PICT, PUNE

DSA

23224

Assignment No. 02

Title: Expression conversion

Aim: To implement expression conversion using Stack Data structures

problem statement: Implement stack an abstract data type using singly LL and use this ADT for conversion of infix expression to postfix, prefix and evalution of postfix and prefix exp

objective:

1. To understand the concept and implementation of stack data structure using SSL.
2. To understand the concept of evalution of exp
3. To understand the concept of evalutn of exp

Outcomes:

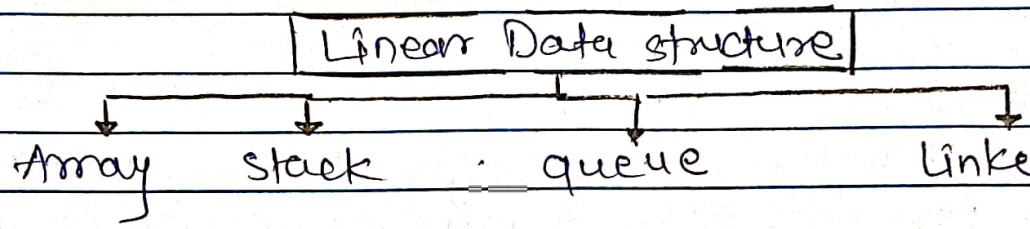
1. Implement stack as an ADT.
2. Implement applications of stack i.e expression, conversion and evaluation.

Theory:

1) Concept of Linear data structure:
The data structure where data items are organized sequentialy or linearly one after another is called linear data structure

PICT, PUNE

These kind of data structures are very easy to implement because memory of computer also has been organized in linear fashion



2) Example of linear data structure:

Example: Array
Stack
Queue
Linked list

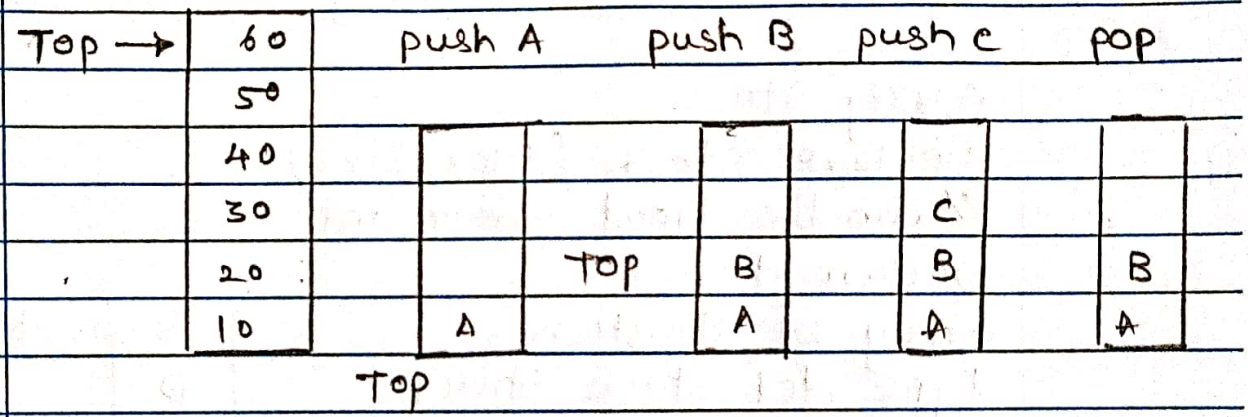
3) Stack:

(i) Concept:
ordered list in which insertion and deletion are made at one end called the top.
Stack principle: LIFO (last in first out)
To retrieve o^{th} element needs to remove n-1 element

(ii) Defination of stack:
A stack is an ordered list in which all insertion and deletions are made at the end called the top

If we have to make stack of elements 10, 20, 30, 40, 50, 60 then 10 will be bottom most element and 60 will be top most element in stack

(iii) Terminology and diagram:



(iv) ADT of Stack

precondition:

1) Stack_full(): check stack is full or not. If full then we can't insert any element in stack.

2) Stack_empty(): check stack is full or not. If not the we can insert element.

Operations:

1) push: by this operation we can push elements into the stack before performing push we must check the stack_full() condition

PICT, PUNE

2) pop: by this operation can remove the elements from stack. before popping elements we should check stack_empty() condition.

⑦ Realisation of ADT using:

Array Stack:

Declare stack (Max_Size)

// can be float, char, int

Declare top

Array of structure:

type def stack struct

{

x [Max_Size]

top

}

Initialize top = -1

S	top=4
D	
C	
B	
A	0

PICT, PUNE

2) pop : by this operations can remove the elements from stack before popping elements we should check stack_empty() condition.

Application of stack :

- Expression conversion
  - Infix to postfix
  - Infix to prefix
  - prefix to infix
  - prefix to infix
- Expression evalution
- passing
- Simulation of recursion
- Game playing find paths
- exhaustive searching
- well format paranthesis
- Reversal of string
- Tree or graph traversal
- Backtracing

Expression conversion and stack :

1) **Need for expression conversion :**
Usually infix notation is used but the problem with infix is that is lot of one head in computing the result for infix expression which result in loss of efficiency.
In the alternative form there is no need for paranthesis which in no need for repeated scanning of tree expression alternative forms procedure and associativity among operator is already accounted for

2) **Advantages of polish notations :**
- Expression can be shown without paranthesis
- It is convient to evalute formula on stacks
- polish notation timinates problems faced by precedence.
- The complete expression can be passed in one Traversal

PICT, PUNE

Working:

1) Infix to postfix conversion:

During the scan the expression an append is immediatly added to the output string while the operator stack stores the operators & left paranthesis as soon as they appear handles sub expressions and mangoes the order of precedence and associtivity of operations.

2) Infix to infix conversion:

For this conversion the infix expression is scanned from right instead of left to right like a postfix conversion. The operand are added directly to output string while the stack contains the operators and parenthesis

3) Postfix evalution:

This is done by adding the operand onto a stack and removing two whenever an operator is read in postfix expression string. The results of the operation is then added to the stack and process is repeated until there is only one element remaining in the stack

4) Prefix evaluation

This is done very similar to postfix evaluation except instead of traversing the string left to right. The presence is repeated until there is only one element left in stack.

Test cases / validations:

validations:

1) No of operator & operand relationship
2) well formed peranthesis

Test cases:

1) Based on precedence of operators:

Declaration of Data structures:

In this program, the data structures used are stack and linked list.

Conclusion:

Succesfully implemented the assignment using stack as stack as ADT to do the infix to post-fix, prefix expression
learned stack data structures, its implemented using stack & its application in expression conversion as a temporary data structures.