Library System

The program consists of the following classes:

1) Book – for store information about a some book;
2) LibStore – for store and manipulate an array of books;
3) IoLibrary – has input/output methods for communicate with user.

Assigning all the methods easy to understand based on their names.
Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which may contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known as methods.
OOP based on polymorphism, encapsulation and inheritance.

Examples:

1.Encapsulation

If a class disallows calling code from accessing internal object data and forces access through methods only, this is a strong form of abstraction or information hiding known as encapsulation. In our code for example it is hidden fields in Book class and "getters" and "setters" methods to allow access to them.

```
class Book
{
        string title;
        string author;
        int year;
        bool in_library; // hidden fields
. . .
        void setTitle(string s_title); // “setters”
        void setAuthor(string s_author);
        void setYear(int s_year);
        void setInLib(bool in_lib);

        string getTitle();// “getters”
        string getAuthor();
        int getYear();
        bool getInLib();
};
```

2.Inheritance

This allows classes to be arranged in a hierarchy that represents "is-a-type-of" relationships.
In our code, class IoLibrary publicly inherits class LibStore. IoLibrary has access to class fields LibStore with access modifier: protected and to public methods.

```
class LibStore
{
protected:
        vector<Book> books;
. . .
};

class IoLibrary: public LibStore
{. . . };
```

3. Polymorphism

Example of polymorphism can be a overloaded constructor of Book class.

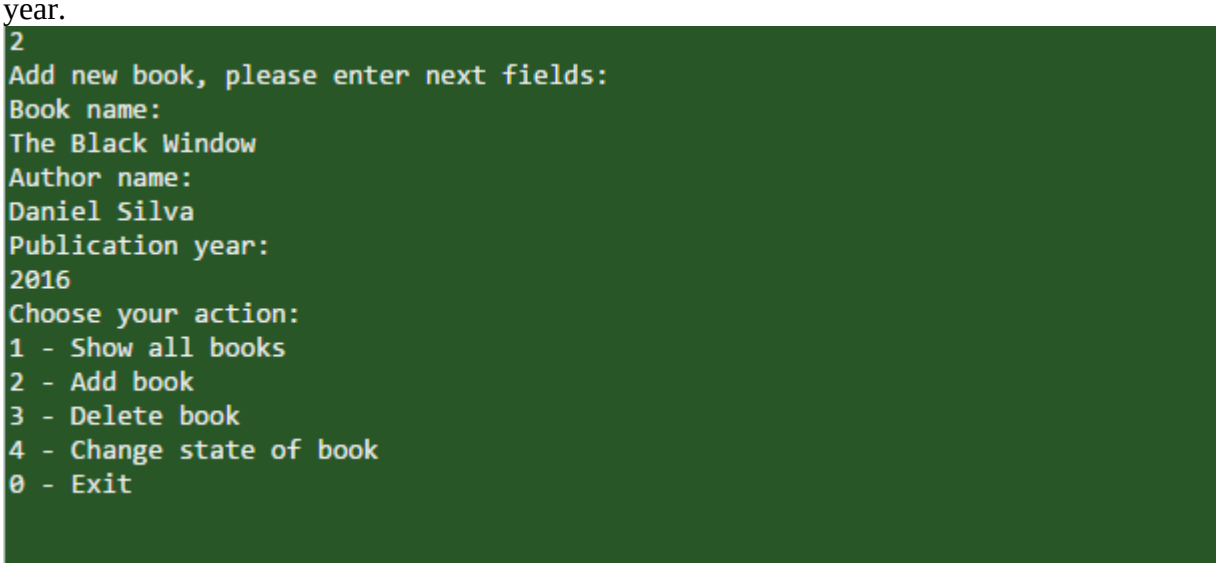
```
class Book
{
        . . .

public:
        Book();
        Book(string s_title, string s_author, int s_year, bool in_lib);
        . . .
};
```

The work of program

t the beginning, the program shows the menu for modeling the library. The menu is as follows.

For add new book, you should type 2 and then write book name, author name and publication year.



Type 1 for show all book in library.

```
Books list:
    Number           Book name         Author name    Publication year              State
         1    The Black Window        Daniel Silva                2016         In library
         2           After tou          Jojo Moyes                2016         In library
Choose your action:
1 - Show all books
2 - Add book
3 - Delete book
4 - Change state of book
0 - Exit
```

For delete book you should type a number of book, which you want to delete.
And for changing book states type number of book and its state.

main.c

```
#include <iostream>
#include "IoLibrary.h"

using namespace std;

int main()
{
	IoLibrary* lib = new IoLibrary();
	lib->run();

	return 0;
}
```

book.h

```
 #pragma once
#include <string>

using namespace std;

class Book
{
	string title;
	string author;
	int year;
	bool in_library; //state which show where is book: in library or in use by readers

public:
	Book();
	Book(string s_title, string s_author, int s_year, bool in_lib);
	~Book();

	void setTitle(string s_title);
	void setAuthor(string s_author);
	void setYear(int s_year);
	void setInLib(bool in_lib);

	string getTitle();
	string getAuthor();
	int getYear();
	bool getInLib();

};
```

book.cpp

```
#include "book.h"

Book::Book()
{
	year = 0;
	in_library = false;
}
Book::Book(string s_title, string s_author, int s_year, bool in_lib)
{
	title = s_title;
	author = s_author;
	year = s_year;
	in_library = in_lib;
}
Book::~Book()
{

}

void Book::setTitle(string s_title)
{
	title = s_title;
}
void Book::setAuthor(string s_author)
{
	author = s_author;
}
void Book::setYear(int s_year)
{
	year = s_year;
}

void Book::setInLib(bool in_lib)
{
	in_library = in_lib;
}

string Book::getTitle()
{
	return title;
}
string Book::getAuthor()
{
	return author;
}
int Book::getYear()
{
	return year;
}
bool Book::getInLib()
{
```

```
        return in_library;
    }
```

IoLibrary.h

```
#pragma once
#include <iostream>
#include <iomanip>
#include "LibStore.h"

class IoLibrary: public LibStore
{

public:
	IoLibrary();

	void run();

	void showAll();
	void addBookIn();
	void deleteBookIn();
	void changeState();
};
```

IoLibrary.cpp

```
#include "IoLibrary.h"

IoLibrary::IoLibrary():LibStore()
{
	cout<<"Library modulation system..."<<endl;
}

void IoLibrary::run()
{
	int act;
	do
	{
		cout<<"Choose your action:"<<endl;
		cout<<"1 - Show all books"<<endl;
		cout<<"2 - Add book"<<endl;
		cout<<"3 - Delete book"<<endl;
		cout<<"4 - Change state of book"<<endl;
		cout<<"0 - Exit"<<endl;

		cin>>act;
		switch(act)
		{
		case 0:
			exit(0);
			break;
		case 1:
			showAll();
			break;
		case 2:
			addBookIn();
			break;
		case 3:
			deleteBookIn();
			break;
		case 4:
			changeState();
			break;
		}


	}while(act != 0);
}

void IoLibrary::showAll()
{
	cout<<"Books list:"<<endl;
	if(books.empty())
	{
	   cout<<"list empty"<<endl;
	   return;
	}
```

```
	cout<<setw(10)<<"Number"<<setw(20)<<"Book name"<<setw(20)<<"Author
name"<<setw(20)<<"Publication year"<<setw(20)<<"State"<<endl;
	for(int i=0;i<books.size();i++)
	{

	cout<<setw(10)<<i+1<<setw(20)<<books[i].getTitle()<<setw(20)<<books[i].getAuthor()
<<setw(20)<<books[i].getYear();
		if(books[i].getInLib())
		{
			cout<<setw(20)<<"In library"<<endl;
		}else
		{
			cout<<setw(10)<<"In use"<<endl;
		}
	}
}
void IoLibrary::addBookIn()
{
	string name,author;
	int year;
	cout<<"Add new book, please enter next fields:"<<endl;
	cin.ignore();
	cout<<"Book name:"<<endl;
	getline(cin,name);
	cout<<"Author name:"<<endl;
	getline(cin,author);
	cout<<"Publication year:"<<endl;
	cin>>year;
	addBook(name,author,year);
}
void IoLibrary::deleteBookIn()
{
	showAll();
	cout<<"Write number of book for delete:"<<endl;
	int num;
	cin>>num;
	deleteBook(num-1);
}
void IoLibrary::changeState()
{
	showAll();
	cout<<"Write number of book for edit state:"<<endl;
	int num, state;
	cin>>num;
	cout<<"Write 1 - if book in library, 0 - if book in use"<<endl;
	cin>>state;
	setBookState(num-1,state==1);

}
```

LibStore.h

```
#pragma once
#include <vector>
#include "book.h"

class LibStore
{
protected:
	vector<Book> books;

public:

	LibStore();
	~LibStore();

	void addBook(Book b);
	void addBook(string s_title, string s_author, int s_year);

	void deleteBook(int n);

	void setBookState(int n, bool in_lib);


};
```

LibStore.cpp

```
#include "LibStore.h"

LibStore::LibStore()
{

}
LibStore::~LibStore()
{

}

void LibStore::addBook(Book b)
{
	books.push_back(b);
}
void LibStore::addBook(string s_title, string s_author, int s_year)
{
	Book b(s_title,s_author,s_year,true);
	books.push_back(b);
}

void LibStore::deleteBook(int n)
{
	if(n >= 0 && n < books.size())
		books.erase(books.begin()+n);
}

void LibStore::setBookState(int n, bool in_lib)
{
	if(n >= 0 && n < books.size())
		books[n].setInLib(in_lib);
}
```