27-04-2024

INCREMENT & DECREMENT :- ++, -- [UNARY]

There are two notations called

1. PREFIX : b = ++a

   a = a+1, b = a

2. POSTFIX : b = a++

   b = a, a = a+1

Here a will observe change to $\boxed{+1}$ a++/++a

only b will not change in both cases

b = 6, b = 5

Ex of Prefix :-

```
-> main()
{
  int a=8, b;
  b = ++a;
  printf("a=%d, b=%d", a, b);
}
```

O/P :-

a = 9

b = 9

Ex of Postfix :-

```
-> main()
{
  int a=8, b;
  b = a++;
  printf("%d %d", a, b);
}
```

O/P :- a = 9

b = 8

int a = 5

O/P :- a = 6

b = 5

```
-> main()
{ int a=1, b=2, c=3, d;
  d = ++a + b++ + ++c;
  printf("%d %d %d %d", a, b, c, d);
}
```

a = %d, b = %d, c = %d, d = %d

O/P :- a = 2

b = 3

c = 4

d = 8 = 2+2+4

-> main()

```
{
  int a=1, b=2, c=3, d;
  d = ++a + b++ + c++;
  printf("%d %d %d %d", a, b, c, d);
}
```

O/P:- a = 2
b = 3
c = 4
d = 7 = 2 + 2 + 3

-> main()

```
{
  int a=1, b=2, c=3, d;
  d = a++ + b++ + c++;
  printf("%d %d %d %d", a, b, c, d);
}
```

O/P:- a = 2
b = 3
c = 4
d = 6

-> main()

```
{
  int a=1, d;
  d = a++ + a++ + a++;
  printf("%d %d", a, d);
}
```

~~1+2+3+4~~
~~2+3+4~~

O/P:- a = 4
d = 6

-> main()

```
{
  int a=1, b;
  b = ++a + ++a + a++;
  printf("%d", b);
}
```

2+3+3

O/P:- a = 4
b = 4

-> main()

```
{
  int a=1;
  a++;
  printf("%d", a);
}
```

O/P:- a = 2

-> main ()

```
{ int a = 1, b;
  b = a++;
  printf ("%d", b);
}
```

o/p :- b = 1

* We shouldn't shouldn't apply increment & decrement operators on constants

DECREMENT OPERATORS =>

-> main ()

```
{ int a = 5, b;
  b = a--;
  printf ("a=%d, b=%d", a, b);
}
```

o/p :-

a = 4

b = 5

-> main ()

```
{ int a=5, b;
  b = --a;
  printf ("a=%d, b=%d", a, b);
}
```

o/p :-

a = 4

b = 4

-> main ()

```
{ int a=5, b;
  b = ++a + a++ + --a + a--;
  printf ("%d", b);
}
```

o/p :-

b = 25

a = 5

--a ++a

5 7

19

b = 6 + 7 + 6 + 6

= 25

29-04-2024

RELATIONAL :- <, >, <=, >=, ==, !=

BINARY OPERATORS

SYNTAX :- op1 Rop op2

These operators check the relation between 2 operands. If it is true gives a true value means '1', otherwise gives a false value means '0'.

EX :-

```
main()
{ int a;
  a = 5>=2;
  printf("%d", a);
}
```

O/P :- 1

```
main()
{ int a;
  a = 5==2;
  printf("%d", a);
}
```

O/P :- 0

LOGICAL :-

Logical And && (binary operator)

A	B	A && B
0	0	0
0	1	0
1	0	0
1	1	1

EX :-

```
main()
{ int a;
  a = 5<6 && 6>5;
  printf("%d", a);
}
```

O/P :- 1 True

a = 5<6 && 6<5

O/P :- 0 False

Logical OR || (binary operator)

A	B	A || B
0	0	0
0	1	1
1	0	1
1	1	1

Logical Not ! (unary)

A	!A
0	1
1	0

Ex:-
```
main ()
{ int a;
  a=5<6||6<5;
  printf ("%d", a);
}
```
o/p:- 1

Ex:-
```
main ()
{ int a;
  a=5&&5;
  printf ("%d", a);
}
```
o/p:- 1

Ex:-
```
main ()
{ int a;
  a=6!=5;
  printf ("%d", a);
}
```
o/p:- 1

Ex:-
```
main ()
{ int a;
  a=!5;
  printf ("%d", a);
}
```
o/p:- 0

BIT WISE OPERATOR =>

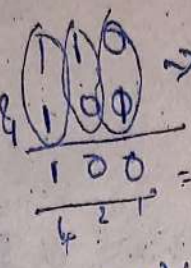
- Bitwise And & (binary)
- bitwise or | (binary)
- bitwise xor ^ (can be binary)
- 1's complement ~ (tilda) (unary)
- left shift << (binary)
- right shift >> (binary)

we should not apply Bitwise operators on float

Ex:-

```
main()
{ int a, b, c;
  a = 5 & 6;
  b = 5 | 6;
  c = 5 ^ 6;
  printf("%d %d %d", a, b, c);
}
```

O/P:-



⇒ INDIVIDUAL OPERATION

5×2^1

Ex:-

```
main()
{ int a, b, c, d, e;
  a = 8 & 12;
  b = 8 | 12;
  c = 8 & 12;
  d = 8 | 12;
  e = 8 ^ 12;
  printf("%d %d %d %d %d", a, b, c, d, e);
}
```

O/P:-

```
8  = 1 0 0 0
12 = 1 1 0 0
-----------
0
```

& = 1000 → 8

| = 1100 → 12

^ = 0100 → 4

O/P:- 1 1 8 12 4

TRUTH TABLE OF THE BITWISE OPERATORS =>

X	Y	X & Y (AND)	X | Y (OR)	X ^ Y (XOR)
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

LEFT SHIFT OPERATOR :-

$$a << n = a * 2^n$$

-> main ()

{ int a=5, b;

b = a << 3;

printf ("%d", b);

}

O/P :- $5 * 2^3 = 40$ [LeftShift]

b = a >> 3

$5/2^3 = 0$ [RightShift]

RIGHT SHIFT OPERATOR :-

$$a >> n = \frac{a}{2^n}$$

-> main ()

{ int a=50, b;

b = a >> 3;

printf ("%d", b);

}

O/P :- $\frac{50}{2^3} = 6$ [RightShift]

Left shift :-

b = a << 3

$50 * 2^3 = 400$

$19 \times 2^1 = 38$

$\frac{19}{2^1} = 9$

LEFT SHIFT & RIGHT SHIFT OPERATORS =>

-> main ()

{

int x;

printf ("enter the value of x : ");

scanf ("%d", &x);

printf ("Left shift operation : \n x << 1 = %d \n", x << 1);

printf ("Right shift operation : \n x >> 1 = %d \n", x >> 1);

}

O/P :- enter the value of x : 19

Left shift operation :

x << 1 = 38

Right shift operation :

x >> 1 = 9

$19 * 2^1 = 38$

$19/2^1 = 9$

Q:- To demonstrate the use of bitwise operations :-

```
-> main ()
{
  int a, b;
  printf ("Enter the Value of a: ");
  scanf ("%d", &a);
  printf ("Enter the Value of b: ");
  scanf ("%d", &b);
  printf ("the Bitwise operations are as below: \n");
  printf ("a&b = %d \n", a&b);
  printf ("a|b = %d \n", a|b);
  printf ("a^b = %d \n", a^b);
  printf ("a<<1 = %d \n", a<<1);
  printf ("a>>1 = %d \n", a>>1);
  printf ("b<<1 = %d \n", b<<1);
  printf ("b>>1 = %d \n", b>>1);
}
```

O/P:-

Enter the Value of a: 5
Enter the Value of b: 9
The Bitwise operations are as below:
a&b = 1
a|b = 13
a^b = 12
a<<1 = 10
a>>1 = 2
b<<1 = 18
b>>1 = 4

$a << b = a \times 2^b$
$a >> b = \frac{a}{2^b}$
$b << a = b \times 2^a$
$b >> a = \frac{b}{2^a}$

a: 5
b: 9

a&b = 1
a|b = 13
a^b = 12
a<<b = 2560 => 5×2^9
a>>b = 0 => $5/2^9 = 0$
b<<a = 288 => 9×2^5
b>>a = 0 => $\frac{9}{2^5} = 0$

BITWISE OPERATIONS:-

```
main ()
{ int a, b;
  printf ("Enter the value of a: ");
  scanf ("%d", &a);
  printf ("Enter the value of b: ");
  scanf ("%d", &b);
  printf ("The Bitwise operations are as below: \n");
  printf ("a&b = %d\n", a&b); // Bitwise AND operation
  printf ("a|b = %d\n", a|b); // Bitwise OR operation
  printf ("a||b = %d\n", a||b); // Logical OR operation
  printf ("a^b = %d\n", a^b); // Bitwise XOR operation
  printf ("a<<b = %d\n", a<<b); // Bitwise left-Shift operation
                                   a<<b = a*(2^b)
  printf ("a<<1 = %d\n", a<<1);
  printf ("a>>b = %d\n", a>>b); // Bitwise right-Shift operation
                                   a>>b = a/(2^b)
  printf ("a>>1 = %d\n", a>>1);
  printf ("b<<a = %d\n", b<<a); // Bitwise left-Shift operation
                                   b<<a = b*(2^a)
  printf ("b<<1 = %d\n", b<<1);
  printf ("b>>a = %d\n", b>>a); // Bitwise Right-Shift operation
                                   b>>a = b/(2^a)
  printf ("b>>1 = %d\n", b>>1);
}
```

O/P:-

←