Introduction

- Stacks: A stack is a linear list of elements in which an element may be inserted or deleted only at one end called the top of the stack.

- This means, the elements are removed from a stack in the reverse order of that in which they were inserted into the stack.

  Example: a stack of dishes, a stack of pennies etc.

- Stacks are also called Last in First out(LIFO) list.

- Push: is the term used to insert an element into a stack.
- Pop; is the term used to delete an element from a stack.

Stacks Continue…

- Suppose we have five data to be inserted to the stack:

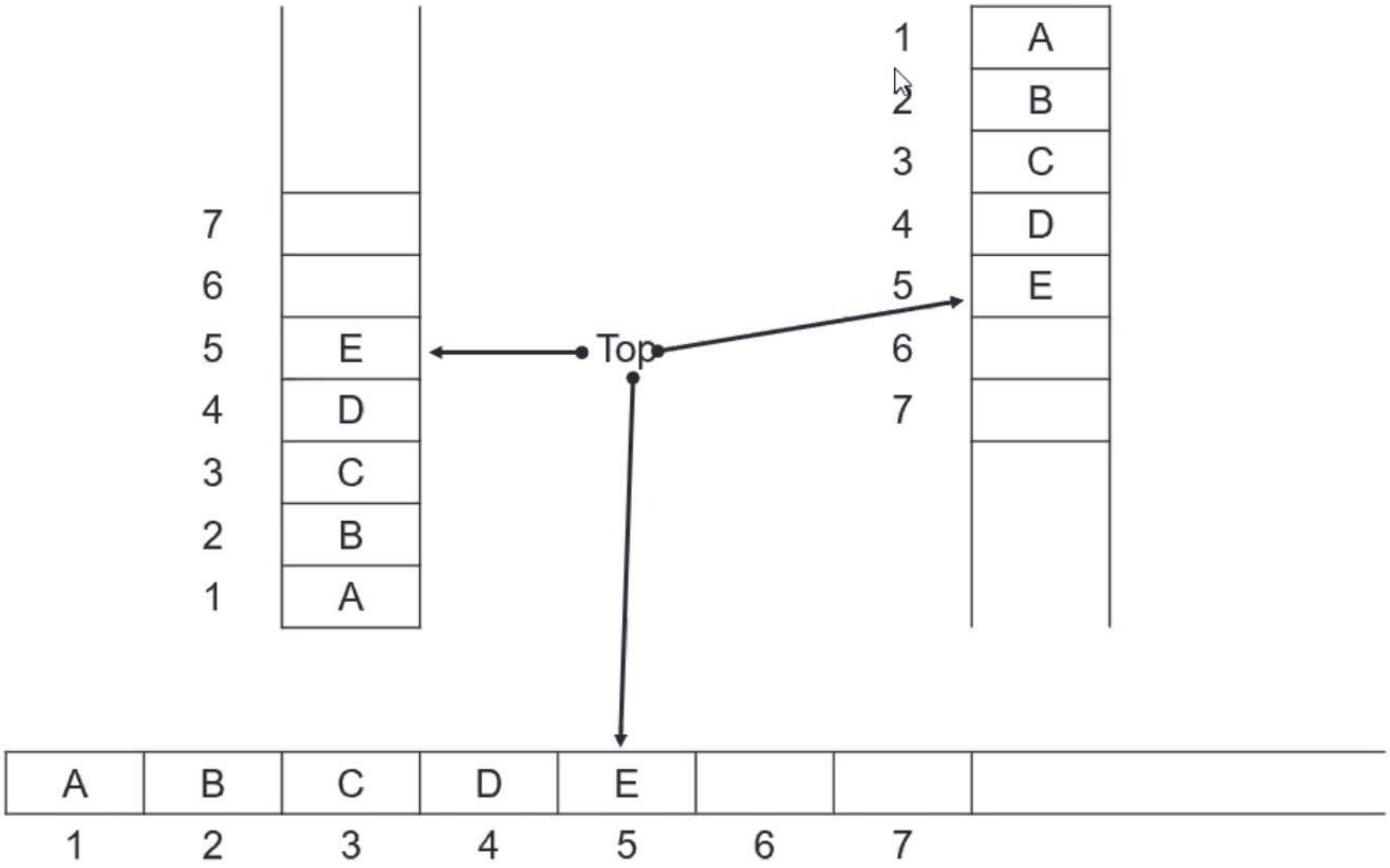
  A, B, C, D, E
- The diagram of the stack is shown on the next slide. There we show the three ways to picture a stack.

- However, we normally use the following notation to represent a stack.

  Stack: A, B, C, D, E (The right most is the top).

- 'C', can not be deleted from the list before 'E' and 'D'.

Diagrams of Stacks



Array Representation of Stacks

- Stacks may be represented in the computer in various ways usually by means of linear array.

- For this reason, we require a linear array Stack; a pointer variable Top which contains the location of the top element and a variable MAXSTK which indicates the maximum no. of elements that can be held by the stack.

- Top=0 or NULL indicates that the stack is empty and no elements can be deleted from the stack.

- Top=MAXSTK indicates that the stack is holding maximum elements and no further elements can be inserted.

Arithmetic Expression: Polish Notation

- Polish notation, named after the Polish Mathematician Jan Lukasiewicz, refers to the notation in which the operator symbol is placed before its two operand.
- Example: A+B can be written as +AB.
- This notation also known as prefix notation.

- Another notation which is reverse of Polish notation is postfix notation.
- Example: A*B can be written as AB*.
- The common mathematical expression such as (A+B)*C is known as infix notation.

Polish Notation Continue…

- Advantage: To evaluate infix expression, we need to follow the precedence of the operators; otherwise correct result can not be obtained.

- For example: The following two expressions (A+B)*C and A+(B*C) almost same but result is different, because of the operator precedence.

- In case of prefix or postfix notation, parentheses never used, to determine the order of the operations.

Infix to postfix and prefix

- (A + B ↑ D) / (E – F) + G to postfix

=(A + [B D ↑]) / [E F -] + G

=[A B D ↑ +] / [E F -] + G

=[A B D ↑ + E F - /] + G

=A B D ↑ + E F - / G +

- (A + B ↑ D) / (E - F) + G to prefix

=(A + [↑ B D]) / [- E F] + G

=[+ A ↑ B D] / [- E F] + G

=[/ + A ↑ B D - E F] + G

=+ / +A ↑ B D - E F G

Infix to postfix and prefix

- A * (B + D) / E – F * (G + H / K) to postfix

=A * [B D +] / E – F * (G + [H K /])

=[A B D + *] / E – F * [G H K / +]

=[A B D + * E /] - [F G H K / + *]

=A B D + * E / F G H K / + * -

- A * (B + D) / E – F * (G + H / K) to prefix

=A * [+ B D] / E – F * (G + [/ H K])

=[* A + B D] / E – F * [+ G / H K]

=[/ * A + B D E] - [* F + G / H K]

=- / * A + B D E * F + G / H K

Postfix Expression Evaluation

- Evaluate postfix expression: 5, 6, 2, +, *, 12, 4, /, -

=5, [6+2], *, 12, 4, /, -

=[5*8],12, 4, /, -

=40, 12, 4, /, -

=40, [12/4], -

=40-3

=37

Note: Postfix expression can be easily evaluated using computer. How ever infix expression can also be evaluated with some extra cost (Infix to postfix then postfix evaluation).

Postfix Evaluation Example:

Once again we evaluate postfix expression: 5, 6, 2, +, *, 12, 4, /, - by showing Stack's contents as each element is scanned.

Symbol Scanned	Stack
5	5
6	5, 6
2	5, 6, 2
+	5, 8
*	40
12	40, 12
4	40, 12, 4
/	40, 3
-	37
)	

Algorithm: Postfix Evaluation

This algorithm finds the value of an arithmetic expression P written in postfix notation.

1. Add a right parenthesis ")" at the end of P.
2. Scan P from left to right and repeat steps 3 and 4 until ")" is encountered.
3. If an operand is encountered put it onto stack.
4. If an operator θ is encounter, then
   a) Remove two top elements of Stack, where A is the top element and B is the next-to-top element.
   b) Evaluate B θ A.
   c) Place the result of (b) back on Stack.
5. Set value equal to the top element on the Stack.
6. Exit.

Transforming Infix to Postfix Expression

This algorithm finds the equivalent expression P from infix Q.

1. Push "(" onto stack and add ")" to the end of Q.
2. Scan Q from left to right and repeat steps 3 to 6 until 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 θ is encountered, then:
   a) Repeatedly pop from Stack and add to P each operator (on the top of the Stack) which has the same precedence as or higher precedence than θ.
   b) Add θ to Stack.
6. If right parenthesis is encountered, then:
   a) Repeatedly pop from Stack and add to P each operator (on the top of the stack) until a left parenthesis is encountered.
   b) Remove the left parenthesis. [Do not add it to P].
7. Exit