CS-2001 Data Structures

National University of Computer and Emerging Sciences

Course Instructor: Syed Zain Ul Hassan

Application of Stack

Convert Infix to postfix

Expression Representation Techniques

Expression is a collection of operators and operands that represents a specific value.

1.Infix Expression 2. Prefix Expression 3. Postfix Expression

Expression	Example	Note
Infix	a + b	Operator Between Operands
Prefix	+ a b	Operator before Operands
Postfix	a b +	Operator after Operands

To evaluate an infix expression, we need to consider Operators' Priority and Associative property.

For example:

Expression 3+5*4 evaluate to 32 i.e. (3+5)*4 or to 23 i.e. 3+(5*4).

Generally, postfix expressions are free from Operator Precedence that's why they are preferred in Computer system. Computer System Uses Postfix form to represent expression.

Table below show the Priority for operators that needs in conversion method.

Input Character	priority
(	5
^ , Not	4
*, / ,AND , DIV, MOD	3
+ , - , OR	2
= , < , > , < > , <=,>=	1
#	0

Explanation the conversion in steps:

1. Read symbol from expression and it **may** be –
    - Alphabet from A-Z or a-Z
    - Numerical Digit from 0-9
    - Operator

- Opening And Closing Braces (,)

2. **If Entered Character is Alphabet** or **Digit** then Following Action Should be taken:
   - Print Alphabet and Digit as Output
3. **If Entered Character is Opening Bracket** then Following Action Should be taken-
   - Push '(' onto Stack
   - If any Operator Appears before ')' then Push it onto Stack.
   - If Corresponding ')' bracket appears then Start Removing Elements [Pop] from Stack till '(' is removed.
4. **If Entered Character is Operator** then Following Action Should be taken:
   - Check Whether There is any Operator Already present in Stack or not.
   - If Stack is Empty then **Push Operator onto Stack**.
   - If Present then Check Whether **Priority of Incoming Operator** is greater than **Priority of Topmost Stack Operator**.
   - If Priority of Incoming Operator is Greater than **Push Incoming Operator onto Stack**.
   - Else Pop (Operator that have high Priority or equal) From Stack and push incoming operator to stack, again go to Step 1.

Example 1:- Convert the following infix expression to postfix: A/B^C-D #.

Current Symbol	Stack	Output	Comment
	#	–	Initially Stack is Empty
A	#	A	Print Operand
/	/	A	Push Operator Onto Stack
B	/	AB	Print Operand
^	/^	AB	Push Operator Onto Stack because Priority of ^ is greater than Current Topmost Symbol of Stack i.e '/'
C	/^	ABC	Print Operand
–	/	ABC^	**Step 1 :** Now '^' Has Higher Priority than Incoming Operator So We have to Pop Topmost Element . **Step 2 :** Remove Topmost Operator From Stack and Print it
–	NULL	ABC^/	**Step 1 :** Now '/' is topmost Element of Stack Has Higher Priority than Incoming Operator So We have to Pop Topmost Element again. **Step 2 :** Remove Topmost Operator From Stack and

			Print it
–	–	ABC^/	**Step 1:** Now Stack Becomes Empty and We can Push Operand onto Stack
D	–	ABC^/D	Print Operand
NULL	–	ABC^/D-	Expression Scanning Ends but we have still one more element in stack so pop it and display it

Example 2:- Convert the following infix expression to postfix: A + (B /C)#.

Infix	Stack	Postfix
	#	
A	#	A
+	# +	A
(	#+(	A
B	#+(	AB
/	#+(/	AB
C	#+(/	ABC
)	#+	ABC/
#	#	ABC/+

Example 3:- Convert the following infix expression to postfix a+b*c+d*e#

input	stack	postfix
	#	
a	#	a
+	#,+	a
b	#,+	ab
*	#,+,*	ab
c	#,+,*	abc
+	#,+	abc*+
d	#,+	abc*+d
*	#,+,*	abc*+d
e	#,+,*	abc*+de
#	#	abc*+de*+

Example 4: - Convert the following infix expression to postfix

((A – (B + C)) * D) / (E +F) #

Infix	Stack	Postfix
	#	
(	#(	
(	#((	
A	#((	A
-	#((-	A
(	#((-(	A
B	#((-(	AB
+	#((-(+	AB
C	#((-(+	ABC
)	#((-	ABC+
)	#(	ABC+-
*	#(*	ABC+-
D	#(*	ABC+-D
)	#	ABC+-D*
/	#/	ABC+-D*
(	#/(	ABC+-D*
E	#/(	ABC+-D*E
+	#/(+	ABC+-D*E
F	#/(+	ABC+-D*EF
)	#/	ABC+-D*EF+
#	#	ABC+-D*EF+/

Example 5:- Convert the following infix expression to postfix (a+b^c^d)*(e+f/d)#

input	stack	postfix
	#	
(	#,(	
a	#,(	a
+	#,(,+	a
b	#,(,+	ab
^	#,(,+,^	ab
c	#,(,+,^	abc

^	#,(,+,^	abc^
d	#,(,+,^	abc^d
)	#	abc^d^+
*	#,*	abc^d^+
(	#,*,(	abc^d^+
e	#,*,(	abc^d^+e
+	#,*,(,+	abc^d^+e
f	#,*,(,+	abc^d^+ef
/	#,*,(,+,/	abc^d^+ef
d	#,*,(,+,/	abc^d^+efd
)	#,*	abc^d^+efd/+
#		abc^d^+efd/+*

2- Evaluation of Postfix Expression

Explanation the Evaluation in steps:

1. At first, stack is empty.

2. Read symbol from expression one after another until reach the end of expression:

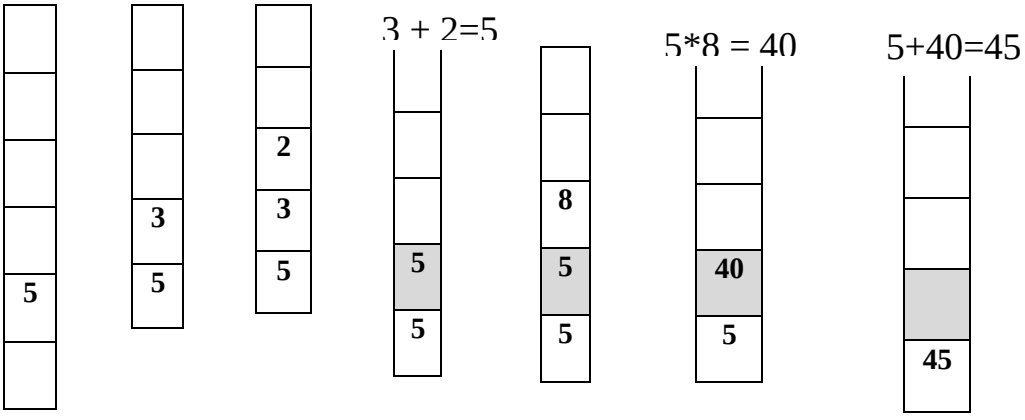
2.1 if the symbol is operand push into stack.

2.2 if the symbol is operator pop two operands from stack.

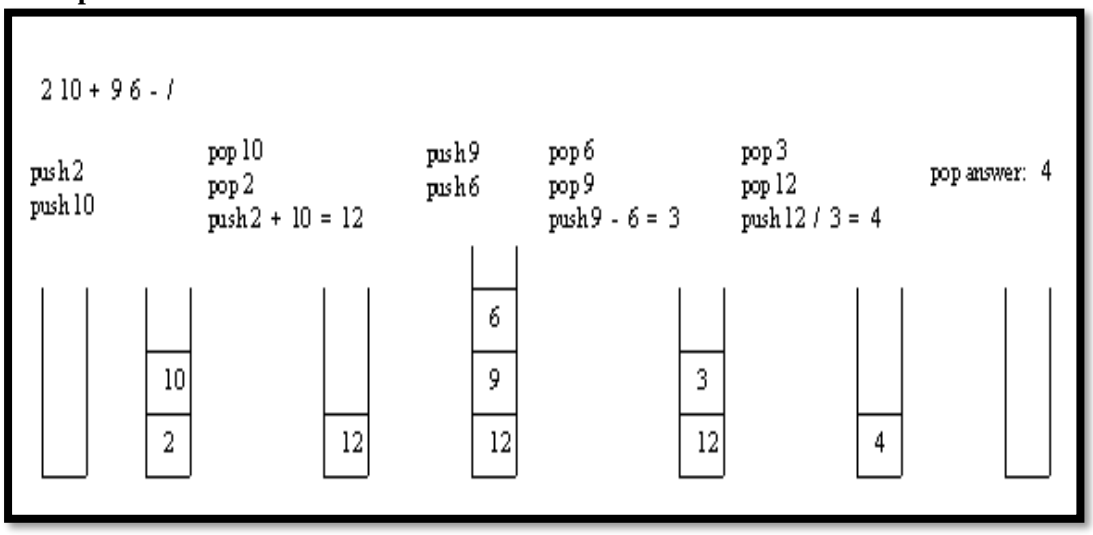
3. Perform the operation(operator) between the two operands and push result into stack.

4. Back into step 2, until the end.

Example 1: Find the evaluation of postfix expression: **5 3 2 + 8 * +**



Example 2:


2 10 + 9 6 - /
push 2
push 10
pop 10
pop 2
push 2 + 10 = 12
push 9
push 6
pop 6
pop 9
push 9 - 6 = 3
pop 3
pop 12
push 12 / 3 = 4
pop answer: 4
10
2
12
6
9
12
3
12
4