2024 OOP Course and Laboratory

Catalin Boja, catalin.boja@ie.ase.ro

Assignment 1

Write a C++ program that manages information about books in a small library. Each book has the following details: title (char array), author (char array), year of publication (integer), and number of pages (integer). The program should provide functionality for managing the collection of books using static and dynamic arrays, as well as demonstrating the use of pointers and parameter passing to functions.

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.**
- 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 strncpy().
- 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 (from easiest to most difficult)

1. Basic Setup (Easy)

- Define a Book structure containing:
    - char title[100];

    - char author[100];
    - int year;
    - int pages;
- Write a function **void initializeBook(Book* b, const char* title, const char* author, int year, int pages)** that takes a pointer to a Book structure and initializes its fields using the provided parameters. This function demonstrates pointer usage and parameter passing.

2. Static Array Management (Medium)

- Create a static array of 5 Book structures in the main function.
- Use a loop to fill the array with user input by calling initializeBook for each element.
- Write a function **void displayBooks(const Book books[], int size)** that prints out the details of each book in the array. Use pointers for traversing the array within the function.

3. Dynamic Array and Memory Management (Advanced)

- Extend the program to manage a dynamic array of books instead of a static one.
- Write a function **Book* addBook(Book* books, int& size, const Book& newBook)** that:
    - Takes the dynamic array of books and its size as arguments.
    - Allocates new memory to increase the array size by one.
    - Adds a new book to the array and returns the updated array.
- Ensure that the function correctly handles memory allocation and deallocation.

4. Search Function (More Advanced)

- Implement a function **Book* searchByTitle(Book* books, int size, const char* title)** that searches for a book by title and returns a pointer to the found Book structure (or nullptr if not found). This tests the student's ability to work with dynamic arrays, pointers, and string manipulation.

5. Memory Cleanup and Program Modularization (Challenging)

- Write a function **void cleanup(Book*& books)** to deallocate the dynamic memory used for the books array.
- Modularize the program by creating separate functions for each operation (e.g., adding a book, displaying books, searching, and cleaning up memory).

Optional Bonus (Extra Challenge)

- Implement a sort function **void sortByYear(Book* books, int size)** that sorts the books array by the year of publication using pointers.