30-04-2024

BITWISE AND & [BINARY]:-

```
int main()
{
  int a, b;
  printf("Enter the Value of a: ");
  scanf("%d", &a);
  printf("Enter the value of b: ");
  scanf("%d", &b);
  printf("a & b = %d \n", a & b);
}
```

Ex:-

```
main()
{
  int x=1, y=0, z=5, a;
  int a = x < y && ++z;
  printf("%d", z);
}
```

O/P :- 5

```
main()
{
  int x=1, y=0, z=5, a;
  int a = x < y || ++z;
  printf("%d", z);
}
```

O/P :- 6

```
main()
{ int x=1, y=0, z=5, a;
  int a = x>y || ++z;
  printf("%d", z);
}
```

o/p :- 5

⇒
```
main()
{ int x=5, y=6, a, b, c, d, e, f, g, h;
  a = 5 && 6;
  b = 5 || 6;
  c = !5;
  d = 5 & 6;
  e = 5 | 6;
  f = 5 << 3;
  g = 10 >> 3;
  h = 5 ^ 6;
  printf("%d %d %d %d %d %d %d %d", a, b, c, d,
    e, f, g, h);
}
```

o/p :-

a = 5 → 1 0 1
    6 → 1 1 0
    ----
    1 0 0 → 4

a = 1

b = 1

c = 0

d = 5 & 6
    1 0 1
    1 1 0
    -----
    1 0 0 →
    4

e = 5 | 6
    5 = 1 0 1
    6 = 1 1 0
    ----
    1 1 1 → 7

e = 7

f = $5 * 2^3$
  = 40

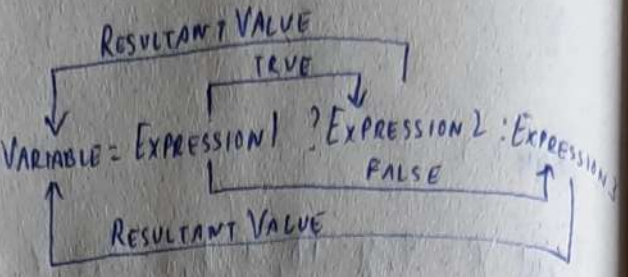
g = $\frac{10}{2^3}$
  = 1

h = 1 0 1
    1 1 0
    -----
    0 1 1 →

CONDITIONAL OPERATOR :- ? : (trinary)

Eg CONDITION ? STATEMENT 1 : STATEMENT 2 ;

If condition is true statement 1 is executed otherwise statement 2 is executed



EX :-

```
main ( )
{
  int n;
  printf ("enter n");
  scanf ("%d", &n);
  n %2 == 0 ? printf ("even") : printf ("odd");
}
```

15 > 5
5 > 8

EX :-

```
main ( )
{
  int n;
  printf ("enter n");
  scanf ("%d", &n);
  n %4 == 0 ? printf ("Leap year") : printf ("Not a leap year");
}
```

```
(n %4 == 0 && n %100 != 0) || n %400 == 0 ? printf ("%d", n)
  : printf (" ");
```

02-05-2024

ASSIGNMENT OPERATOR :- =, +=, -=, *=, /=, %=, &=, |=, ^=, <<=, >>=

-> main()

{ int a=5, b=6, c=7, d=8, e=9, f=10, g=2, h=5, i=20, x=8;

a+=2; // 7

b-=4; // 2

c*=4; // 28

d/=5; // 1

x%=6; // 2

e&=5; // 1

f|=6; // 14

g^=7; // 5

h<<=2; // 20

i>>=3; // 2

printf("%d %d %d %d %d %d %d %d %d %d", a, b, c, d, e, f, g, h, i, x);

}

5 2 8
1 1 0
6 8 0

~~x%=6~~

x&=5; 130
550

a = 5
b = a + 7

a<<n = a * 2^n

a>>n = a/2^n

Ex :-

main()

{ int a, b=110, c=20;

a = b -= c *= 5;

printf("%d", a);

}

Right to Left ←

a = 20 * 5 = 100

110 - 100 = 10

∴ a = 10

→ main()

```
{ int a, b=110, c=20;
  a = b + c + (b *= 5);
  printf("%d", a);
}
```

O/P :- 680;

*&a

↓ ↓

ADDRESS

ADDRESS VALUE

SPECIAL OPERATOR =>

→ ADDRESS : &

SYNTAX : & Variable (Unary)

It gives address of variable.

EX :-

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

O/P :- gives some address

→ VALUE AT ADDRESS : *&

SYNTAX : * & Variable

It gives value assigned to that address.

EX :-

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

O/P :- 5

EXAMPLES OF ASSIGNMENT OPERATORS =>

1. Simple Assignment operator :-

```
#include
int main ()
{ int num;
   num = 10;
   printf ("The value of num is : %d \n", num);
} return 0;
```

2. Addition Assignment operator :-

```
int main ()
{ int num;
   printf ("Enter a number: ");
   scanf ("%d", &num);
   num += 5;
   printf ("%d \n", num);
}
```

3. Subtraction Assignment operator :-

```
int main ()
{ int num;
   printf ("enter a number: ");
   scanf ("%d", &num);
   num -= 3;
   printf ("%d \n", num);
}
```

4- Multiplication Assignment Operator :-

```
int main ()
{ int num;
  printf ("Enter number: ");
  scanf ("%d", &num);
  num *= 4;
  printf ("%d\n", num);
}
```

5. Division assignment operator :-

```
int main ()
{ int num;
  printf ("Enter number: ");
  scanf ("%d", &num);
  num /= 4;
  printf ("%d\n", num);
}
```

6. Modulus assignment operator :-

```
int main ()
{ int num;
  printf ("Enter number: ");
  scanf ("%d", &num);
  num %= 5;
  printf ("%d\n", num);
}
```

7. main( )

```
{ int a=5, b=10, c;
  c = a+b;
  printf ("c=a+b= %d\n", c);
  c+=a;
  printf ("c+=a= %d\n", c);
  c-=a;
  printf ("c-=a= %d\n", c);
  c*=a;
  printf ("c*=a= %d\n", c);
  a=10;
  c=15;
  c/=a;
  printf ("c/=a= %d\n", c);
  c%=a;
  printf ("c%=a= %d\n", c);
  c&=a;
  printf ("c&=a= %d\n", c);
  c|=a;
  printf ("c|=a= %d\n", c);
  c<<=a;
  printf ("c<<=a= %d\n", c);
  c>>=a;
  printf ("c>>=a= %d\n", c);
  c<<=2;
  printf ("c<<=2= %d\n", c);
  c>>=2;
  printf ("c>>=2= %d\n", c);
  c^=a;
  printf ("c^=a= %d\n", c);
}
```

O/P:-

```
c=a+b = 15
c+=a = 20
c-=a = 15
c*=a = 75
c/=a = 1
c=a = 1
c&=a = 0
c|=a = 10
c<<=a = 10240
c>>=a = 10
c<<=2 = 40
c>>=2 = 10
c^=a = 0
```

EXAMPLES FOR SPECIAL OPERATOR =>

-> ADDRESS OPERATOR :- &

* int main ()
{
int x = 100;
printf ("The address of x is %d", &x);
}

o/p :- The address of x is 6487580

* int main ()
{
int x = 100;
printf ("The address of x is %p", &x);
}

P => POINTER

o/p :- The address of x is 000000000062FE1C

* int main ()
{
int x = 100;
printf ("The address of x is %xh", &x);
}

ch => Character

o/p :- the address of x is h

* int main ()
{ int x = 100;
printf ("The address of x is %o", &x);
}

O -> OCTAL

o/p :- The address of x is 30577034

* int main ()
{ int x = 100;
printf ("The address of x is %x", &x);
}

X => HEXADECIMAL

o/p :- The address of x is 62fe1c

→ VALUE AT ADDRESS :- *&

* int main ()
{ int x = 100;
printf ("The value assigned to that address is %d", *&x);
}
o/p :- The value assigned to that address is 100

* int main ()
{ int x = 100;
printf ("The value assigned to that address is %p", *&x);
}
o/p :- the value assigned to that address is 0000000000000064

* int main ()
{ int x = 100;
printf ("The value assigned to that address is %ch", *&x);
}
o/p :- the value assigned to that address is dh

* int main ()
{ int x = 100;
printf ("The value assigned to that address is %o", *&x);
}
o/p :- the value assigned to that address is 144

* int main ()
{ int x = 100;
printf ("The value assigned to that address is %x", *&x);
}
o/p :- The value assigned to that address is '64'