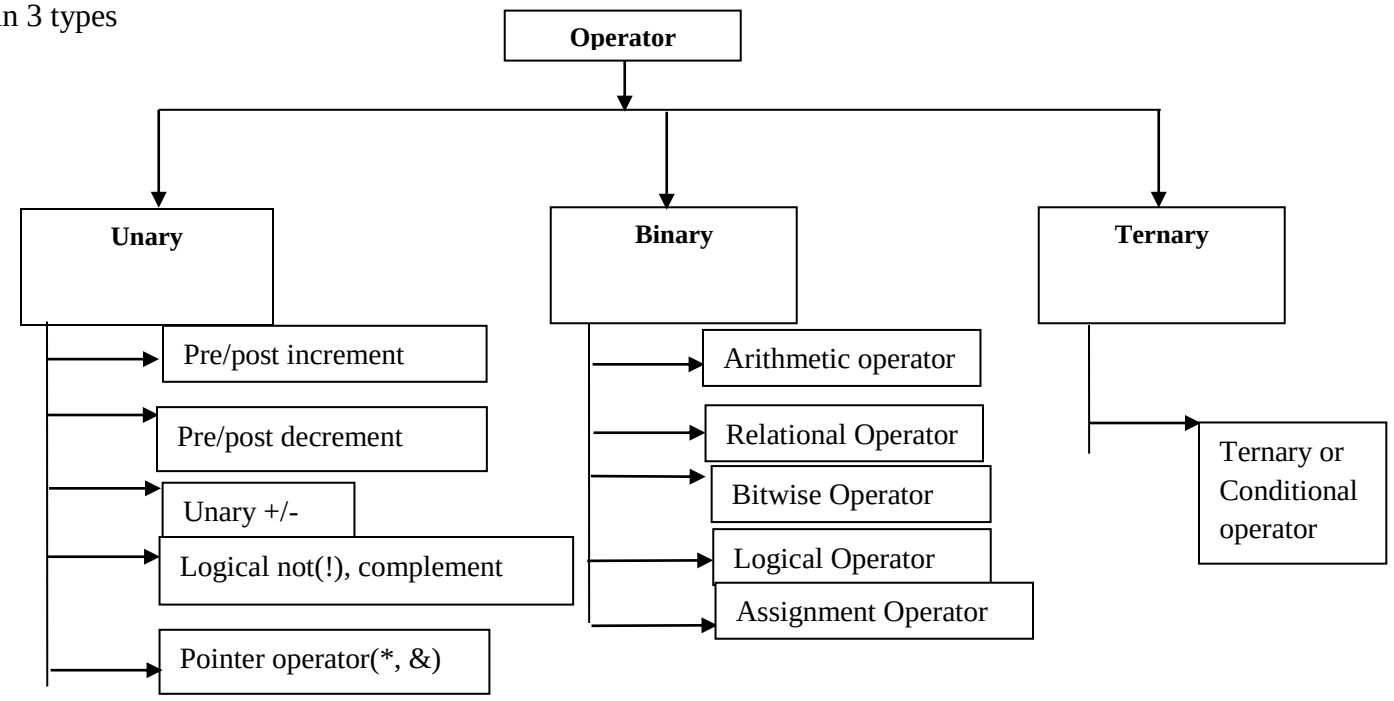
Expressions : An expression is a combination of variables, constants, operators. It can be arithmetic, logical and relational for example:-

- int z= x+y // arithmetic expression
- b //relational expression
- a= =b // logical expression

Operator:

This is a symbol use to perform some operation on variables, operands or with the constant. Some operator required 2 operand to perform operation, Some required single operand to perform operation and some require 3 operand to perform operation.

Depending upon the numbers of operands required to perform operations, operands are divided in 3 types



Unary Operator: unary operator takes only one operand to perform its operation. Example are pre increment(++i), post increment(i++), pre decrement(--i),post decrement(i++), unary +/-, logical NOT operator, pointer operator.

Pre Increment/ Decrement:- In the prefix increment/decrement the value of the variable is incremented/decremented 1st , then the new value is used.

EXAMPLE

```
void main() {
 int y=12, z;
z= ++y;
printf("y =%d, z=%d",y,z);
}
```

Output:- y=13,z=13

In this example value of y is incremented 1st then the incremented value is assigned to z. That's why value of both variables are 13.

Post Increment/Decrement: - In the postfix increment and decrement operator is used in the operation . And then increment and decrement is perform.

EXAMPLE

```
void main() {
 int y=12, z;
z= y++;
printf("y =%d, z=%d",y,z);
}
```

Output:- y=13,z=12

In this example value of y is assigned to z 1st then the value of y is incremented.That's why value y=13 and z=12.

Distinguish between i++ and ++i with suitable example.

i++	**++i**
1. i++ is called post increment operator	1. ++i is called pre increment operator
2. void main(){ int i=5,y;	2. void main(){ int i=5,y;

y=++i;
prntf("y=%d",y)
}
Output: - y=6
	y=i++;
	prntf("y=%d",y)
	}
	Output: - y=5

3. i++ first assign the value to the vaiable n left and then increments the operand.	3. ++i first adds 1 to the operand and then the result is assigned to the variable on the left.

Binary Operator

Arithmetic Operator:

This operator used for numeric calculation. Arithmetic operators are addition(+), subtraction(-), multiplication(*), division(/) and modulus(%). But modulus cannot applied with floating point operand as well as there are no exponent operator in c.

When both the operands are integer then it is called integer arithmetic and the result is always integer. When both the operands are floating point then it is called floating arithmetic and when operand is of integer and floating point then it is called mix type or mixed mode arithmetic. And the result is in float type.

```
void main()
{
int a=8,b=3,c;
c=a+b;
printf("Result of addition=%d",c);
c=a-b;
printf("Result of subtraction=%d",c);
c=a*b;
printf("Result of multiplication=%d",c);
c=a/b; // fractional part of the result is removed as both of the operands are integer
```

```
printf("Result of division=%d",c);
c=a%b; // returns the remainder of the division
printf("Remainder is=%d",c);
}
```

Relational Operator

It is use to compared value of two expressions depending on their relation. Expression that contain relational operator is called relational expression. Here the value is assigning according to true or false value. The followings are relational operator.

Operator Symbol	Operator Name
<	less than
<=	less than equal to
>	greater than
>=	greater than equal to
==	equal to equal
!=	not equal to

```
void main(){
int a= 6, b = 4, c;
c = a >= b;
printf("The comparison returns %d ", c);
c = a < =b;
printf("The comparison returns %d ", c);
c = a == b;
printf("The comparison returns %d ", c);
c = a != b;
```

```
printf("The comparison returns %d ", c);
}
```

Output: The comparison returns 1

The comparison returns 0

The comparison returns 0

The comparison returns 0

Logical or Boolean Operator

Operator used with one or more operand and return either value zero (for false) or one (for true). The expression that combines two or more expressions is termed as logical expression. C has three logical operators

Operator Symbol	Operator Name
&&	Logical AND
||	Logical OR
!	Logical NOT

Among which Logical NOT operator is an unary operator. Remaining are binary operator.. Logical AND gives result true if both the expression are true, otherwise result is false. And logical OR gives result false if both the expression false, otherwise result is true.

Expression 1	Expression 2	&&	||
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Bit-wise Operator:-

Bit-wise operator permit programmer to access and manipulate of data at bit level. Various bit-wise operator enlisted are

Operator Symbol	Operator Name
one's complement	(~)
bitwise AND	(&)
bitwise OR	(|)
bitwise XOR	(^)
left shift	(<<)
right shift	(>>)

These operators can operate on integer and character value but not on float and double.

Example

```
void main() {
int = 50, b=5, r;
r = a & b;
printf("Bitwise AND  operator- result is %d",r);
r = a | b;
printf("Bitwise OR  operator- result is %d",r);
r = a ^ b;
printf("Bitwise XOR  operator- result is %d",r);
r = a << 2;
printf("Bitwise left shift operator- result is %d",r);
r = a >> 2;
printf("Bitwise right shift  operator- result is %d",r);
```

}

Output:

Bitwise AND operator- result is 0

Bitwise AND operator- result is 55

Bitwise XOR operator- result is 55

Bitwise left shift operator- result is 200

Bitwise right shift operator- result is 12

Assignment Operator:-

A value can be stored in a variable with the use of assignment operator. The assignment operator(=) is used in assignment statement and assignment expression. Operand on the left hand side should be variable and the operand on the right hand side should be variable or constant or any expression.

Example

- int x= y; //right hand side is variable
- int x =10; //right hand side is constant
- int p = q+r; //right hand side is expression

When variable on the left hand side is occur on the right hand side then we can avoid by writing the compound statement.

*=, /=, %= ,&=, ^=, <<=,>>=

A=A*B

=>A*=B

Precedence of operators: - If more than one operators are involved in an expression then, C language has predefined rule of priority of operators. This rule of priority of operators is called operator precedence. The operator at the higher level of precedence are evaluated first.

Associativity of operator: - Associativity is only used when there are two or more operators of same precedence. Associativity doesn't define the order in which operands of a single operator are evaluated. Associativity is either left to right (L->R) or right to left (R->L).

Operators	Description	Precedence level	Associativity
() [] --> .	function call array subscript arrow operator dot operator	1	Left to right
+ - ++ - - ! ~ * & sizeof	unary plus unary minus increment decrement logical not 1's complement indirection address size in byte	2	Right to left
% * /	modulus multiplication division	3	Left to right
Addition Subtraction	+	4	Left to right

	-		
<< >>	left shift right shift	5	left to right
<= >= < >	less than equal to greater than equal to less than greater than	6	left to right
== !=	equal to not equal to	7	left to right
&	bitwise AND	8	left to right
|	bitwise XOR	9	left to right
|	bitwise OR	10	left to right
&& ||	logical AND logical OR	11 12	left to right
?:	conditional operator	13	Right to left
=, *=, /=, %= &=, ^=, <<=,>>=	assignment operator	14	right to left
,	comma operator	15	

Ternary Operator

It sometimes called as conditional operator. Since it required three expressions as

operand and it is represented as (? , :).

SYNTAX

exp1 ? exp2 :exp3

Here exp1 is first evaluated. It is true then value return will be exp2 . If false then exp3.

EXAMPLE

```
void main()
{
 int a=10, b=2
 int s= (a>b) ? a:b;
 printf("value is:%d");
 }
```

Output:

Value is:10