UNIT - 2 (ANURAG GUPTA)

OPERATOR & OPERANDS

An operator is a symbol that tells the compiler to perform specific mathematical or logical operations.
And on which operator is applied is known as operands.

ex: a + b → operator, operands

C Language provides the following types of operator:

1) Unary operator
- (which applied on single operand only).
- Are of following type
  1. Increment ++
  2. Decrement --
  3. unary minus -
  4. Bitwise Complementation ~
  5. Logical NOT !

2) Binary operator
- (which applied on two operands)
- Are of following type
  1. Arithmetic Operator:
     - Addition +
     - subtraction -
     - multiplication *
     - Division /
     - Modulus %
  2. Relational Operator:
     - less than <
     - less than equal to <=
     - Greater than >
     - Greater than equal to >=
     - Double equal to ==
     - Not equal to !=

3) Ternary operator (conditional)
(which applied on Three operands.)
- Only one operator in this:

  ? :

  ↑
  Ternary or conditional operator

3) Logical Operator:
- logical AND &&
- logical OR ||
- logical NOT !

4) Bitwise Operator:
- Bitwise AND &
- Bitwise OR |
- Bitwise X-OR ^
- Bitwise left shift <<
- Bitwise Right shift >>
- Bitwise Complementation ~

5) Assignment Operator
- =
- +=
- -=
- *=
- /=
- %=

Use & Meaning of all operators:

① Increment (++) / Decrement (--) Operator: Increment operators are used to increase the value of the variable and decrement operators are used to decrease the value of the variable by one (1).

Syntax: ++variableName or variableName++
↳ pre increment ↳ post increment:

eg: ++i or i++
+1+i i+1

pre decrement ← --variableName or variableName-- → post Decrement
eg: --i = -1+i i-- = i-1

Binary Operators: ① Arithmetic operators:

S.NO.	ARITHMETIC OPERATORS	OPERATION	EXAMPLE	
1.	+	Addition	A+B, 5+4=9	* In modulus if numerator is less than denominator so numerator becomes answer (No need to find remainder) & only we check the sign of Numerator for answer.
2.	—	Subtraction	A-B, 5-4=1	
3.	*	Multiplication	A*B, 5*4=20	
4.	/	Division	A/B, 5/4=1.25	
5.	%	Modulus	A%B, 5%4=1, 4%5=4 (Gives remainder as output) -4%5=-4, 4%-5=4	

② Relational Operator: Are used to find relation between two variables. i.e compare two variables.
and * if values are true (After compare) so it will return 1.
* if values are false (After compare) so it will return 0.

SNO	OPERATORS	EXAMPLE	DESCRIPTION
1	>	x > y	5>3, so it will return 1 (as 5 is greater than 3, condition is True)
2	<	x < y	5<3, so return 0. (false)
3	>=	x >= y	5>=3, so return 1 (True). Here 5 is not equal to 3, but showing greater, so if one condition is satisfied it will take it as true.
4	<=	x <= y	5<=3, so return 0 (false)
5	!=	x != y	5 != 3, so return 1 (True) (Here 5 is not equal to 3).
6	==	x == y	5==3, so return zero (False) * Double equal only used to compare two variable. And if we want to give value to variable, we need to use = (single equal).

(3) Logical Operators : These operators are used to perform logical operations on expression.

* Here if value is non-zero (+or-) $\xrightarrow{\text{convert}}$ 1
* and if value is zero $\xrightarrow{\text{convert}}$ 0

and then apply truth table of logic gates.

S.NO.	OPERATORS	EXAMPLE	DESCRIPTION
1.	&&	(x>5) && (y<5)	It will return true (1) when both conditions are true. let (9>5) && (4<5) → 1 && 1 = 1 (True) [1&&1=1 according to AND gate truth table]
2.	||	(x>=10) || (y>=10)	It return true when at-least one of the condition is true. (12>=10) || (9>=10) → 1 || 0 = 1 (True)
3.	!	!(x>5)	It reverse the state of the operand. * !(9>5) → !1 = 0

Truth Table for operators

x	y	x & y (AND)	x ! y (OR)	x ^ y (X-OR)	! x (NOT)	! y (NOT)
0	0	0	0	0	1	1
0	1	0	1	1	1	0
1	0	0	1	1	0	1
1	1	1	1	0	0	0

(4) **Bitwise operator**: These operators are used to perform bit operations.
Decimal values are converted into binary (0,1) values and bitwise operators work on these binary bits.

* Here 1st convert given decimal number into binary.
2nd apply given bitwise operation [If necessary use truth Table]
3rd convert result from binary to decimal.

ex: $x = 5$, $y = 3$
=
so $x \& y$

5 & 3 ↓ convert into binary (1)
101 & 011
↓ apply AND truth Table (2)
001
↓ convert into decimal (3)
1 Ans.
=

$x ! y$

5 ! 3
101 ! 011
111
= 7 Ans.

$x \wedge y$

5 ^ 3
101 ^ 011
110
= 6 Ans.

~x or ~y
↳ Bitwise complementation
This is a unary operator so applied either on x or y.
we will not use truth table here.
$\sim x = -(x+1)$
or $\sim y = -(y+1)$

eg: ~5
= ~(5+1)
= -6 Ans.

~3
~(3+1)
= -4 Ans.

Bitwise left shift and Right shift:

① $x << n$ [Means Here x (after converting into binary) will be left shifted by n place]

② $x >> n$ [Here x will be Right shifted by n place]

⑤ **Assignment Operators:** In C Programs, values for the variables are assigned using assignment operators.

* If value 14 is to be assigned for the variable cs, it can be assigned as $cs = 14$. [*In C value assigned always from Right to left]

OPERATORS		EXAMPLE	EXPLANATION
SIMPLE ASSIGNMENT OPERATOR	=	cs = 10	10 is assigned to variable cs.
COMPOUND ASSIGNMENT OPERATORS	+=	cs += 10	same as cs = cs + 10.
	-=	cs -= 10	same as cs = cs - 10.
	*=	cs *= 10	same as cs = cs * 10.
	/=	cs /= 10	same as cs = cs / 10.
	%=	cs %= 10	same as cs = cs % 10
	&=	cs &= 10	same as cs = cs & 10
	^=	cs ^= 10	same as cs = cs ^ 10

Special operators:

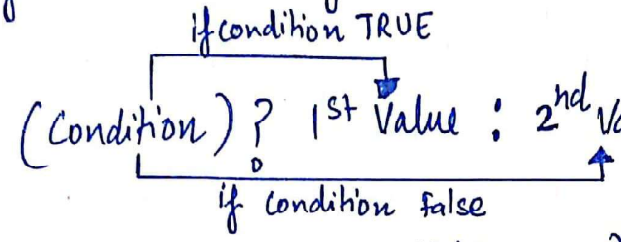
① & This is used to get the address of variable. (&a)

② * This is used as pointer to a variable.

③ sizeof() This gives the size of the variable.
eg: sizeof(int) will give us 2 Bytes.

Ternary or conditional Operators: which applied on 3 operands and return one value if condition is true and returns another value if condition is false.

Syntax:



for ex:

① (50 > 20) ? printf("Hello") : printf("Hi");

② (A > 100 ? 0 : 1)

Here if A is greater 100, 0 is returned (as 0 is 1st value) else 1 is returned.

* This operator is equal to if else conditional statement.

Type Casting in C: Type casting is a way to convert a variable from one data type to another data type. For example, if you want to store a 'long' value into a simple integer then you can type cast 'long' to 'int'.

Syntax: (datatype name) variable.

- Type casting is also known as explicit conversion.
- Type conversion (Implicit conversion) can be performed by the compiler automatically.

Eg: of typecasting:

```
int sum = 17, count = 5;
float mean;
mean = (float) sum / count;
```

Here mean = 3.400000 Ans.

*Here we done typecasting for sum variable. Hence 17 becomes 17.000../5 = 3.4000..
*If we not used typecasting here so this treated as 17/5 = 3
[C Rule is int/int = int]