

CS 12 • Winter 2003

In-lab Programming Exercise 5

Separate Compilation

Kun Yan & Wagner Truppel
kyan, wagner@cs.ucr.edu
Department of Computer Science and Engineering
University of California, Riverside

March 2, 2003

In this assignment, you are going to write **two** classes to maintain information about books in a library. The complete program should consist of 5 source files: **book.h**, **book.cpp**, **library.h**, **library.cpp**, and **main.cpp**.

Details

- **book.h**: This file contains the *interface* for the book class. The class `Book` holds the following information:
 - **the book title**: this should be a `string` object.
 - **the author's name**: you may assume a single author, also to be represented by a `string` object.
 - **the number of pages**: how many pages there are in the book.

The class should also contain some member functions:

- one or more appropriate **constructors**.
- `setInfo()`: a function to ask the user for a book's title, author, and number of pages.
- `showInfo()`: a function to display the complete information about a book.

- `compareTitle()`: this function takes a title as an argument and compares it to the title of the `Book` object it was called on. If they match, the function returns `true`, otherwise it returns `false`. **Hint:** use the facilities available in the `string` class.
- **book.cpp:** This file contains the *implementation* for the class `Book`.
- **library.h:** This file contains the *interface* for the library class. The class `Library` holds the following information:
 - the **maximum number of books** that can be held in the library.
 - the **actual (current) number of books** in existence in the library.
 - an **array of book objects**. **Hint:** `Book *booksA`.

This class should also contain some member functions:

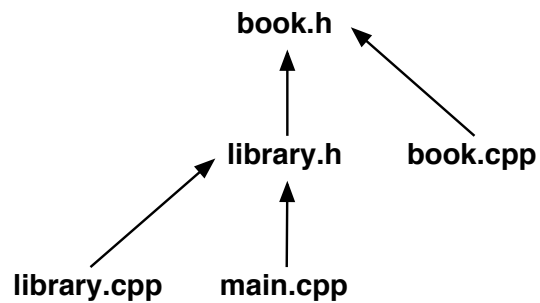
- an appropriate **constructor**: the constructor takes an argument, passed from `main()`, to initialize the maximum size of the library. It also initializes the other member variables. **Hint:** you need to use dynamic allocation.
- `add()`: this function adds a new book to the library. Be sure not to overfill the library. **Hint:** it should call `setInfo()`.
- `showAll()`: this function should call `showInfo()` for each book in the library.
- `searchByTitle()`: this function prompts the user for a title and searches for a book in the library having that title. If the book is found, its information should be displayed; if the book is not found, an error message should be displayed explaining that a book with the given title was not found in the library.
- `getNumberOfBooks()`: this function returns the number of books currently in the library.
- an appropriate **destructor**: the destructor should properly dispose of the array holding the books.
- **library.cpp:** This file contains the *implementation* for the class `Library`.
- **main.cpp:** This file contains the main function. It should prompt the user for the maximum number of books in the library, then create a `Library` object. It should also present a menu of options to the user:
 1. Input the information for a book.
 2. Search for a book, by title.
 3. Display the information for a book.
 4. Display the information for all books.

5. Display the number of books currently in the library.
6. Exit.

Depending on which option the user chooses, the corresponding function should be called. Be sure to let the user choose options from the menu as often as he or she likes, until the exit option is chosen.

File Organization

The following picture illustrates the dependence between the various files, that is, which files should include which other files. Thus, for example, `main.cpp` should have an include directive for the file `library.h`, and so on.



Compilation

You should compile your program using the following command:

```
g++ -g -Wall -W -Werror book.cpp library.cpp main.cpp
```

■