2024 OOP Course and Laboratory
Catalin Boja
catalin.boja@ie.ase.ro

Assignment 4

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. All attributes are private and are accessible by public accessor methods
2. Implement:

Attributes:

1. std::string title – The title of the book.
2. std::string* authors – A dynamic array of authors.
3. int authorCount – The number of authors.
4. int pages – The number of pages in the book.
5. int id – Constant id of the book

Constructors:

1. **Default Constructor**:
    - Initializes the title to "Unknown".
    - Initializes the authors pointer to nullptr.
    - Sets authorCount to 0.
    - Sets pages to 0.
2. **Parameterized Constructor**:

- Accepts title, a dynamic array of authors, authorCount, and pages as parameters.
- Dynamically allocates memory for the authors and copies them.

3. **Copy Constructor**:
   - Creates a deep copy of another Book object, ensuring dynamic memory is handled correctly.
4. **Destructor**:
   - Frees the dynamically allocated memory for the authors array.

Methods:

1. **Dynamic Memory Management**:
   - Properly handle dynamic memory allocation and deallocation for the authors array.
2. **Display Details**:
   - Print all book attributes, including title, authors, and pages.
3. **Generic Methods**:
   - bool isAuthor(const std::string& name): Checks if the provided name exists in the authors list. Returns true if found, otherwise false.
   - int getAuthorPosition(const std::string& name): Returns the index of the author in the list if they exist, or -1 if they do not.

Overloaded Operators:

1. **Assignment Operator (=)**:
   - Ensures deep copying of the authors array and proper cleanup of existing data.
2. **Arithmetic Operators**:
   - Book operator+(int extraPages) – Adds extra pages to the book and returns a new Book object.
   - (int extraPages) + Book – Adds extra pages with an integer on the left-hand side.
   - Book& operator+=(int extraPages) – Modifies the current book by adding extra pages.
3. **Logical Operators**:
   - bool operator==(const Book& other) – Compares two books for equality based on title, authors, and pages.
   - bool operator>(const Book& other) – Compares two books based on the number of pages.
4. **Unary Operators**:
   - bool operator!() – Returns true if the book has no authors.
   - Book& operator++() – Pre-increment to increase the page count.
   - Book operator--(int) – Post-decrement to decrease the page count.
5. **Array Access Operator ([])**:
   - Returns the author at a given index. Throws an exception for out-of-range indices.
6. **Function Operator (operator())**:
   - int operator()() – Returns the number of pages.
   - std::string operator()(int index) – Returns the author's name at a given index or throws an *out_of_range* exception if out of range.
7. **Type Conversion Operator**:
   - Explicit operator std::string() – Converts the book to its title as a string.
8. **Stream Operators**:
   - friend std::ostream& operator<<(std::ostream& os, const Book& book) – Outputs the book’s attributes to a stream.

- friend std::istream& operator>>(std::istream& is, Book& book) – Inputs book details from a stream.

Note for stream operators.

The >> operator allows reading a Book object's data from a standard input stream. It should prompt the user for:
>> Title of the book
>> Number of authors
>> Each author name
>> Number of pages. in this particular order.

Expected Input and Output
Input (via >>):
Enter title: Test Book
Enter number of authors: 2
Enter author 1: Author A
Enter author 2: Author B
Enter number of pages: 350

Operator >> outputs the Book object's details in a structured format. It should display:

- Title of the book on a separate line
- Number of pages on a separate line
- List of authors separated by commas on a separate line

Expected Input and Output
Output (via <<):
Title: Test Book
Pages: 350
Authors: Author A, Author B