SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING-IOT Including CS&BCT

COURSE NAME : 19ITT201 - DATA STRUCTURES

II YEAR / III SEMESTER

Unit 1- LINEAR STRUCTURES AND TREES

Topic 10 : Applications of stack and queue

Problem

1. Write the postfix notation for following expression

   (A+B)*C- (D-E)^F

2.The following expression 6 5 2 3 + 8 * + 3 + *

3. Give the postfix form for the infix expression (m-n) / p * q + r ^ s * t

4. Evaluate the infix expression

   3+8*4/2-(8-3) covert to postfix operation

5. Evaluate the postfix expression

   752+*415-/- convert to infix operation

Applications of Stack

- Following is the various Applications of Stack in Data Structure
- Evaluation of Arithmetic Expressions
- Tower of Hanoi
- Reverse a Data
- Balance Paraenthesis

Applications of stack –Cont..

- An arithmetic expression can be written in three different but equivalent notations, i.e., without changing the essence or output of an expression
- These notations are –
- Infix Notation
- Prefix (Polish) Notation
- Postfix (Reverse-Polish) Notation

Applications of stack –Cont..

- We write expression in **infix** notation, e.g. a - b

+ c, where operators are used **in**-between operands.

- It is easy for us humans to read, write, and speak in infix notation but the same does not go well with computing devices.
- An algorithm to process infix notation could be difficult and costly in terms of time and space consumption.

Applications of stack –Cont..

- In this notation, operator is **prefix**ed to operands, i.e. operator is written ahead of operands. For example, **+ab**. This is equivalent to its infix notation **a + b**. Prefix notation is also known as **Polish Notation**.

Applications of stack –Cont..

- This notation style is known as **Reversed Polish Notation.** In this notation style, the operator is **postfix**ed to the operands i.e., the operator is written after the operands. For example, **ab+.** This is equivalent to its infix notation **a + b.**

The following table briefly tries to show difference in all three notations

Sr.No.	Infix Notation	Prefix Notation	Postfix Notation
1	a + b	+ a b	a b +
2	(a + b) * c	* + a b c	a b + c *
3	a * (b + c)	* a + b c	a b c + *
4	a / b + c / d	+ / a b / c d	a b / c d / +
5	(a + b) * (c + d)	* + a b + c d	a b + c d + *
6	((a + b) * c) - d	- * + a b c d	a b + c * d -

Applications of stack –Cont..

Parsing Expressions

- It is not a very efficient way to design an algorithm or program to parse infix notations. Instead, these infix notations are first converted into either postfix or prefix notations and then computed.
- Precedence
- When an operand is in between two different operators, which operator will take the operand first, is decided by the precedence of an operator over others. For example –
- As multiplication operation has precedence over addition, b

* c will be evaluated first. A table of operator precedence is provided later.

Applications of stack –Cont..

- Associatively describes the rule where operators with the same precedence appear in an expression.
- For example, in expression a + b − c, both + and – have the same precedence, then which part of the expression will be evaluated first, is determined by associatively of those operators.
- Here, both + and − are left associative, so the expression will be evaluated as **(a + b) − c**.
- Precedence and associatively determines the order of evaluation of an expression. Following is an operator precedence and associatively table (highest to lowest)

–

Sr.No.	Operator	Precedence	Associativity
1	Exponentiation ^	Highest	Right Associative
2	Multiplication (∗) & Division (/)	Second Highest	Left Associative
3	Addition (+) & Subtraction (−)	Lowest	Left Associative

Applications of stack –Cont..

Sr.No.	Arithmetic Expression			
1	infix	Operand	Operator	Operand
2	Prefix	Operator	Operand	Operand
3	Postfix	Operand	Operand	Operator

Algorithm for Prefix to Postfix

- Read the Prefix expression in reverse order (from right to left)
- If the symbol is an operand, then push it onto the Stack
- If the symbol is an operator, then pop two operands from the Stack

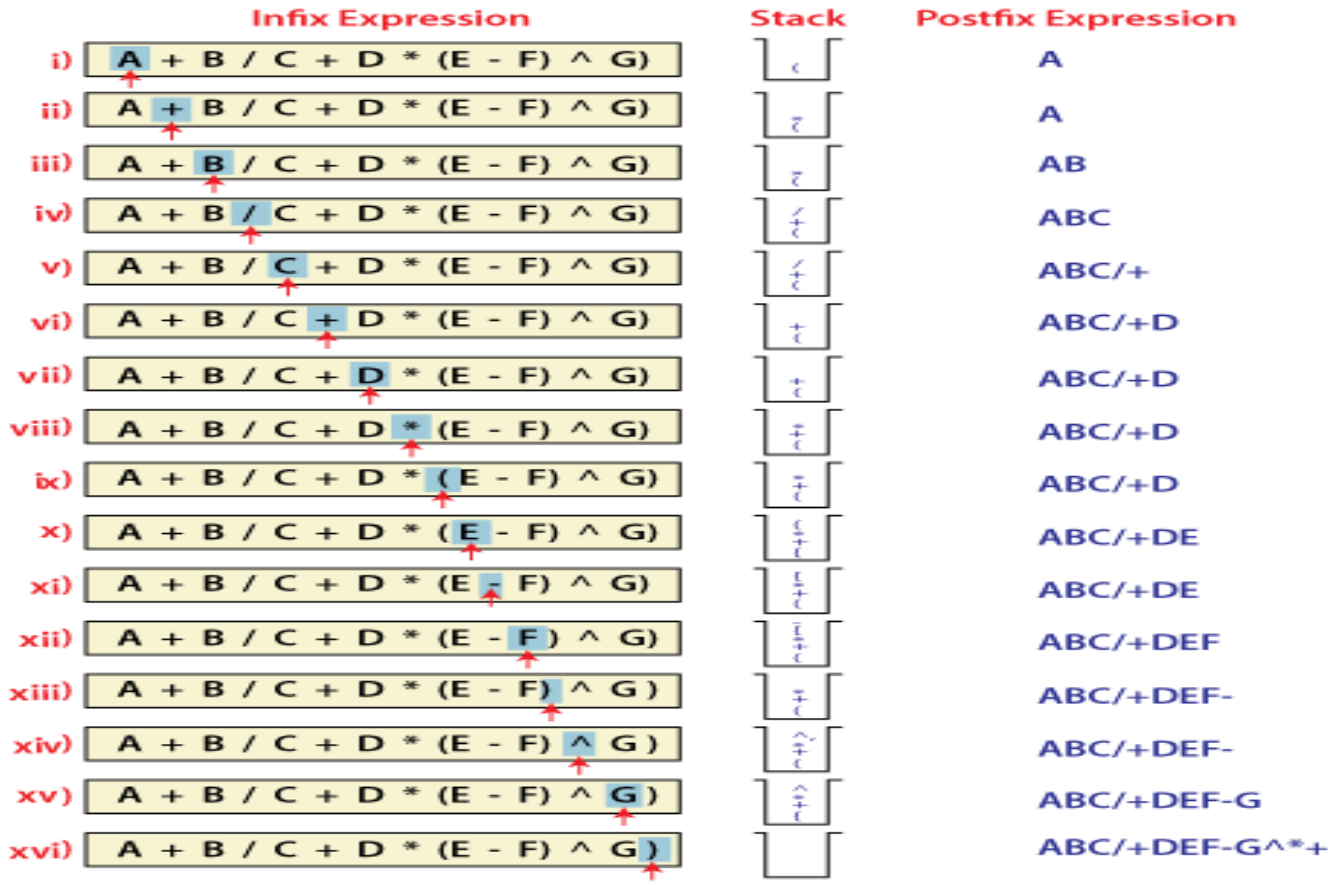
Create a string by concatenating the two operands and the operator after them.

string = operand1 + operand2 + operator

And push the resultant string back to Stack

- Repeat the above steps until end of Prefix expression.

Let's take the example of Converting an infix expression into a postfix expression.

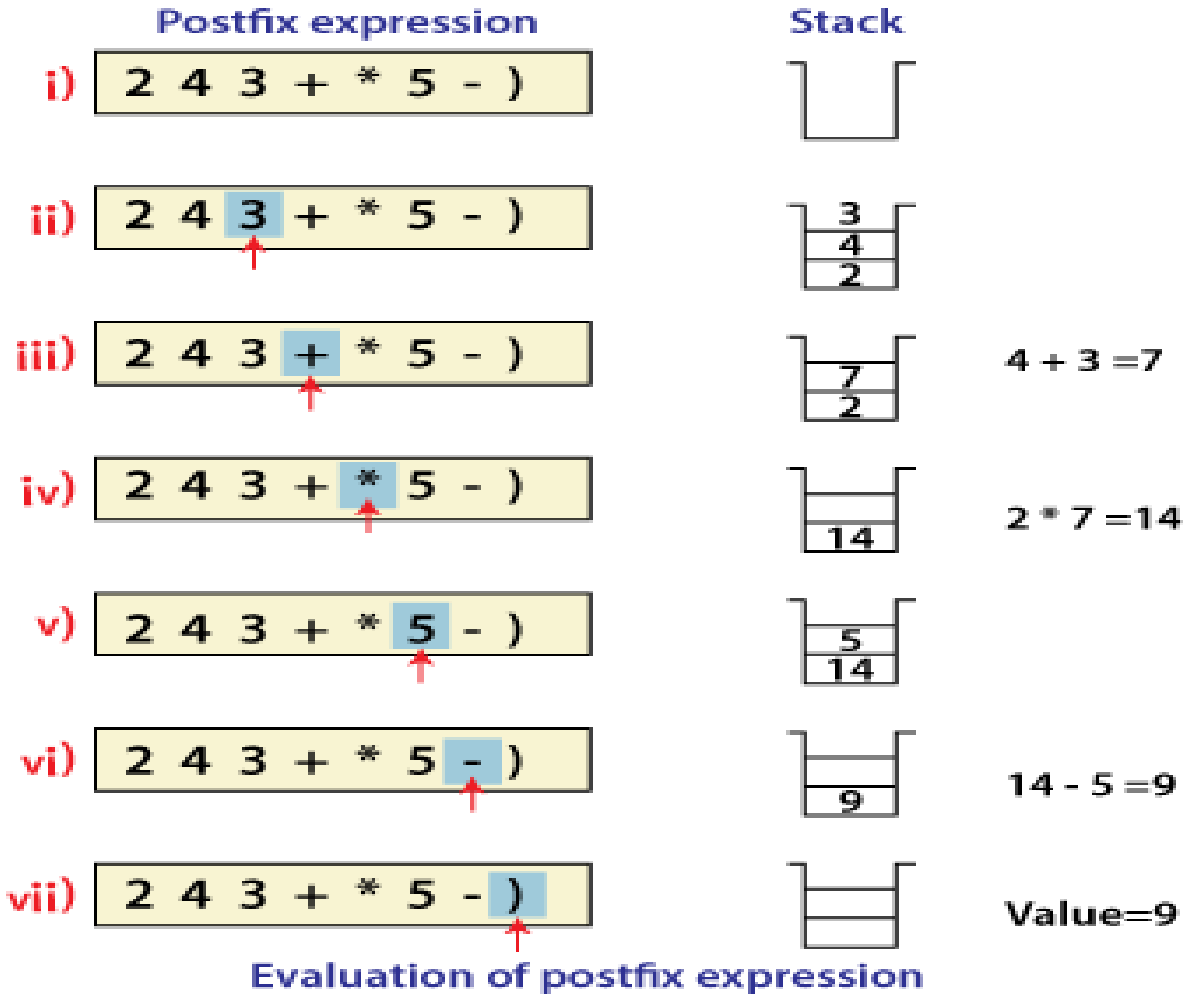


Infix to Postfix

Example:

Now let us consider the following infix expression 2 * (4+3) - 5. Its equivalent postfix expression is 2 4 3 + * 5.

The following step illustrates how this postfix expression is evaluated.



Evaluation of Postfix Expression

Postfix → a b c * + d - Let, a = 4, b = 3, c = 2, d = 5

Postfix → 4 3 2 * + 5 -

Operator/Operand	Action	Stack
4	Push	4
3	Push	4, 3
2	Push	4, 3, 2
*	Pop (2, 3) and 3*2 = 6 then Push 6	4, 6
+	Pop (6, 4) and 4+6 = 10 then Push 10	10
5	Push	10, 5
-	Pop (5, 10) and 10-5 = 5 then Push 5	5

Result of Expression

Step No.	Value of i	Operation	Stack
1	2	Push 2 in stack	**2**
2	3	Push 3 in stack	**3** **2**
3	4	Push 4 is stack	**4** **3** **2**
4	+	Pop 2 elements from stack and perform addition operation. And push result back to stack. i.e. 4+3 = 7	7 **2**
5	*	Pop 2 elements from stack and perform multiplication operation. And push result back to stack. i.e. 7 * 2 = 14	**14**
6	6	Push 6 in stack	**6** **14**
7	-	Pop 2 elements from stack and perform substraction operation. And push result back to stack. i.e. 14 - 6 = 8	**8**
8		Pop result from stack and display	

Activity

MCQ

1. What data structure is used when converting an infix notation to prefix notation?
   a) Stack
   b) Queue
   c) B-Trees
   d) Linked-list
2. What would be the Prefix notation for the given equation? A+(B*C)

a) +A*CB

b) *B+AC

c) +A*BC

d) *A+CB

Advantages

1. A stack is used when a variable is not used outside that function.
2. It allows you to control how memory is allocated and deallocated.
3. Stack automatically cleans up the object.
4. Not easily corrupted
5. Variables cannot be resized.

Disadvantages

1. Stack memory is very limited.
2. Creating too many objects on the stack can increase the risk of stack overflow.
3. Random access is not possible.
4. Variable storage will be overwritten, which sometimes leads to undefined behavior of the function or program.
5. The stack will fall outside of the memory area, which might lead to an abnormal termination.

Assessment 1

1. List out the advantages of application of stack based implementation

a)_____

b)_____

c)_____

d)_____

2. Identify the disadvantages of application of stack based implementation

a)_____

b)_____

c)_____

d)_____

REFERENCES

1. M. A. Weiss, “Data Structures and Algorithm Analysis in C”, Pearson Education, 8th Edition, 2007. [Unit I, II, III, IV,V]

2. A. V. Aho, J. E. Hopcroft and J. D. Ullman, “Data Structures and Algorithms”, Pearson Education, 2nd Edition, 2007 [Unit IV].

3. A.M.Tenenbaum, Y. Langsam and M. J. Augenstein, “Data Structures using C”,PearsonEducation, 1st Edition, 2003.(UNIT I,II,V)

4.https://www.geeksforgeeks.org/queue-set-1introduction-and-array-implementation/

THANK YOU