Data Structure
(Prefix, Infix and Postfix notation)
CSE-207

Md. Manowarul Islam, Dept. of CSE, Jagannath University, Dhaka-1100.

Infix Notation

- To add A, B, we write

  A+B

- To multiply A, B, we write

  A*B

- The operators ('+' and '*') go in between the operands ('A' and 'B')
- This is *"Infix"* notation.

Prefix Notation

- Instead of saying "A plus B", we could say "add A,B " and write

  + A B

- "Multiply A,B" would be written

  * A B

- This is *Prefix* notation.

Postfix Notation

- Another alternative is to put the operators after the operands as in

A B +

and

A B *

- This is *Postfix* notation.

- The terms infix, prefix, and postfix tell us whether the operators go between, before, or after the operands.

Pre A In B Post

- Infix expression is called polish notation
- Postfix expression is called reverse polish notation

Parentheses

- Evaluate 2+3*5.
- + First:

  (2+3)*5 = 5*5 = 25
- * First:

  2+(3*5) = 2+15 = 17
- Infix notation requires Parentheses.

What about Prefix Notation?

- + 2 * 3 5 =

  $= + 2 \ \underline{* \ 3 \ 5}$

  $= \underline{+ \ 2 \ 15} = 17$

- * + 2 3 5 =

  $= * \ \underline{+ \ 2 \ 3} \ 5$

  $= \underline{* \ 5 \ 5} = 25$

- No parentheses needed!

Postfix Notation

- 2 3 5 * + =

  $= 2\ \underline{3\ 5\ *}\ +$

  $= \underline{2\ 15\ +} = 17$

- 2 3 + 5 * =

  $= \underline{2\ 3\ +}\ 5\ *$

  $= \underline{5\ 5\ *} = 25$

- No parentheses needed here either!

- Infix is the only notation that requires parentheses in order to change the order in which the operations are done.

Fully Parenthesized Expression

- A FPE has exactly one set of Parentheses enclosing each operator and its operands.
- Which is fully parenthesized?

(A + B) * C

$\boxed{((A + B) * C)}$

((A + B) * (C))

Infix to Prefix Conversion

Move each operator to the left of its operands & remove the parentheses:

$$(\underleftarrow{(A +} B) * (C + D))$$

Infix to Prefix Conversion

Move each operator to the left of its operands & remove the parentheses:

$$(+ \underset{\longleftarrow}{A\ \ B\ \ *}\ (C + D))$$

Infix to Prefix Conversion

Move each operator to the left of its operands & remove the parentheses:

$$* + A\ B\ \underleftarrow{(\ C\ +}\ D\ )$$

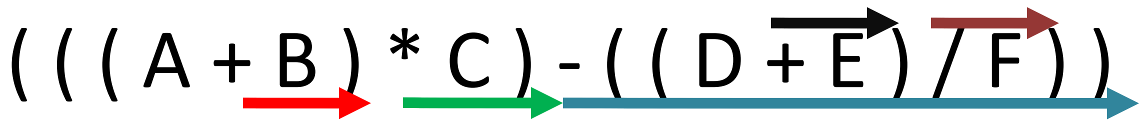
Infix to Prefix Conversion

Move each operator to the left of its operands & remove the parentheses:

* + A B + C   D

Order of operands does not change!

Infix to Postfix


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

A B + C * D E + F / -

- Operand order does not change!
- Operators are in order of evaluation!

Infix to Postfix

Algorithm: Polish (Q, P)

Suppose Q is an arithmetic expression 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 step 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 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 that operator.
        b) Add that operator to STACK.
            [End of if structure]
6. If a right parenthesis is encountered, then:
        a)Repeatedly pop from the STACK and add to P each operator until a left parenthesis is encountered.
        b). Remove the left parenthesis.[Do not add the left parenthesis to P].
            [End of if structure]
        [End of step 2 loop].
7. EXIT.
```

FPE Infix to Postfix

(((A + B) * (C - E)) / (F + G))

- stack: <empty>
- output: []

FPE Infix to Postfix

(((A + B) * (C - E)) / (F + G)))

▲

Symbol	stack	Output
	(	[]

FPE Infix to Postfix

((A + B) * (C - E)) / (F + G)))

Symbol	stack	Output
(	((	

FPE Infix to Postfix

(A + B) * (C - E)) / (F + G)))

Symbol	stack	Output
(	((	
(	(((	

FPE Infix to Postfix

A + B) * (C - E)) / (F + G)))

Symbol	stack	Output
(	((	
(	(((	
(	((((	

FPE Infix to Postfix

+ B) * (C - E)) / (F + G)))

Symbol	stack	Output
(	((	
(	(((	
(	((((	
A	((((	A

FPE Infix to Postfix

B) * (C - E)) / (F + G)))

Symbol	stack	Output
(	((	
(	(((	
(	((((	
A	((((	A
+	((((+	A

FPE Infix to Postfix

) * (C - E)) / (F + G)))

Symbol	stack	Output
(	((	
(	(((	
(	((((	
A	((((	A
+	((((+	A
B	((((+	AB

FPE Infix to Postfix

) * (C - E)) / (F + G)))

Symbol	stack	Output
B	((((+	AB

FPE Infix to Postfix

* (C - E)) / (F + G)))

Symbol	stack	Output
B	((((+	AB
)	(((	AB +

FPE Infix to Postfix

(C - E)) / (F + G)))

Symbol	stack	Output
B	((((+	AB
)	(((	AB+
*	(((*	AB+

FPE Infix to Postfix

C - E)) / (F + G)))

▲

Symbol	stack	Output
B	((((+	AB
)	(((	AB+
*	(((*	AB+
(	(((*(	AB+

FPE Infix to Postfix

- E)) / (F + G)))

Symbol	stack	Output
B	((((+	AB
)	(((	AB+
*	(((*	AB+
(	(((*	AB+
C	(((*(	AB+C

FPE Infix to Postfix

E)) / (F + G)))

▲

Symbol	stack	Output
B	((((+	AB
)	(((	AB+
*	(((*	AB+
(	(((*	AB+
C	(((*	AB+C
-	(((*(-	AB+C

FPE Infix to Postfix

E)) / (F + G)))

Symbol	stack	Output
-	(((*(-	AB+C

FPE Infix to Postfix

)) / (F + G)))

Symbol	stack	Output
-	(((*(-	AB+C
E	(((*(-	AB+CE

FPE Infix to Postfix

) / (F + G)))

Symbol	stack	Output
-	(((*(-	AB+C
E	(((*(-	AB+CE
)	(((*	AB+CE -

FPE Infix to Postfix

/ (F + G)))

Symbol	stack	Output
-	(((*(-	AB+C
E	(((*(-	AB+CE
)	(((*	AB+CE-
)	((	AB+CE-*

FPE Infix to Postfix

(F + G)))

Symbol	stack	Output
-	(((*(-	AB+C
E	(((*(-	AB+CE
)	(((*	AB+CE-
)	((	AB+CE-*
/	((/	AB+CE-*

FPE Infix to Postfix

F + G)))

Symbol	stack	Output
-	(((*(-	AB+C
E	(((*(-	AB+CE
)	(((*	AB+CE-
)	((	AB+CE-*
/	((/	AB+CE-*
(	((/(	AB+CE-*

FPE Infix to Postfix

F + G)))

Symbol	stack	Output
(	((/(	AB+CE-*

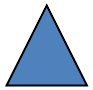
FPE Infix to Postfix

+ G)))

Symbol	stack	Output
(	((/(	AB+CE-*
F	((/(	AB+CE-*F

FPE Infix to Postfix

G)))



Symbol	stack	Output
(	((/(	AB+CE-*
F	((/(	AB+CE-*F
+	((/(+	AB+CE-*F

FPE Infix to Postfix

)))

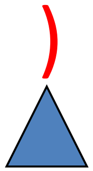
Symbol	stack	Output
(	((/(	AB+CE-*
F	((/(	AB+CE-*F
+	((/(+	AB+CE-*F
G	((/(+	AB+CE-*FG

FPE Infix to Postfix

))

Symbol	stack	Output
(	((/(	AB+CE-*
F	((/(	AB+CE-*F
+	((/(+	AB+CE-*F
G	((/(+	AB+CE-*FG
)	((/	AB+CE-*FG +

FPE Infix to Postfix



Symbol	stack	Output
(	((/(	AB+CE-*
F	((/(	AB+CE-*F
+	((/(+	AB+CE-*F
G	((/(+	AB+CE-*FG
)	((/	AB+CE-*FG+
)	(	AB+CE-*FG+ /

FPE Infix to Postfix

)

Symbol	stack	Output
)	(	AB+CE-*FG+/

FPE Infix to Postfix

Symbol	stack	Output
)	(	AB+CE-*FG+/
)		AB+CE-*FG+/

Example: Q: 4 * (5 + 3) - 24 / 6 and P: ?

Infix Expression Q	Stack	Postfix Expression P
4	(	4
*	(*	4
(	(* (	4
5	(* (	4, 5
+	(* (+	4, 5
3	(* (+	4, 5, 3
)	(*	4, 5, 3, +
-	(-	4, 5, 3, +, *
24	(-	4, 5, 3, +, *, 24
/	(- /	4, 5, 3, +, *, 24
6	(- /	4, 5, 3, +, *, 24, 6
)	(-)	4, 5, 3, +, *, 24, 6, /
		4, 5, 3, +, *, 24, 6, /, -

Postfix Expression P : 4, 5, 3, +, *, 24, 6, /, -

END