OBJECT ORIENTED PROGRAMMING

SEMINAR 2

OBJECTIVES

The main objective of this seminar is to practice the concepts we learned at the lecture related to the stack and the heap, pointers and dynamic memory allocations. Finally, at the end of the seminar, to test your skills and understanding related to memory management in C/C++, we'll organize a contest: you will pair up with a colleague and try to find all the errors in the provided code.

PROPOSED PROBLEMS

1. Let's start by remembering the memory layout of a C program, and the common memory. Find all the memory errors in the solution MemoryErrors (found in the archive MemoryErrors.zip).

2. Write a *module* for a dynamic array of integer numbers.
   You know that in C/C++ when you define an array on the stack, its size is fixed. More specifically, if you write *int arr[100];* , this means that the maximum number of values your array can store is 100.
   Your task is to write a module (define an abstract data type and the operations you can perform on that data type) for a dynamic array, where the storage is handled automatically, being expanded and contracted as needed. This is something similar to the container *std::vector* from STL: https://en.cppreference.com/w/cpp/container/vector .
   Your dynamic array should have the following fields:
   - *capacity* – the current capacity of the array; this tells you the maximum number of elements that you can store in the array;
   - *length* – the current length of the array; this tells you how many elements you are storing in the array;
   - *data* – the actual array (allocated on the heap) where you store the elements;

   Now you should write functions for the following operations:
   - insert an element at the end of the array;
   - erase the element from the end of the array;
   - get/set the value of the element from a given position in the array;
   - erase the element from a given position in the array;

- search for an element x in the array. This function returns the index of the first occurrence of the element x, or -1 if that element is not present in the array;
- resize the array.

You should use error codes to report the errors to the user (when the preconditions of the functions do not hold).

3. How would you define a static array (on the stack) that stores pointers to integers?

4. How would you define a dynamic array that stores pointers to integers?

5. Modify the code you wrote such that the dynamic array can store elements of any type. Yes, you should use void pointers to functions for this!
   For this part, the signature of your search function should change. Now this function also takes as parameter a pointer to a function that is used to compare two elements.

<u>Optional</u>

6. Now is time to observe the benefits of using modular programming! More specifically, you will reuse the module that you wrote at exercise 5 for a toy application.
   Write a menu based application *MyMovieDatabase* that allows you to manage all the movies that you've seen.

   A movie is defined by its title, the director, the year it was released and your rating for the movie. You should have the following options:
   - insert a movie in your database;
   - search for a movie in your database;
   - modify your rating for the movie.