Burlacu Natalia,

PhD, Associate Professor

DATA STRUCTURS AND ALGORITHMS

Laboratory Work no. 6 & Individual Work no. 6

Chisinau, TUM, 2024

6A. Solving the problems using STACK / QUEUE ADT, FILE data type & UDT (user-defined data type) in C / C++

Average 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 **dynamic Stack**;
- The version that will capitalize on your structure: should be implemented in a **dynamic Queue.**

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

I. Stack / Queue creating, crossing, and displaying.
II. Insert an element into the Stack / Queue.
III. Search for an element by position or by value.
IV. Deleting an element from the Stack / Queue.
V. Registering the newly formed Stack / Queue after (adding/removing an element into the Stack / Queue) 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).

6B. Solving the problems using LINKED LISTS ADT, FILE data type & UDT (user-defined data type) in C / C++

Average 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:

D. use the pointers;
E. have to be presented in your report, emerging from the content of the problem statement.
F. 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:

C. 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.

D. 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 **SIMPLE LINKED LIST**;
- The version that will capitalize on your structure: should be implemented in a **DOUBLE LINKED LIST;**
- The version that will capitalize on your structure: should be implemented in a **CIRCULAR LINKED LIST.**

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

VI. **SIMPLE / DOUBLE LINKED LIST / CURCULAR LINKED LIST** creating, crossing, and displaying.

VII. Insert a node element into the **SIMPLE / DOUBLE LINKED LIST / CURCULAR LINKED LIST** *(the position and the value of the inserting node will be read from the keyboard).*

VIII. Search for an element by position or by value.

IX. Deleting a node element from the **SIMPLE / DOUBLE LINKED LIST / CURCULAR LINKED LIST** *(the position and the value of the inserting node will be read from the keyboard).*

X. Registering the newly formed **SIMPLE / DOUBLE LINKED LIST / CURCULAR LINKED LIST** after (adding/removing a node element into

the **SIMPLE / DOUBLE LINKED LIST / CURCULAR LINKED LIST**) 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).