CS 303 Extra Credit Project 1-A

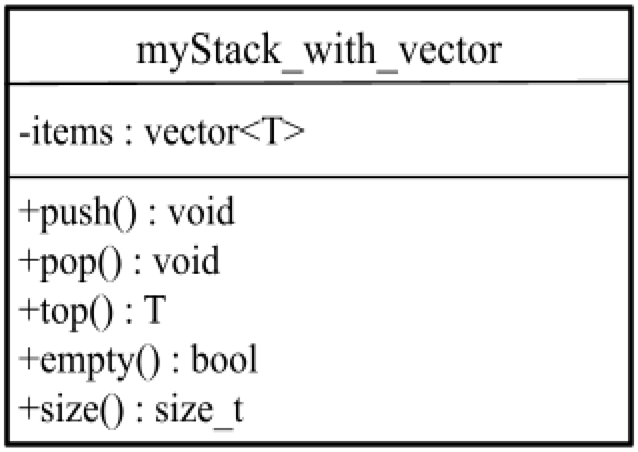
UMKC - Professor Syed Jawad Hussain Shah

Alyssa Barbee

Saw Nwe

Aarohi Patel

UML Class Diagram:



Within our project, we used a class called myStack_with_vector, for implementation. This class had a private member attribute, a vector of items of type integer. We used a template class so the items could be any type. For the public attributes of the class, there are only functions in this function. There is a push, pop, top, empty, and size function. All of these functions are defined as standard and basic. We also have some helper functions.

Efficiency of Algorithms:

1. Precedence Function:
    a. Worst Case: O(1); Best Case: O(1)
    b. Since this is just a function that makes comparisons to figure out what the precedence of the evaluator is (within PEMDAS), there is not really a way to make it better. It just needs to be checked to make sure what evaluator has been given. It is a constant function because there are no loops or anything that would make it linear or something more complex.
2. Apply Operator Function:
    a. Worst Case: O(1); Best Case: O(1)

   b. This is yet another function that makes comparisons until it figures out which characteristic is related to the input. Because of this, the comparisons are necessary, because without them, we cannot figure out which kind of operator is given, and therefore which calculation to perform. It can be a constant operation if the operation given is a member of the first comparison. For example, if the string "a+b" is given, and the first line is checking for addition and adding the two sides of the equation. In this case it would be the best case situation. Also, this same logic can be applied to the precedence function above. It is a constant function because there are no loops or anything that would make it linear or something more complex.

3. Infix Evaluator Function:
   a. Worst Case: $O(n^2)$; Best Case: $O(n)$
   b. This is the main function that calls almost all other functions. Because of this, it is pretty complex and involves both a for and while loop, making it $O(n^2)$, for worst and average cases. However, if all conditions are met ideally, then it can be $O(n)$. There could possibly be a better way to get the same functionality, but we did what we could think of and

Individual Contributions:

- Alyssa
  - As part of the group project I added and updated the code to fit the requirements for the project. For example, I added any missing aspects of the code (ex. Beyond addition and subtraction I added operators such as '<' and '!'). I made sure the code was consistent and added any other necessary comments.

- Saw
  - As part of the group project, my responsibilities were to design and implement the header file, the infix expression parser, evaluator and other functions. I designed and implemented the header file, the essential classes, functions and data structures required for the project. This includes declaring the stack using vector, utility functions for operator, precedence and methods for evaluating expressions. I built the general evaluation functionality using two stacks: operands and operators ensuring proper handling of precedence rules.
- Aarohi - **Please do not apply a grade to the assignment on Canvas for this member**
  - As a member of this group, my responsibilities included creating the components of the main.cpp file and putting together most of the report file. I created the vector of expressions to evaluate through our infix operator and created the for loop that called the evaluator function for each iteration. I also formatted the report file, created the UML for our class, recorded complexities of functions, cited our sources, and described my contributions to the project.

References:

Copilot/ChatGPT

Lecture 7 Slides

Knowledge from all other slides throughout the class

Koffman, Elliot B., and Paul A. T. Wolfgang. *Objects, Abstraction, Data Structures and Design Using C++*. John Wiley & Sons, Inc, 2006.

"GeeksforGeeks | a Computer Science Portal for Geeks." *GeeksforGeeks*, geeksforgeeks.org.