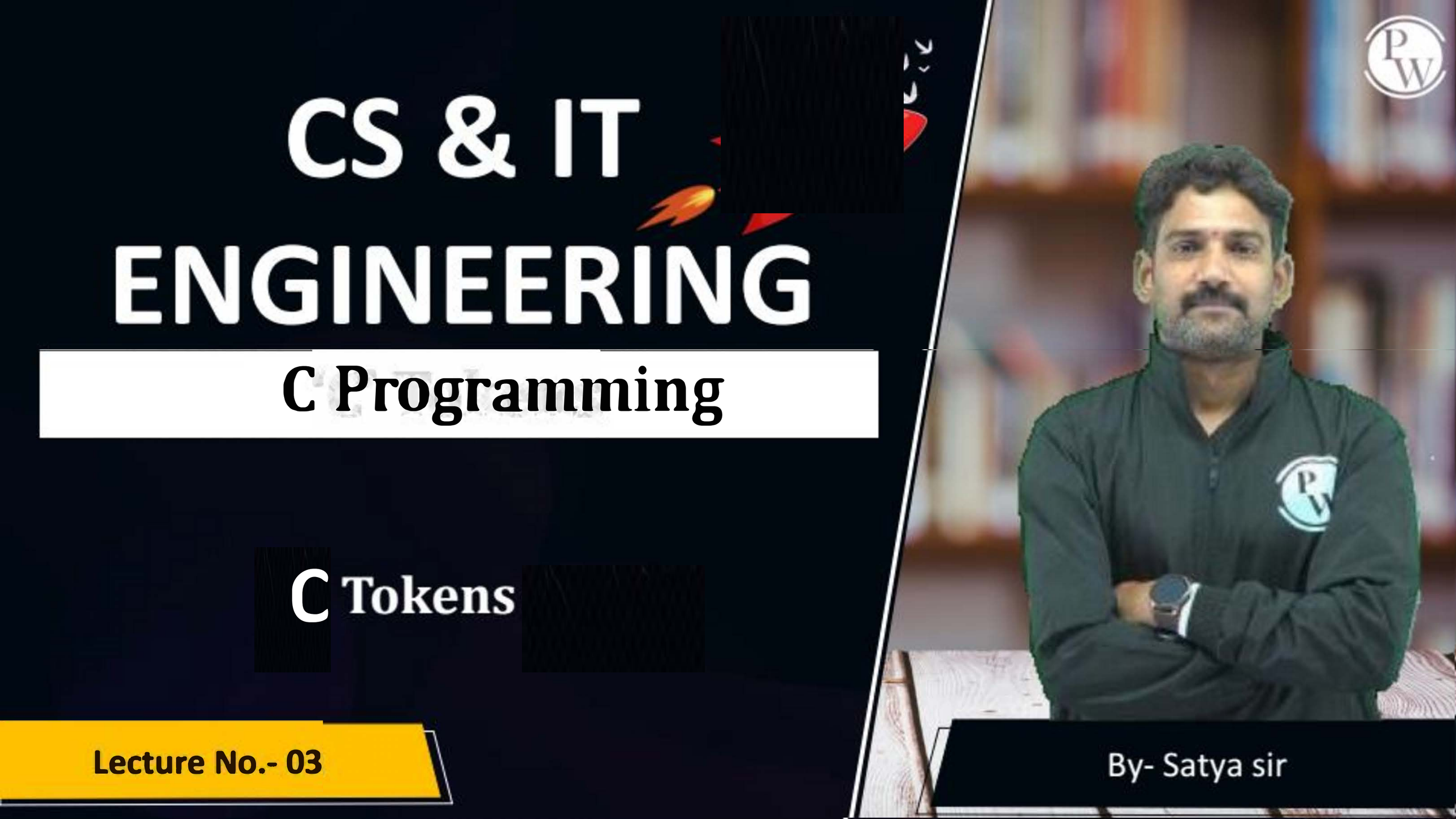
# CS & IT ENGINEERING

## C Programming

## C Tokens



Lecture No.- 03

By- Satya sir

# Recap of Previous Lecture

Operators

- Unary (++, --, +, -, *, &, !, ~, sizeof)
- Binary
- Ternary

- Increment (++), Decrement (--)

# Topics to be Covered

- Binary operators
- Ternary operator
- Operator Precedence & Associativity

# Topic : 'C' Tokens - 3

**Binary operators** : The operator, that Perform operation Using two operands.

- These are sub classified into 5 types

1) Arithmetic operators
2) Logical "
3) Bitwise & shift "
4) Relational / Comparison "
5) Assignment "

# Topic : 'C' Tokens - 3

## Arithmetic operators

| Operator | Meaning / Operation | Operand1 + Operand 2 |
|---|---|---|
| + | Addition | |
| - | Subtraction | Operand1 - Operand 2 |
| * | Multiplication | Operand1 * Operand 2 |
| / | Division | |
| % | Modulus | |

Two operands
⇒ Binary operator

+ Operand
- Operand

⇒ Sign plus
⇒ Sign minus

Unary

* Operand

Pointer Indirection
unary

# Topic : 'C' Tokens - 3

## / and %

Division ( / ) operator ⇒ It Performs division operation and returns Quotient as Result.
The Type of result depends on input values type.

$$3\overline{)17}\quad 5=Q,\quad 15,\quad (2)=R$$

a = 17, b = 3

c = a / b (input values a, b are Integer Constants, so, Quotient also will be Integer only)

c = 5

---

a = 17., b = 3
(OR)
a = 17, b = 3.
(OR)
a = 17., b = 3.

If atleast one input is Real Constant, Then Quotient also will be Real type.

c = a / b ⇒ Result will be in Real form (0.000000 (OR) 5.000000 (OR) 5.666666)

# Topic : 'C' Tokens - 3

**%. (Modulus) operator** : It also Performs division, but returns remainder as Result. Result type always Integer, because % operator can be applied only on Integer constants.

```
    5 = Q
3 | 17
    15
   (2) = R
```

Ex: a = 17 b = 3

c = a/b ⇒ c = 5

d = a % b ⇒ d = 2

x = 17.5, y = 3.5

q = x % y ⇒ [Error, % operator cannot be applied on Real Constants]

NOTE : To Perform mod operation on Real numbers, fmod(op1, op2) is Used. Ex: fmod(x,y)

# Topic : 'C' Tokens - 3

| | Result sign |
|---|---|
| +ve / +ve | +ve |
| +ve / -ve | -ve |
| -ve / +ve | -ve |
| -ve / -ve | +ve |

| | Result sign |
|---|---|
| +ve % +ve | +ve |
| +ve % -ve | +ve |
| -ve % +ve | -ve |
| -ve % -ve | -ve |

Ex: a = +17 b = -3

c = a/b ⇒ c = -5

d = a % b ⇒ d = +2

% Operator Consider Numerator sign only, It ignores denominator sign.

# Topic : 'C' Tokens - 3

## Logical Vs Bitwise Operators

| Logical | Bitwise |
|---|---|
| && Logical AND | & Bitwise AND |
| \|\| Logical OR | \| Bitwise OR |
| ! Logical NOT (Negation) | ~ Bitwise NOT (Complement) |

- Logical operator Perform operation on Truth values (T/F)
  - If Numbers ⇒ Any Non-zero number (±Ve) = True
    only zero = False
  - If characters ⇒ Any NON-NULL characters = True
    only NULL ('\0') = False

### TRUTH TABLE

| A | B | A && B | A \|\| B | !A | !B |
|---|---|---|---|---|---|
| F | F | F | F | T | T |
| F | T | F | T | T | F |
| T | F | F | T | F | T |
| T | T | T | T | F | F |

**Example** d=1 e=1 f=0 g=0

a=-3 b=2 C=0

d=1

d= a && b = -3 && 2 = T && T = T

e=1

e = b || C = 2 || 0 = T || F = T

f= !a && !C = !T && !F = F && T
= F = 0

g= !b || !a
= !T || !T = F || F = F = 0

# Topic : 'C' Tokens - 3

## Bitwise Operators ( Bitwise AND (&), Bitwise OR (|), Bitwise NOT (~), Bitwise XOR (^), Left shift (<<), Right shift (>>) ]

- These Perform operation on Binary Digits or bits (1/0)

- Remember to represent Binary values in Standard forms.
  [16-bit form, 32-bit form, 64-bit form ...]

**Example :** a = 29 b = 18

c = a & b

d = a | b

e = a ^ b

c = a & b

29 = 0000 0000 0001 1101

18 = 0000 0000 0001 0010

29 & 18 = 0000 0000 0001 0000

c = 16

29 | 18 = 0000 0000 0001 1111

d = 31

29 ^ 18 = 0000 0000 0000 1111

e = 15

[2's comp]

| A | B | A&B | A\|B | A^B | ~A |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | |
| 0 | 1 | 0 | 1 | 1 | 2's comp value |
| 1 | 0 | 0 | 1 | 1 | |
| 1 | 1 | 1 | 1 | 0 | |

## Shift operators

### Left shift (<<)

Syntax: `Value << Count`

$\cong$ Value $* 2^{Count}$

Example

A = 13

B = A<<1

C = A<<2

D = A<<3

B = 26, C = 52

D =

| | | |
|---|---|---|
| A = | 0000 0000 0000 1101 | = 13 |
| B = A<<1 | 0000 0000 0001 1010 | = 26 = 13 × $2^1$ |
| C = A<<2 | 0000 0000 0011 0100 | = 52 = 13 × $2^2$ |
| D = A<<3 | 0000 0000 0110 1000 | = 104 = 13 × $2^3$ |

### Right shift (>>)

Syntax: `Value >> Count`

$\cong$ Value / $2^{Count}$

| | | |
|---|---|---|
| A = 57 | 0000 0000 0011 1001 | = 57 |
| B = A>>1 | 0000 0000 0001 1100 | = 28 = 57/$2^1$ |
| C = A>>2 | 0000 0000 0000 1110 | = 14 = 57/$2^2$ |
| D = A>>3 | 0000 0000 0000 0111 | = 7 = 57/$2^3$ |

# 2 mins Summary

- Arithmetic operators

  ( +, -, *, /, % )

- Logical operators ( &&, ||, ! )

- Bitwise operators & shift operators ( &, |, ~, <<, >>, ^ )

To be Contd ...

# THANK - YOU