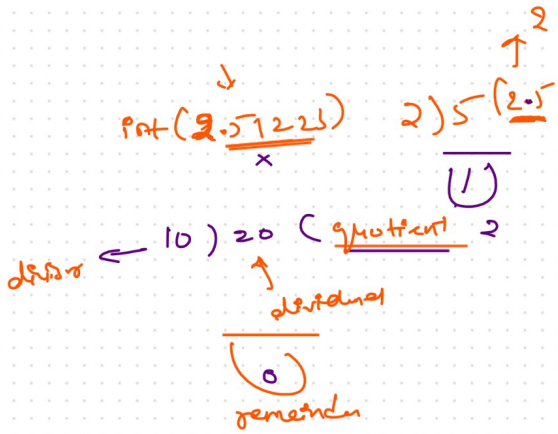
Operators

* special symbols used for calculation, decision making inorder to creat a flow/process

- Data manipulation → Calculations
- Decision making
- Optimization

Types :
1. Arithmetic
2. Relational (a.k.a) Comparison
3. logical (a.k.a) conditional
4. Bitwise
5. Unary Operator

1. Arithmetic : +, −, *, /, %, ← operators



(/) we get only integer part when it encounter decimal it stop proceeding

(%) if we get remainder value as '0' so dividend value is equally divided by divisor, otherwise it is not equally divided

Relational Operator : >, <, >=, <=, ==, !=

* '=' already for assignment operator

* we use it for comparison between two values
* the yield of comparison will be always boolean value i.e True / False

Logical Operators : && , || , !

AND OR NOT

Logical Gate

A	B	Logical AND
1	1	1
1	0	0
0	1	0
0	0	0

1 — True
0 — False

a=4 b=1

expression → ((a > 2) && (b > 2))

exp3 = exp1 && exp2

(4 > 2) && (1 > 2)

T F

T && F = F

5 + 2 / 2 - 1

= (5 + 1 / 2 - 1)

+, /, - → 5 + 0 - 1

5 + -1

(5 + (1/2) - 1)

(5 + 2 * 4 + (1/2) - 1)

exp2 = 2 * 4

exp1 = (1/2)

exp3 = (5 + 8 + 0 - 1)

→ 5 + 8 + 0 - 1

13 + 0 - 1

= 12

(OR operator ||) :

Logical OR

A	B	
1	1	T
1	0	T
0	1	T
0	0	0

a = 1
b = 3

a > 2 || b > 2

1 > 2 || 3 > 2

F || T

True

2 2 2

2 > 2

(NOT operator !)

Logical NOT

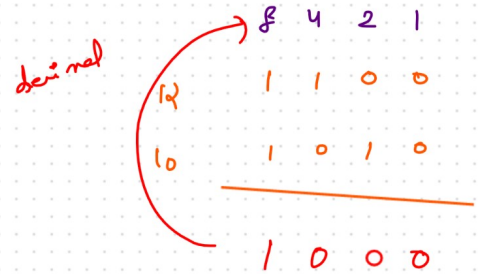
1 → 0

0 → 1

Bitwise Operator : &, |, ^, ~, <<, >>

Bitwise AND OR XOR NOT leftshift rightshift

Bitwise AND 12 & 10



decimal → 12, 10 → Binary representation

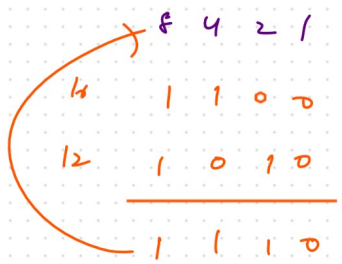
64 32 16 8 4 2 1

Base 2

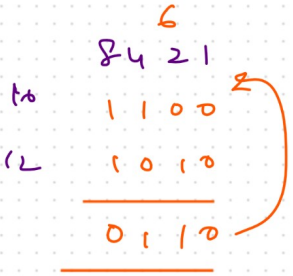
1 1 0 0 } Binary repres

1 0 1 0

Bitwise OR



Bitwise (XOR)



Logical Bitwise

1 1 0

1 0 1

0 1 1

0 0 0

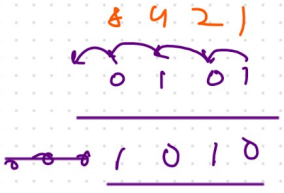
Bitwise NOT (~)

0 → 1

1 → 0

leftshift (<<) → multiplication

5 << 1



operand → 5 x 2 operand

operator

5×2^n

$5 \times 2^1 = 10$

$5 \times 2^4 \rightarrow 5 \times 16 \Rightarrow 80$

5 x 16

32 8 4 2 1 0

1 0 1 1 . . . .

Rightshift (>>) → Division

5 >> 1 5>>2

0010

2

$5/2^1$

$5/2 = 2$

$5/4 = 1$

Unary Operator : ++, --, +, -

a=10

++a → a=a+1 → a=11

a++ → a=a+1 (10) → a=11

pre increment : int a=5

b = ++a → increment first & assignment

post increment : int a=5

b = a++

↑ assign first & then increment

(2) 2's complement

if MSB = 1 its negative

Step 1 In 8 bit system

1. invert bit
2. Add 1
3. Result

128	64	32	16	8	4	2	1
0	0	0	0	1	1	0	0

Step 2 flip

-ve

1	1	1	1	0	0	1	1

MSB → 0 +ve

Step 3 Invert 0 0 0 0 1 1 0 0

Step 4 Add 1 bit

1

Figure → 0 0 0 0 1 1 0 1

8 4 2 1

= -13