C++

Semester 3 SD

Stacks & Queues Assignment 4

Nov 24, 2022.

Student: Andre Sanao

Course: Technology

Introduction

In the previous assignment, we were to create our own Stack & Queue algorithm. This was created in C. Now we are task to do the same but in C++. This time we have to use OOP to create classes and functions for the algorithm. With C++, we will be using the “new” keywords in C++ to allocate memory instead of malloc in C. In the following section will show the implementation of the algorithm.

Design

To be able to translate the C code into C++, we need to start making classes for the algorithm. Each person has their way of translating their code. To start it off, we needed nodes just like in the Linked List assignment. I used the Linked List that was given as an example. In the example, a struct was used to create a node (figure 1).

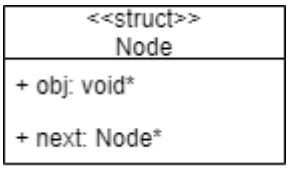

Figure 1 struct Node

In the Stack class is where I started to implement OOP. As you can see in figure 2, we started creating private members initialize them in the constructor. Objsize is used as a parameter because we want to be able to change the data type. For the destructor, we iterate through the node and deallocate them using the delete keyword. It is required to encapsulate these data members because encapsulating makes it that it is hidden from the users. To read the private members, I created get functions to read the values of the private members. These get functions will be used in the testing cpp file. The functions for the Stack class will be the same but this time memory is allocated/deallocated using the new and delete keyword.

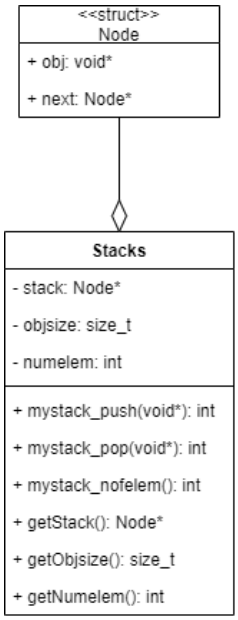

Figure 2 Relationship between Node and Stacks

In the Queue class, it is the same where we create constructor and initializes 2 Stack objects and the data type. For the destructor, we use the delete keyword to deallocate the objects and that will call the destructor of the Stack class. Lastly is creating the get function to return the private members and functions for enqueue and dequeue. The final UML diagram can be seen below (figure 3).

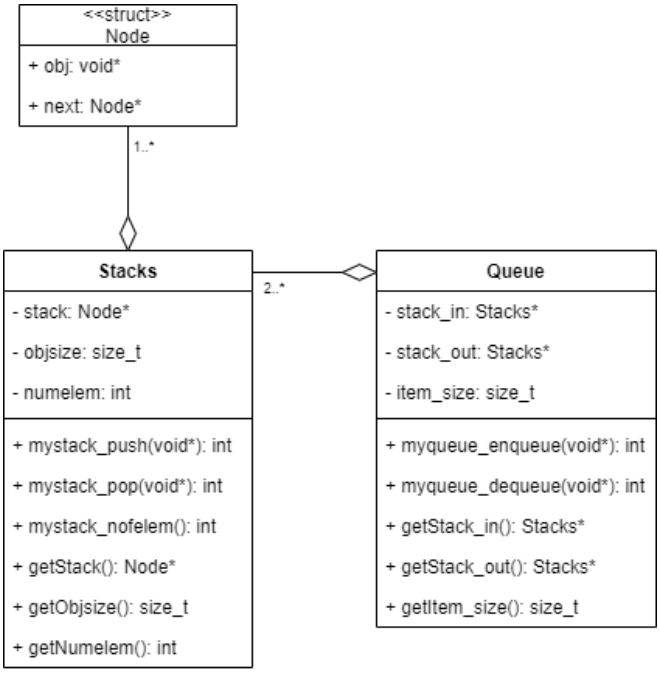

Figure 3 UML diagram with Queue class added

The relationship between these is aggregation because Stacks can have one or more of the struct Node and Queue can have two or more of Stacks because the implementation consist of 2 Stacks object. In this case the child class can still remain to exist even if the parent class is not present.

Conclusion

To conclude this assignment, it was a bit of a challenge because I have to remember what I have learned for OOP back in semester 2. But after looking back, it became clearer to me and implemented this knowledge to the assignment. Some of them like the new and delete keyword needed more research to understand more that I have to look up online. Luckily there is an eBook for C++ OOP by Bjarne Stroustrup that I can use whenever it is needed.

Bibliography

Isocpp (2022) *CppCoreGuidelines/cppcoreguidelines.md at master · ISOCPP/cppcoreguidelines*, *GitHub*. Available at: https://github.com/isocpp/CppCoreGuidelines/blob/master/CppCoreGuidelines.md