Bitwise Operators & Loops

Bitwise Operator (bit Level)

- → AND → &
- → OR → |
- → NOT → ~
- → XOR → ^

Truth Table

{&}

a	b	O/P
0	0	→ 0
0	1	→ 0
1	0	→ 0
1	1	→ 1

{|}

a	b	O/P
0	0	→ 0
0	1	→ 1
1	0	→ 1
1	1	→ 1

{~}

a	O/P
0	→ 1
1	→ 0

why·it came -2 in bool also?
sa

{^}

a	b	o/p
0	0	0
0	1	1
1	0	1
1	1	0

e.g → 5^5 → 0

e.g →
a = 2
b = 3

a&b ⇒ 2

How its works?

$$\begin{array}{r} 1\ 0 \\ \&\ \underline{1\ 1} \\ \boxed{1\ 0} \end{array}$$

Answer

example ⇒
Let take num = 1
~num
o/p → -2

steps ⇒
1. All bits flipped
2. -ve humber is accessed using 2's compliment
3. Hence -2.

Homwork → $(\sim a)$ vs $\sim(a)$
output verify

Left and Right shift operator
"<<" ">>"

suppose,
a = 2
0000 000010
If we do,
a<<1
then, 0000....000100

a>>1
then divide by 2^1

a>>n
then divide by 2^n

Note:
If -ve number is there, and if try to do right shift then compiler will handle.
If we shift by negative number then it will give a garbage value.
If -ve number is there, and if it is signed and if try to do right shift then compiler will give a big value.

Pre/Post Increment / Decrement operator

pre-increment → ++a

post-increment → a++

pre-decrement → --a

pre-decrement → a--

e.g →

```
main() {
  int a=5;
  ++a;
  cout<<a;
  return 0;
}
```

o/p → 6

Comparision ⤵

```
main() {                         | main() {
  int a=10;                      |   int a=10;
  cout<< (--a) *10;              |   cout<< (a--) *10;
}                                | }
```

$\boxed{109}$ $\boxed{109}$

o/p → 90 | o/p → 100

Home Work :

```
main() {
    int a=10;

   cout << (++a) * (a++);
}
```

Loops

for(i=0; i<10; $\underbrace{\text{i++ or ++i}}$)

Break & continue

To exit loop, break is used.

To skip iteration, continue is used.

Variable Scoping

Local variable
Global variable

e.g →

```
main() {
for(int i=0; i<5; i++){
     cout<<i;
}
 cout<<i;  ← Here 'i' cannot
               be accessed.
}
```

Local variable is more prioritize then global variable.

Global variable is bad practice.

Expression solving

Operator Precedence Table

To avoid this table, use brackets.