Lecture 5

operators, function & macros

* **Bitwise operators :-**

The operators which are used in o.p. used four bit's called bitwise operators. Their are four types.

1. AND (&)
2. OR (|)
3. Not (~)
4. XOR (^)

Bit's → 0 1

→ AND (&) :-

a	b	a&b
0	0	0
0	1	0
1	0	0
1	1	1

→ OR (|) :-

a	b	a|b
0	0	0
1	0	1
0	1	1
1	1	1

→ ~ (Tilda / Negation) Not :-

a	~a
0	1
1	0

→ XOR (^) :-

a	b	a^b
0	0	0
1	0	1
0	1	1
1	1	0

Diff value 1

Same value 0

Ex:-

```
bool a = true;
bool b = false;
cout << (a & b);
```

o/p → 0

```
bool a = true;
bool b = true;
cout << (a & b);
cout << (a | b);
```

o/p → 1

o/p → 1

```
bool a = true;
bool b = false;
cout << (a | b);
```

o/p → 1

```
bool a = false;
bool b = false;
cout << (a & b);
cout << (a | b);
cout << (a ^ b);
cout << ~a
```

o/p → 0

o/p → 0

o/p → 0

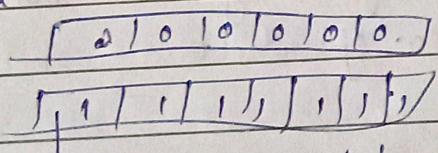
o/p → -1

why (-1)

it is done by 2's compilement

a = false

a = 0



cout << (2 & 3); o/p - 2

cout << (5 & 10); o/p - 0

↳ 5 → 101

10 → 1010

5 & 10

0 0 0 1 0 1

& 1 0 1 0

0 0 0 0 0 → 0//

cout << (3 | 4); o/p - 7

* Left & Right shift operators :-

⊙ Left shift << ← (1 High value)

⊙ Right shift >> → (1 Low value)

→ Left shifting :-

Ex :- a = 5 → 101

0 0 0 0 0 - - - - 101

<< 1 (left shift by 1)

0 0 0 0 0 - - - 0 1 0 1 0

o/p - 10

Generic formula :- = $2^n \times n$.

multiple of 2^n of number

n = no. of shift

Code :- int a = 12;

a = a >> 1;

cout << a << endl;

→ Right shift :- (>>) ⟶

a = 8 → 1000

0000000 ---- 1000

>>1 (Right shift by one)

000000 ---- 0100

o/p → 4 ↳

Generic formula :- $\frac{a^n}{2^n}$

divisible by 2^n with number of shift

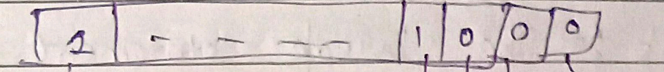
Q- Can't I say condifential that Right shift provide divide by 2 ?

Why

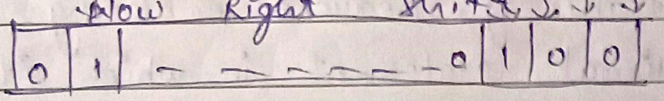
Ans/-

If there is any -ve no. then

let a = -8

-ve 

Now Right shift



Then it create high value of +ve no. So, that's why didn't say it is divisible by 2.

Post / Pre → Increment & Decrement.

(a) ++a → Pre Increment

(b) a++ → Post Increment

(c) a-- → Post Decrement

(d) --a → Pre Decrement /decrease

Pre :- First increase then use

Post :- First use then increase/decrease

Exi- int main(){
int a=5;
cout << a++;
cout >> ++a;
o/p → 5, 7

Q- Homework must check
Exi- int a=5;
cout << (++a) * (++a);
↳ < → 7 ↙
o/p - 42
but 49 be output

Note:- i=i+1;
i+=1;
i++;
} These are same type but method to write.

* Break :- The break statement ends execution of nearest enclosing loop or conditional statement in which it appears.
Syntax:-
break;

n=5 Exi- for(int i=0; i<n; i++){
cout << "Harsh";
}break;
It runs only one time but not run by 5 times.

⊙ Break is also called fifth ji in shaddi (mujhe nhi jana barat chal

* Continue:- The continue statement breaks one iteration (in the loop), if a specified condition occurs, & continue with next iteration in the loop.

"Continue keyword is used for skip our miss the iteration in code"

Syntax:-

```
continue;
```

Ex:-
```
for (int i=0; i<5; i++){
    cout << i;
    continue;
}
```
o/p - 0, 2, 2, 3, 4

Ex:-
```
for (int =0; i<5; i++){
    continue;
cout << "babbar";
}
```
o/p → Blank (Nothing)

⊙ Continue is also called tuji in shaddi (chalo chalo)

Variable Scoping:- Variable scoping define the scoped of declare variable in the code. They are two types:-

① Local variable (Gooli ka dor)
② Global variable (Behar ka dor)

Global variable:-

A global variable can be accessed from anywhere in the entire program. It is usually declared at the top of the program outside of all blocks & functions of the program.

Local variable:-

A variable defined inside a function (defined inside function body b/w braces) is called a local variable.

It's scope is only limited to the function where it is defined.

Ex:- int a; // declare
int b = 5; // initialize

We can declare a one variable at only one time.

Ex:- int b = 5; → Global variable
if (true) {
int b = 5; } → Local variable
}

* Best practice we must use local variables

Operator precedence -

Category	operator	Associativity
Brackets	(), [], ->, ., ++, --	Left to Right
Unary	!, ~, +, -, --, &, *, sizeof	Right to left
Multiplicative	*, /, %	Left to Right
Additive	+, -	Left to Right
Bitwise shift	>>, <<	Left to Right
Relational	<, >, <=, >=	Left to Right
Equality	==, !=	Left to Right
Bitwise AND	&	Left to Right
Bitwise XOR	^	Left to Right

Precedence: High ↓ Low

① Must use (Brackets) to solve problems.

Switch Case :- Switch case is used when we have to decide only one case from multiple case.

Syntax :-

```
switch (expn) {
    case 1; - - -
            break;
    case 2; - - -
            break;
    default; - - -
            - - -
}
```

⊙ We can also use multiple if case as switch case would do.

Ex:- if {
    }
if {
    }
if {
    }
if {
    }

→ Ex:- switch (val) {
    case 1: cout << "Babbar";
        break;
    case 2: cout << "Love";
        break;
    case 3: cout << "Harsh";
        break;
    default: cout << "Gupta";
        break;

Q- Can we use "continue" in switch case?

Ans:-

We can not use continue statement with switch case as it is compatible with them.

18. Can I write these following in switch case statement?

Ex: case 1 ⟶ case 'a';

a) 'a' → It provide default value

b) 'Navrah' → It shows an error

c) Expression → We can use expression

d) Can I write same two case → Error (No)

e) -ve value → We can use -ve value

f) float value → We can not float value

g) sentence → We can not use sentence

h) operators → we can not use operators or symbol.

i) boolean → We can use but we have declare boolean variable and switch case in true or false only.

j) continue; → We can not use continue it shows error

Q. Find output? int a = 5;

a = ~a → -6

a = ~(a) → -6

a = (~a) → -6

a = ~(~a) → 5

a = ~~a → 5

a = (~~a) → 5