Project 1: Borders Books

CPSC 131 Fall 2020

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

### Introduction

Welcome to Borders Books, the bookstore ready to offer the most popular or fashionable books you want to read. We pride in providing books cheaper than anywhere else, new or used. We have books for all ages, and whether it is a New York Times bestseller or a little known gem, if you need it, either we have it, or we will get it for you in 48 hours.

You are provided our database of books in our store called data.csv, and each book has a title, an author or a list of authors, an ISBN (ISBN-10 or ISBN-13), and a price in US dollars. We need a program to create and maintain a collection of books to form a book list. This project is to reinforce the new data structure learnt, fixed-size array, using dynamic memory.

### Objective

The header file, BookList.hpp, contains two classes Book and BookList and a number of ordinary functions.

Book is a class that holds the information for each book, is completely defined and implemented.

BookList is a class that holds an internal listing of many Book objects. This list should hold using a dynamically allocated array at most as many Book objects as specified by the constructor in the private data member \_capacity. A complete BookList class interface and partial implementation have been provided. You are to complete the implementation of BookList. You are to modify only designated TO-DO sections. Do not modify anything outside such designated areas. Implement all functions in the source (.cpp) file, not the header (.hpp) file. Header file only solutions will not be accepted.

Your code is tested in the provided main.cpp.

For the class BookList you will need to implement the following functions:

* **Constructors/Destructors** - Initialize your data. Allocate memory for data member \_bookArray using a dynamically allocated array. The constructor should take a single std::size\_t variable to indicate the capacity, i.e. the maximum number of books that can be stored in the dynamic array. If no size is specified (default constructor), then the size should be set to 30. The destructor should deallocate the memory.
* **operator+=(const BookList & rhs)** - This concatenates the rhs list to the end of this list.
* **operator[](size\_t index)** - This returns the book at the specified index.
* **std::size\_t size()** - This returns the number of books in the list.
* **std::size\_t find( const Book & book )** - This returns the returns the (zero-based) offset from top of list where the book is found, or the size of the list if the book is not found

Initially the given code will not compile. As you complete the code, the tests should start to pass in main.cpp.

### Detailed Description

Class Book has:

1. Attributes:
   1. \_title: indicates the name of the book (Ex: An Introduction to Programming with C++, Data structures for particle physics experiments)
   2. \_author: indicates the book’s author (Ex: Diane Zak, Alison "Ally" Uttley)
   3. \_isbn: ISBN is a 10 or 13 character international standard book number uniquely identifying this book (Ex: 9790619213090, 979010181X).
   4. \_price: indicates the cost of the book in US Dollars (Ex: 74.99, 115.50). Code this as type double.
2. Construction: Allow books to be constructed with zero, one, two, three, or four arguments. Title must be the first argument, author the second, ISBN the third, and price the fourth. Initialize each attribute with member initialization and in the constructor’s initialization list. Do not set the attribute’s value in the body of the constructor.
3. Function members include setting and retrieving each of the attributes. Name your overloaded functions get/set\_isbn, get/set\_title, get/set\_author, and get/set\_price. For example:   
   void set\_isbn( const std::string & isbn); //mutator or getter  
   std::string get\_isbn() const; //query or setter
4. Friend functions for reading and writing, aka the insertion and extraction operators . For example, main() may read from standard input and write to standard output a Book object like this:  
   Book book;  
   std::cin >> book; // extraction (reading)  
   std::cout << book; // insertion (writing)

Insertion and extraction shall be symmetrical. That is, you shall be able to read what you write. Assume fields are separated by commas and string fields are always enclosed with double quotes. The first field must be the ISBN, the second the title, the third the author, and the fourth the price. For example:

"0001005340", "Little Grey Rabbit's Birthday", "Alison \"Ally\" Uttley", 31.57

Incomplete books or books with errors shall be ignored.

Don’t try to add and remove the quotes yourself. See and use std::quoted()

Two relational operators for class Book, == and !=, have been defined and implemented as ordinary functions.

|  |  |
| --- | --- |
| The class BookList builds on the Book class by considering a dynamic array of books. Here you create and maintain a collection of books to form a book list. A complete BookList class interface and partial implementation have been provided. You are to complete the implementation.  Class BookList has:   1. Attributes:    1. \_capacity: indicates the maximum number of books that can be stored in the dynamic array    2. \_books\_array\_size: indicates the number of books in the list    3. \_bookArray: indicates the dynamic array of books |  |

2. Constructors:

1. The constructor with a single argument, with default values, allows to set \_capacity to the specified value, with the default value of 30.
2. The copy and move constructors do the usual.
3. The assignment operator = does the usual.
4. The assignment operator += appends another object of type BookList to the current object, as long as it does not exceed the capacity of the current object. It stops when the capacity is reached and does not add any more books.

3. Destructor needs to release to the heap the dynamic memory.

4. Function members include:

1. find() takes a book as a parameter and returns the zero-based offset of that book, or the total number of books in the book list if the book was not found
2. size() takes no parameters and returns the number of books in the book list.
3. operator[] takes as an argument a non-negative integer and returns the book stored at that index
4. readInFile() reads the books from a given file (data.csv) and populated the calling object

4. Friend functions for reading and writing, aka the insertion and extraction operators . For example, main() may read from standard input and write to standard output a BookList object like this:  
BookList list5;  
std::cin >> list5; // extraction (reading)  
std::cout << list55; // insertion (writing)

Two relational operators for class BookList, == and !=, have been defined and implemented as ordinary functions.

### Source Code Files

You are given “skeleton” code files with declarations that may be incomplete and without any implementation. Implement the code and ensure that all the tests in main.cpp pass successfully.

* LICENSE: This is completed. It contains the MIT license that allows you to use the instructor’s provided code as the skeleton code.
* BookList.hpp: This file is completed
* **BookList.cpp: This file is to have the function members with // TO DO to be completed**
* main.cpp: The main function tests the output of your functions. You may wish to add additional tests. During grading your main.cpp file will be replaced with the one you were provided with.

### Hints

This is an introductory assignment meant to test your knowledge of constructing a class, dynamically allocating memory with a pointer, and operator overloading. **You should not use the std::vector for this first project.**

Make sure your code compiles, and then try and solve the logic. Focus on solving one test at a time.

### Obtaining and submitting code

The skeleton code is available in Project 1 folder in Titanium. When you have completed the project, upload the completed files in the Project 1 folder.

#### Submitting the project

Only submit the following one file on Titanium in the link provided for Project 1:

1) BookList.cpp

### Testing (either of the two below)

1. **On Tuffix:**

Use the following command to compile your program:  
**clang++ -g -std=c++17 main.cpp BookList.cpp -o test**  
  
To attempt to run the compiled test program, use the following command:  
**./test**

##### **You can use Visual Studio:**

Free download: Community 2019

<https://visualstudio.microsoft.com/downloads/>

You can write, debug, run your project in Visual Studio.

### Grading rubric

Your grade will consist of two parts, *Form* and *Function*.

*Function* refers to whether your code works properly as tested by the main function (80%).

**Points will be deducted if vectors are used for this assignment (see Hints above)**

*Form* refers to the design, organization, and presentation of your code. An instructor will read your code and evaluate these aspects of your submission (20%).

### Deadline

The project deadline is **October 2, 2020, 11:59 PM**.

Late submissions will not be accepted.

**Your code must compile/build for it to be tested and graded. If you only complete part of the project, make sure that it compiles before submitting.**

### Blurb for your resume

Use your GitHub account as a ready-to-show portfolio of your programming projects to potential employers. You can also add this blurb to your resume after successfully completing this project:

*Completed from a point-of-departure baseline the implementation of a C++ program that reads books from a CSV file and then queries the book list by index.*