

# Arithmetic and Logical Expressions

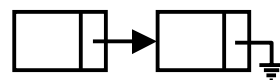
---

---

## THE POLISH NOTATION

Dr.Sudhirkumar J. Patel

---



# Arithmetic and Logical Expressions

---

- repeatedly scan through the expression
- take parentheses and priorities of operators into account

$a + b + c * d - e / g$

$a + b + ( c * d ) - ( e / g )$

$a + ( ( b + c ) * d - e ) / g$

$a + b \leq c \ \&\& \ a + b \leq d$

$( a + b \leq c ) \ || \ ( a + b \leq d )$

# The Polish Notations

---

Q : How can a compiler accept an expression and produce correct code ?

A : Transforming the expression into a form called Polish notation

Infix form	Prefix form	Postfix form
a * b	* a b	a b *
a + b * c	+ a * b c	a b c * +
(a + b) * c	* + a b c	a b + c *

Reverse Polish  
notation

# Expression Evaluations : Stacks

---

$5 * ( ( ( 9 + 8 ) + ( 4 * 6 ) ) - 7 )$

Postfix form : **5 9 8 + 4 6 \* + 7 - \***

Push ( **5** )

Push ( **9** )

Push ( **8** )

Push ( Pop() **+** Pop() )

Push ( **4** )

Push ( **6** )

Push ( Pop() **\*** Pop() )

Push ( Pop() **+** Pop() )

Push ( **7** )

Push ( Pop() **-** Pop() )

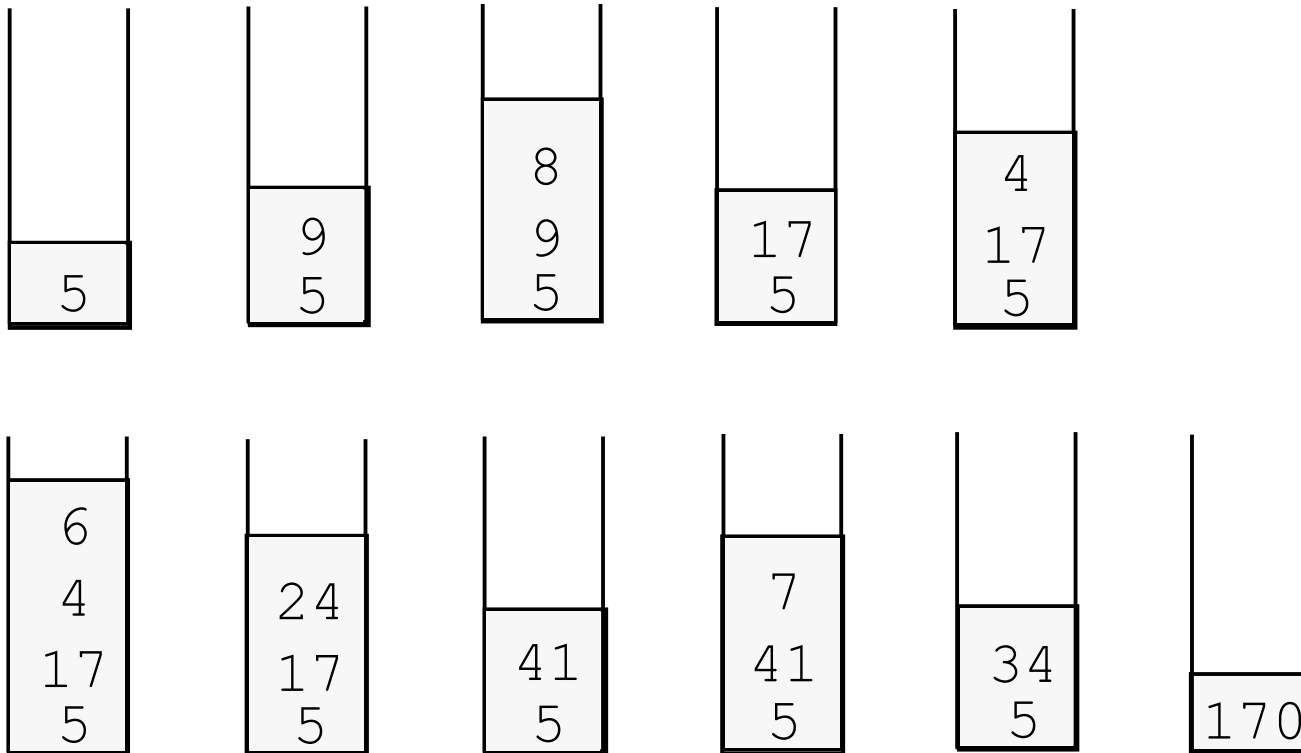
Push ( Pop() **\*** Pop() )

# Expression Evaluations : Stacks

---

5 \* ( ( ( 9 + 8 ) + ( 4 \* 6 ) ) - 7 )

Postfix form : **5 9 8 + 4 6 \* + 7 - \***




# Infix Form $\rightarrow$ Postfix Form

---

A / B ? C + D \* E - A \* C

( ( ( A / ( B ? C ) ) + ( D \* E ) ) - ( A \* C ) )



The diagram illustrates the conversion of the infix expression  $( ( ( A / ( B ? C ) ) + ( D * E ) ) - ( A * C ) )$  to postfix notation. Curved arrows show the operators moving to the right of their operands: the division operator  $/$  moves after  $C$ , the multiplication operator  $*$  moves after  $E$ , the addition operator  $+$  moves after the closing parenthesis of the  $( D * E )$  sub-expression, and the subtraction operator  $-$  moves after the final closing parenthesis.

A B C ? / D E \* + A C \* -

# Infix Form $\rightarrow$ Postfix Form

---

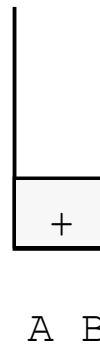
↓  
A + B \* C



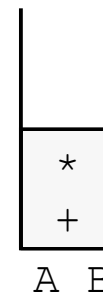
↓  
A + B \* C



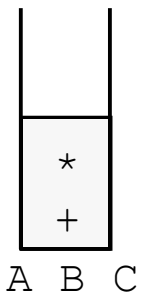
↓  
A + B \* C



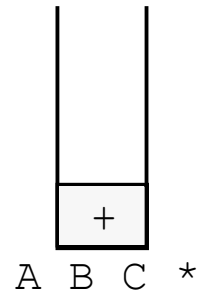
↓  
A + B \* C



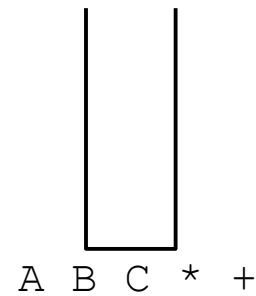
↓  
A + B \* C



↓  
A + B \* C



↓  
A + B \* C

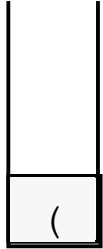


# Infix Form $\rightarrow$ Postfix Form

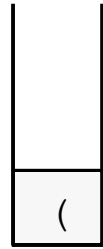
---



$(A+B) * C$



$(A+B) * C$



A



$(A+B) * C$



A



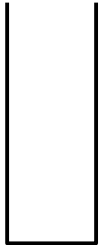
$(A+B) * C$



A B



$(A+B) * C$



A B +



$(A+B) * C$



A B +



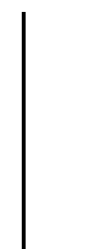
$(A+B) * C$



A B + C



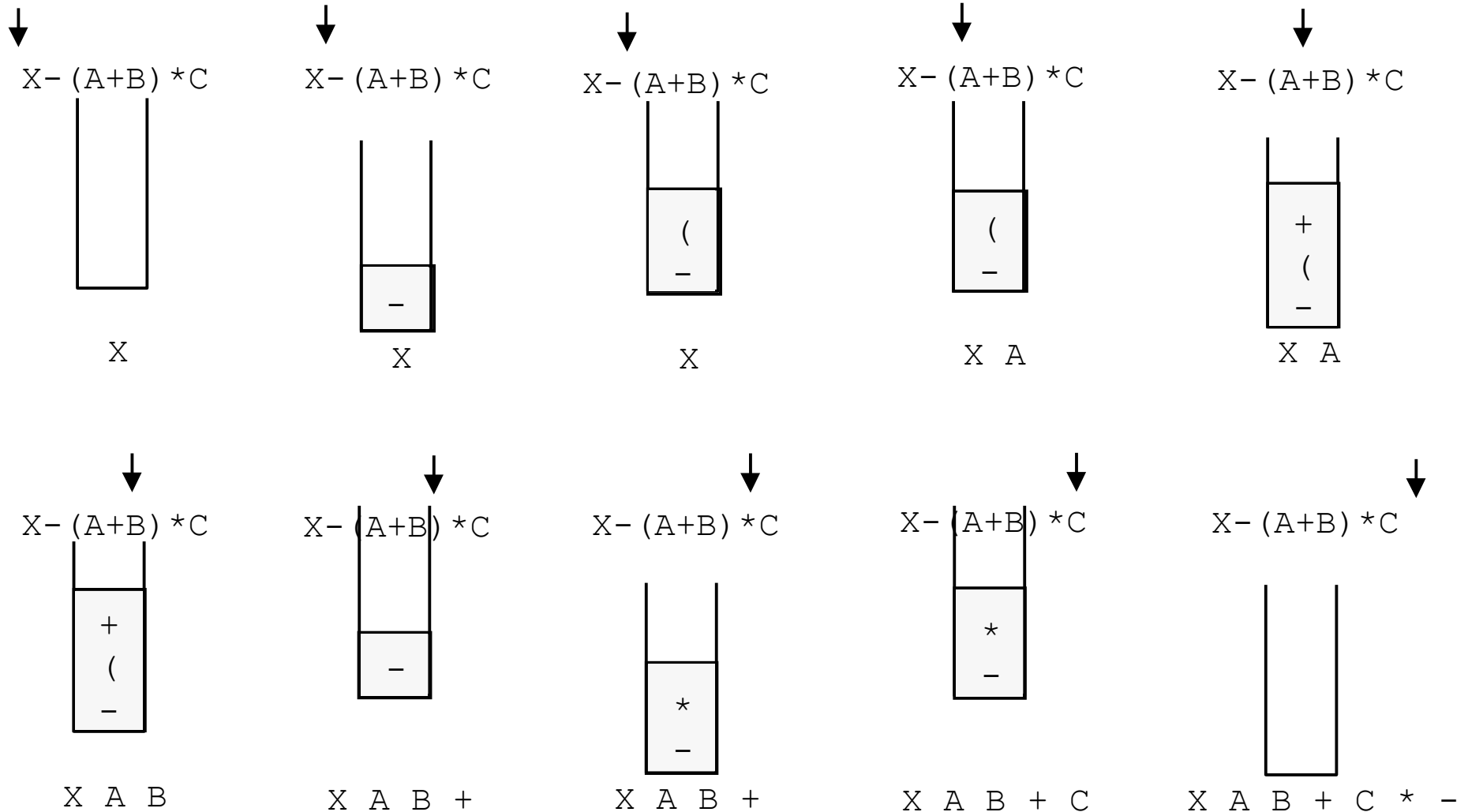
$(A+B) * C$



A B + C \*



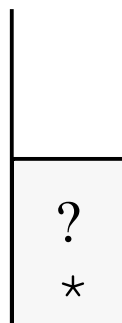
# Infix Form $\rightarrow$ Postfix Form



# Operator Priorities

Symbol	In-Stack Priority	In-Coming Priority
)	-	-
?	3	4
*, /	2	2
+, -	1	1
(	0	4

Operators are taken out of the stack as long as the in-stack priority is greater than or equal to the in-coming priority of the new operator.



↓

input : a\*b?2 + 3  
 output: ab2  
           : ab2?\*