Subject : Data structure
Topic: Infix to postfix conversion
Name of the teacher: Lisna Thomas
Academic year: 2020-2021

INFIX NOTATION

- **Infix notation** is the common arithmetic and logical formula notation, in which **operators are written** infix-style **between the operands** they act on
- E.g. **A + B**

POSTFIX NOTATION

- In Postfix notation, the **operator** comes **after the Operand**.
- For example, the Infix expression **A+B** will be written as **AB+** in its **Postfix Notation**.
- Postfix is also called '**Reverse Polish Notation**'

Prefix notation

- In Prefix notation, the **operator** comes **before the operand**.
- The Infix expression **A+B** will be written as **+AB** in its Prefix Notation.
- Prefix is also called '**Polish Notation**'

CONVERSION FROM INFIX TO POSTFIX ALGORITHM

Step1

- **Scan the Infix expression** from **left to right** for tokens (Operators, Operands & Parentheses) and perform the steps 2 to 5 for each token in the Expression

ALGORITHM

Step2

- If token is **operand, Append it** in postfix expression

Step3

- If token is a **left parentheses “(“**, **push it** in stack.

ALGORITHM

Step4

- If token is an **operator**,
  - **Pop all the operators** which are of higher or equal precedence then the incoming token and **append them** (in the same order) to the output Expression.
  - After popping out all such operators, **push the new token** on stack.

ALGORITHM

Step5

- If **“)”** right parentheses is found,
  - **Pop all the operators** from the Stack and append them to Output String, **till** you **encounter the Opening Parenthesis “(“**.
  - **Pop the left parenthesis** but don’t append it to the output string (Postfix notation does not have brackets).

ALGORITHM

Step6

- When all tokens of Infix expression have been scanned. **Pop all the elements from the stack** and **append** them to the Output String.

- The Output string is the Corresponding **Postfix Notation**.

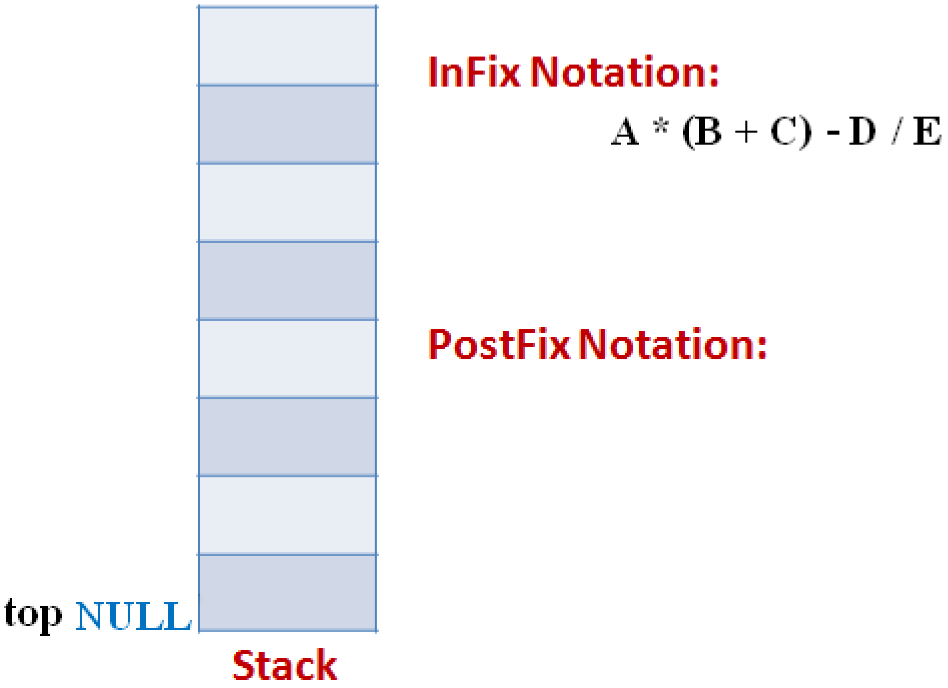
Example

- Let the incoming the Infix expression be:

A * (B + C) – D / E

Stage 1: **Stack is empty** and we only have the Infix

Expression.



InFix Notation:

A * (B + C) - D / E

PostFix Notation:

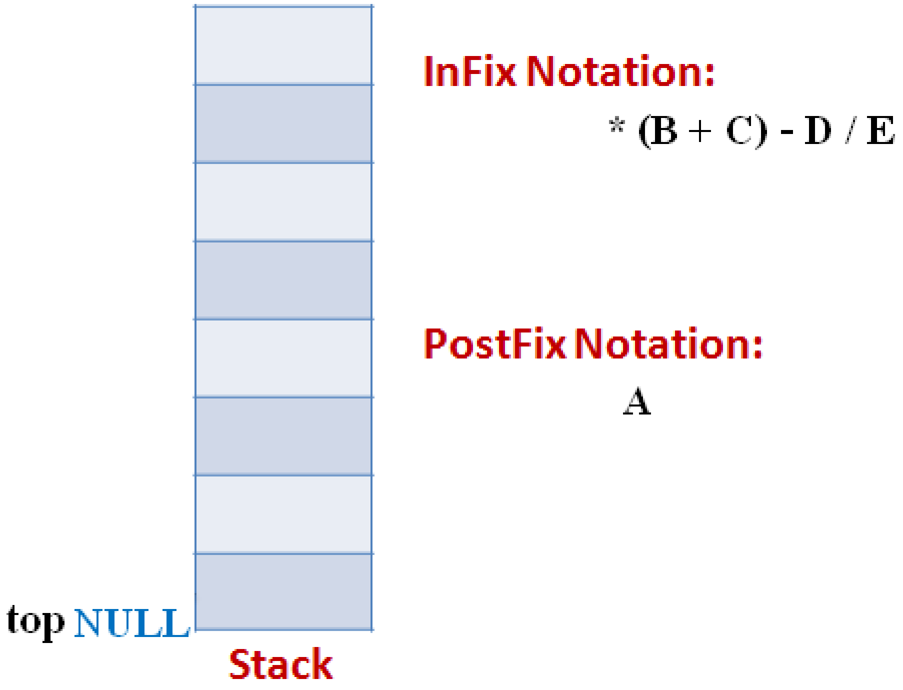
top NULL

Stack

EXAMPLE

Stage 2

- The first token is **Operand A** Operands are Appended to the Output as it is.



InFix Notation:

* (B + C) - D / E

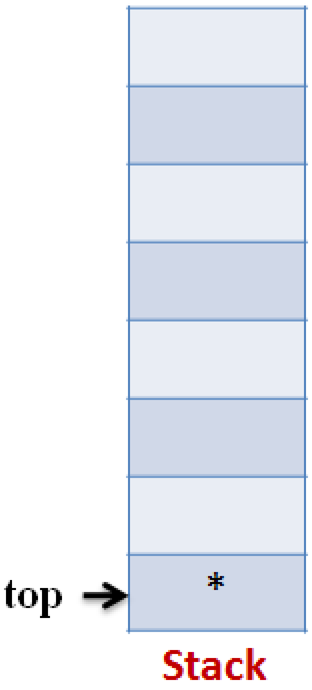
PostFix Notation:

A

EXAMPLE

Stage 3

- Next token is ***** Since **Stack is empty (top==NULL)** it is **pushed into the Stack**



InFix Notation:

(B + C) - D / E

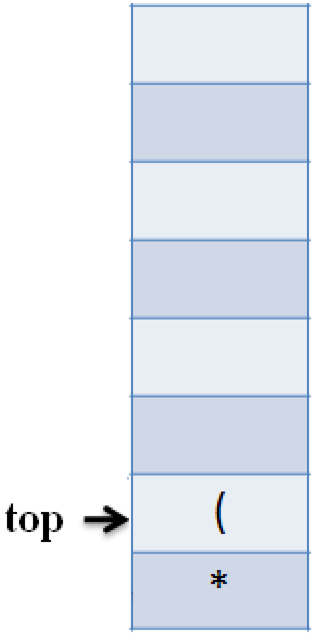
PostFix Notation:

A

EXAMPLE

Stage 4

- Next token is **(** the precedence of open-parenthesis, when it is to go inside, is maximum.
- But when another operator is to come on the top of ,,(,,then its precedence is least.



Stack

InFix Notation:

B + C) - D / E

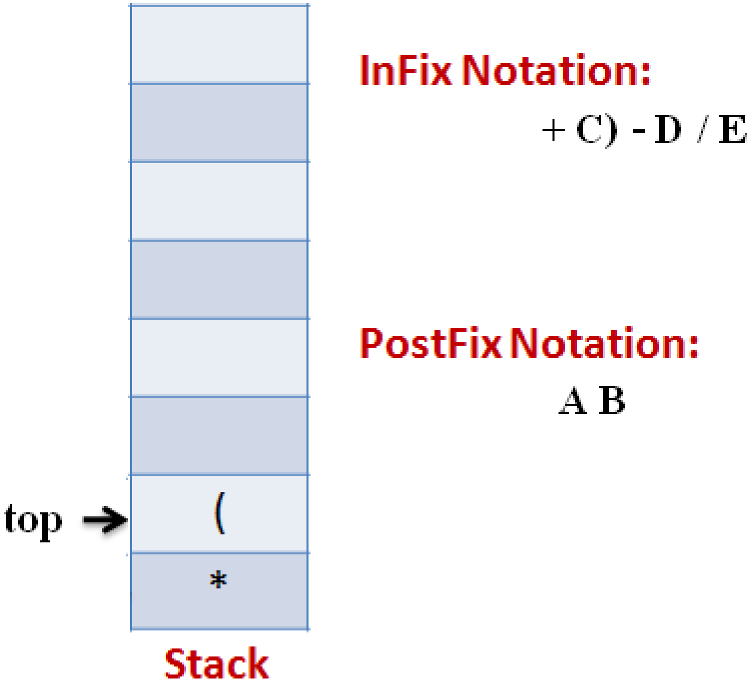
PostFix Notation:

A

Example

Stage 5

- Next token, **B** is an operand which will go to the Output expression as it is



InFix Notation:

+ C) - D / E

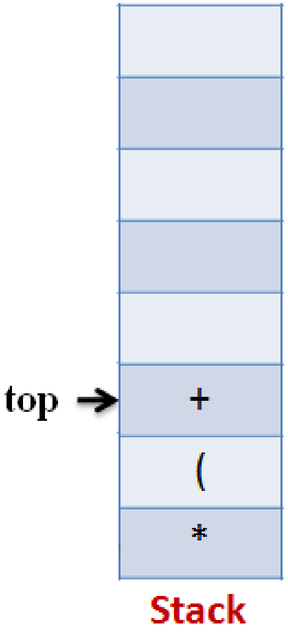
PostFix Notation:

A B

EXAMPLE

Stage 6

- Next token, **+** is operator, We consider the precedence of **top element in the Stack**, **„(‘**. The outgoing precedence of open parenthesis is the least (refer point 4. Above). So **+** gets **pushed into the Stack**



InFix Notation:

C) - D / E

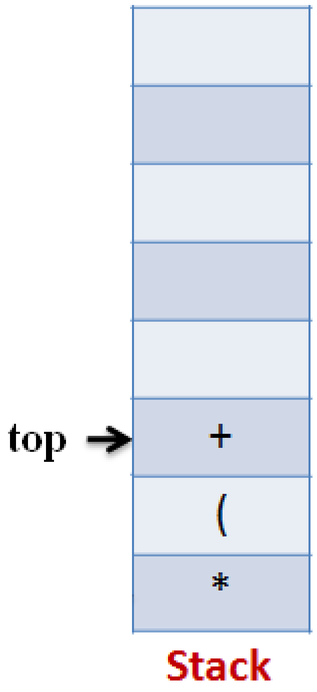
PostFix Notation:

A B

EXAMPLE

Stage 7

- Next token, **C**, is appended to the output



InFix Notation:

) - D / E

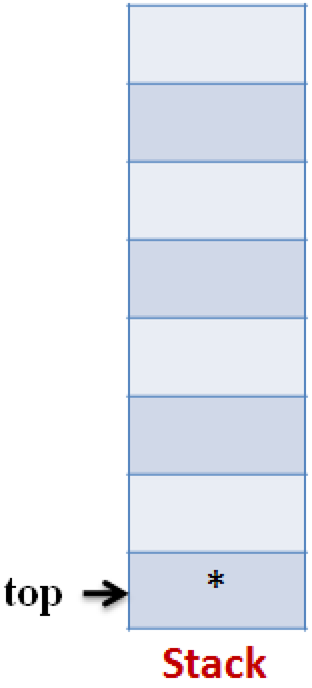
PostFix Notation:

A B C

Example

Stage 8

- Next token **)**, means that **pop all the elements from Stack** and **append them to the output** expression till we read an opening parenthesis.



InFix Notation:

- D / E

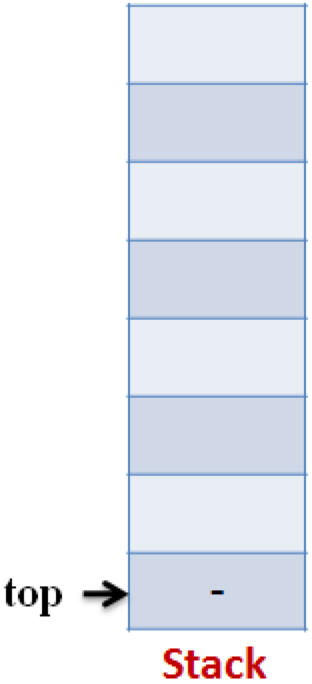
PostFix Notation:

A B C +

EXAMPLE

Stage 9

- Next token, **-**, is an operator. The precedence of operator on the top of Stack „*****‘ is more than that of Minus. So we **pop multiply** and **append it to output** expression. Then **push minus in the Stack**.



InFix Notation:

D / E

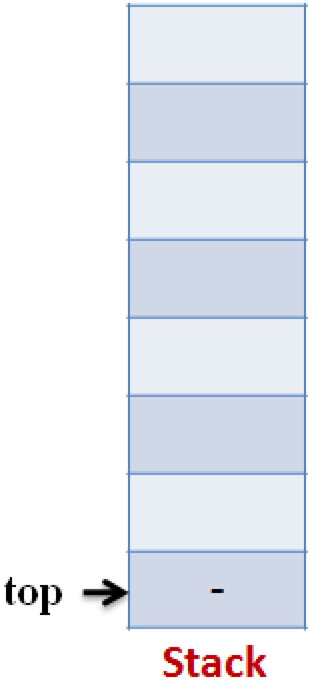
PostFix Notation:

A B C + *

EXAMPLE

Stage 10

- Next, Operand ‘**D**‘ gets **appended to the output**.



InFix Notation:

/ E

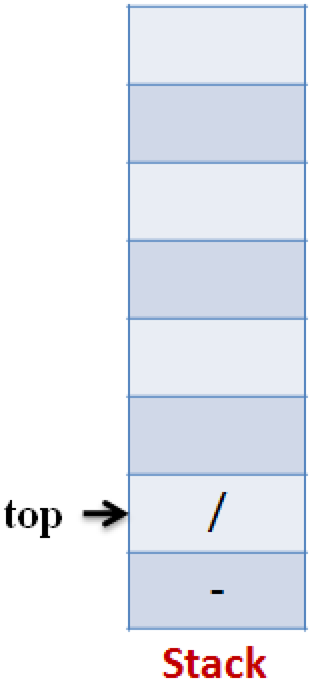
PostFix Notation:

A B C + * D

Example

Stage 11

- ⋆ Next, we will insert the **division** operator into the Stack because its precedence is more than that of minus.



InFix Notation:

E

PostFix Notation:

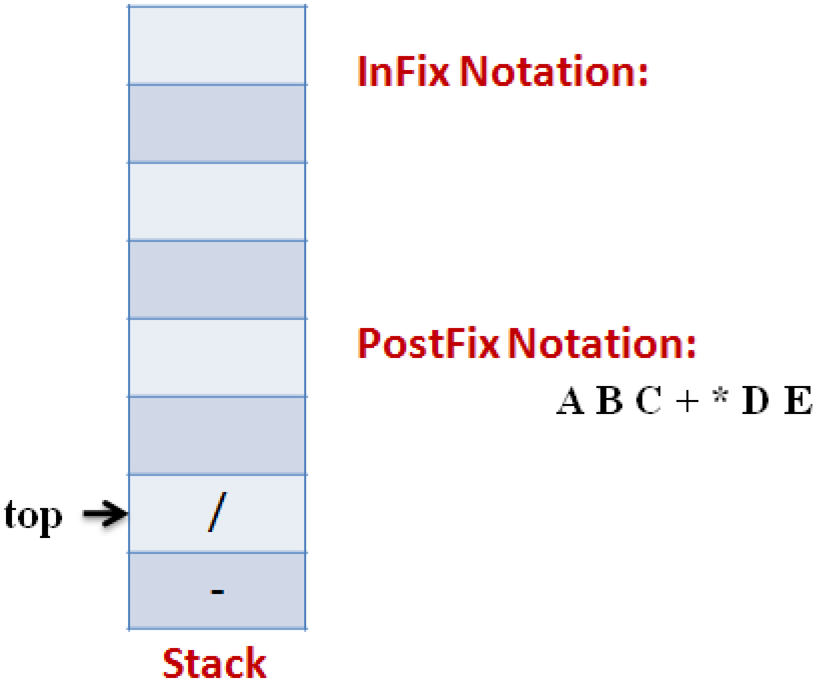
A B C + * D

Example

Stage 12

- The last token, **E**, is an operand, so we **insert it to the output**

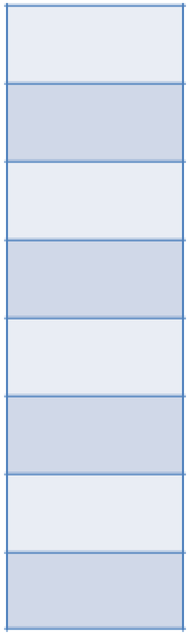
  Expression as it is.



EXAMPLE

Stage 13

- The input Expression is complete now. So we **pop the Stack** and **Append it to the Output Expression** as we pop it.



InFix Notation:

PostFix Notation:

A B C + * D E / -

top = NULL

Stack

The End

……. Thank You ……..

www.ustudy.in