IT-1201: Data Structures

for

1st Year 2nd Semester of B.Sc (Honors) in IT (5th Batch)

Lecture: 08

Stack-1

Prepared by:

K M Akkas Ali
akkas_khan@yahoo.com, akkas@juniv.edu
Assistant Professor
Institute of Information Technology (IIT)
Jahangirnagar University, Dhaka-1342

Prepared by: K M Akkas Ali, Assistant Pro

Lecture-08: Stack-1

Objectives of this Lecture:

- ❖ What is a Stack?
- ❖ Array implementation of stacks
- ❖ Operations on a Stack
- ❖ Arithmetic expressions
- ❖ Infix expressions into postfix Expressions

Prepared by: K M Akkas Ali, Assistant Professor, IIT, JU

What is Stack?

- ➢ A stack is a homogeneous collection of elements in which an element may be inserted or deleted only at one end, called the top of the stack.
- ➢ Formally this type of stack is called a Last In, First Out (LIFO) stack.
- ➢ Terminology is used for two basic operations associated with stacks are:
  - ❑ “Push” is the term used to insert an element into a stack.
  - ❑ “Pop” is the term used to delete an element from a stack.
- ➢ The elements may be popped from the stack only in the reverse order of that in which they were pushed onto the stack.

Prepared by: K M Akkas Ali, Assistant Professor, IIT, JU

Diagram of Stack:

- Suppose, the following 6 elements are pushed, in order, onto an empty stack labeled as STACK:

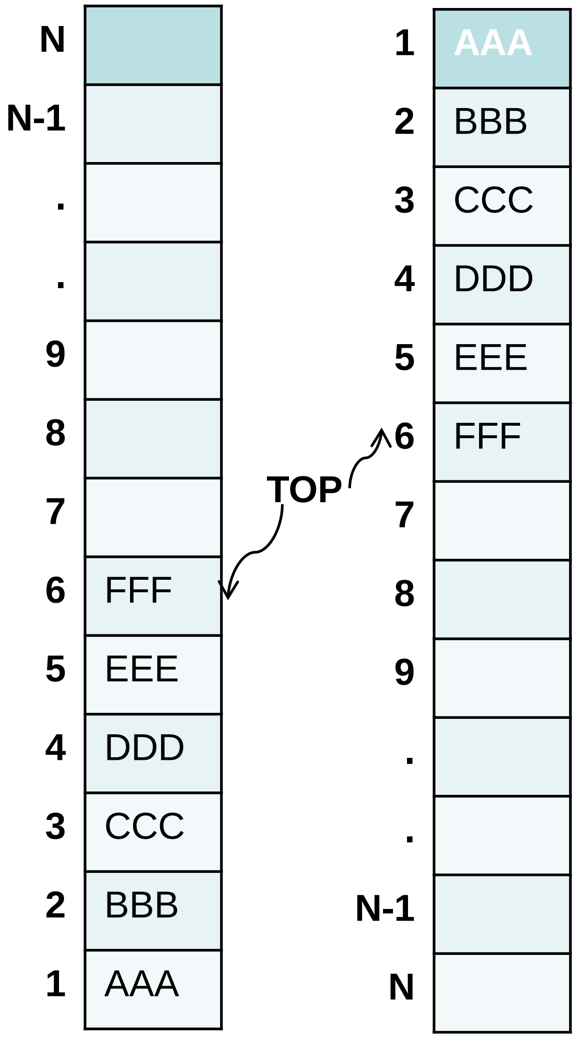
  STACK: AAA, BBB, CCC, DDD, EEE, FFF

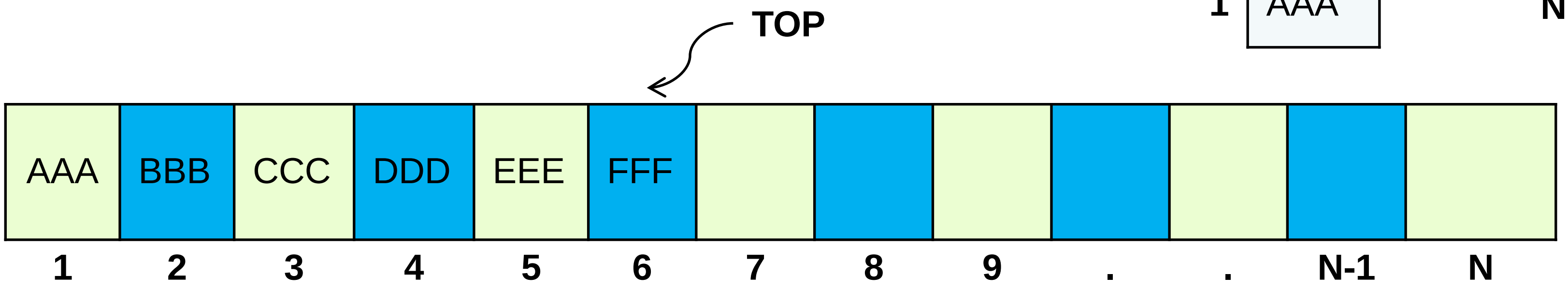
- Figure below shows three ways of depicting such a stack.

- Note that, regardless of the way a stack is described, its underlying property is that-

  - insertions and deletions can occur only at the top of the stack.

- This means that, item EEE can not be deleted from the stack before FFF is deleted. Similarly, DDD can not be deleted before EEE and FFF are deleted, and so on.





Representing Stack Using Array

- Stacks may be represented in computer in various ways. Two popular ways are-

  1. By means of one-way list or linked list
  2. By means of a linear array

- To implement a stack using array, items are inserted and removed at the same end (called the TOP of the stack)

- To implement a stack using array, we need-

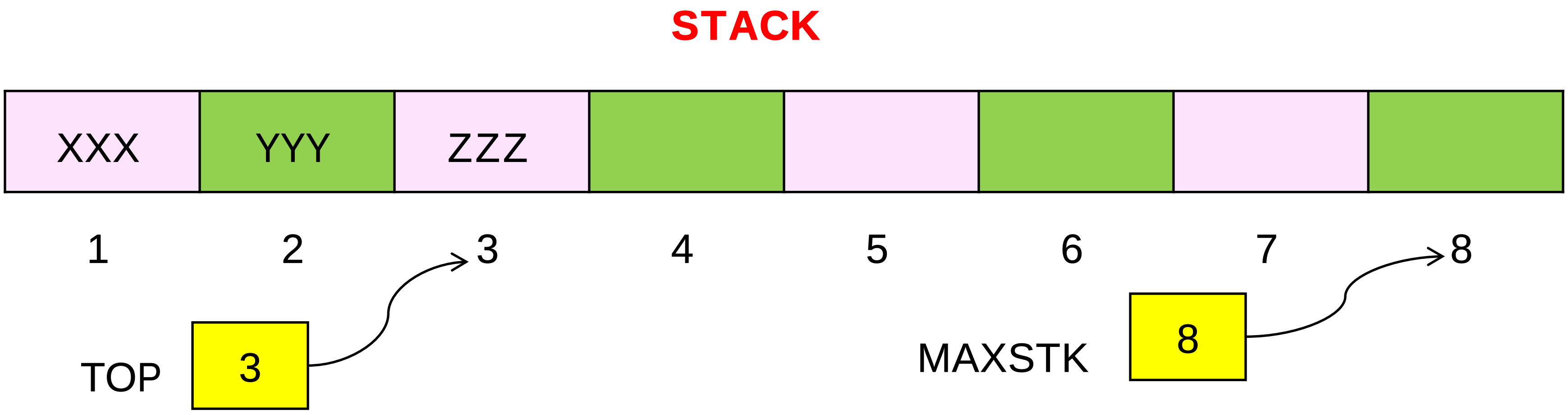
  - the name of the linear array itself (e.g. **STACK**).
  - a pointer variable **TOP**, which contains the location of the top element of the stack or which indicates how many elements are there in the stack.
  - a variable (e.g. **MAXSTK**) which gives the maximum number of elements that can be held by the stack.

Prepared by: K M Akkas Ali, Assistant Professor, IIT, JU

Array Representation of Stacks

- ➢ Figure below shows the representation of stack using a linear array.

STACK



- ➢ Since TOP=3, the stack has three elements- XXX, YYY and ZZZ. It also means, the location of the top element is 3.
- ➢ Since, MAXSTK=8, maximum number of elements that can be held by the stack is 8, so five more items can be added in the stack.
- ➢ Following two conditions satisfy whether the stack is overflow or underflow.
  - ❖ If TOP=0 or TOP=NULL, then the stack is empty.
  - ❖ If TOP=MAXSTK, then the stack is already filled.

Prepared by: K M Akkas Ali, Assistant Professor, IIT, JU

Operations on a Stack

Let us assume that maximize size of an stack is n.

Then following basic operations are usually performed on an stack:

Push:

- ❑ This operation adds or pushes another item onto the stack.
- ❑ The number of items on the stack is less than n.

Pop:

- ❑ This operation removes an item from the stack.
- ❑ The number of items on the stack must be greater than 0.

Top:

- ❑ This operation returns the value of the item at the top of the stack.
- ❑ Note: It does not remove that item.

Is Empty:

- ❑ This operation returns true if the stack is empty and false if it is not.

Is Full:

- ❑ This operation returns true if the stack is full and false if it is not.

Prepared by: K M Akkas Ali, Assistant Professor, IIT, JU

Overflow and Underflow

- The operation of adding (pushing) an item onto a stack may be implemented by the PUSH procedure.
    - In executing the PUSH procedure, one must first test whether there is room in the stack for the new item; if not, then we have the condition known as overflow.
- The operation of removing (popping) an item from a stack may be implemented by the POP procedure.
    - In executing the POP procedure, one must first test whether there is an element in the stack to be deleted; if not, then we have the condition known as underflow.

Prepared by: K M Akkas Ali, Assistant Professor, IIT, JU

Push Operation on a Stack

➢ The operation of adding (pushing) an item onto a stack is implemented by the following PUSH procedure.

```
PUSH(STACK, TOP, MAXSTK, ITEM)
This procedure pushes an ITEM onto a stack

          1. [Stack already filled?]
             If TOP = MAXSTK,  then : Print :"OVERFLOW", and return.
          2.  Set  TOP := TOP+1  [Increase TOP by 1]
          3. STACK[TOP]) := ITEM  [Insert ITEM in new TOP position]
          4. Return.
```

➢ In order to understand the algorithm, let's break it apart line by line.

PUSH(STACK, TOP, MAXSTK, ITEM):

- ❑ First, PUSH accepts a parameter - ITEM. This parameter is of the same type as the rest of the stack. Item is the data to be added to the stack.

❑ **IF TOP = MAXSTK:**

- ❑ This line performs a check to see whether or not the stack is full.

❑ **TOP := TOP+1;**

- ❑ If the stack is not full, TOP is increased by a value equal to the size of another item (here by 1). This allocates room for the insertion.

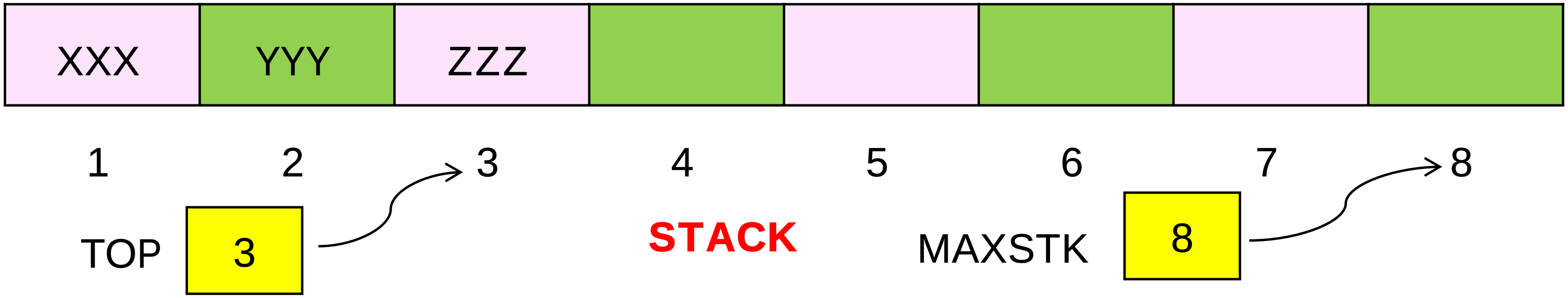
❑ **STACK[TOP] := ITEM;**

- ❑ Using this assignment, ITEM is pushed in the new TOP position.

Push Operation on a Stack

Example:

- Consider the stack shown in the figure below where TOP=3 and MAXSTK=8. Suppose we want to push an item WWW onto this stack.



1	2	3	4	5	6	7	8
XXX	YYY	ZZZ					

TOP 3

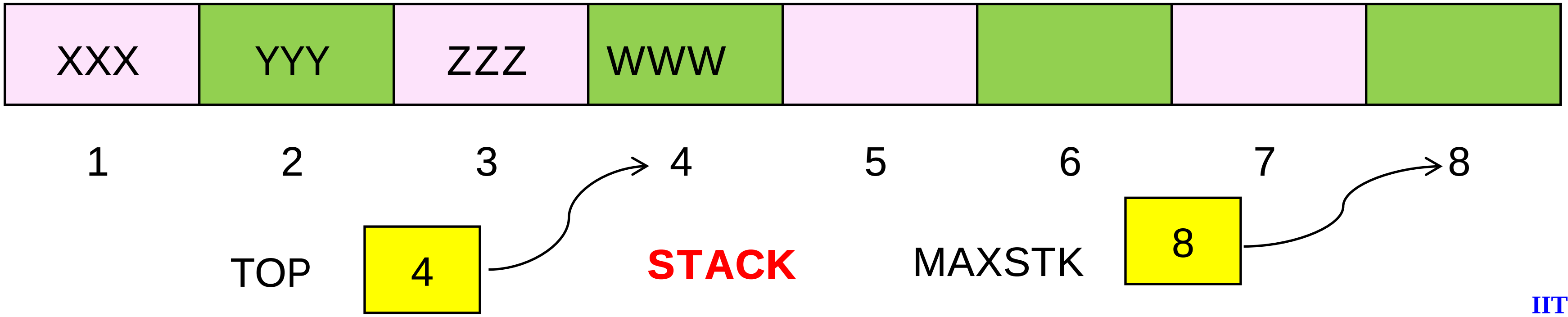
STACK

MAXSTK 8

- PUSH procedure is simulated below.

1. Since TOP=3 and MAXSTK=8; i.e. TOP≠MAXSTK, therefore control is transferred to step 2.
2. TOP=TOP+1=3+1=4
3. STACK[TOP]=STACK[4]=WWW
4. Return

- After the PUSH operation, the stack will be look like as follows.



1	2	3	4	5	6	7	8
XXX	YYY	ZZZ	WWW				

TOP 4

STACK

MAXSTK 8

Pop Operation on a Stack

➢ The operation of removing (popping) an item from a stack is implemented by the following POP procedure.

```
POP(STACK, TOP, ITEM)

This procedure  deletes the top elements of STACK and assigns it to the variable
ITEM.

          1.  [Stack has an item to be removed?]
              If TOP = 0,  then : Print :"UNDERFLOW", and return.
          2.  Set  ITEM := STACK[TOP]  [Assign TOP element to item]
          3.  Set TOP := TOP-1  [Decrease TOP by 1 point]
          4.  Return.
```

➢ In order to understand the algorithm, let's break it apart line by line.

PUSH(STACK, TOP, ITEM):

❑ First, PUSH accepts a parameter - ITEM. This parameter is of the same type as the rest of the stack. Item is the data to be added to the stack.

❑ **IF TOP = 0;**

❑ This line performs a check to see whether the stack is empty.

❑ **ITEM : =STACK[TOP];**

❑ If the stack is not empty, TOP element is assigned to ITEM.

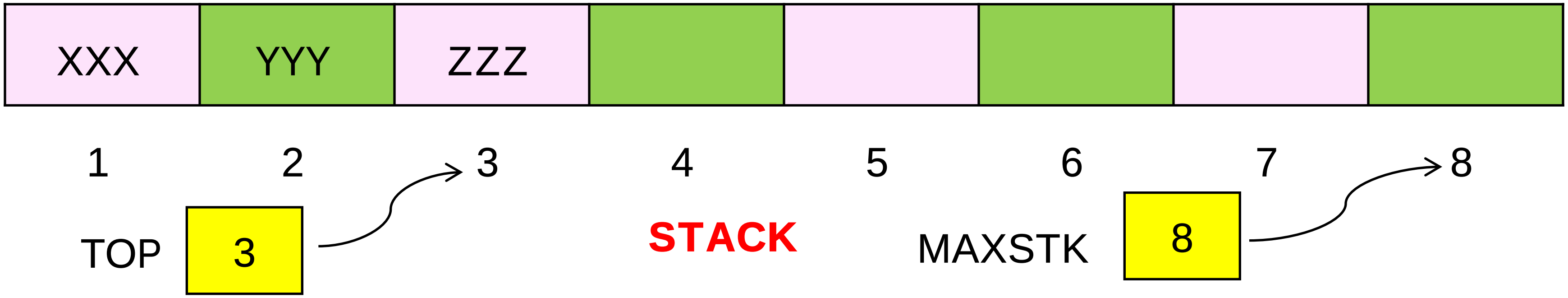
❑ **TOP := TOP-1;**

❑ Using this assignment, value of TOP is decreased by 1.

Pop Operation on a Stack

Example:

- Consider the stack shown in the figure below where TOP=3 and MAXSTK=8. Suppose we want to delete an item ZZZ from this stack.



XXX	YYY	ZZZ					
1	2	3	4	5	6	7	8

TOP 3

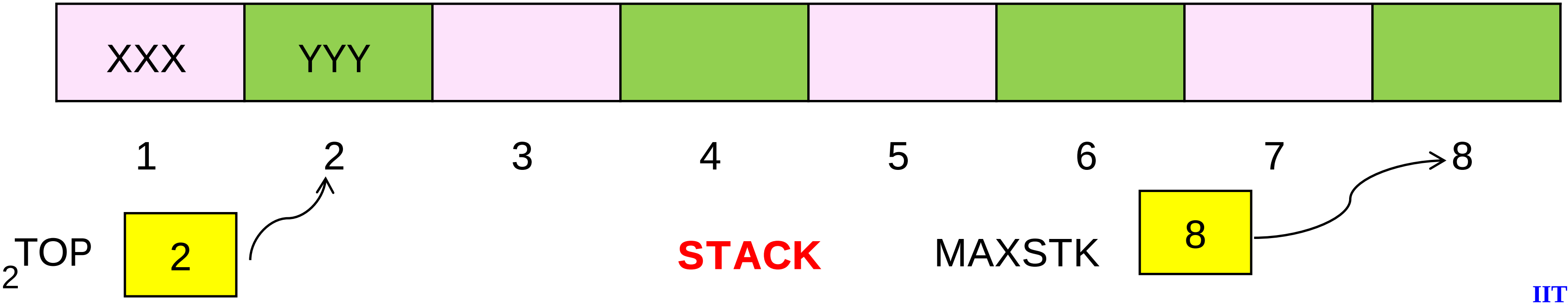
STACK

MAXSTK 8

- POP procedure is simulated below.

1. Since TOP=3, i.e. TOP≠0, therefore control is transferred to step 2.
2. ITEM=ZZZ
3. TOP=3-1=2
4. Return

- After the POP operation, the stack will be look like as follows.



XXX	YYY						
1	2	3	4	5	6	7	8

TOP 2

STACK

MAXSTK 8

Notations of Arithmetic Expression-an Application of Stack

- Let Q be an arithmetic expression involving operators and operands (e.g. Q: A+B/C-D).
- An arithmetic expression may be represented by any one of the following three notations:

1. Infix notation:

- This notation is the conventional notation for arithmetic expression. It is called infix notation because, the operator symbol is placed between its two operands. For example: A+B, C-D, E*F, G/H etc. This is called infix notation.
- When parsing expressions written in infix notation, you need parentheses and precedence rules to remove ambiguity. For example, with this notation, we must distinguish between (A+B)*C and A+(B*C) by using either parentheses or by precedence of operators.

2. Prefix or Polish notation:

- Prefix notation, also known as Polish notation (named after the Polish mathematician Jan Lukasiewiez), refers to the notation in which the operator symbol is placed before its two operands. For example, the above infix expressions can represented by Polish notations as +AB, -CD, *EF, /GH etc.
- With this notation, we do not need parentheses or other brackets to distinguish between (A+B)*C and A+(B*C). It will be just expressed by Polish notations as *+ABC and +A*BC respectively.

3. Postfix or Reverse Polish notation (RPN):

- In postfix or reverse Polish notation, the operator symbol is placed after its two operands. We do not need parentheses or other brackets here.
- For example, the above infix expressions can represented by reverse Polish notations as AB+, CD-, EF*, GH/ etc.

Prepared by: K M Akkas Ali, Assistant Professor, IIT, JU

Levels of Precedence

- Let Q be an arithmetic expression involving constants and operations.
- The binary operation (e.g. addition, multiplication, subtraction etc) in Q may have different levels of precedence:

Highest	: Exponentiation(↑)
Next highest	: Multiplication(*) and division(/)
Lowest	: Addition (+) and subtraction (-)

Example:

- Suppose we want to evaluate the following parenthesis-free arithmetic expression: 2 ↑ 3 + 5 *2 ↑ 2 – 12 / 6

Solution:

- First, evaluate the exponentiations to obtain:

  8 + 5 * 4 -12 / 6

- Then evaluate the multiplication and division to obtain:

  8 + 20 -2

- Finally, evaluate the addition and subtraction to obtain the final result:

  26

- Note that, the above expression is traversed three times, each time corresponding to a level of precedence of the operation.

Prepared by: K M Akkas Ali, Assistant Professor, IIT, JU

Transforming One Notation to Another

- Let Q be an arithmetic expression involving operators and operands (e.g. Q: A+B/C-D).
- An arithmetic expression may be given by any one of three notations described earlier.
- We may transform an arithmetic expression given in a notation to another notation either:

1. By direct observation method, or
2. By implementing algorithm with the help of stack

Translation by observation method:
Example-1:

- Suppose, Q: (A+B)/(C-D) is an arithmetic expression given in infix notation. We want to translate it into Polish or prefix notation.

Solution:

- In Polish notation, the operator symbol is placed before its two operands. For example, A+B is translated as +AB. For compound expression using parentheses, like (A+B)/(C-D), we use brackets [] to indicate a partial translation:

(A+B)/(C-D)=[+AB]/[-CD]=/+AB-CD

Prepared by: K M Akkas Ali, Assistant Professor, IIT, JU

Transforming One Notation to Another

Translation by observation method:

Example-2:

- Suppose, Q: (A+B)/(C-D) is an arithmetic expression given in infix notation. We want to translate it into reverse Polish or postfix notation.

Solution:

- In postfix notation, the operator symbol is placed after its two operands. For example, A+B is translated as AB+. For compound expression using parentheses, like (A+B)/(C-D), we use brackets [] to indicate a partial translation:

  (A+B)/(C-D)=[AB+]/[CD-]=AB+CD-/

Example-3:

- Suppose, P: /+AB-CD is an arithmetic expression given in Polish notation. We want to translate it into infix notation.

Solution:

- In infix notation, the operator symbol is placed between its two operands. For example, +AB is translated as A+B. For expressions using more operators, we use brackets [] and precedence of operators to indicate a partial translation:

  /+AB-CD=[+AB]/[-CD]=(A+B)/(C-D)

Prepared by: K M Akkas Ali, Assistant Professor, IIT, JU

Transforming Infix Expression into Postfix Expression

- Let Q be an arithmetic expression given in infix notation.
- Besides operators and operands, Q may also contain left and right parentheses.
- We also assume that operators on the same level are performed from left to right unless otherwise indicated by the parentheses.
- The following algorithm transforms the infix expression Q into its equivalent postfix expression P.
- The algorithm uses a stack to temporarily hold operators and left parentheses.
- The algorithm is completed when the stack is empty.

```
     POLISH(Q,P)
     Suppose Q is an arithmetic expressions written in infix notation. This algorithm
     finds the equivalent postfix expression P.
1.   Push "(" onto STACK, and add ")" to the end of Q.
2.   Scan Q from left to right and repeat steps 3 to 6  for each element of Q until
     the STACK is empty.
3.   If an operand is encountered, add it to P.
4.   If a left parenthesis is encountered, push it onto STACK.
5.   If an operator $\otimes$ is encountered, then :
       (a) Repeatedly pop from STACK and add to P each operator (on the top of
           STACK) which has the same precedence as or higher precedence than $\otimes$
       (b) Add $\otimes$ to STACK
     [End of if structure]
6.   If a right parenthesis is encountered, then :
       (a) Repeatedly pop from STACK and add to P each operator (on the top of
           STACK)  until a left parenthesis is encountered.
       (b) Remove the left parenthesis
     [End of if structure]
[End of step 2 loop]
7.   Exit
```

Prepared by: K M Akkas Ali, Assistant Professor, IIT, JU

Transforming Infix Expression into Postfix Expression

Example-2:

- Consider the following arithmetic expression Q given in infix notation.

  Q: A+(B*C-(D/E ↑ F)*G)*H
- Transform Q into its equivalent postfix expression P by simulating the related algorithm.

Solution:

- At first we push a left parenthesis “(“ onto STACK, and then we add a right parenthesis “)” to the end of Q to obtain:

Q:	A	+	(	B	*	C	-	(	D	/	E	↑	F	)	*	G	)	*	H	)
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

- The elements of Q have been labeled from left to right for easy reference.
- Note that-
  1. Each operand encountered is simply added to P and does not change STACK.
  2. The subtraction operator (-) in row 7 sends * from STACK to P before it (-) is pushed onto STACK.
  3. The right parenthesis in row 14 sends ↑ and then / from STACK to P, and then removes the left parenthesis from the top of STACK.
  4. The right parenthesis in row 20 sends * and then + from STACK to P, and then removes the left parenthesis from the top of STACK.
  5. After step 20 is executed, the STACK is empty and the algorithm is completed.
  6. The equivalent postfix expression found is:

     P: A B C * D E F ↑ / G * - H * +

Prepared by: K M Akkas Ali, Assistant Professor, IIT, JU

Transforming Infix Expression into Postfix Expression

Symbol scanned Q		STACK							Postfix expression P														
1.	A	(							A														
2.	+	(	+						A														
3.	(	(	+	(					A														
4.	B	(	+	(					A	B													
5.	*	(	+	(	*				A	B													
6.	C	(	+	(	*				A	B	C												
7.	-	(	+	(	-				A	B	C	*											
8.	(	(	+	(	-	(			A	B	C	*											
9.	D	(	+	(	-	(			A	B	C	*	D										
10.	/	(	+	(	-	(	/		A	B	C	*	D										
11.	E	(	+	(	-	(	/		A	B	C	*	D	E									
12.	↑	(	+	(	-	(	/	↑	A	B	C	*	D	E									
13.	F	(	+	(	-	(	/	↑	A	B	C	*	D	E	F								
14.	)	(	+	(	-				A	B	C	*	D	E	F	↑	/						
15.	*	(	+	(	-	*			A	B	C	*	D	E	F	↑	/						
16.	G	(	+	(	-	*			A	B	C	*	D	E	F	↑	/	G					
17.	)	(	+						A	B	C	*	D	E	F	↑	/	G	*	-			
18.	*	(	+	*					A	B	C	*	D	E	F	↑	/	G	*	-			
19.	H	(	+	*					A	B	C	*	D	E	F	↑	/	G	*	-	H		
20.	)								A	B	C	*	D	E	F	↑	/	G	*	-	H	*	+

Prepared by: K M Akkas Ali, Assistant Professor, IIT, JU