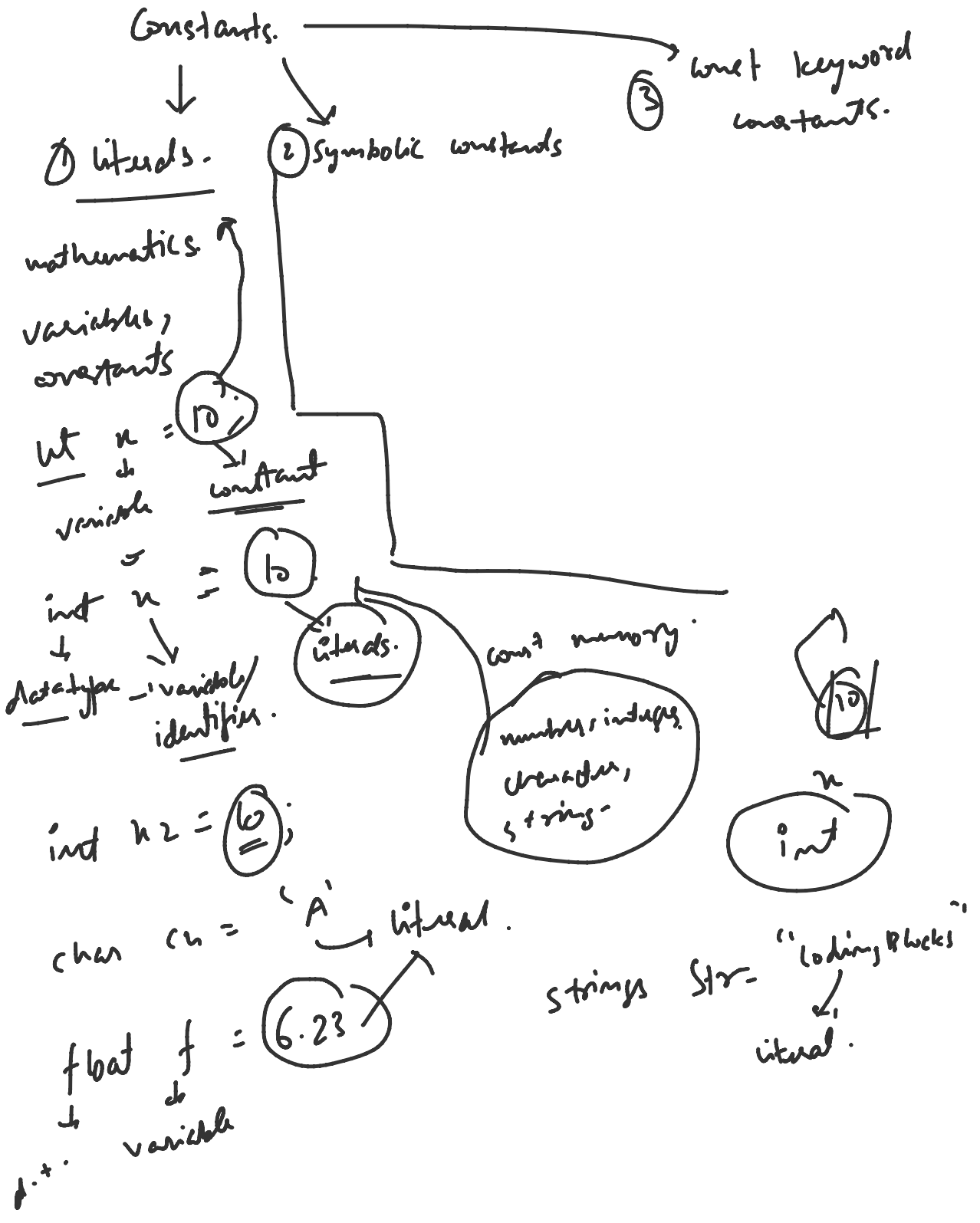
Basics of C++

05 June 2023 19:19



2 Symbolic constants.

Preprocessor Directives / Macros.

#define — —

#define ——— ———

pi → 2.14 + 1

cout << pi * r * r.

<< 2.14 + 1 * r * r. r = 4

<< 16 = 1 + 2.14

18.14

Operators

Unary :- It requires only 1 operand to fulfill its task.

int a = +10; , int b = -10;

++a; a++; --b; b--;

sizeof(1 entity)

Arithmetic o/p. → requires 2 operands and only works when both sides are numbers / easily convertible to number

5 + 9, ✓

'A' + 1 = → 66.

65 ↘ implicit type conversion → ASCII value.

12.0 + 6.0 ✓

+, −, *, /, %.

( ) → condition check,

( ) → condition check, to give priority to the expression inside brackets.

Assignment (=) to assign values

Relational → to compare / comparison.

'A' >= 1
↳ ASCII value.

(same rules as Arithmetic)

↳ it has a return type
it returns either true / false.

Logical operators. ∴ conditions to check know it up.

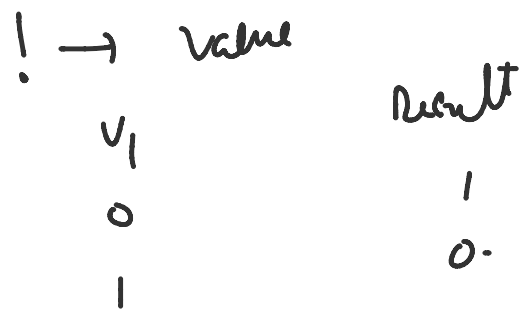
&& → AND.
= → If all true / 1 then 1, else false.

v1	v2	result
0/F	0/F	0/F
1/T	0/F	0/F
0/F	1/T	0/F
1/T	1/T	1/T

If AND encounters any false value, it will come out of that condition instantly without further evaluating rest of the expressions/conditions.

|| → logical OR. → Any 1 → 1.

v1	v2	Result
0	0	0
0	1	1
1	0	1
1	1	1



---

true in C++ / C.J

1 / any non zero value is always considered true.

if (true) ⟶ execute

if (!false) ⟶ execute

if (0) ⟶ not execute

if (!true) ⟶ ———— ;

if (0.1) ⟶ execute

if (-10) ⟶ execute.

only false values are false and 0.

only false
Rest all are true.

if ('A') → execute

Increment

2 types

int a = 10;

(i) ++a.
pre increment.
↳ Pehle increment hoga
fir task hoga.

(ii) a++;
post increment
↳ Pehle task fir increment.

Decrement.

2 types.

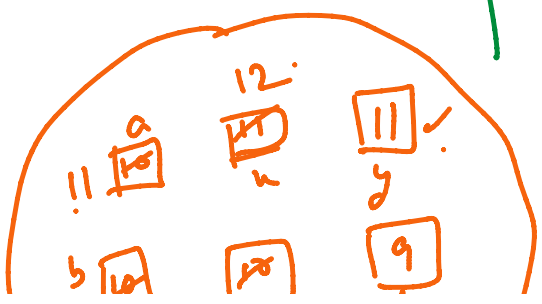
int b = 10;

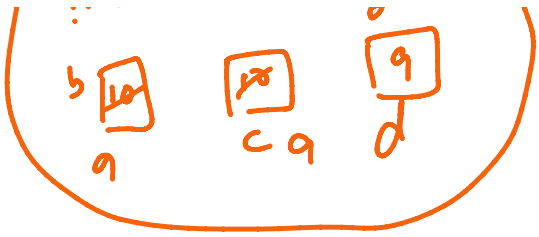
(i) --b
↳ pehle decrement
fir task

(ii) b--;
pehle task
fir decrement.

int a = 10;
int x = ++a; pre
int y = x++; post

int b = 10;
int c = b--; post
int d = --c; pre.





Building Blocks

Amit / Gayatri

a = 0
b = 1
c = 0
d = 1

Kavi

0
0
6
1

Infana 1, 1, 0 2.

Anjali 1, 0, -1, 0.

Bitwise operators.

operation

- right
- left

Binary no answer.

6 + 7 = 13

Binary

6.23 + 7.23

DNS 6 + 7

Binary

V1 + V2

Arithmetic operators needs/works on decimal numbers.

Bitwise operators work on BNS.

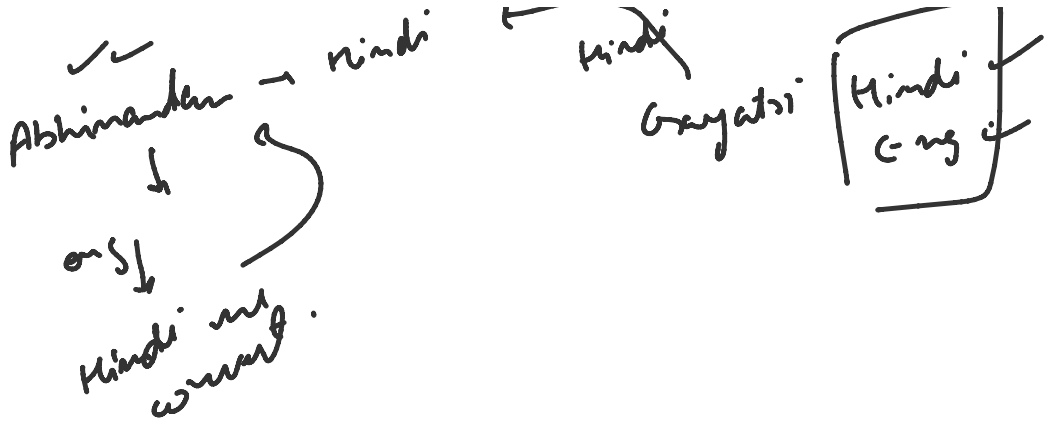
Tuk.

Hindi

Hindi

2 choice

Hindi



Binary digit.
=
It will understand quickly and also act quickly. ← BNS (Generally DNS.)

In Bitwise operators, which they are performing their task, both numbers are present in their binary form.

Bitwise operators → &, |, ~, ^, <<, >>

Bitwise AND '&'

All 1 → 1, else 0.

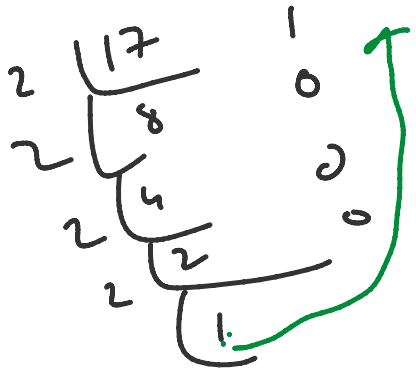
3 & 5

128 64 32 16 8 4 2 1
1 0 1
$2^0 + 2^1 = 3$

$1 \times 2^2 + 0 \times 2^1 + 1 \times 2^1$

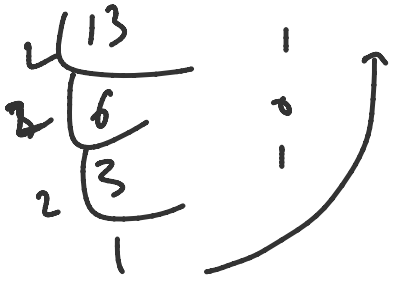
101 = (5)

2 | 5 — 1
2 | 2 — 0
2 | 1 — 1



$10001 = (17)_2$

10001



$(1101) = (13)_2$

$1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 = 13$

(8) (4) (1)

$13 - 8 = 5$

$5 - 1$

$\frac{1}{16}\ \frac{0}{8}\ \frac{1}{4}\ \frac{0}{2}\ \frac{1}{1}$

4

$12 - 8 = 4$

$\frac{0}{16}\ \frac{1}{8}\ \frac{1}{4}\ \frac{0}{2}\ \frac{0}{1}$

385

$385 = (1)$

$\frac{0}{1}\ \frac{1}{0}\ \frac{1}{1}$

8 bits.

0 0 1

9B13

$= (9)$

$9 = \frac{1}{1}\ \frac{0}{1}\ \frac{0}{0}\ \frac{1}{1}$

= (9) ✓

9: 1 _ _ _

13: 1 1 0 1

9 ans = 1 0 0 1

17, 13, 7

17 & 13 & 7 = (1)

17 1 0 0 0 1

13 0 1 1 0 1

7 0 0 1 1 1

8 0 0 0 0 1

Bitwise OR '|'

- Any 1 → 1. / All zero → 0.

3 | 5 = (7)

0 1 1

1 0 1

4' res 1 1 1

9 | 13 → (13)

9 1 0 0 1

13 1 1 0 1

res. 1 1 0 1

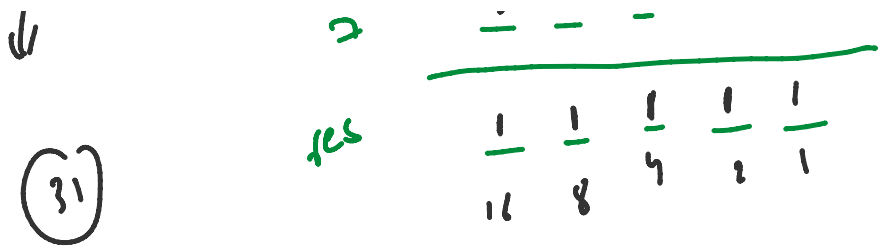
17 | 13 | 7 ↓

17 1 0 0 0 1

13 0 1 1 0 1

7 0 0 1 1 1

1 1 1



(31)

Bitwise NOT ~ (Negation)

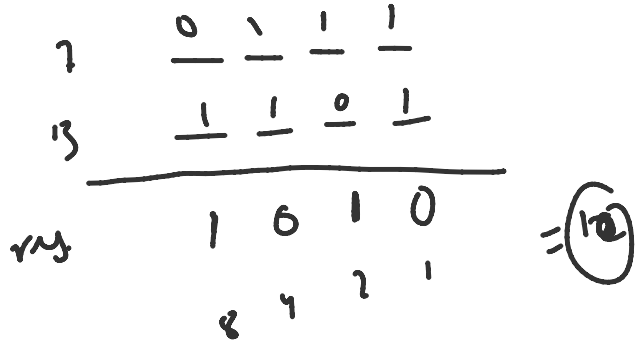
0 → 1
1 → 0

~7 = 000
~7 = 0.

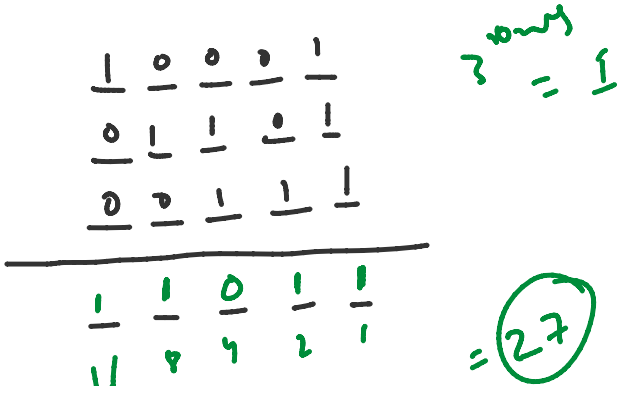
~13 : 1101
0010

Bitwise XOR. ^
if odd no. of 1 → 1
else 0.

7 ^ 13 =



17 ^ 13 ^ 7 =



3 ones = 1

1 1 0 1 1 = 27

<< left shift.

x << y → Shift all bits of x by y places.

S = 101

y = 2

S << 2

1 0 1 << 2 = 20.

S x 2 x 2

1 0 1 0 0

16 8 4 2 1

7 3

→ 1 1 1 << 3

7 x 2 x 2 x 2

1 1 1 0 0 0

32 16 8 4 2 1

56

int → 4 bytes → 32 bits.

1

0 0 . . . . . 1 1 0 1 1 1 = 7.

29 bits

fall off.

If I say x = x x 2^y

⇒ x = x x 2^y

>> right shift operator.

shift y bits to the right.

x >> y

>>

7 >> 2

0 0 1 = 1

If x >> y => $x = \frac{x}{2^y}$

~ (~13)

signed / unsigned

int

range

32 bits

o/p → 1 1 1 ... ... 1 0 0 1 0

2's complement of (14).

-ive representation.

~1 → -2

~(-2) → 1

0 0 1

If signed int, and we are doing
Bitwise NOT on it → -(num+1)
~num → -(num+1)

Bitmasking → concept / questions solved.

7 ^ 7 →

```
  1 1 1
  1 1 1
-------
  0 0 0
```

5 ^ 5 →

```
  1 0 1
  1 0 1
-------
  0 0 0
```

Any number Xored with itself will always be 0.

int a = 10;

a++ / ++a; (Increase value of a by 1)

(Increase value of a by 1000)

n = a + 1000;

by 1000

a = a + 1000;

compound Assignment operators.

a += 1000

assignment

a _= — value

original operator to be performed

a = 10 x 10

equal. a = a x 10;
a *= 10;

a = 10 , << 2.

a <<= 2;

---

ques 0, 1, ~, ^, <<, >>

You are given some numbers in which every no. appears twice except (1) number. Find out that number.

n = 5

1, 2, 1, 3, 2. → 1 time. ?

we will XOR each no. with

f_no = 1

0 1 1
0 0 0
0 1 1

0 ^ 1 ^ 2 ^ 3 ^ 2 =) task.

0 ^ 3 = 3

2 ^ 2 = 0 ?

0 1 0
0 1 0

13 = (1101) = 3

you are given a number and you have to find out no. of 1's in its binary form.

17 = 10001 = 2

!