OOP Course and Laboratory
Catalin Boja
catalin.boja@ie.ase.ro

Assignment

Implement a C++ class Book that demonstrates the use of dynamic memory, constructors, destructors, getters, setters, and generic processing methods.

Notes for the Assignment

- **You can use only pointers and classic C array manipulation** (**NO** STL containers like vector).
- **You can use only iostream, string.h and stdio.h libraries.**
- **Don't change given methods definition**
- For char array processing you can use ONLY *string.h* functions (like strncpy, strlen, strcmp, etc). Because is vulnerable to different buffer overflow attacks, **we don't use** strcpy(). Use strcpy_s().
- The solution must be modularized by using functions.
- Manage memory carefully, especially when dealing with dynamic arrays.
- Solutions that use advanced C/C++ syntax will not be considered for evaluation (like dynamic cast, other libraries, etc)
- Required time to code it, from 80 minutes to 240 minutes
- Please don't waste your time by coding this with ChatGPT or other code generation tools. You gain nothing. Try to code it, even if you don't know how to solve everything or don't have time to finish it. Coding is learned by writing a lot of bad code. Copy & paste does not help. Use Google Search and other tools to learn how to do on your own different implementations.

Requirements

1. Class Definitions

1. **Book Class (Base Class)**:
    - Attributes:
        - title: A char* to store the title of the book.
        - author: A char* to store the author's name.
        - isBorrowed: A bool indicating if the book is currently borrowed.
    - Methods:
        - A **constructor** to initialize title and author.
        - A **virtual destructor** to properly deallocate dynamically allocated memory.
        - borrowBook(): Marks the book as borrowed.
        - returnBook(): Marks the book as returned.
        - displayInfo(): A **pure virtual method** to display the book's details (must be implemented by derived classes).
        - getType(): A **virtual method** to return the type of the book (e.g., "Book" for the base class, overridden in derived classes).

2. **Derived Classes (TextBook, Novel)**:
    - **TextBook**:
        - Additional attribute: subject (e.g., "Physics", "Math").
        - Override displayInfo() to include subject.
        - Override getType() to return "TextBook".
    - **Novel**:
        - Additional attribute: genre (e.g., "Fiction", "Mystery").
        - Override displayInfo() to include genre.
        - Override getType() to return "Novel".
3. **Borrower Class**:
    - Attributes:
        - name: A char* for the name of the borrower.
        - borrowerId: An int for a unique identifier.
    - Methods:
        - A **constructor** to initialize the attributes.
        - A **destructor** to free dynamically allocated memory.
        - displayInfo(): Displays borrower information.
4. **LibrarySection Class**:
    - Represents a **1:1 composition** with a Book (each section is tied to one book).
    - Attributes:
        - sectionName: A char* for the name of the section (e.g., "Shelf A").
        - book: A **pointer** to a Book object.
    - Methods:
        - A **constructor** to initialize the section and associate it with a book.
        - A **destructor** to handle cleanup.
        - displayInfo(): Displays the section's name and information about the associated book.
5. **Library Class**:
    - Represents a **1:N composition** with Book objects.
    - Attributes:
        - books: A **fixed-size classic array** of pointers to Book objects (e.g., Book* books[50]).
        - bookCount: An integer to track the number of books.
    - Methods:
        - addBook(Book* book): Adds a book to the array.
        - displayBooks(): Displays information about all books.
        - borrowBook(int index): Marks a book at the given index as borrowed.
        - returnBook(int index): Marks a book at the given index as returned.
        - **searchByAuthor(const char* author):**Searches for books by a specific author. Returns the books written by that author.
6. **countBorrowedBooks()**:
    - Counts the total number of borrowed books in the library.
    - Returns an integer representing the count.

---

2. Functional Requirements

1. **Inheritance:**
    - Use inheritance for the Book hierarchy (TextBook, Novel).

    - Demonstrate polymorphism with virtual methods:
        - displayInfo(): Polymorphic behavior for displaying details.
        - getType(): Overridden in derived classes.
2. **Composition:**
    - Use a **1:N composition**:
        - The Library class holds an array of Book* objects (maximum size 50).
    - Use a **1:1 composition**:
        - The LibrarySection class associates one book with one section.
3. **Memory Management:**
    - Use **classic pointers** and ensure proper cleanup using destructors.
4. **Error Handling:**
    - Ensure borrowing is only possible if the book is available.
    - Ensure returning is only possible if the book is borrowed.

---

3. Tasks for Students

1. Implement all classes with the specified attributes and methods.
2. Populate the library with at least 5 books (mix of TextBook and Novel).
3. Add books to different library sections.
4. Borrow a book, return it, and display the updated library status.
5. Demonstrate polymorphism by iterating through the books array in Library and calling displayInfo().

---

4. Bonus Tasks

1. Add a new book type (ReferenceBook) that cannot be borrowed. Override borrowBook() to throw an exception.
2. Implement a searchByTitle() method in the Library class to find a book by its title.
