**LAB REPORT ON**

**Data Structures**



**Lab No. 2**

**Date: 2020/09/11**

**Topic: Stack**

**Submitted By**  **Submitted To**

Tapendra Pandey

BCT-D Department of Electronics

075BCT093 and Computer Engineering

**INTRODUCTION**

**Stack:** Stack is the linear data structure, which follows the Last in First out Order (LIFO). The last element that inserted is the one, which removed first. When an element is inserted in a stack, the concept is called as push and when an element is removed from the stack, the concept is called as pop.

Trying to pop out an empty stack is called as underflow and trying to push an element in a full stack is called as overflow. The pointer that tracks the topmost element of stack is known as top pointer.

**Applications of stack:**

1. Balancing of symbols

2. Infix to Postfix /Prefix conversion

3. Redo-undo features at many places like image editors

4. Forward and backward feature in web browsers

5. Used in many algorithms like Tower of Hanoi, tree traversals, stock span problem, and histogram problem.

**Infix, Postfix and Prefix notation**

**Infix:**

An infix expression takes only one single letter or it has two letters with operators (+,-,\*,/)in between them or complete two infix expression with operators (+, -,\*,/) in between them.

**Prefix:**

An prefix expression takes, only one single letter or it has two letters in sequence with (+,-,\*,/ ) before them or complete two prefix expression with operators (+, -,\*,/) in between them.

**Postfix:**

A postfix expression takes only one single letter or it has two letters in sequence with (+,-,\*,/) after them or complete two postfix expression with operators (+, -,\*,/) in between them.

**ALGORITHMS**

**I. Push and pop in stack**

1: Declare stack ‘abc’

2: Push to stack

3: While ‘abc’ is not empty, Get top element

4: Pop from stack

**II. Convert infix to postfix expression**

1. Scan the infix expression from left to right.

2. If the scanned character is an operand, output it.

3. Else,

a) If the precedence of the scanned operator is greater than the precedence of the operator in the stack (or the stack is empty or the stack contains a ‘(‘ ), push it.

b) Else, pop all the operators from the stack, which are greater than or equal to in precedence than that of the scanned operator. After doing that push the scanned operator to the stack. (If parenthesis is encountered while popping then stop and push the scanned operator in the stack.)

4. If the scanned character is an ‘(‘, push it to the stack.

5. If the scanned character is an ‘)’, pop the stack and and output it until a ‘(‘is encountered, and discard both the parenthesis.

6. Repeat steps 2-6 until infix expression is scanned.

7. Print the output

8. Pop and output from the stack until it is not empty.

**III. Convert infix operation to prefix operation**

1: Reverse the infix expression, ‘(‘= ‘)’ and ‘)’ =‘(‘.

2: Obtain the postfix expression of the modified expression

3: Reverse the postfix expression

**IV: Evaluate postfix operation**

1: Create a stack to store operands (or values).

2: Scan the given expression and do following for every scanned element.

a) If the element is a number, push it into the stack

b) If the element is a operator, pop operands for the operator from stack. Evaluate the operator and push the result back to the stack

3: When the expression is ended, the number in the stack is the final answer

**V: Check for balanced parenthesis**

1: Declare a character stack S.

2: Now traverse the expression string exp.

a) If the current character is a starting bracket (‘(‘or ‘{‘or ‘[‘) then push it to stack.

b) If the current character is a closing bracket (‘)’ or ‘}’ or ‘]’) then pop from stack and if the popped character is the matching starting bracket then fine else parenthesis are not balanced.

3: After complete traversal, if there is some starting bracket left in stack then “not balanced”.

**Discussion and Conclusion**

In this lab, we learnt about the ADT- Stacks and its some applications. We implemented the knowledge and did push pop operations, converted infix to prefix, infix to postfix, evaluated the postfix expression and checked for balanced parentheses in lab.