Lab Report

Autor: John Eury

Topic: Implementation of a Stack in C++

The objective of this exercise was to understand the principles and implementation of the stack data structure. A stack is a LIFO (Last In, First Out) structure in which the most recently added element is the first to be removed.

During the lab session, a stack was implemented using a custom C++ class. The stack was based on a dynamic array and included the following methods:

- `push(int value)` – adds a value to the top of the stack,
- `pop()` – removes the top element,
- `top()` – returns the value at the top without removing it,
- `isEmpty()` – checks whether the stack is empty,
- `size()` – returns the number of elements currently in the stack.

The implementation was tested using a variety of integer input values. Functionality tests were performed on `push` and `pop` operations, as well as edge cases such as popping from an empty stack.

The program was written in Visual Studio Code and compiled with `g++`. The GDB debugger was used to step through the code and verify proper memory management and correctness of the logic.

Conclusion: The exercise provided a deeper understanding of stack behavior and memory handling in C++. Implementing the stack from scratch helped reinforce the theoretical knowledge with practical experience.

In the next session, we plan to implement a stack using a singly linked list and compare its performance and memory usage with the array-based approach.