Chapter

6B FORMS OF AN EXPRESSION

- *Prefix, Post fix, Infix Forms of an Expression*
- *Expression Tree & its traversal*

[A PRELIM TO CHAPTER 13-NON-LINEAR DATA STRUCTURE]

Expressions can be represented in various forms. Some use operators, some use functions and so on. For example, (a + b) and Sum(a,b) express the same meaning.

Arithmetic expressions can be represented in various forms. The three of the forms of expression are Infix, Prefix and Postfix.

Infix Expression

This is the form which is most commonly used. It places an operator in between two operands. In case of varying precedence of operators, this form requires brackets to group them.

For Example : A + B; A * (B + C)

Prefix Expression

In this form, the operator is placed in front of two operands. In this form, brackets are not required to express any expression. This form is also called **Polish Form**.

For Example : + A B ; + A * B C

Postfix Expression

In this form, the operator is placed behind two operands. In this form, brackets are not required to express any expression. This form is also called **Reverse-Polish Form**.

For Example : A B + ; A B + C *

Rule of Priority : There are 5 arithmetic operators used in programming (+ – * / %). Priorities of the operators in expressions are as follows:

(i) ( ) for multiple brackets, inner brackets first

(ii) * /% same priority, sub-expression taken from left to right

(iii) + – same priority, sub-expression taken from left to right

Conversion from Infix Form to Prefix Form

For performing this conversion, make a group of two operands that has an operator in middle.The prefix form asks the operator to be placed in front. Use the priority chart to make the groups. Each group represents an operand in the next round. Postfix form does not require brackets to represent priorities of various operators.

1. A* B + C

Ans.

$$\begin{aligned} A*B + C &= (A*B) + C \\ &= (*AB) + C \qquad [\equiv P + C \text{ ; where } P = *AB] \\ &= + *AB\ C \end{aligned}$$

2. A* (B + C)

Ans.

$$\begin{aligned} A * (B + C) &= A * (+ B\ C) \\ &= A * (+BC) \qquad [A * P \text{ ; where } P = BC+] \\ &= * A + BC \end{aligned}$$

3. A / (B + C * D)

Ans.

$$\begin{aligned} A / (B + C * D) &= A / (B + (C * D)) \\ &= A / (B + (*CD)) \\ &= A / (+ B *CD) \\ &= / A + B *C\ D \end{aligned}$$

4. A * B / C – C % D * E

Ans.

$$\begin{aligned} A * B / C - C \% D * E &= (A * B) / C - (C \% D) * E \\ &= ((*A\ B) / C) - ((\% C\ D) * E) \\ &= (/ *AB\ C) - (* \% CDE) \\ &= - / *ABC * \% CDE \end{aligned}$$

5. (A + B / C) * D – C * B + A

Ans.

$$\begin{aligned} (A + B / C) * D - C * B + A &= (A + (B / C)) * D - (C * B) + A \\ &= (A + (/BC)) * D - (*CB) + A \\ &= ((+ A / BC) * D) - (*CB) + A \\ &= ((* + A / BC\ D) - (*CB)) + A \\ &= (- * + A / BC\ D *CB) + A \\ &= + - * + A / BC\ D *CB\ A \end{aligned}$$

6. M / (A * R) + K * (E – T)

Ans.

$$\begin{aligned} M / (A * R) + K * (E - T) &= M / (A * R) + K * (E - T) \\ &= M / (*AR) + K * (- ET) \\ &= (M / (*AR)) + (K * (- ET)) \\ &= (/ M *AR) + (*K - ET) \\ &= + / M *AR *K - ET \end{aligned}$$

7. D * (R – (A + S)) / (T * (I – C))

Ans. D * (R – (A + S)) / (T * (I – C)) = D * (R – (A + S)) / (T * (I – C))
= D * (R – (+ AS)) / (T * (– IC))
= (D * (– R + AS)) / (*T – IC)
= (* D – R + AS) / (*T – IC)
= / * D – R + AS *T –IC

Conversion from Prefix Form to Infix Form

For performing this conversion, make a group of two operands that is preceeded by an operator. Each group represents an operand in the next round. The Infix form uses brackets to represent priorities of various operators.

1. + * A B C

Ans. + * A B C = + (* A B) C
= + (A*B) C [+PC ; where P = A*B]
= (A * B) + C

2. +A *B C

Ans. +A * B C = + A (*B C)
= + A (B * C) [+AP ; where P = B*C]
= A + (B * C)

3. / A + B *C D

Ans. / A + B * C D = / A + B (*C D)
= / A + B (C*D)
= / A (+ B (C*D))
= / A (B + (C*D))
= A / (B + (C*D))

4. – / *ABC * % CDE

Ans. – / * ABC * % CDE = – / (*AB) C * (% CD) E
= – / (A * B) C * (C % D) E
= – (/ (A * B) C) (* (C % D) E)
= – ((A * B) / C) ((C % D) * E)
= ((A * B) / C) – ((C % D) * E)

The brackets in the infix form is allowed in the answer step.

The above Infix expression can be written by removing the brackets without disturbing the priorities of the operators as A * B / C – C % D * E

5. % + * - / A B C D E F

Ans.

$$\begin{aligned} \%+*-/ABCDEF &= \%+*-(/AB)CDEF \\ &= \%+*-(A/B)CDEF \\ &= \%+*((-(A/B)C)DEF \\ &= \%+(*((A/B)-C)D)EF \\ &= \%+(((A/B)-C)*D)EF \\ &= \%(+(((A/B)-C)*D)E)F \\ &= \%(((A/B)-C)*D)+E)F \\ &= (((A/B)-C)*D)+E)\%F \end{aligned}$$

The Infix expression after reducing the brackets is

$$= ((A/B-C)*D+E)\%F$$

6. / – *CU *RT *+AI N

Ans.

$$\begin{aligned} /-*CU*RT*+AIN &= /-(*CU)(*RT)*(+AI)N \\ &= /-(C*U)(R*T)*(A+I)N \\ &= /(-(C*U)(R*T))(*(A+I)N) \\ &= /((C*U)-(R*T))((A+I)*N) \\ &= ((C*U)-(R*T))/((A+I)*N) \end{aligned}$$

The Infix expression after reducing the brackets is (C*U - R*T) / (A+I) * N

7. + * – CHE * / + M I S % + TRY

Ans.

$$\begin{aligned} +*-CHE*/+MIS\%+TRY &= +*(-CH)E*/(+MI)S\%(+TR)Y \\ &= +(*(C-H)E)*(/(M+I)S)(\%(T+R)Y) \\ &= +((C-H)*E)(*((M+I)/S)((T+R)\%Y)) \\ &= +((C-H)*E)(((M+I)/S)*((T+R)\%Y)) \\ &= ((C-H)*E)+(((M+I)/S)*((T+R)\%Y)) \end{aligned}$$

The Infix expression after reducing the brackets is (C – H) * E + (M + I) / S * (T + R) % Y

Conversion From Infix Form to Postfix Form

For performing this conversion, make a group of two operands that has an operator in middle. Use the priority chart to make the groups. Each group represents an operand in the next round. Postfix form does not use brackets to represent priorities of various operators.

1. A* B + C

Ans.

$$\begin{aligned} A*B+C &= (A*B)+C \\ &= (AB*)+C \qquad [\equiv P+C \text{ ; where } P = AB*] \\ &= AB*C+ \end{aligned}$$

2. A* (B + C)

Ans.

$$\begin{aligned} A*(B+C) &= A*(BC+) \\ &= ABC+* \qquad [A*P \text{ ; where } P = BC+] \end{aligned}$$

3. A / (B + C * D)

Ans.

$$A / (B + C * D) = A / (B + (C * D))$$
$$= A / (B + (CD*))$$
$$= A / (B\,CD* +)$$
$$= A\,BCD* + /$$

4. A * B / C – C % D * E

Ans.

$$A * B / C - C \% D * E = (A * B) / C - (C \% D) * E$$
$$= ((A\ B*) / C) - ((C\,D\%) * E)$$
$$= (A\ B*\ C /) - (C\,D\%\,E*)$$
$$= A\,B*\,C /\ C\,D\%\,E* -$$

5. (A + B / C) * D – C * B + A

Ans.

$$(A + B / C) * D - C * B + A = (A + (B / C)) * D - (C * B) + A$$
$$= (A + (B\,C /)) * D - (C\,B*) + A$$
$$= ((A\,B\,C / +) * D) - (C\,B*) + A$$
$$= ((A\,B\,C / +\,D*) - (C\,B*)) + A$$
$$= (A\,B\,C / +\,D*\,C\,B* -\ ) + A$$
$$= A\,B\,C / +\,D*\,C\,B* - A +$$

6. P / (E * N) + C * (I – L)

Ans.

$$P / (E * N) + C * (I - L) = P / (E * N) + C * (I - L)$$
$$= (P / (E\,N*)) + (C * (I\,L -))$$
$$= (P\,E\,N* /) + (C\,I\,L - *)$$
$$= P\,E\,N* /\,C\,I\,L - * +$$

7. D * (A – (R+K)) / (E * (S – T))

Ans.

$$D * (A - (R+K)) / (E * (S - T)) = D * (A - (RK +)) / (E * (S\,T -))$$
$$= D * (A\,R\,K + -) / (E\,S\,T - *))$$
$$= (D\,A\,R\,K + - *) / (E\,S\,T - *\ )$$
$$= D\,A\,R\,K + - *\,E\,S\,T - * /$$

Conversion From Postfix Form to Infix Form

For performing this conversion, make a group of two operands that is followed by an operator. Each group represents an operand in the next round. The Infix form uses brackets to represent priorities of various operators.

1. A B * C +

Ans.

$$A\,B * C + = (A\,B*) C +$$
$$= (A * B) C + \quad [\ PC+\ ;\ \text{where}\ P = A+B\ ]$$
$$= (A * B) + C$$

2. A B C * +

Ans.

$$A\,B\,C * + = A\ (B\,C*\ ) +$$
$$= A\ (B * C) + \quad [\ PC+\ ;\ \text{where}\ P = B*C\ ]$$
$$= A + (B * C)$$

3. A B C – * D +

Ans. A B C – * D + = A (B C –) * D +

= (A (B – C) *) D +

= (A * (B – C)) D +

= (A * (B – C)) + D

4. A B C D + – *

Ans. A B C D + – * = A B (C D +) – *

= A (B (C + D) –) *

= A (B – (C + D)) *

= A * (B – (C + D))

5. A B % C D / – E F G * + -

Ans. A B % C D / – E F G * + - = (A B %) (C D /) – E (F G *) + –

= ((A % B) (C / D) –) (E (F * G) +) –

= ((A % B) – (C / D)) (E + (F * G)) –

= ((A % B) – (C / D)) – (E + (F * G))

The brackets in the infix form is allowed in the answer step.

The above Infix expression can be written by removing the brackets without disturbing the priorities of the operators as A % B – C / D – (E + F * G)

6. P Q R S T * – / + W Y + /

Ans. P Q R S T * – / + W Y + / = P Q R (S T *) – / + (W Y +) /

= P Q (R (S * T) –) / + (W + Y) /

= P (Q (R – (S * T)) /) + (W + Y) /

= (P (Q / (R – (S * T))) +) (W + Y) /

= (P + (Q / (R – (S * T)))) (W + Y) /

= (P + (Q / (R – (S * T)))) / (W + Y)

The Infix expression after reducing the brackets is (P + Q / (R – S*T)) / (W+Y)

7. PHY+ / T % IC* S – *

Ans. PHY+ / T % IC* S – * = P (HY+) / T % (IC*) S – *

= P (H + Y) / T % (I* C) S – *

= (P (H + Y) /) T % ((I* C) S –) *

= ((P / (H + Y)) T %) ((I* C) – S) *

= ((P / (H + Y)) % T) ((I* C) – S) *

= ((P / (H + Y)) % T) * ((I* C) – S)

The Infix expression after reducing the brackets is (P / (H+Y)) % T * (I*C – S)

Expression Tree of a Mathematical Expression

A mathematical expression can be expressed in form of an Expression Tree. It is a special form of a Binary Tree which contains a collection of nodes. Each node stores either an operator or an operand. Each node containing an operator can have two types of sub-nodes

(i) an operand

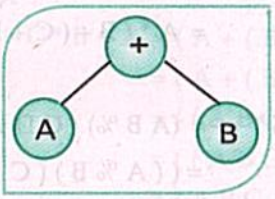
(ii) an operator which in turn can store in the same way

Note :

- More details of Trees and Non-Linear Data Structure is in Chap-13).

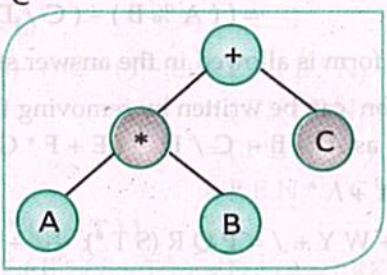
The following examples explain how to make Expression Trees.

Example 1 : A + B

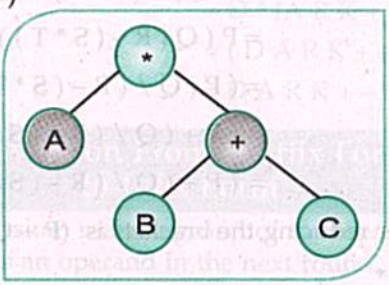


Example 2 : A * B + C

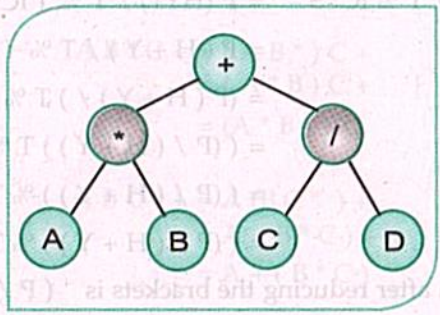
= (A*B) + C



Example 3 : A * (B + C)

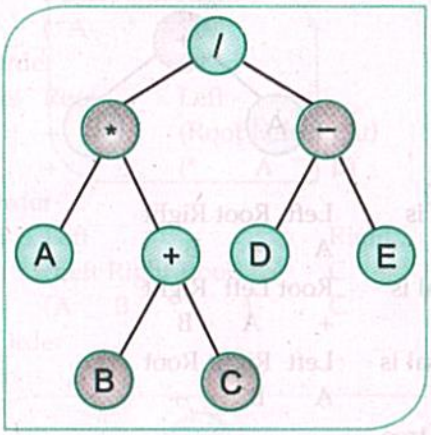


Example 4 : A*B + C/D



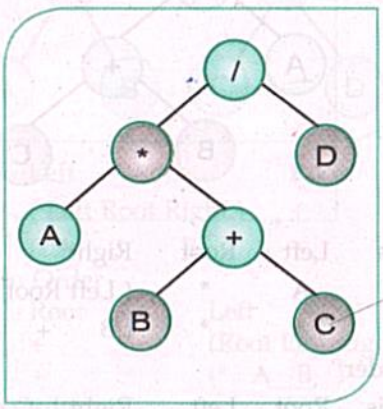
Example 5 : A * (B + C) / (D – E)

= (A * (B + C)) / (D – E)



Example 6 : A * (B + C) / D

= (A * (B + C)) / D



Rules of forming an Expression from an Expression Tree

1. The variables are grouped according to the priority or precedence of the operators.
2. The Root (topmost node of a Tree) gets an operator.
3. The priority of the operators is as follows :
   - Brackets
   - * / % Left to Right
   - + – Left to Right

Reading an Expression Tree

There are 3 ways in which an Expression Tree can be read :

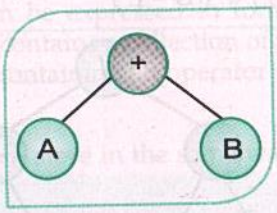
(i) In Order - Left Root Right

(ii) Pre Order - Root Left Right

(iii) Post Order - Left Right Root

The following examples explain the concept.

Example 1 : Given the tree

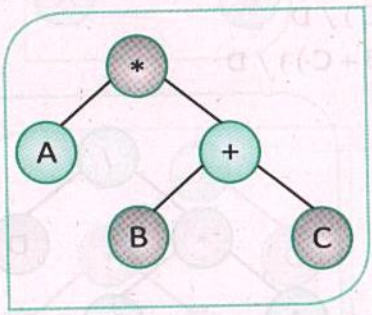


The In Order traversal is Left Root Right
A + B

The Pre Order traversal is Root Left Right
+ A B

The Post Order traversal is Left Right Root
A B +

Example 2 : Given the tree



The **In Order** traversal is Left Root Right
A * (Left Root Right)
A * (B + C)

A * (B + C) .....In Order

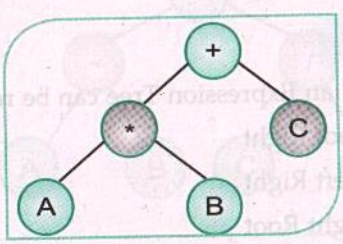
The **Pre Order** traversal is Root Left Right
* A (Root Left Right)
* A (+ B C)

* A + B C .....Pre Order

The **Post Order** traversal is Left Right Root
A (Left Right Root) +
A (B C +) *

A B C + * .....Post Order

Example 3 : Given the tree



The **In Order** traversal is	Left	Root	Right
	(Left Root Right)	+	C
	(A * B)	+	C

(A * B) + C .....In Order

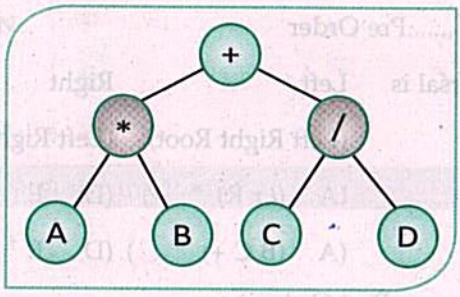
The **Pre Order** traversal is	Root	Left	Right
	+	(Root Left Right)	C
	+	(* A B)	C

+ * A B C .....Pre Order

The **Post Order** traversal is	Left	Right	Root
	(Left Right Root)	C	+
	(A B *)	C	+

A B * C + ...Post Order

Example 4 :



The **In Order** traversal is	Left	Root	Right
	(Left Root Right)	+	(Left Root Right)
	(A * B)	+	(C/D)

(A * B) + (C / D) .....In Order

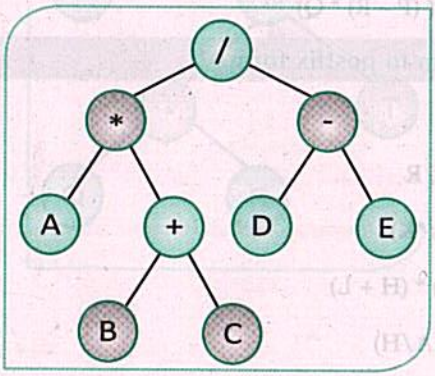
The **Pre Order** traversal is	Root	Left	Right
	+	(Root Left Right)	(Root Left Right)
	+	(* A B)	(/ C D)

+ * A B / C D .....Pre Order

The **Post Order** traversal is	Left	Right	Root
	(Left Right Root)	(Left Right Root)	+
	(A B *)	(C D /)	+

A B * C D / + ...Post Order

Example 5 :



The **In Order** traversal is

Left	Root	Right
(Left Root Right)	/	(Left Root Right)
(A * (*l* R *r*))	/	(D – E)
(A * (B + C))	/	(D – E)

(A * (B+C)) / (D – E) .....In Order

The **Pre Order** traversal is

Root	Left	Right
/	(Root Left Right)	(Root Left Right)
/	(* A (R *l r*))	(– D E)
/	(* A (+ B C))	(– D E)

/*A + B C – D E .....Pre Order

The **Post Order** traversal is

Left	Right	Root
(Left Right Root)	(Left Right Root)	/
(A (*l r* R) *)	(D E –)	/
(A (B C +) *)	(D E –)	/

A B C + * D E – / .....Post Order

Exercise

1. Draw the Expression Tree of the following expressions given in Infix form.

(a) A * (B + C / D)

(b) (A / (B*C – D)) + (E*F)

(c) (P + Q) / R – (P + Q / R)

(d) P * Q + (P – Q) / (P + Q)

(e) P – (Q + R) + (P + R)/((P – R) * Q)

2. Convert from infix form to postfix form.

(a) (A + B) * C / (D – E)

(b) P / Q*R + P*Q/R – P/R

(c) M % (N + L) – N / (L *K)

(d) K – (L+ H/G + H/L) * (H + L)

(e) P / (A – M*H) * (M + A/H)

3. Convert from infix form to prefix form.

(a) D * A * (R – K) + P / L / (U + M)

(b) (M – (N + P/Q)) * (M – P) / Q

(c) N * U + M – (B – (E – R))

(d) K – (Y – (R / T * Z + W))

(e) (Q + P + R) * S – (D – G * S)

4. Convert from prefix form to infix form.

(a) + * / + * A B C D E F

(b) + * – N A U / H / V M

(c) * – / Y E + A R / I N

(d) – H + L * S / T N

(e) * * P A / + F N – T H

5. Convert from postfix form to infix form.

(a) A N * T – H E / R / +

(b) T I + G * R S + –

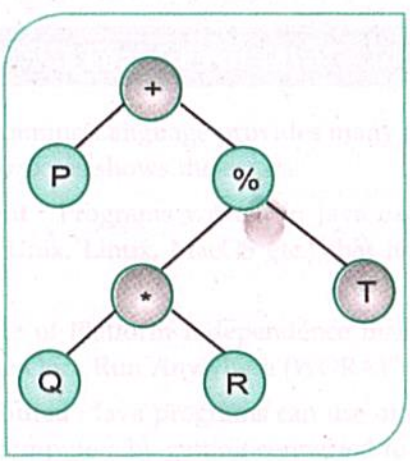
(c) K L R + * K L + K R – / –

(d) N C + C N – * N C / + C N / –

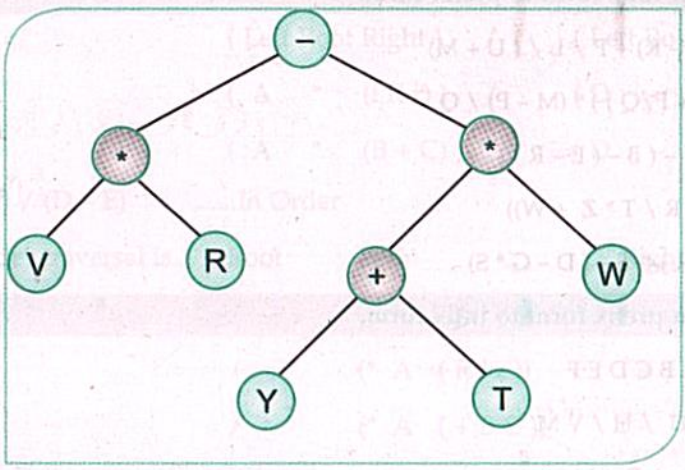
(e) U K * G / U K / G * + U K + G + –

6. Write the Infix, Prefix, Postfix forn from the given Expression Trees :

(a)



(b)



(c)