Authors

- Califf McBride YYG532

- CPSC 5010

- 4-29-2023

##BookStore Documentation

Bookstore is a basic program for managing a university type bookstore. It includes features to create and record information about books/textbooks. There is also functionality to maintain an inventory list of all the books in the shop. Included is an account with the ability to update the bookstore balance as books are sold.

Getting Started/Adding Your First Book

When you run the program in your editor, an account an inventory list will be created for you. It is recommended that you add one or more books before exploring other features of the program. To add a book

1. Run the program and select 1 to add a book.

2. From there you must enter if it is a traditional fiction/nonfiction book or a textbook

3. You will be prompted to type in a title, author, enter a price and ISBN identification number. It is very important that each title has a unique ID number. This is the most reliable way to perform search functions in the system. If a duplicate id number is entered, the existing one will be overridden by the latest book.

Features

Once you have added one or more books, the other features can be used.

Search for a book-You can search for a book by title or by ISBN. Note searching by ISBN is more reliable. Search by title IS CASE SENSITIVE

*Buy a book-You can buy a book using the title or ISBN. Makes changes and updates the stores bank account and stock numbers accordingly.

*Display Inventory-Displays the linked list of the book store's Inventory

*Bookstore Stats-Uses linked list to display either the most expensive, cheapest, or average book cost

*View Account Balance-Shows the bookstores balance

Important Function Descriptions

class InventoryList

The InventoryList is implemented using linked list. Each node in the list contains a pointer to a book object. This approach was used instead of storing book objects directly so that virtual functions and inheritance can be used. Because of this, InventoryList is a friend class to node, so that it can access the information pointed to by the book pointers in the node class.

InventoryList()

//Default constructor for list, head/tail pointers set to null

InventoryList::addBook(Node* headPointer)

//You pass in the current headpointer of the linked list and a new node is added

InventoryList::findBookTitle(string bookToFind)

InventoryList::findBookTitle(inte bookToFind)

//Overloaded definition, finds book based on the parameter

double InventoryList::findBookISBNSell(int bootToFind)

//Enter book ISBN, returns the price of the book

InventoryList::removesStock(double ISBN)

//Removes 1 book stock unit

InventoryList::printList()

//Traverses linked list pointer by pointer and outputs each books information

class Account

Mains bookstore balance and has static variable taxrate

double transaction(double price)

//Takes in a book price, adds tax and makes change based on the amount tendered. It returns change if the transaction is valid, returns -1 if not enough money is given.

class Book

Has data members title, fullName, Publication date, price, numberInStock and ISBN. Also has been implemented with all of the getters/setters.

Book()

//Default constructor prompts the user to enter all of the information

Book(string title, string fullName, int numberInStock, int ISBN, double price)

//Parameterized version of the default constructor

virtual void output()

//Virtual function used to output the correct information for either a book or textbook which is a child class of book.

class Textbook

Child class of book, however it adds private member variables course and subject

##Source Code

```
// Califf McBride YYG532
// CPSC 5010 Final Project: Bookstore

#include <iostream>
#include "Book.h"
#include "InventoryList.h"
#include "Account.h"

/*Main user interface for book store. Uses switch statements to pick between 6 options*/
int main()
{
    bool exit = false;
    bool statsExit = false;
    InventoryList newList;
    Account newAccount;

    while (!exit)
    {
        cout << "Please select the desired option: " << endl;
        cout << "1 - Add a book " << endl;
        cout << "2 - Search for a Book " << endl;
        cout << "3 - Buy a Book " << endl;
        cout << "4 - Display Inventory" << endl;
        cout << "5 - Bookstore Stats" << endl;
        cout << "6 - View Store Bank Account Balance" << endl;
        cout << "0 - Exit" << endl;

        int choice; //What do they pick?
        cin >> choice;

        switch (choice)
        {
            //Adds a book to the linked list at the head
            case 1:
            {
                cout << "Add a book " << endl;
                newList.addBook(newList.getHeadPtr());
                break;
            }
            //Searches the linked list to see if the entered book is contained within
            case 2:
            {
                cout << "1- Search by title " << endl;
                cout << "2- Search by ISBN " << endl;
                bool validChoice = false;
                int searchChoice;

                while (!validChoice)
                {
                    cin >> searchChoice;

                    if (searchChoice == 1 || searchChoice == 2)
                    {
                        validChoice = true;
                        break;
                    }
```

```
                    else
                    {
                        cout << "Invalid Choice....try again";
                    }
                }

                if (searchChoice == 1)
                {
                    cout << "Enter the title of the book you would like to search: " <<
endl;
                    string bookToSearch;
                    cin >> bookToSearch;
                    newList.findBookTitle(bookToSearch);
                    break;
                }
                else if (searchChoice == 2)
                {
                    cout << "Enter the ISBN of the book you would like to find: " <<
endl;
                    int bookToSearch;
                    cin >> bookToSearch;
                    newList.findBookISBN(bookToSearch);
                    break;
                }
            }
            //Buy a book
            case 3:
            {
                int bookToFind;
                cout << "Enter the ISBN of the book: " << endl;
                cin >> bookToFind;

                //Get the price of the book
                double bookPrice = newList.findBookISBNSell(bookToFind);

                if (bookPrice == -1)
                {
                    cout << "Sorry we don't have any of that book" << endl;
                    break;
                }
                else
                {
                    double change = newAccount.transaction(bookPrice); //Account class
handles transaction

                    if (change == -1)
                    {
                        cout << "That is not enough money....." << endl;
                        break;
                    }
                    else
                    {
                        cout << "Change: " << change << endl;
                        newList.removeStock(bookToFind); //Removes 1 Book
                        newAccount.addFunds(bookPrice); //Add money to the bank account
                        break;
                    }
```

```
            }
        }
        //Prints each node in the linked list to display inventory
        case 4:
        {
            newList.printList();
            break;
        }
        //Different stats can be viewed, mainly most expensive, cheapest and average
book cost for the store
        case 5:
        {
            while (!statsExit)
            {
                cout << "What stat would you like to see? " << endl;
                cout << "1-Most Expensive Book: " << endl;
                cout << "2-Cheapest Book: " << endl;
                cout << "3-Average Cost: " << endl;
                cout << "0-Exit";
                int statsChoice;
                cin >> statsChoice;

                switch (statsChoice)
                {
                    case 1:
                    {
                        newList.mostExpensive();
                        break;
                    }

                    case 2:
                    {
                        newList.leastExpensive();
                        break;
                    }

                    case 3:
                    {
                        newList.averageCost();
                        break;
                    }
                    case 0:
                    {
                        cout << "Exiting to main menu....." << endl;
                        statsExit = true;
                        break;
                    }
                }
            }
            break; //Return control back to the main menu
        }
        //Displays account balance
        case 6:
        {
            newAccount.output();
            break;
        }
        //Exit the list
```

```
            case 0:
            {
                cout << "Thank you for using Bookstore Manager 2023. Have a nice day!";
                exit = true;
                break;
            }
        }
    }
}
```

```
#pragma once

/*
* This class handles the stores funds. Contains balance and static variable taxRate
* Contains function for buying a book, adding funds and getting the balance
*/
class Account
{
private:
	double balance;
	static double taxRate; //TN tax rate 7%
public:
	Account();
	//Default Constructor

	Account(double balance);
	//Start with a specific balance

	double getBalance();
	//getter for balance

	void addFunds(double funds);
	//Update balance after transaction

	double transaction(double price);
	//Performs transaction: returns change and updates balance and stock value

	void output();

	~Account();
};
```

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

using namespace std;

double Account::taxRate = 0.07; //TN tax rate

//Default constructor
Account::Account()
{
	this->balance = 0;
}
```

```cpp
//Parameterized constructor with starting balance as an argument
Account::Account(double balance)
{
	this->balance = balance;
}

//Getter for balance
double Account::getBalance()
{
	return balance;
}

//Add funds to the account
void Account::addFunds(double funds)
{
	balance = balance + funds;
}

/*
* Function for conducting a transaction. After the price is returned from inventory list
* It is passed in as an argument. Uses the tax rate to determine the final sale value.
* Also, it makes change based on the amount tendered. If enough money is not tendered,
the
* function returns -1
*/
double Account::transaction(double price)
{
	cout << "Price before tax: " << price << endl;
	double tax = price * taxRate;
	double totalPrice = price + tax;

	cout << "Total price: " << totalPrice << endl;
	double amountTendered;
	cout << "Enter the amount tendered: " << endl;
	cin >> amountTendered;

	if (amountTendered >= totalPrice)
	{
		double change = amountTendered - totalPrice;
		return change;
	}
	else
	{
		return -1;
	}
}

Account::~Account() {}; //Unimplemented Destructor

void Account::output()
{
	cout << "Balance: " << balance << endl;
}
```

```cpp
#pragma once
#include <iostream>
#include "Date.h"

using namespace std;

/*Main book class, used as the base class for textbook. */

class Book
{
private:
	string title;
	string fullName;
	Date* publicationDate;
	double price;
	bool inStock = false;
	int numberInStock = 0;
	int ISBN;
public:
	Book();
	//Default constructor, prompts user for data

	Book(string title, string fullName, int numberInStock, int ISBN, double price);
	//Parameterized constructor

	string getTitle();
	//Getter for title

	string getAuthor();
	//Getter for author

	int getISBN();
	//Getter for ISBN

	int getNumberInStock();
	//Getter for number in stock

	double getPrice();
	//Getter for book price

	void setTitle(string title);
	//Setter for title

	void setAuthor(string name);
	//Setter for author

	void setISBN(int ISBN);
	//Setter for ISBN

	bool isInStock();
	//Returns true of the book is instock

	void addStock(int stockAmount);
	//Adds stock based on the argument

	void subtractStock(int stockAmount);
	//Removes specified amount
```

```
    virtual void output();
       //Outputs information based on the class that is used

       ~Book();
};



#include "Book.h"
#include <iostream>
#include <string>

using namespace std;

/*Base class book: Class stores basic information common to all books.*/

/*Default constructor builds a book object prompt by prompt*/
Book::Book()
{
       cout << "Enter book title: ";
       cin.ignore(); //clear the input stream from \n
       getline(cin, title);

       cout << "Enter the author in the format (last name, first name): ";
       getline(cin, fullName);

       cout << "How many are in stock? "; cin >> numberInStock;
       cout << "What is the ISBN for " << title << "?: "; cin >> ISBN;
       cout << "Price: "; cin >> price;
       publicationDate = new Date();
}

Book::Book(string title, string fullName, int numberInStock, int ISBN, double price)
{
       this->title = title;
       this->fullName = fullName;
       this->numberInStock = numberInStock;
       this->ISBN = ISBN;
       this->price = price;
       inStock = true;
       publicationDate = new Date();
}

/*Getters and Setters Boiler Plate to access member variables*/
string Book::getTitle()
{
       return title;
}

string Book::getAuthor()
{
       return fullName;
}

int Book::getNumberInStock()
{
       return numberInStock;
}
```

```
int Book::getISBN()
{
	return ISBN;
}

double Book::getPrice()
{
	return price;
}

void Book::setTitle(string title)
{
	this->title = title;
}

void Book::setAuthor(string name)
{
	this->fullName = name;
}

void Book::setISBN(int ISBN)
{
	this->ISBN = ISBN;
}

bool Book::isInStock()
{
	return inStock;
}

void Book::addStock(int stockAmount)
{
	numberInStock = numberInStock + stockAmount;
}

//Pass stock amount by reference
void Book::subtractStock(int stockAmount)
{
	numberInStock = numberInStock - stockAmount;
}

void Book::output()
{
	cout << "Book Title: " << title << endl;
	cout << "Author: " << fullName << endl;
	cout << "Number in Stock: " << numberInStock << endl;
	cout << "ISBN: " << ISBN << endl;
	cout << "Price: " << price << endl;
	publicationDate->output();
	cout << endl;
}

Book::~Book() {}; //Unimplemented Distructor
```

```h
#pragma once
#include <iostream>
using namespace std;

//Basic date class containing year, month and day
class Date
{
private:
    int year;
    int month;
    int day;
public:
    Date();
    //Destructor: Not implemented
    ~Date();
    //Output date info
    void output();
};
```

```cpp
#include "Date.h"
using namespace std;

Date::Date()
{
    cout << "Enter the year the book was published: "; cin >> year;
    cout << "Enter the month the book was published :" ; cin >> month;
    cout << "Enter the day the book was published: "; cin >> day;
    cout << endl;
}//Constructor

Date::~Date() {};//Unimplemented Destructor

void Date::output()
{
    cout << "Published Date: " << month << "/" << day << "/" << year;
}
```

```h
#pragma once
#include "Node.h"
#include "Book.h"
#include "TextBook.h"
#include "Account.h"

class InventoryList
{
private:
    Node* head;
    Node* tail;
    int numberOfBooks;
public:
    InventoryList();

    ~InventoryList();
    //Unimplement Destructor

    void addBook(Node* headPointer);
    //Adds a book to the linked list inventory

    void mostExpensive();
    //Searches the List of the most expensive book

    void leastExpensive();
    //Displays Least Expensive title

    void averageCost();
    //Creates an Average cost for all items in the inventory

    void printSpacers();
    //Used in formatting Output

    Node* getHeadPtr();
    //void removeBook();
    //Removes Book from the list

    void findBookTitle(string bookToFind);
    //Finds a book base on title.

    void findBookISBN(int bookToFind);
    //Finds a book using the ISBN

    double findBookISBNSell(int bookToFind);

    void removeStock(double ISBN);
    //Removes 1 item for the stock

    virtual void printList();
    //Virtual Function Prints the appropriate class output
};


#include "InventoryList.h"
#include <string>

using namespace std;
```

```
/*Class to handle and display the book store's inventory*/
InventoryList::InventoryList()
{
    //Create head and tail pointer
    this->head = nullptr;
    this->tail = nullptr;
    numberOfBooks = 0;
}

/*
* Invetory list is the main function for adding books to the linked list. It takes a head
pointer a parameter
* If the head pointer is a nullptr, the linked list is established
* Otherwise, the book is added to the front of the list
* It is important to note, objects can't be stored directly in this linked list because
Book and Textbook
* are different sizes.
*/
void InventoryList::addBook(Node* headPointer)
{
    int bookType;
    cout << "1-Enter Fiction or Non-Fiction Book" << endl;
    cout << "2-Enter Textbook" << endl;
    cin >> bookType;

    if (bookType == 1)
    {
        string title, fullName;
        double price;
        int numberInStock, ISBN;
        cout << "Enter book title: ";
        cin.ignore(); //clear the input stream from \n
        getline(cin, title);

        cout << "Enter the author in the format (last name, first name): ";
        getline(cin, fullName);

        cout << "How many are in stock? "; cin >> numberInStock;
        cout << "What is the ISBN for " << title << "?: "; cin >> ISBN;
        cout << "Price: "; cin >> price;

        Book* newBook = new Book(title, fullName, numberInStock, ISBN, price);

        //Create new node and put in data
        Node* bookNode = new Node(*newBook);

        //bookNode->setNext(headPointer);
        bookNode->setPrevious(nullptr);

        //Set the previous head pointer so that it points to the new node in the back
        if (headPointer == nullptr)
        {
            head = bookNode;
        }
        else
        {
            bookNode->setNext(headPointer);
            headPointer->setPrevious(bookNode);
```

```
            head = bookNode;
        }

        numberOfBooks++;
    }
    else if (bookType == 2)
    {
        string title, fullName, subject, course;
        int numberInStock, ISBN;
        double price;

        cout << "Enter book title: ";
        cin.ignore(); //clear the input stream from \n
        getline(cin, title);

        cout << "Enter the author in the format (last name, first name): ";
        getline(cin, fullName);

        cout << "How many are in stock? "; cin >> numberInStock;
        cout << "What is the ISBN for " << title << "?: "; cin >> ISBN;
        cout << "What subject? ";
        cin.ignore();
        getline(cin, subject);

        cout << "Course: ";
        getline(cin, course);

        cout << "Price: ";
        cin >> price;

        Book* newBook = new TextBook(title, fullName, numberInStock, ISBN, subject,
course, price);

        //Create new node and put in data
        Node* bookNode = new Node(*newBook);

        //bookNode->setNext(headPointer);
        bookNode->setPrevious(nullptr);

        //Set the previous head pointer so that it points to the new node in the back
        if (headPointer == nullptr)
        {
            head = bookNode;
        }
        else
        {
            bookNode->setNext(headPointer);
            headPointer->setPrevious(bookNode);
            head = bookNode;
        }

        numberOfBooks++;
    }
}

//Finds a book based on a parameter and determines if the book is instock
void InventoryList::findBookTitle(string bookToFind)
{
```

```
    Node* current = head;
    bool isFound = false;

    if (head == nullptr)
    {
        cout << "There are no books in the list. This operation is impossible...Add some
books first.";
    }
    else
    {
        while (current != nullptr)
        {
            if (current->information->getTitle() == bookToFind)
            {
                cout << "We have " << bookToFind << endl;
                cout << "There are " << current->information->getNumberInStock() << "in
stock. " << endl;
                isFound = true;
                break;
            }
            current = current->getNext();
        }
    }

    if (!isFound)
    {
        cout << "Sorry, it looks like we do not have the book: " << endl;
    }
}

//Uses the ISBN of a book to determine if the book is instock. Displays the title and the
number in stock
void InventoryList::findBookISBN(int bookToFind)
{
    Node* current = head;
    bool isFound = false;

    if (head == nullptr)
    {
        cout << "There are no books in the list. This operation is impossible...Add some
books first.";
    }
    else
    {
        while (current != nullptr)
        {
            if (current->information->getISBN() == bookToFind)
            {
                cout << "We have " << current->information->getTitle() << endl;
                cout << "There are " << current->information->getNumberInStock() << "in
stock. " << endl;
                isFound = true;
                break;
            }
            current = current->getNext();
        }
    }
```

```
    if (!isFound)
    {
        cout << "Sorry, it looks like we do not have the book: " << endl;
    }
}

//Used to buy a book.If the book is instock it the funtion returns the price. If it isnt
-1 is returned
double InventoryList::findBookISBNSell (int bookToFind)
{
    Node* current = head;
    bool isFound = false;

    if (head == nullptr)
    {
        cout << "There are no books in the list. This operation is impossible...Add some
books first.";
    }
    else
    {
        while (current != nullptr)
        {
            if (current->information->getISBN() == bookToFind)
            {
                isFound = true;
                return current->information->getPrice();
                break;
            }
            current = current->getNext();
        }
    }

    if (!isFound)
    {
        //cout << "Sorry, it looks like we do not have the book: " << endl;
        return -1;
    }
}

//Removes 1 book from stock after a sale
void InventoryList::removeStock(double ISBN)
{
    Node* current = head;

    while (current != nullptr)
    {
        if (current->information->getISBN() == ISBN)
        {
            current->information-> subtractStock(1);
            break;
        }
        current = current->getNext();
    }
}

//Getter for head pointer
Node* InventoryList::getHeadPtr()
{
```

```
    return head;
}

/*
* Traverses the linked list and uses the gladiator strategy to find the most expensive
book
*/

void InventoryList::mostExpensive()
{
    Node* current = head;
    if (head == nullptr)
    {
        cout << "There are no books in the list. This operation is impossible...Add some
books first.";
    }
    else
    {
        double maxValue = current->information->getPrice();
        string maxTitle = current->information->getTitle();

        while (current != nullptr)
        {
            if (current->information->getPrice() > maxValue)
            {
                maxValue = current->information->getPrice();
                maxTitle = current->information->getTitle();
            }
            current = current->getNext();
        }
        printSpacers();
        cout << "The most expensive book is: " << maxTitle << endl;
        cout << "It costs: " << maxValue << endl;
        printSpacers();
    }
}

/*
* Traverses the linked listand finds the least expensive book.Uses the logic of the
gladiator strategy
* in reverse.
*/
void InventoryList::leastExpensive()
{
    Node* current = head;
    if (head == nullptr)
    {
        cout << "There are no books in the list. This operation is impossible";
    }
    else
    {
        double minValue = current->information->getPrice();
        string minTitle = current->information->getTitle();

        while (current != nullptr)
        {
            if (current->information->getPrice() < minValue)
            {
```

```
                minValue = current->information->getPrice();
                minTitle = current->information->getTitle();
            }
            current = current->getNext();
        }
        printSpacers();
        cout << "The cheapest book is: " << minTitle << endl;
        cout << "It costs: " << minValue << endl;
        printSpacers();
    }
}

void InventoryList::averageCost()
{
    Node* current = head;
    if (head == nullptr)
    {
        cout << "There are no books in the list. This operation is impossible";
    }
    else
    {
        double averageCost = 0;

        while (current != nullptr)
        {
            averageCost += current->information->getPrice();
            current = current->getNext();
        }

        averageCost = averageCost / (double)numberOfBooks;

        printSpacers();
        cout << "The average cost for a book is: " << averageCost << endl;
        printSpacers();
    }
}

void InventoryList::printList()
{
    printSpacers();
    Node* current = head;
    while (current != nullptr)
    {
        current->information->output();
        current = current->getNext();
        printSpacers();
    }
}


void InventoryList::printSpacers()
{
    cout << "----------------------------------------" << endl;
    cout << "----------------------------------------" << endl;
}
InventoryList::~InventoryList() {};
```

```h
#pragma once
#include "Book.h"

/*
* Node contains a pointer to a book object, this is done so to allow for the use of
polymorphism
* Virtual function used for book and textbook rely on the ability to create both book and
textbook variables
* of type Book*
*/
class Node
{
private:
    Book* information; //Node data consists of a pointer to a book
    Node* next;
    Node* previous;
public:
    Node();
    //Default constructor

    Node(Book& bookObjectPtr);
    //Constructs a node with it's data being a pointer to a book object

    Node* getNext();
    //Returns next node pointer

    Node* getPrevious();
    //Gets previous node pointer

    void setNext(Node* next);
    //Set next node

    void setPrevious(Node* previous);
    //set Previous node

    friend class InventoryList;
    //Inventory list is allowed access to Book* information
};
```

```cpp
#include "Node.h"
#include "Book.h"

//Default constructor initializes all pointers to null
Node::Node()
{
    this->information = nullptr;
    this->next = nullptr;
    this->previous = nullptr;
}

//Intilizes a node using book pointer as data
Node::Node(Book& bookObjectPtr)
{
    this->information = &bookObjectPtr;
    this->next = nullptr;
    this->previous = nullptr;
}
```

```
//Gets next pointer
Node* Node::getNext()
{
    return next;
}

//Gets previous pointer
Node* Node::getPrevious()
{
    return previous;
}

//Sets the next pointer
void Node::setNext(Node* next)
{
    this->next = next;
}

//Sets previous pointer
void Node::setPrevious(Node* previous)
{
    this->previous = previous;
}
```

```
#pragma once
#include "Book.h"

/*
*  Textbook is a child class of book with added functionality, it also includes subject
course and price
*  It also utilizes book's virtual function output to output the basic information plus
the textbook specific info
*/

class TextBook :
    public Book
{
private:
    string subject;
    string course;
    double price;
public:
    TextBook(string title, string fullName, int numberInStock, int ISBN, string subject,
string course, double price);
    ~TextBook();
    void output();
};
```

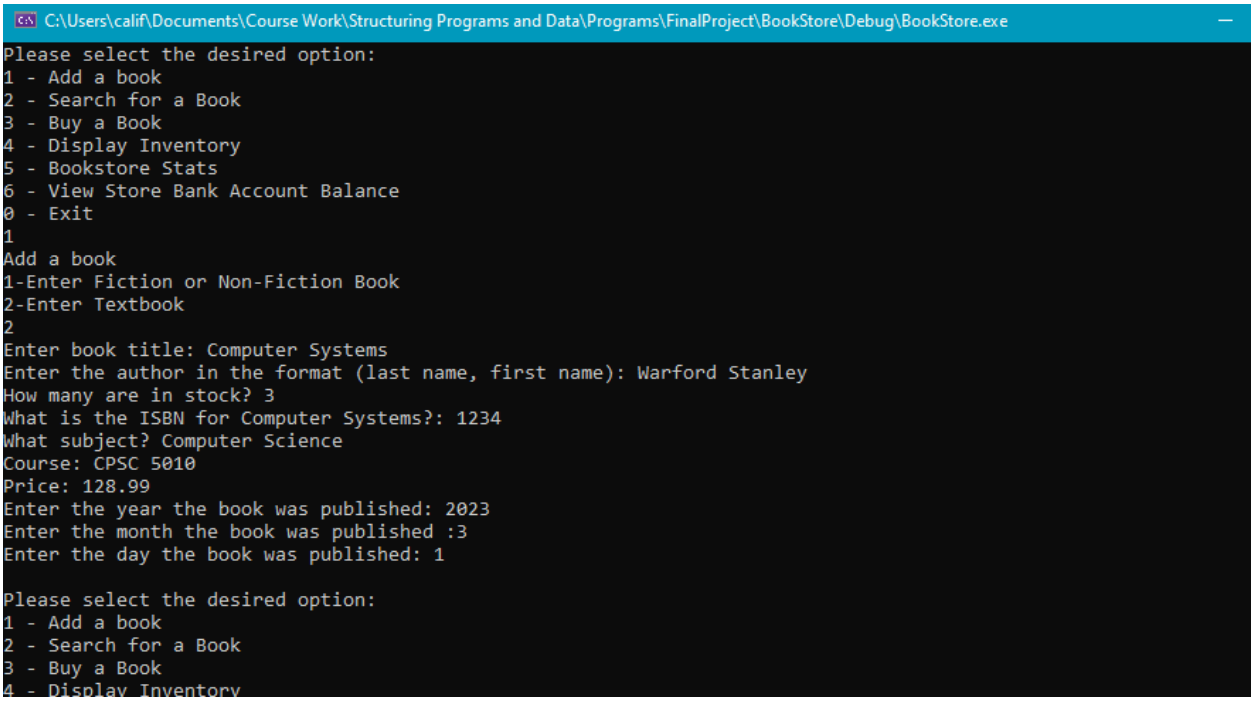
```
#include "TextBook.h"

TextBook::TextBook(string title, string fullName, int numberInStock, int ISBN, string
subject, string course, double price) : Book(title, fullName, numberInStock, ISBN, price)
{
       this->subject = subject;
       this->course = course;
}
```

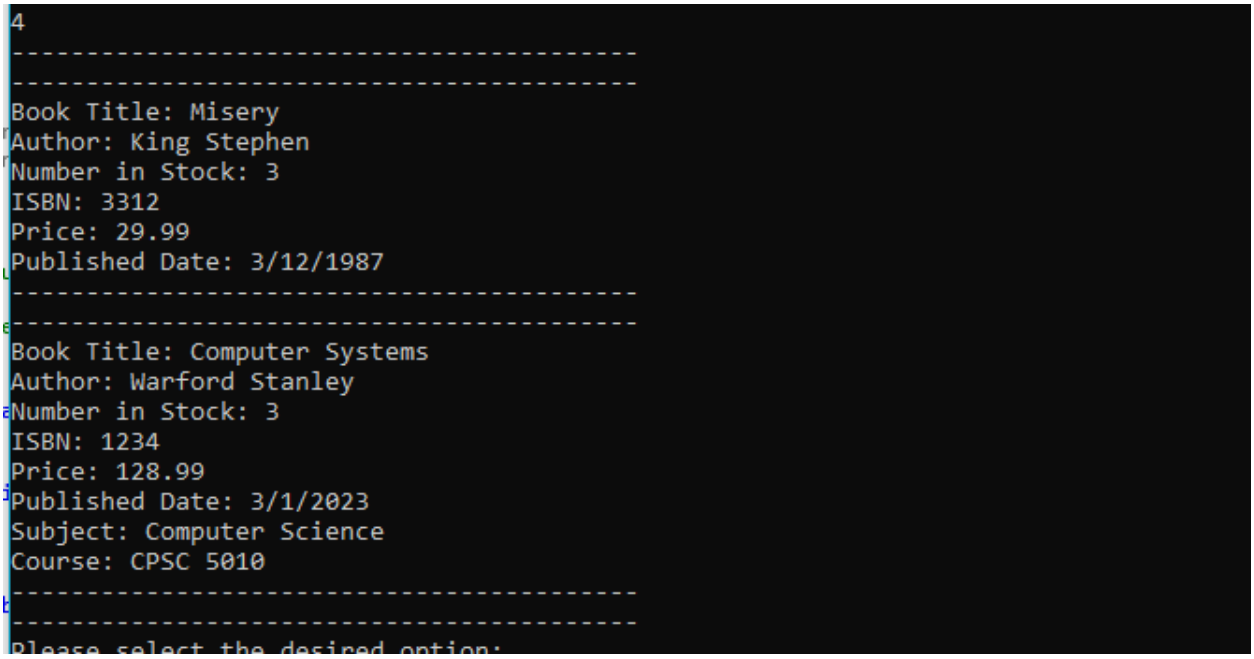
```
void TextBook::output()
{
	Book::output();
	cout << "Subject: " << subject << endl;
	cout << "Course: " << course << endl;
}

TextBook::~TextBook() {};
```

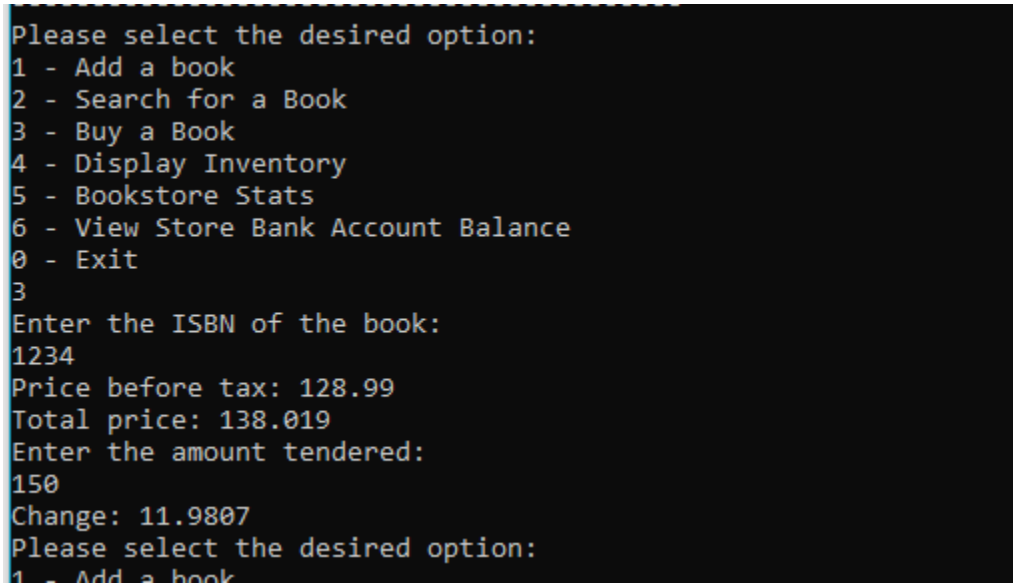
Add a book



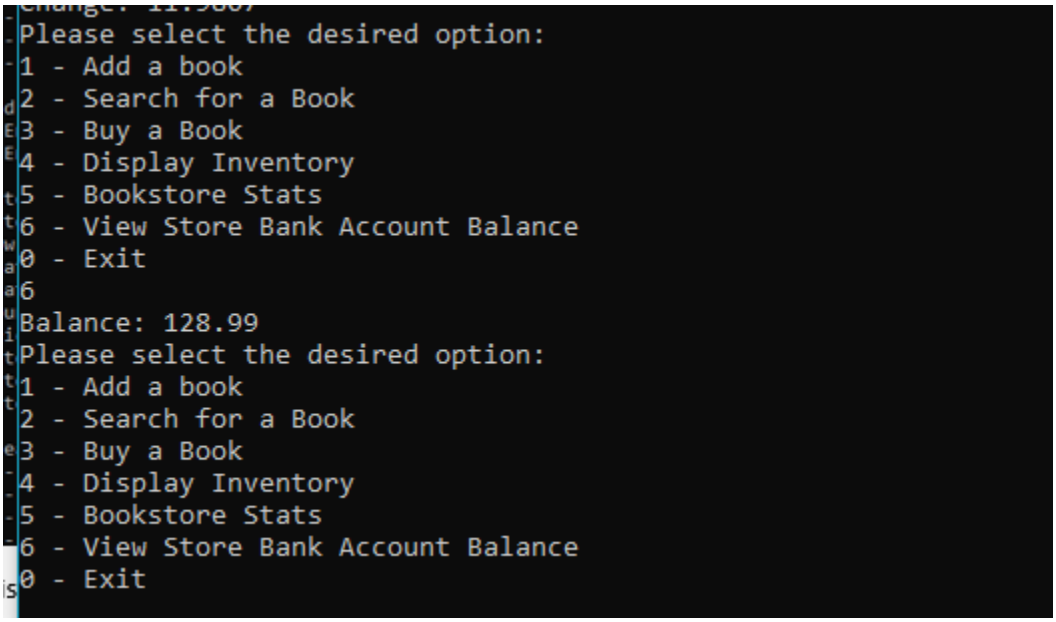
Display Inventory



Buy a book



Display account balance



YouTube Demonstration:

https://youtu.be/FGAd3LoxBMI