Burlacu Natalia,

PhD, Associate Professor

DATA STRUCTURS AND ALGORITHMS

Laboratory Work no. 7 & Individual Work no. 7

Chisinau, TUM, 2024

7. Solving the problems using THE TREE DS, as AVL TREE, B TREE & B+ TREE in C / C++

High-medium level / high level

Task:

Develop a procedural-style program in C / C++, using own written functions. Data processing in your program should be organized according to a given length of input records based on memory allocation functions.

Your solution should:

A. use the pointers;
B. have to be presented in your report, emerging from the content of the problem statement.
C. To draw the block diagram corresponding to the solved problem.

Attention! The grading will take into account the originality, complexity, and quality of the proposed and solved changes.

Using the custom data type (CDT) (structs and structs members (both the initial ones and the output dataset)), that have been assigned to you in problem set no.5 (see the preview set of laboratories (available in: CDT from Set of Labs No. 5)) develop a procedural-style program in C / C++, using own written functions.

Data processing in your program should be organized according to a given length of input records based on memory allocation functions.

Your solution should:

1. use the pointers;

2. have to be presented in your report, emerging from the content of the problem statement, in two versions:

A. The procedural version of your solution should be developed 1. using the structures (declared by the keyword STRUC and typedef or some UNION); 2. if necessary, nested structures (ore union) will be created; 3. using the notations specific to pointers.
B. The presented solution will be unified through a header file that will be called in the main function to allow the program to run, using the following options regarding the user's choices:

- The version that will capitalize on your structure: should be implemented in a **AVL TREE**;
- The version that will capitalize on your structure: should be implemented in a **B TREE;**
- The version that will capitalize on your structure: should be implemented in a **B+ TREE.**

The elaborated code will contain the functions to be called in the main, such as:

I. **AVL TREE** / **B TREE** / **B+ TREE** creating, crossing, and displaying.

II. Insert a node element into the **AVL TREE** / **B TREE** / **B+ TREE** *(in the beginning of ADS; in the end of ADS and in some middle position of ADS, where the given position and / or the value of the inserting node will be read from the keyboard).*

III. Search for an element by position and by value. In the searching an element by value should be capitalized the Backtracking and / or Dynamic Programming approaches.

IV. Deleting a node element from the **AVL TREE** / **B TREE** / **B+ TREE**

- *the option of future deletion should be read from the keyboard: by position and by the value of the node;*
- *in the case of the deletion by value of the node this should be done in the beginning of ADS; in the end of ADS and in some middle position of ADS, where the given position and / or the value of the inserting node will be read from the keyboard.*

V. Registering the newly formed **AVL TREE** / **B TREE** / **B+ TREE** after (adding/removing a node element into the **AVL TREE** / **B TREE** / **B+ TREE**) in a *.* file (which will give the user the possibility to record info, both in *.txt and binary mode), the address of which will read from the keyboard (a function to read the full address of the file to be registered will also be developed.

The given file must be possible both to be created, opened, reopened, but also to be deleted).

P.S: For grading, the given laboratory should be presented by each author in person.