# SNS COLLEGE OF TECHNOLOGY

**Coimbatore-35**
**An Autonomous Institution**

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A+' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

# DEPARTMENT OF INFORMATION TECHNOLOGY

# 19ITT101-PROGRAMMING IN C AND DATA STRUCTURES

I YEAR - II SEM

## UNIT 4 – STACK AND QUEUE

## TOPIC 6 – Expression Parsing

# Expression Parsing

- The way to write arithmetic expression is known as a notation. An arithmetic expression can be written in three different but equivalent notations, i.e., without changing the essence or output of an expression.

- These notations are −
  - Infix Notation
  - Prefix Notation
  - Postfix Notation

# Infix Notation

- infix notation, where operators are used in-between operands.
- It is easy for us humans to read, write, and speak in infix notation but the same does not go well with computing devices.
- An algorithm to process infix notation could be difficult and costly in terms of time and space consumption.

  - a - b + c

  - a,b,c→operands
  - - ,+ →operator

# Prefix Notation

- In this notation, operator is prefixed to operands, i.e. operator is written ahead of operands.
- For example, +ab.
- This is equivalent to its infix notation a + b. Prefix notation is also known as Polish Notation.

# Postfix Notation

- This notation style is known as Reversed Polish Notation.
- In this notation style, the operator is postfixed to the operands i.e., the operator is written after the operands.
- For example, ab+. This is equivalent to its infix notation a + b.

# Precedence

- When an operand is in between two different operators, which operator will take the operand first, is decided by the precedence of an operator over others.
- a+b*c → a+(b*c)
- multiplication operation has precedence over addition, b * c will be evaluated first. A table of operator precedence is provided later.

# Associativity

- Associativity describes the rule where operators with the same precedence appear in an expression.
- For example, in expression a + b − c, both + and – have the same precedence, then which part of the expression will be evaluated first, is determined by associativity of those operators.
- Here, both + and − are left associative, so the expression will be evaluated as (a + b) − c.
  - a+b-c → (a+b)-c
  - a+b*c → (a+b)*c
- a + b*c, the expression part b*c will be evaluated first, with multiplication as precedence over addition. We here use parenthesis for a + b to be evaluated first, like (a + b)*c

| Sr.No. | Operator | Precedence | Associativity |
|---|---|---|---|
| 1 | Exponentiation ^ | Highest | Right Associative |
| 2 | Multiplication ( * ) & Division ( / ) | Second Highest | Left Associative |
| 3 | Addition ( + ) & Subtraction ( − ) | Lowest | Left Associative |

# Infix to post fix and prefix

| Sr.No. | Infix Notation | Prefix Notation | Postfix Notation |
|---|---|---|---|
| 1 | a + b | + a b | a b + |
| 2 | (a + b) * c | * + a b c | a b + c * |
| 3 | a * (b + c) | * a + b c | a b c + * |
| 4 | a / b + c / d | + / a b / c d | a b / c d / + |
| 5 | (a + b) * (c + d) | * + a b + c d | a b + c d + * |
| 6 | ((a + b) * c) - d | - * + a b c d | a b + c * d - |