CS & IT ENGINEERING

C Programming

Data Types and Operators

Lec- 05

By- Pankaj Sharma sir

Data Types -2

Result of an operator → operands

12/5 ⇒ 2.4 ⇒ 2

int int

12.0/5 ⇒ 2.4

12/5.0 ⇒ 2.4

12.0/5.0 ⇒ 2.4

Q) int a ;

a = 4.0 × 6 / 3 + 2 ;

printf ("%d", a);

a = 4.0 × 6 / 3 + 2 ;

a = 24.0 % 3 + 2 ;

× , /

+

=

Ud Re laat marega

Relational Operator

binary operators

(i) <

(ii) >

(iii) <=

(iv) >=

(v) ==

(vi) !=

10 < 20 → Is 10 less than 20 ?

10 is less than 20

true

10 < 20 → statement is (true) → value result O/P 1

printf("%d", 10<20); → 1 (true)

printf("%d", 20<10); → 0 (false)

1. The result of every relational operator is either 0 or 1.

$a <= b$ → Either $a < b$ OR a is equal to b

```
pf("%d", 10<=20);   1
pf("%d", 20<=5);    0
pf("%d", 20<=20);   1
```

✓ <, >, <=, >=

a == b : Is a equals to b ?

10 == 10 → true O/P 1

10 == 5 → False 0

!= a != b : Is a not equal to b ?

10 != 9 → true 1

2 != 2 → False 0

Unary operators

Unary	+, −	
Arith.	×, /, % +, −	L to R
Relational	<, <=, >, >= ==, !=	L to R
Assignment	=	R to L

```
int i ;
i = 4 ;
```
valid ✓

```
int i ;
i = printf("Hello");
```
5 Evaluate

Q/

```
printf("%d", 14/2 + 3x3);
```
16

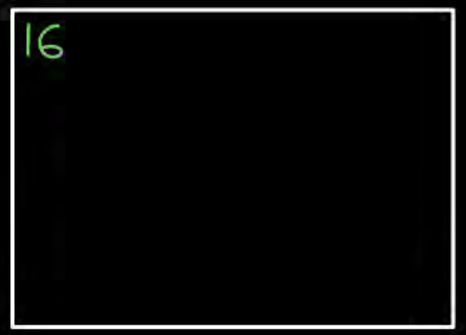
Evaluate

14/2 + 3x3

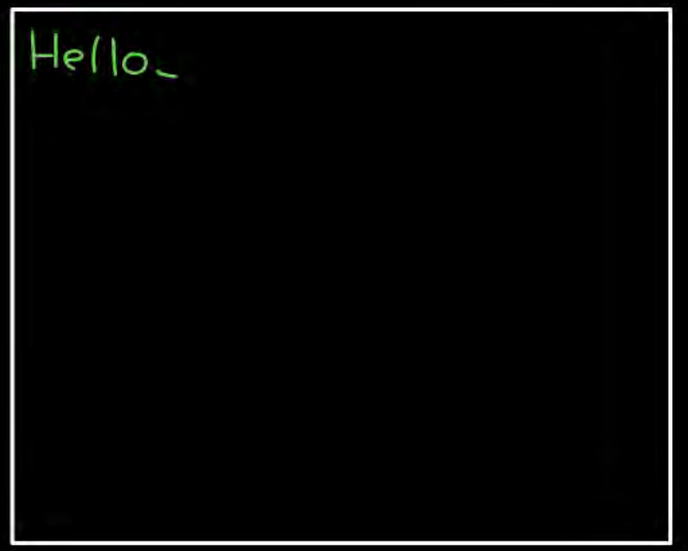
7 + 3x3

7 + 9

16



No. of symbols printed by it.



```
int i;
i = printf("%d", 4/2+3*2);
printf("%d", i);
```

1

8

$4/2 + 3 \times 2$

$2 + 6$

8

81

```
int a;
a = printf("%d", printf("Gate 2023"));
printf("%d",a);
```

Gate 2023 9 1

(i) printf("Gate 2023");

(ii) a = printf("%d", 9); (1)

(iii) printf("%d",a);

Logical Operators

(i) Logical AND (&&)

(ii) Logical OR (||)

] binary

(iii) Logical NOT (!) unary

True ⇒ non-zero

0 ⇒ False

Logical AND(&&)

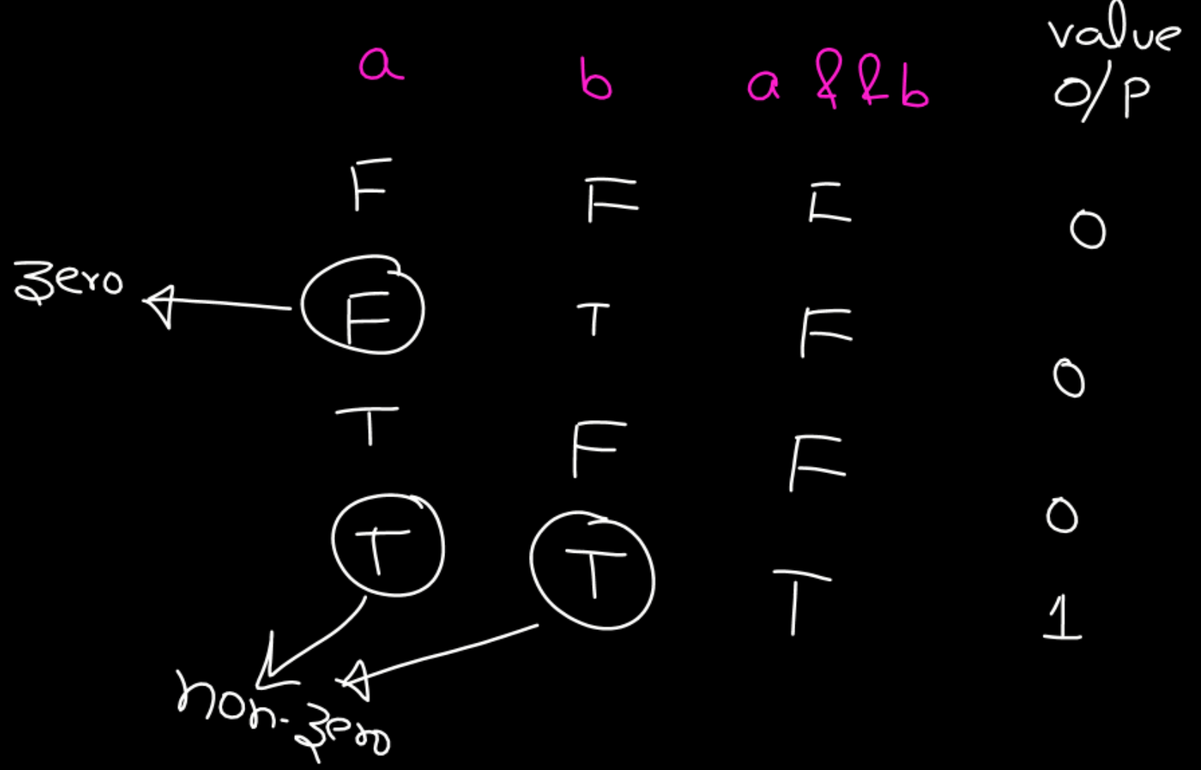
AND → और

a && b

a	b	a && b
F	F	F
F	T	F
T	F	F
T	T	T

$a \times b$ = non-zero

both a, b are non-zero



non-zero (true)

non-zero (true)

```
printf("%d", 2 && -7);
```
1

```
printf("%d", 12.38 && -13);
```
1

```
printf("%d", 1 && 0.0);
```
0

* If both operands are non-zero, the o/p is 1
Otherwise 0

```
int a;
a = 3 && printf("Hello");
printf("%d",a);
```

(non-zero)
5

* Just like relational Operators, the o/p or value of logical operators is either 0 or 1.

```
Hello1
```

Logical OR (||)

The O/P is 1 when atleast one operand is non-zero. otherwise 0.

OR → choice → उपयोग

a	b	a || b	O/P
F	F	F	0
F	T	T	1
T	F	T	1
T	T	T	1

```
printf("%d", 13.78||0.0);
```
1

```
printf("%d", 12||-7);
```
1

```
printf("%d", 0||0.0);
```
0

Logical NOT (!)

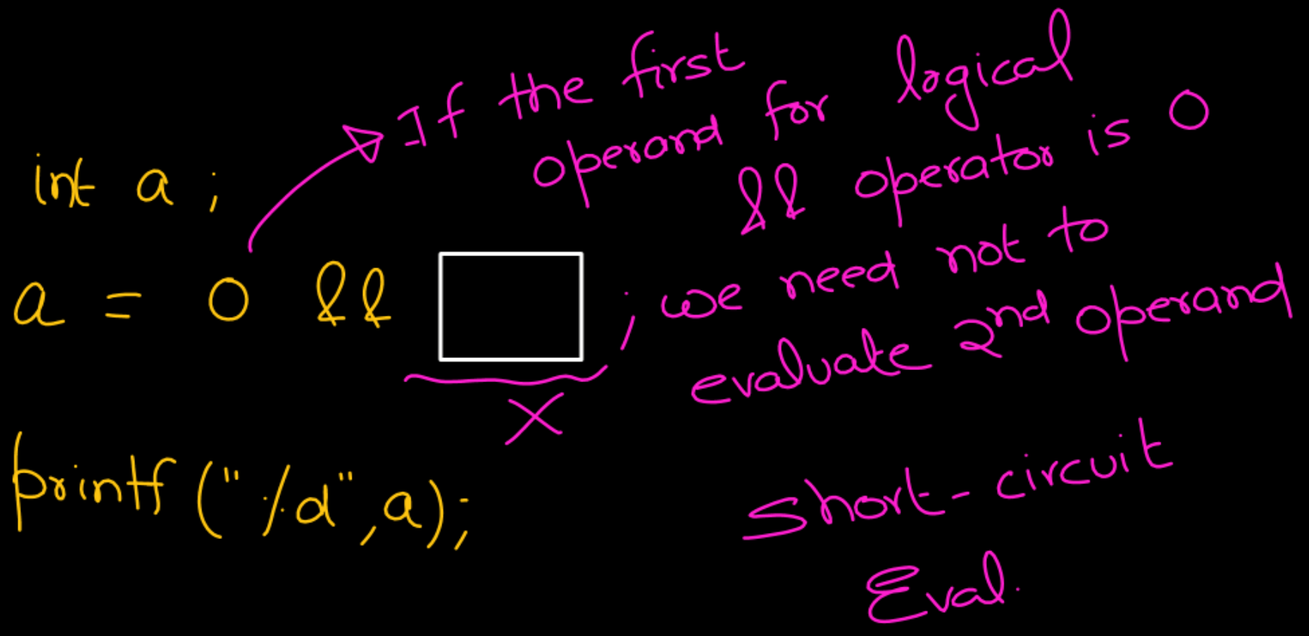
NOT(true) = False

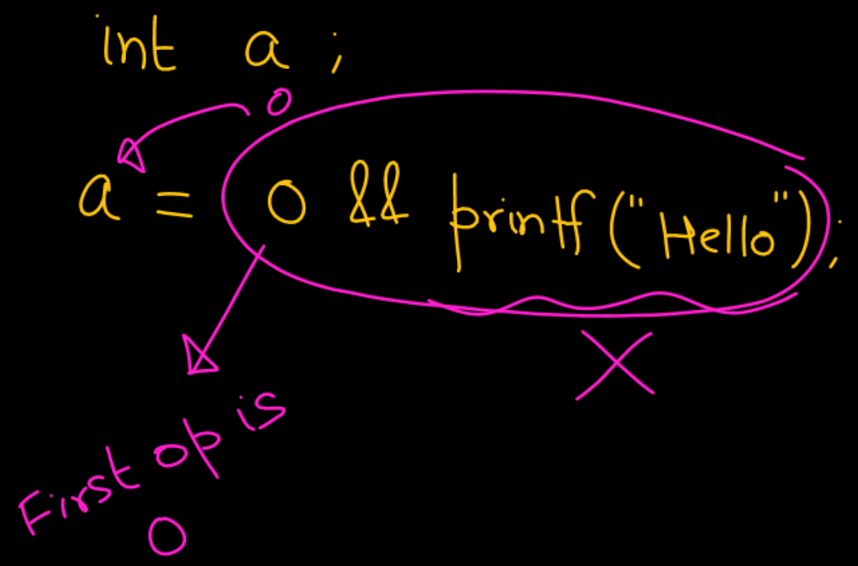
NOT(False) = true

!(5) = !(non-zero) = !(true) = False = 0

!(non-zero) = 0

!0 = !False = true = 1


int a ;
a = 0 && ☐ ;
printf ("%d",a);
If the first operand for logical && operator is 0 we need not to evaluate 2nd operand
X
Short-circuit Eval.


int a ;
0
a = 0 && printf("Hello");
First op is 0
X

```
int a;
a = 10 || ▭ ;
```

If 1st operand for logical OR operator is non-zero, we need not to eval. 2nd operand

do we need to Evaluate ?

No

short-circuit Eval.

2nd op → X Eval.

Q

```
int a;
a = printf("Gate") || printf("2023");
printf("%d", a);
```

1

```
int a;
a = 2<5 == 2<5 && 4>10 != 4>10 ;
printf("%d", a);
```

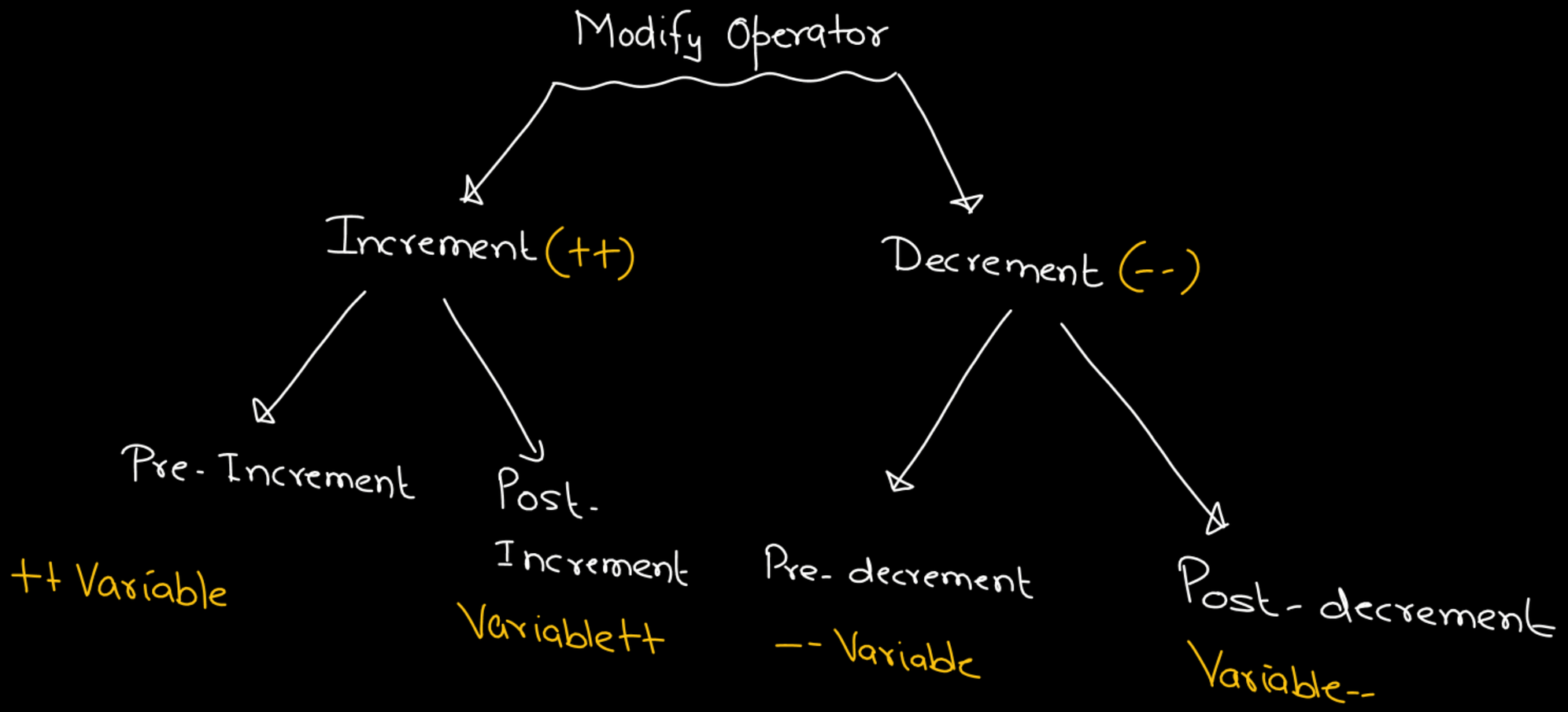
Homework

- Bitwise operators
- Conditional operators

Priority

$3 + 4 \times 6$

$3 + (4 \times 6)$


Modify Operator
Increment (++)
Decrement (--)
Pre- Increment
++ Variable
Post- Increment
Variable++
Pre- decrement
-- Variable
Post- decrement
Variable--

```
int a=5;
  ++a;
printf("%d",a);
```

O/P: 6

$a = a+1$

```
int a=5;
  a++;
printf("%d",a);
```

O/P: 6

```
int a = 5, b;
b = ++a;
printf("%d %d", a, b);
```

(i) a = a+1
(ii) b = a;

O/P: 6 6

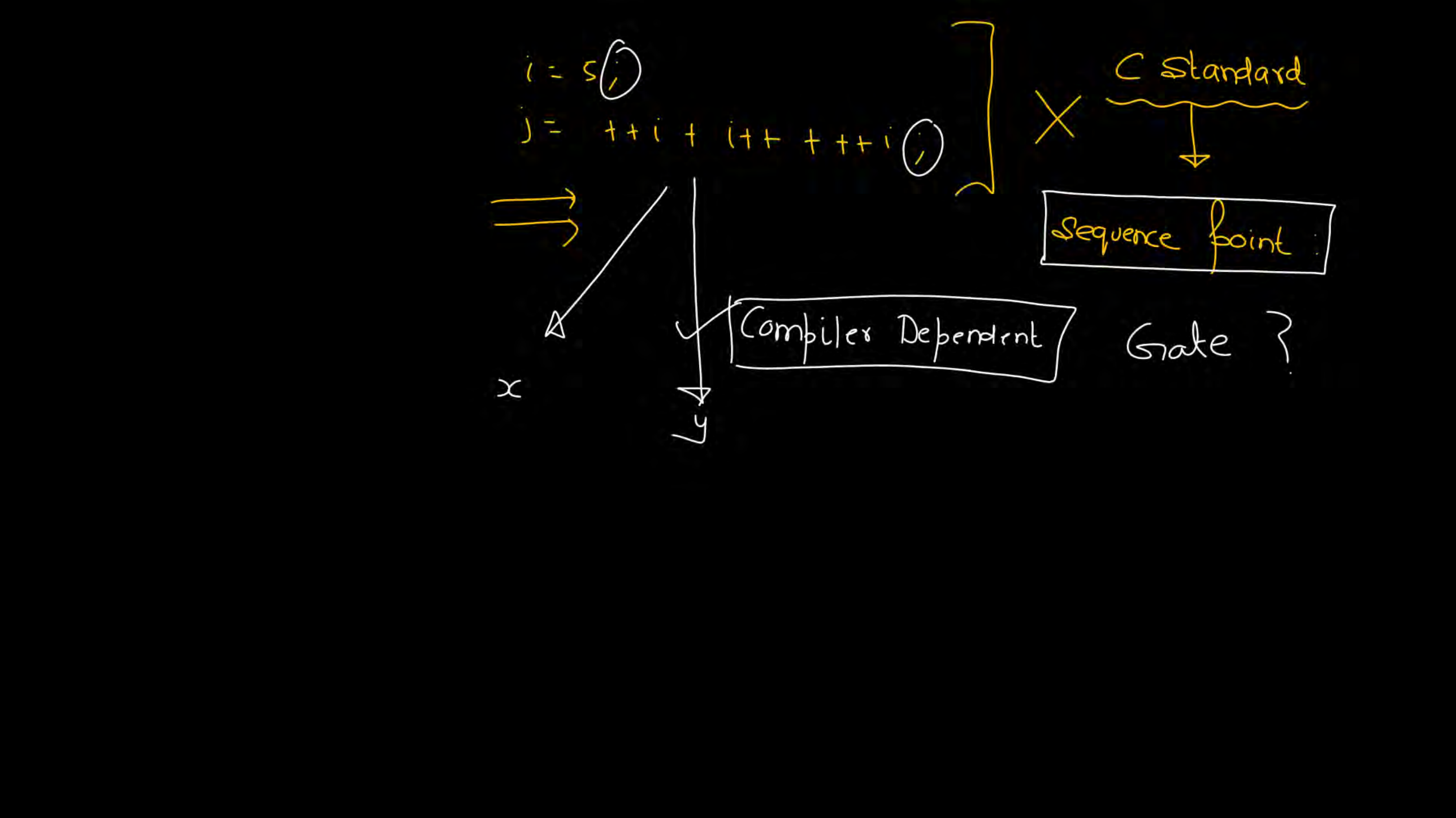
Pre-inc:
(i) First increase the value of variable
(ii) Use the inc value.


```
int a = 5, b;
b = a++;
printf("%d %d", a, b);
```

(i) b = a
(ii) a = a+1

O/P: 6 5

Post-inc: (i) First use the value of var.
(ii) inc. the value


i = 5;
j = ++i + i++ + ++i;
C Standard
Sequence point
x
y
Compiler Dependent
Gate ?

1.

```
int a = 0, b = 1, c;
c = (a++ && ++b);
printf("%d %d %d", a, b, c);
```

① !

② &&

③ ||

c = a++ && ++b;

Post increment

0

c = 0 && ++b;

X

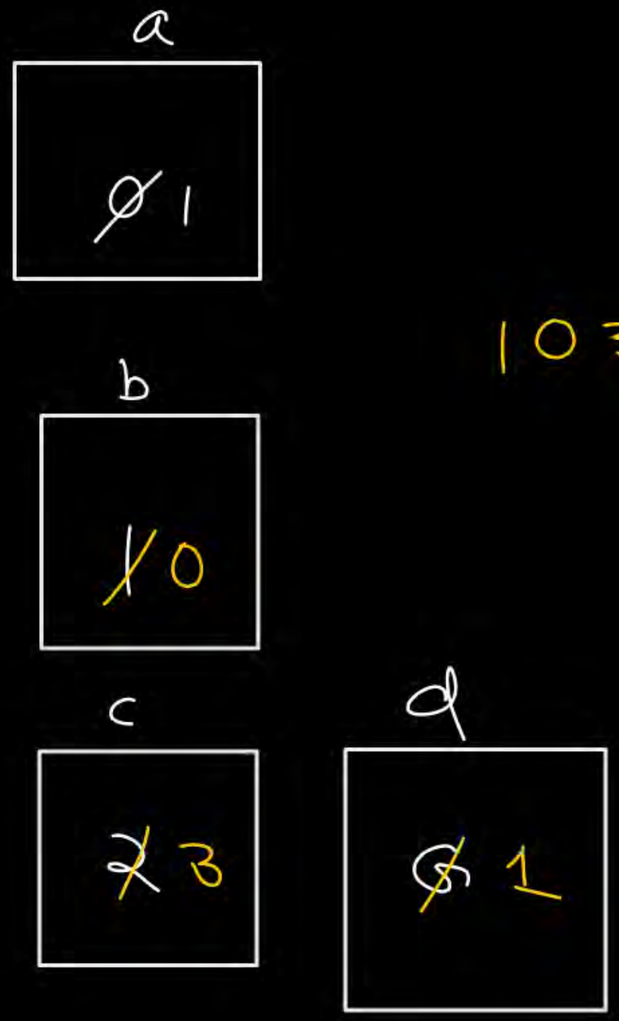
short-circuit Eval

1 || 0

2/

```
int a=0,b=1,c=2,d;
d = ++a && --b || ++c ;
printf("%d %d %d %d", a,b,c,d);
```

1031



(++a && --b) || ++c

(1 && --b) || ++c

(1 && 0) || ++c

0 || ++c

1

d

0 || 3

3 3 1 3 3 3 1

int a=2, b=2, c=0, d=2, e;

e = ((a++ && b++) && c++) || d++ ;

0 || 2

e ← 1

pf("%d %d %d %d %d", a, b, c, d, e);

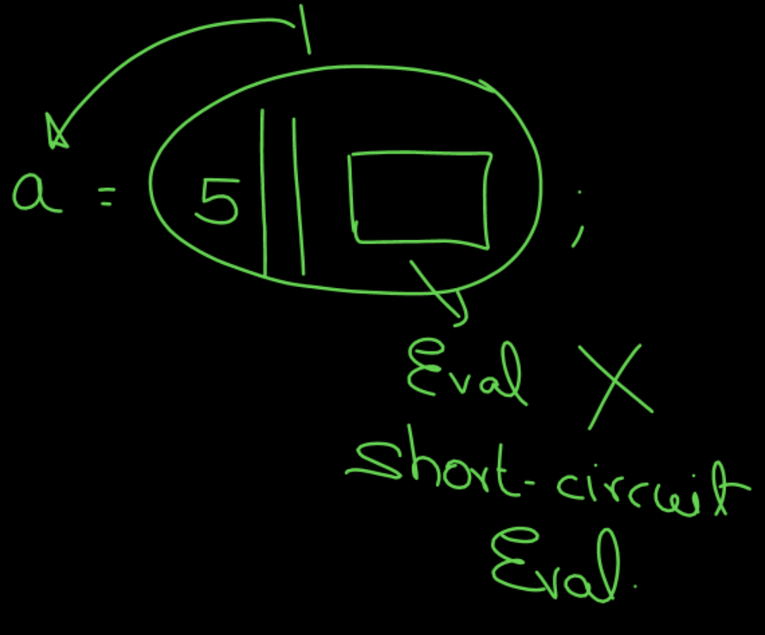
((2 && 2) && c++) || d++

(1 && c++) || d++

1 && 0 0 || d++

```
int a;
a = printf("Hello") || (printf("Gate") && printf("2023"));
printf("%d", a);
```

5




```
Hello1
```

$$a = (pf(\text{"Hello"}) \;\&\&\; pf(\text{"Gate"})) \;||\; pf(\text{"2023"});$$

5 4

HelloGate1

(5 && 4) || —

1 || □ x

unary { ++, --
+, -, ! }

{ /, %, * }
{ +, - }

{ <, <=, >=, > }
{ ==, != }

&&

||

=


THANK YOU GW
SOLDIERS !