Tommy Hollenberg
CS162 Assign2
Design Document
04/23/2017

Understanding the Problem

The purpose of this program is to simulate a simple online library system. The program should contain a library class that contains objects of a cart class to hold books, a librarian class for librarian users, and a patron class for members of the public that are members of the library.

Library data must be stored in auxiliary files that can be accessed and edited each time the program is run. These files should include: a database of books in the library, librarian login information and patron login information, records of the books currently checked out by specific patrons.

The program must contain the following functionality:

- Any user (without logging in) can access the library hours of operation (for a particular day, or for every day of the week) upon request
- Patrons (after logging in with a valid id) can view all the books in the library or search for specific titles, authors, or publication years. They can check out books that they have added to the cart from their searches, view the books that they currently have checked out, and return books to the library.
- Librarians (after logging in with a valid id) can change the library's hours of operation, add new books to the library, remove old books from the library (based on the publication year), and view books that are checked out (and by which patrons).

All class member variables must be private, with public accessor and mutator functions for each, as well as constructors and the Big Three (copy constructor, assignment operator, and destructor) to handle the dynamic memory (i.e. deep copying and freeing memory upon destruction). The program must not have a memory leak.

The program must be broken up in several files, including a driver file with the main function, header and source files for each class, and a Makefile to facilitate compiling and linking files to create the executable.

Devising a Plan

I got carried away and wrote the entire program before starting this design document, so my explanation of the design will focus more on program structure and less on micro-logic.

Books will be stored in a struct with the following members:

- string title
- int num_authors
- string *authors
- int year
- int copies_avail
- int copies_out

The cart class will have a book pointer member and functions to manage dynamically allocated and deallocating memory for an array of books. The library class will not contain this functionality, because it is already implemented in its cart member.

Jennifer repeatedly suggested in lecture that we refrain from simply reading in the entire library of books, and instead focus on repeatedly reading from and writing to the data files to practice file I/O. As such, my program reads in from the book database file for each search/query, and rewrites the entire file after every alteration.

The user interface consists of a web of menus, which is mirrored by my program structure.

Main Menu

Upon login, the user is welcomed and presented with the following main menu options:

1. Login as patron.
2. Login as librarian.
3. View library hours for the entire week.
4. View library hours for a specific day.
5. Exit

User input is handled by the first of two input validation functions that are used repeatedly in this program, called get_pos_integer and get_nonneg_integer. These functions take a prompt string, to be printed to the console repeatedly until valid input is entered, the address of an integer variable to store the input value, and an optional maximum acceptable input integer (the default value is INT_MAX from the climits library). get_pos_integer accepts an integer value between 1 and the max input value, and get_nonneg_integer accepts an integer between 0 and the max input.

Option 5 terminates the program.

Option 3 prints the weekly hours of operation to the console, and option 4 asks the user to input a day of the week and (if a match is

found) prints the hours of operation for that day. The functionality for both options is contained within the library class. Hours of operation are stored in a struct variable that contains two member strings that hold open and close hours in a 24-hr time format (i.e. 0700 and 1600). There is a non-member validation function that checks whether a string is a valid 24-hr time, which is used when a librarian attempts to make a change to the library hours. The function checks that the string contains four characters, that the first character is 0-2, that the second character is 0-9 if the first character is 0 or 1 and 0-4 if the first character is 2. If the first two characters are 2 and 4, respectively, the third and fourth must both be 0. Otherwise, the third character must be 0-5 and the fourth must be 0-9.

Options 1 and 2 allow the user to login as a patron and as a librarian, respectively. The login process is handled by the library class. An id number is requested from the user, and the program reads from the patron_id and librarian_id data files looking for a match. If a match is found, the user data is loaded into the library's patron or librarian member object, and the program proceeds to the patron or librarian menu. If no match is found, the program returns to the main menu shown above.

Patron Menu

The patron menu presents the following options:

1. Search for books to add.
2. View/Manage Cart.
3. Checkout.
4. View/Return books checked out.
5. Logout.

The functionality for these options is contained within the patron class (although functionality from the cart class plays an integral role as well).

Option 5 returns the user to the main menu.

Search Menu

Option 1 brings the user to the search menu, which presents the following options:

1. Search by title.
2. Search by author.
3. Search by year.
4. View all books.
5. Cancel.

Option 5 returns the user to the patron menu.

Options 1-3 allows the user to search for books with a specific title, author, or publication year, respectively, and option 4 sets no search restrictions. Book data is read into a local cart object from the book_database file. If the book does not meet the search criteria, it is overwritten with the next book read in (or removed if it is the

last book in the file). Otherwise, the size of the book array in the cart is increased by one and subsequent books are read into the next book element. Once all books have been read in, the books that matched the search criteria are printed to the console using a display_books function in the cart class that makes frequent appearances in the program. Here is some sample output from option 4 (view all books):

```
1.      Title: hamlet
        Authors: William Shakespeare
        Year of Publication: 1500
        Available Copies: 1
        Copies on Loan: 2

2.      Title: game of thrones
        Authors: George R.R. Martin
        Year of Publication: 2008
        Available Copies: 2
        Copies on Loan: 1

3.      Title: 1984
        Authors: George Orwell
        Year of Publication: 1955
        Available Copies: 1
        Copies on Loan: 1

4.      Title: Coding 4 Dummez
        Authors: Jackson
                 Frederick
        Year of Publication: 1970
        Available Copies: 1
        Copies on Loan: 1
```

The user is then prompted to enter book numbers to add books to the library's cart member object (which will be used for checkout). If there are no copies available, the user is not allowed to add that book to their cart. Inputting 0 will return the user to the previous search menu, and the local cart holding the search results will be emptied.

After adding books to their cart, the patron can select option 2 in the patron menu to view/manage their cart, or option 3 to proceed directly to checkout. If they choose option 2, they will be shown the books in their cart (which are printed using the same display_books function), and given three options:

```
Choose an option:
        1. Remove book.
        2. Proceed to checkout.
        3. Return to menu.
```

Option 3 returns to the patron menu.

Option 1 allows the user to enter a book number to remove a book from the cart (similarly to how books are added to the cart from search results).
Option 2 in this view cart menu is the same as option 3 in the patron menu above, and checks out all the books currently in the cart. This is fairly involved, because three files must be updated, the book_database file (the number of copies available must be decremented and the number of copies on loan must be incremented for each book checked out), the patron_id file (the number of books checked out must be increased so that the correct information is loaded next time the patron logs in), and the books_checked_out file. Three different functions are used to accomplish this, but they all follow the same general 3 step principle. 1. Read in data. 2. Check whether it needs to be updated. 3. Write the data (or the updated version).

Lastly, patrons can view/return books checked out with option 4 of the patron menu. This reads books into a local cart object (the library's cart member object is not used so that books the user added to their cart from search results will not be lost) from the books_checked_out file and overwrites them if they are not associated with the current user. The books checked out to the current patron are printed using the display_books function, and the user is given two options:

Choose an option:
1. Return a book.
2. Return to menu.

Option 2 returns to the patron menu.

Option 1 allows the user to enter a book number to return (similarly to how books are chosen to be added to or removed from the cart). The return process uses some of the same functions as the checkout process to update the book_database, patron_id, and books_checked_out files accordingly. The returned book is removed from the local cart holding the current books checked out, the cart is re-printed, and the user is given the same two options again (unless there are no more books checked out, in which case the user is returned to the patron menu).

Librarian Menu
When the user logs in as a librarian, a different menu is presented. The functionality accessed by this menu is contained within the librarian class, although the cart class is again used frequently.
1. Change library hours.
2. Add new books to the library.
3. Remove old books from the library.
4. View checked out books.
5. Log out.

Option 5 returns to the main menu.

Option 1 allows the user to enter opening and closing times for each in day of the week in the 24-hr time format as discussed above.

Option 2 allows the user to enter information about new books to be added to the book_database file. The user is prompted for how many different books they would like to add, and can indicate the number of copies of each book after entering the title, authors, and year of publication. The new books are simply appended to the book database file, without checking whether any copies of the books already exist.

Option 3 removes all books published prior to a year entered by the user. All of the books in the book_database are read in one at a time, and only written to the new file if their publication year is greater than or equal to the chosen year. This does not delete old books currently checked out by patrons. However, when these books are returned, they are not written back into the book_database file.

Books Checked Out Menu

Option 4 allows the librarian to view the books currently checked out. The following options are give:

1. View all checked out books.
2. View books checked out by a specific user.
3. View which user(s) have checked out a particular book.
4. Return to the menu.

Option 4 returns to the librarian menu.

Option 1 reads and prints out the entire books_checked_out file, organized by patron, as shown in the following output sample:

```
Patron 4 - Dave

1.      Title: 1984
        Authors: George Orwell
        Year of Publication: 1955
        Available Copies: 0
        Copies on Loan: 1

Patron 5 - John

1.      Title: hamlet
        Authors: William Shakespeare
        Year of Publication: 1500
        Available Copies: 0
        Copies on Loan: 1
```

Option 2 lets the librarian enter a patron id, and only the books checked out by that patron are printed (if any).

Option 3 prompts the librarian to enter book data (using the same function used to enter books to add to the database), and prints out the id and name of all patrons who currently have a copy checked out, as shown in the following output sample:

Book title: hamlet
Number of authors: 1
Author 1: William Shakespeare
Year of publication: 1500

Checked out by:
Patron 5 - John
Patron 7 - Tommy

Testing Plan

I have already tested this program quite significantly, and valgrind confirmed that there were no memory leaks after going through all the functionality.

Most of the testing was conducted as the program was being written, piece by piece. At first, this involved editing the auxiliary data files before each run and tweaking the code until the desired results were achieved. As more and more functionality was implemented, I needed to edit the data files less and less. Once the back-end was fully fleshed out, my testing focused instead on the front-end; ensuring that the user interface was correct and properly formatted.