout << "\nFile not found";

}

else

{

fin.read((*char* \*)this, sizeof(\*this));

while (!fin.eof())

{

cout << "\n";

showdata();

fin.read((*char* \*)this, sizeof(\*this)); //internally points to the next record automatically

}

fin.close();

}

}

*int* book::storebook()

{

if (strcmp(title, "NULL") == 0)

{

cout << "\nBook data not found";

return (0);

}

else

{

ofstream fout;

fout.open("bookfile.txt", ios::app | ios::binary);

fout.write((*char* \*)this, sizeof(\*this));

/\*(address of collar object,size of collar object\*/

fout.close();

return (1);

}

}

/\*int menu()

{

cout << "\nBOOK MANAGEMENT";

cout << "\n1. Insert a book";

cout << "\n2. View stock report";

cout << "\n3. Search a book";

cout << "\n4. Delete a book";

cout << "\n5. Edit a record";

cout << "\n6. Buy a book";

cout << "\n7. Exit";

int choice;

cout << "\nEnter a choice";

return choice;

}\*/

*int* main()

{

book b1;

*char* title[20];

*int* choice;

while (1)

{

cout << "\n";

cout << "\n\t\t\t\t------------------------------";

cout << "\n\t\t\t\t| MENU |"

<< "\n\t\t\t\t| 1. Entry of New Book |"

<< "\n\t\t\t\t| 2. View stock report |"

<< "\n\t\t\t\t| 3. Search For Book |"

<< "\n\t\t\t\t| 4. Delete a record |"

<< "\n\t\t\t\t| 5. Edit a record |"

<< "\n\t\t\t\t| 6. Buy a book |"

<< "\n\t\t\t\t| 7. Exit |"

<< "\n\t\t\t\t------------------------------"

<< "\n\nEnter your Choice : ";

cin >> choice;

switch (choice)

{

case 1:

b1.feeddata();

b1.storebook();

cout << "\nRecord stored";

break;

case 2:

cout << "\nSTOCK REPORT\n";

b1.stock\_report();

break;

case 3:

cout << "\nTo search with book's title, press 1\nTo search with Book ID, press 2\n";

*int* num;

cin >> num;

if (num == 1)

{

cout << "\nEnter the title of the book : ";

cin.ignore();

cin.getline(title, 19);

b1.search\_title(title);

}

else if (num == 2)

{

cout << "\nEnter the Book ID : ";

*char* b\_ID[20];

cin.ignore();

cin.getline(b\_ID, 19);

b1.search\_bookID(b\_ID);

}

else

{

cout << "\nInvalid Choice Entered";

}

break;

case 4:

cout << "\nEnter the title of the book which is going to be deleted : ";

cin.ignore();

cin.getline(title, 19);

b1.delete\_book(title);

break;

case 5:

cout << "\nEnter the title of book whose data you want to update : ";

cin.ignore();

cin.getline(title, 19);

b1.update\_data(title);

break;

case 6:

cout << "\nEnter the name of the book which you want to buy : ";

cin.ignore();

cin.getline(title, 19);

*int* noc;

cout << "\nEnter the number of copies of the book : ";

cin >> noc;

b1.buy\_book(noc, title);

break;

case 7:

exit(0);

default:

cout << "\nInvalid Choice Entered";

}

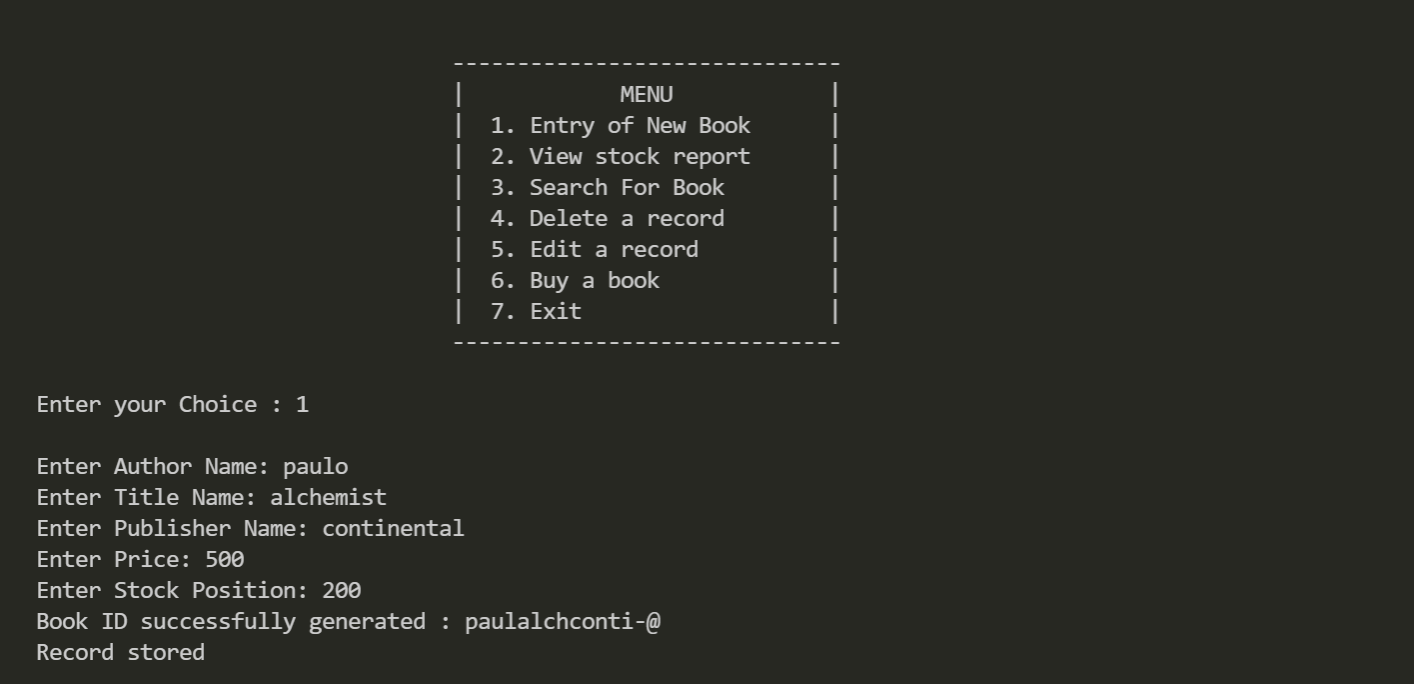
}

return 0;

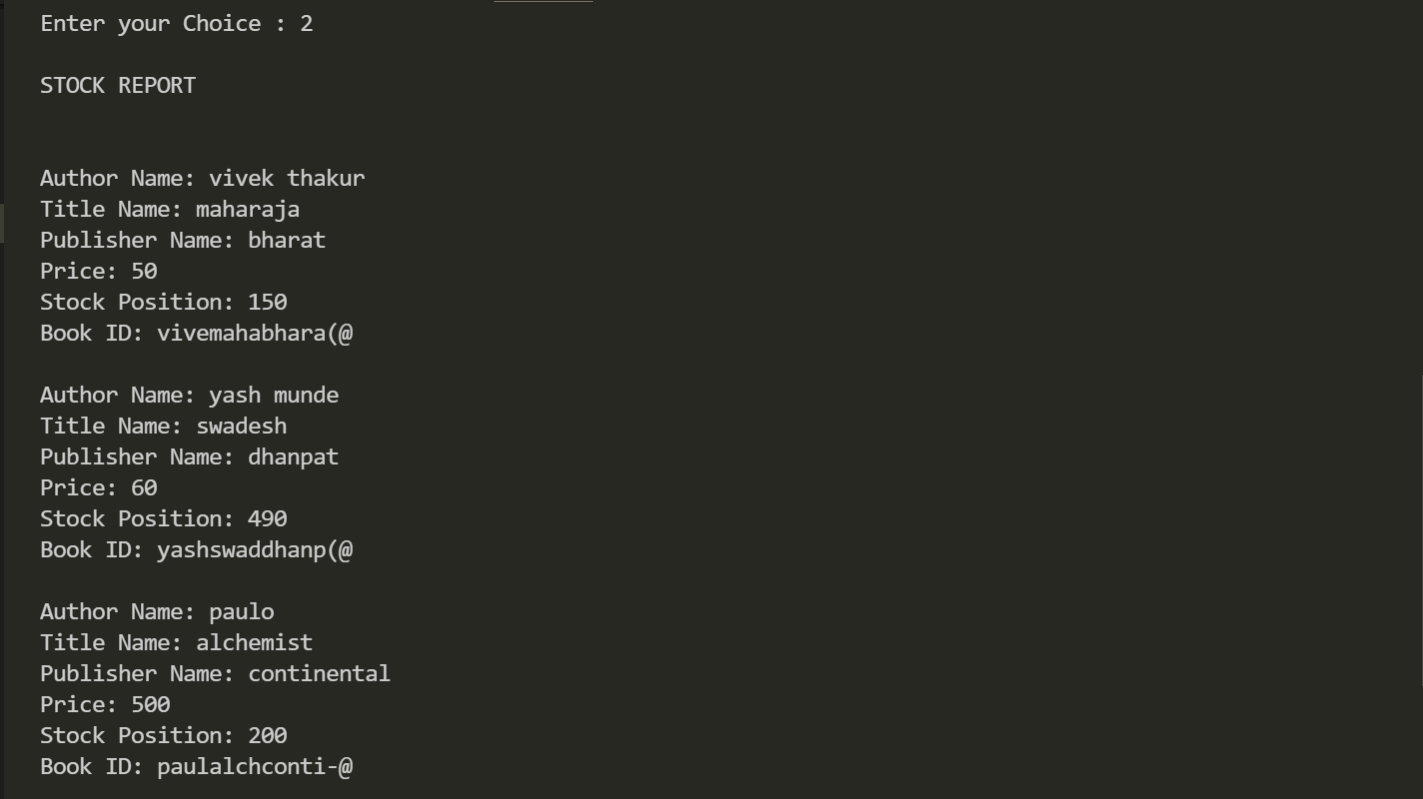
}

OUTPUT :

In the menu driven program, when the user enters 1

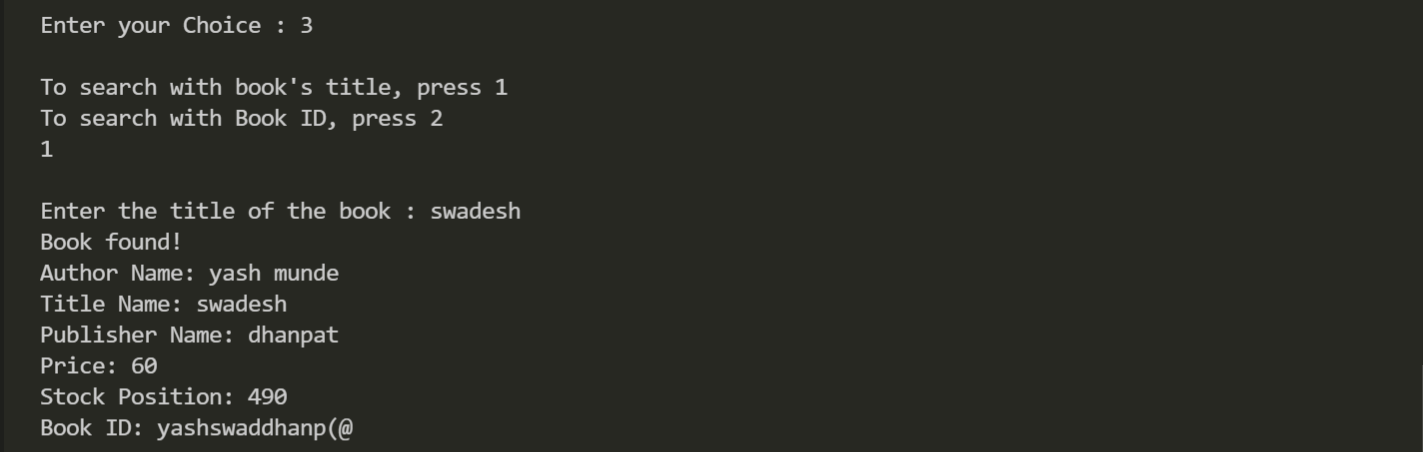


he can enter a record.

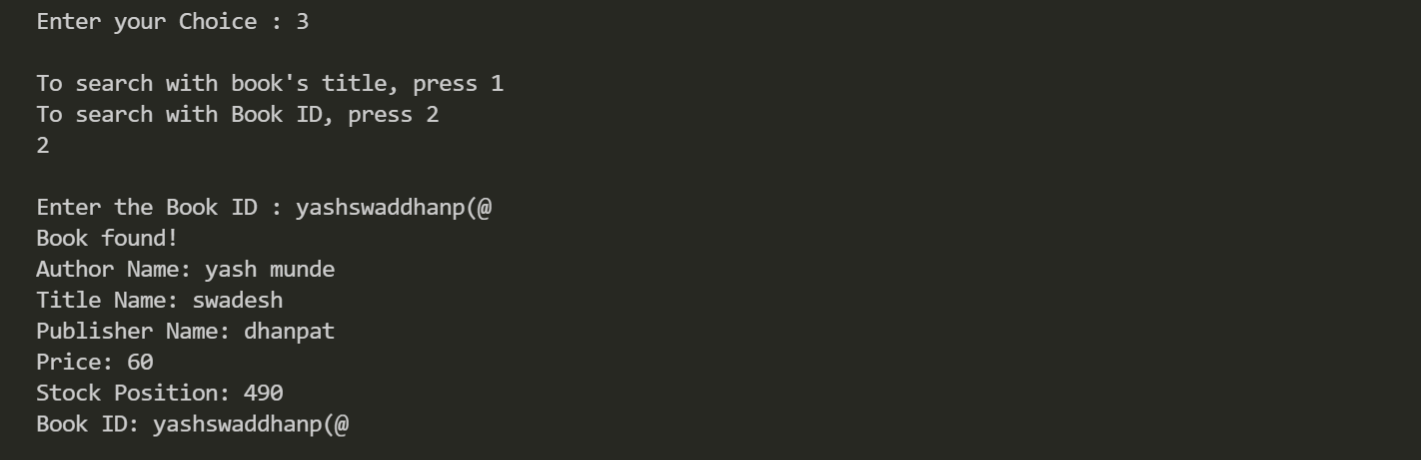
When the user enters 2, he can view stock report

When the user enters 3, he can search for a book, he can either search by the title or by the book ID.

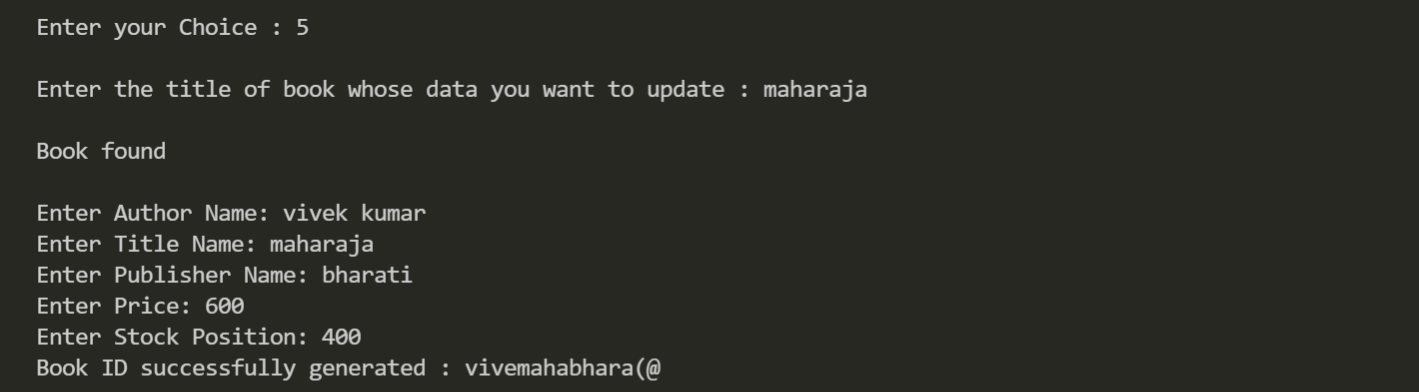
Search using book’s title:



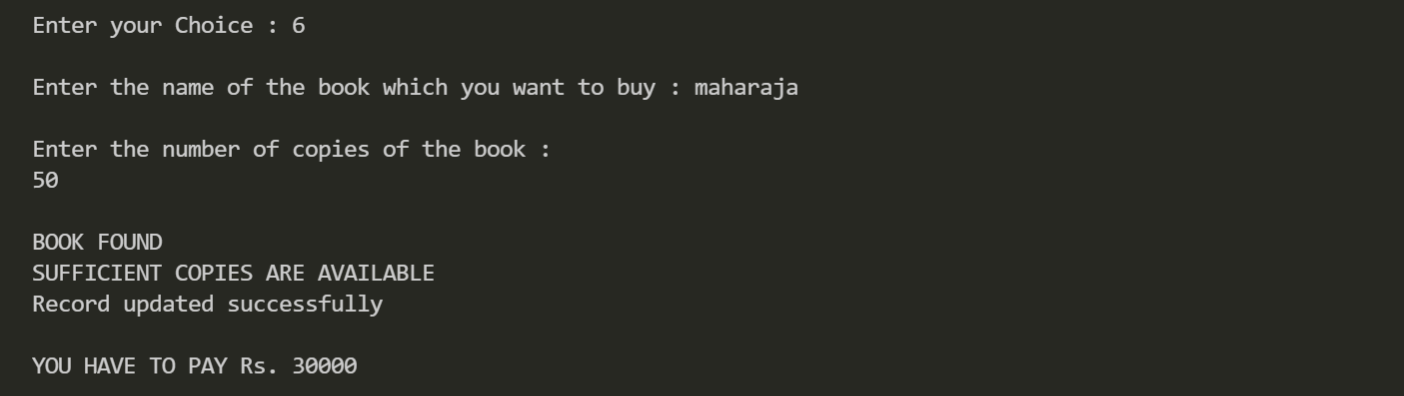
Search using book ID:



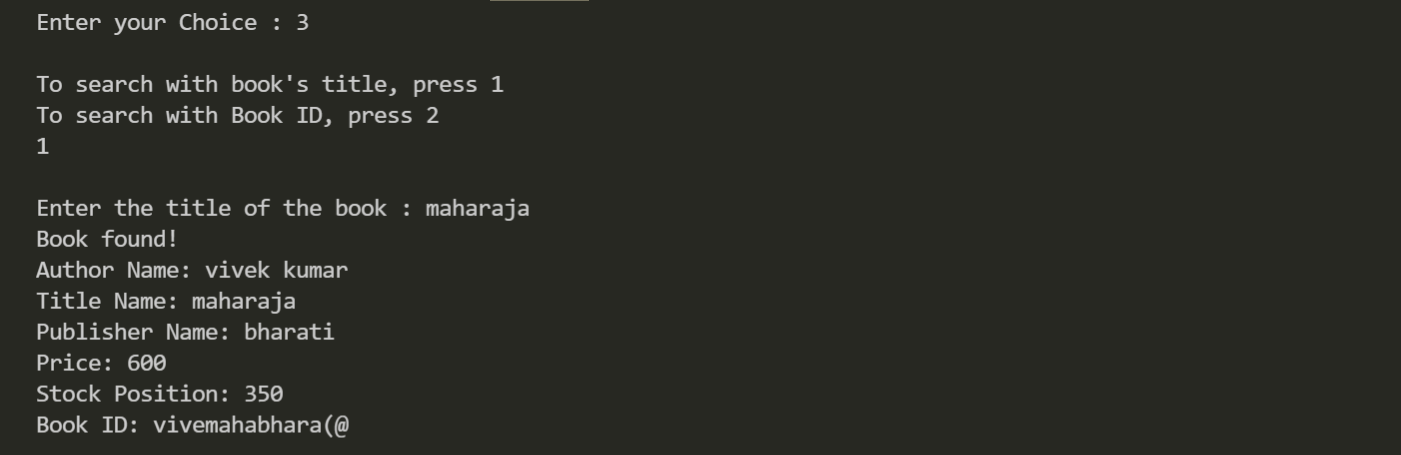
When the user enters 5, he can modify the data of a book:



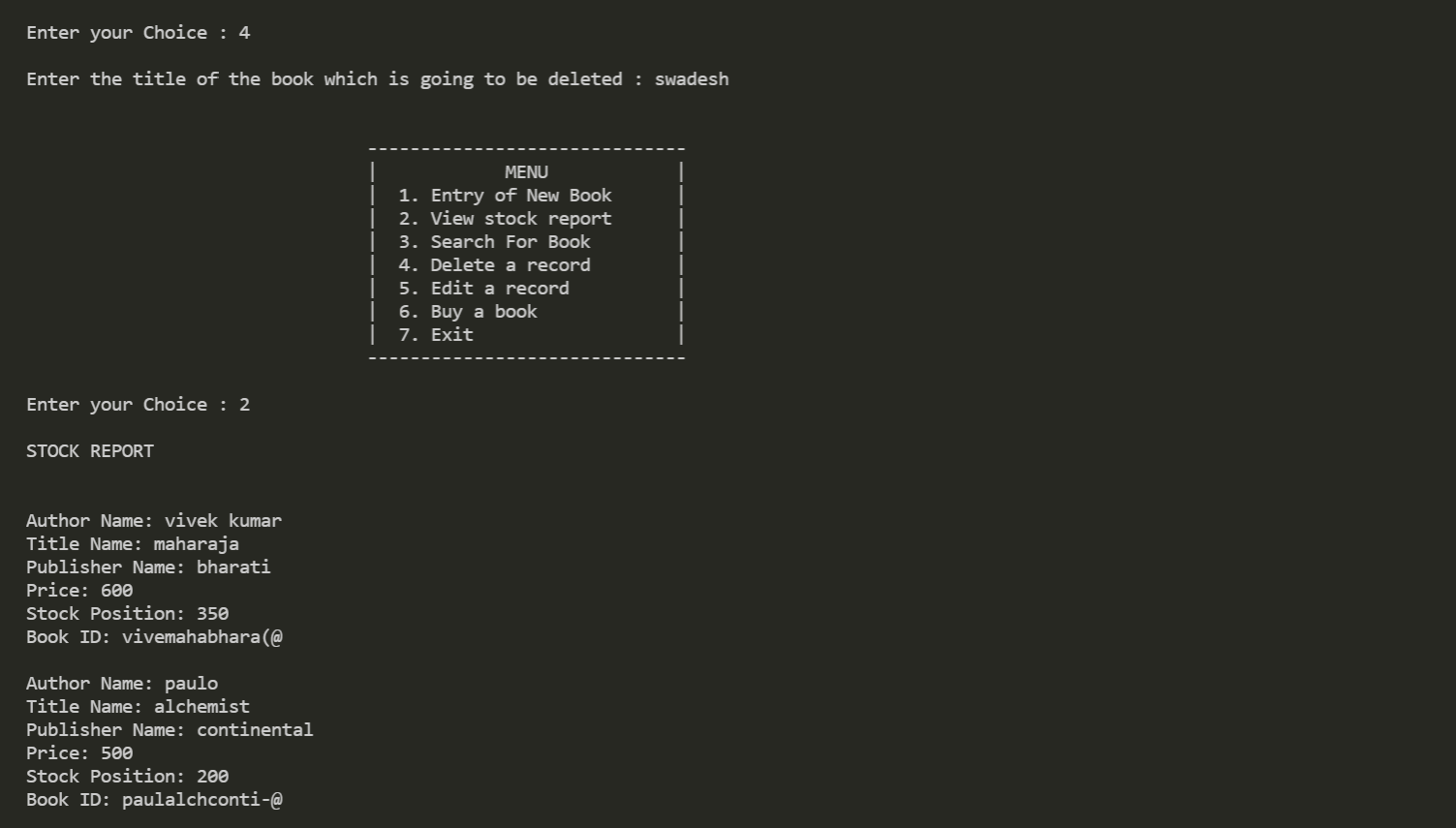
When the user enters 6, he can buy book:

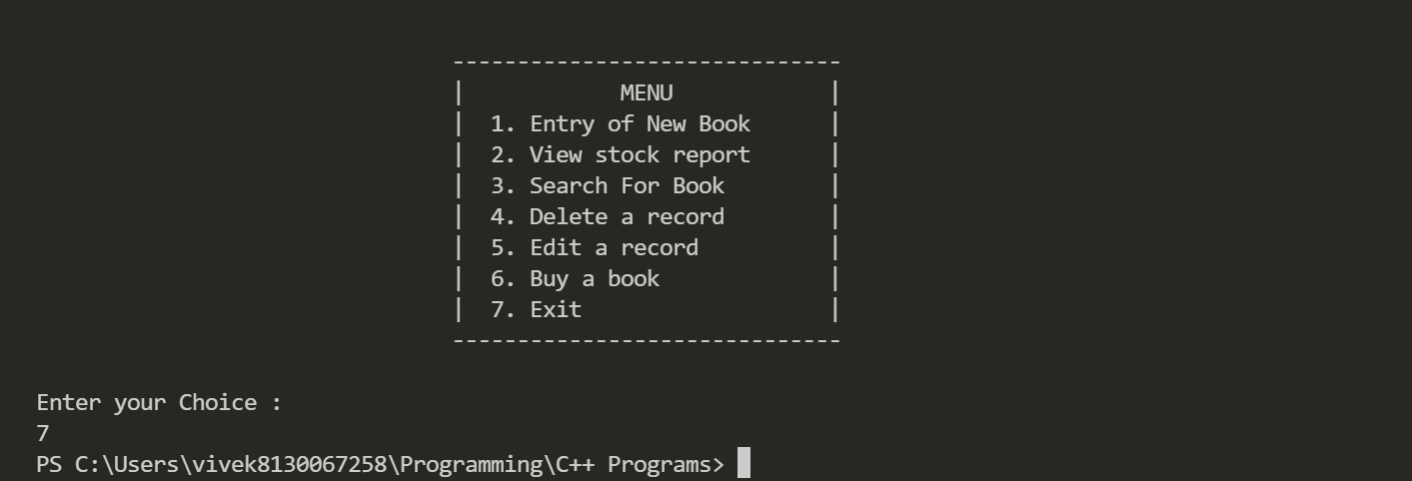


In the following screenshot it can be seen that record has been modified(due to command 5) and number of the books in stock has been decremented(due to sale in command 6):



When the user enters 4, he can delete a record, in the following screenshot, it can be seen that the record of ‘swadesh’ has been deleted:



When the user enters 7, he can exit the program:

**8. MODULES**

1. BOOK DETAILS MODULE : -

In this section we just simply enter the details of the book required by the customer and search for the available outcome like book Author , Title , Publisher , Book\_Id , availability of stock with the help of some of the functions given below .

1. void feeddata(); This function is used to enter data.
2. void showdata(); :- This function is used to display data.
3. void generatebookID(); :- This function is used to generate book Identity.
4. int storebook(); :- This function is used to to store books.
5. void stock\_report(); :- This function is used to check the availability of stock .
6. void search\_title(char \*); :- This function is used to search the Title of the book.
7. void search\_bookID(char \*); :- This function is used to search for the book Id .
8. void delete\_book(char \*); :- This function is used to delete the book from the available stock.
9. void update\_data(char \*); :- This function is used to update the data.
10. void update\_stocks(int); :- This function is used to update the stock availability .
11. void buy\_book(int, char \*); :- This function is used so that if the details match, the customer is able to buy the book.