

CPSC 427 – Model a library

A programming Assignment

- First, a universal truth
 - *Many days of programming can save you hours of planning*
- *Model a library. It has books and patrons. Patrons can register for a card and check out books. Patrons can only checkout 10 books at a time.*

Problem

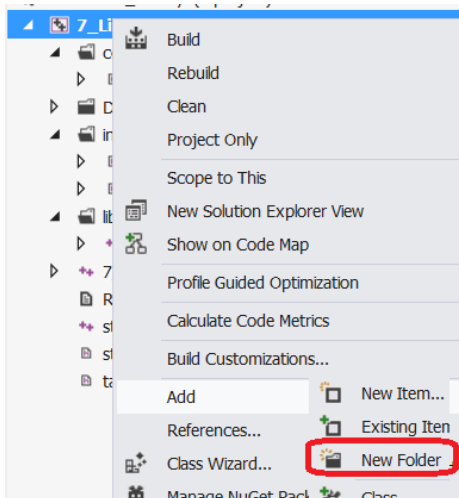
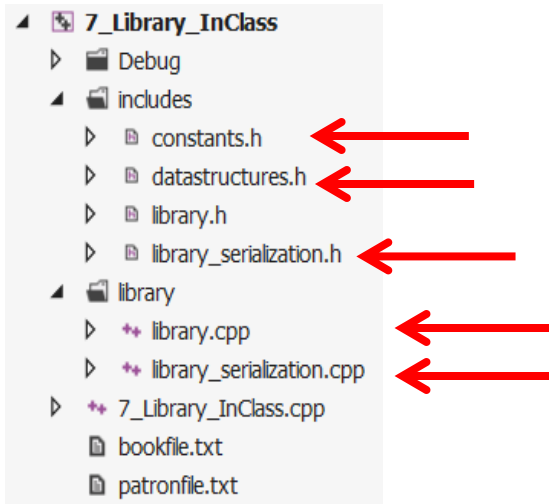
Library has books, patrons
patrons can checkout/in books
patrons can enroll (get a card), can check
out 10 books at a time

Top down design!!



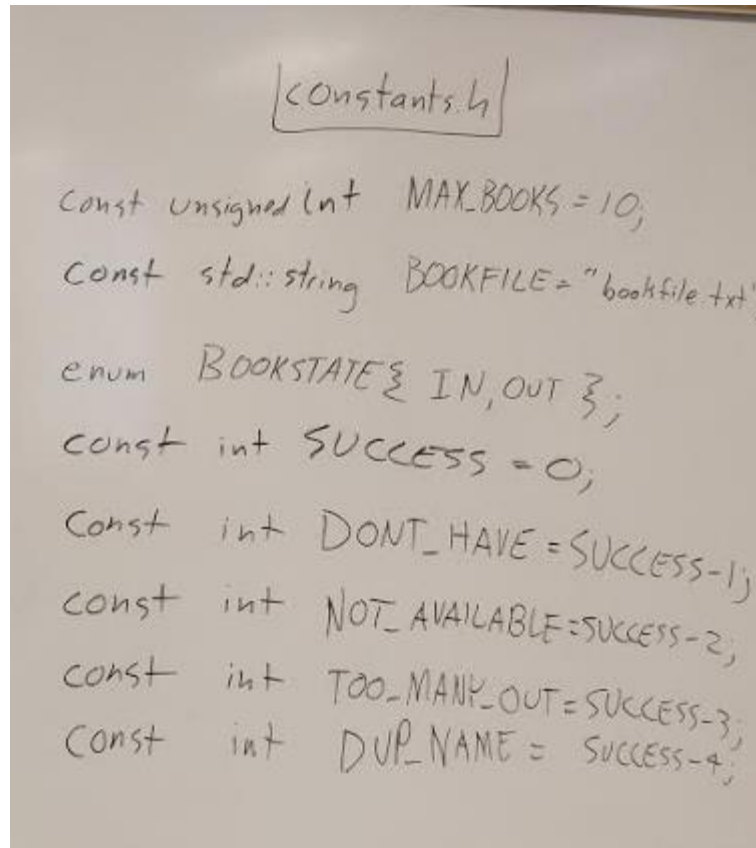
- Layout functions needed
- Generate constants file along the way
- put off defining datastructures until the last minute, hide them so user does not know what you are using (allows you to change them without affecting user)
- Make API easy to use and as simple as possible
- Hide what you can in cpp files (datastructures, containers, initialization).
- Anything you can automatically do for the user ...do

Directory structure



- Several files so lets put them in different directories.
- Helps organize code
- Essential for large projects
- Right click on project then 'Add' then 'New Folder'

constants.h

A photograph of a piece of paper with handwritten C++ code. The code is for a header file named 'constants.h'. It defines several constants: 'MAX_BOOKS' as an unsigned int with value 10, 'BOOKFILE' as a std::string with value 'bookfile.txt', an enum 'BOOKSTATE' with values 'IN' and 'OUT', and four integer constants: 'SUCCESS' (0), 'DONT_HAVE' (SUCCESS-1), 'NOT_AVAILABLE' (SUCCESS-2), 'TOO_MANY_OUT' (SUCCESS-3), and 'DUP_NAME' (SUCCESS-4).

```
constants.h  
  
const unsigned int MAX_BOOKS = 10;  
const std::string BOOKFILE = "bookfile.txt";  
  
enum BOOKSTATE { IN, OUT };  
const int SUCCESS = 0;  
const int DONT_HAVE = SUCCESS - 1;  
const int NOT_AVAILABLE = SUCCESS - 2;  
const int TOO_MANY_OUT = SUCCESS - 3;  
const int DUP_NAME = SUCCESS - 4;
```

- Lets put this in its own folder, called constants
- Don't forget include guards (#pragma once is the easiest)

datastructures.h

Data Structures.h

```
struct Book {
```

```
    int ID;
```

```
    std::string title;
```

```
    std::string author;
```

```
    BOOKSTATE state;
```

```
    int patronID;
```

```
struct Patron {
```

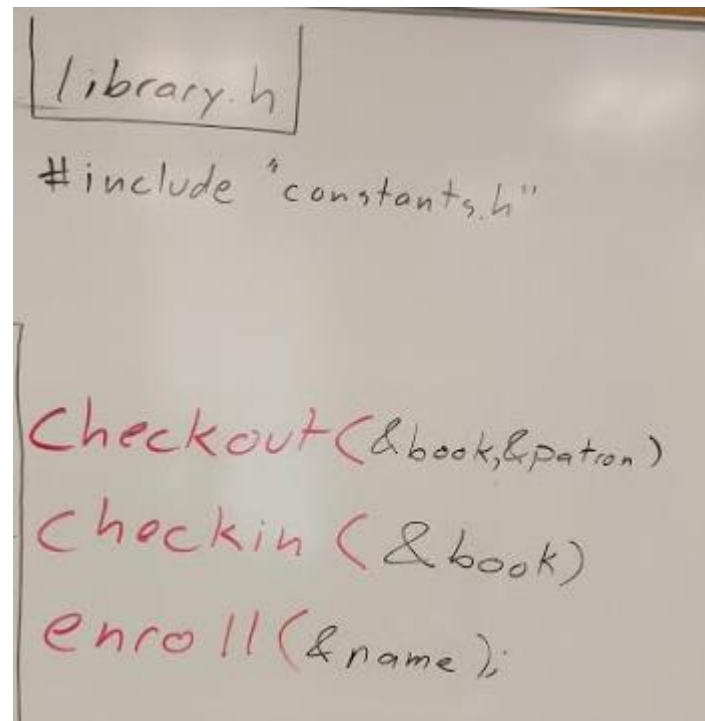
```
    int ID;
```

```
    std::string name;
```

```
    int NBCCO;
```

- Lets put this in includes folder

library.h

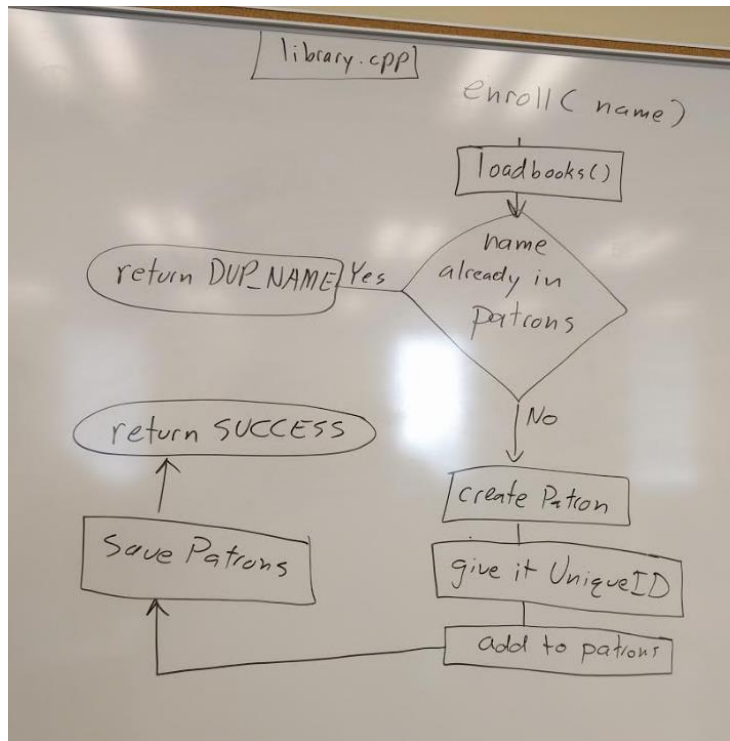


A photograph of a whiteboard with handwritten C code. The code is for a header file named 'library.h'. It includes a preprocessor directive to include 'constants.h'. Below this, three function declarations are written in red ink: 'Checkout(&book, &patron)', 'checkin(&book)', and 'enroll(&name);'. The first two functions are missing closing parentheses, and the third is missing a semicolon.

```
library.h  
#include "constants.h"  
  
Checkout(&book, &patron)  
checkin (&book)  
enroll (&name);
```

- Lets put this in includes folder

library.cpp



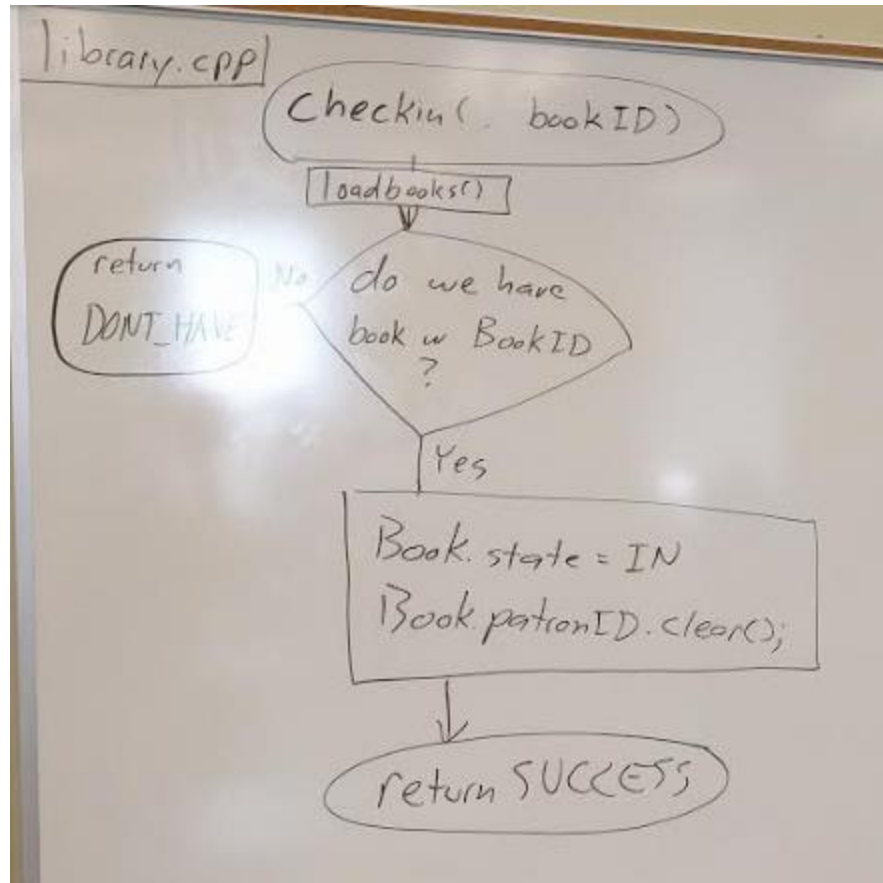
library.cpp

```
loadbooks(filename)
if (isloaded)
    return SUCCESS;

//load books

isloaded = TRUE;
return SUCCESS;
}
```


library.cpp



library.cpp

book is loaded - false.

library.cpp

