Bitwise operator:-

In C, the following 6 operators are bitwise operators (work at bit-level):-

**Bitwise AND operator &**

The output of bitwise AND is 1 if the corresponding bits of two operands is 1. If either bit of an operand is 0, the result of corresponding bit is evaluated to 0.

Let us suppose the bitwise AND operation of two integers 12 and 25.

12=00001100 (in binary)

25=00011001 (in binary)

Bit opetation of 12 and 25

00001100

& 00011001

= 00001000=8(in decimal)

### Example #1: Bitwise AND

#include <stdio.h>

int main()

{

int a = 12, b = 25;

printf("Output = %d", a&b);

return 0;

**Bitwise OR operator |**

### The output of bitwise OR is 1 if at least one corresponding bit of two operands is 1. In C Programming, bitwise OR operator is denoted by |.

### 12 = 00001100 (In Binary)

### 25 = 00011001 (In Binary)

### Bitwise OR Operation of 12 and 25

### 00001100

### | 00011001

### \_\_\_\_\_\_\_\_

### 00011101 = 29 (In decimal)

### **Example #2: Bitwise OR**

### #include <stdio.h>

### int main()

### {

### int a = 12, b = 25;

### printf("Output = %d", a|b);

### return 0;

### }

### **Output**

### Output = 29

### **Bitwise XOR (exclusive OR) operator ^**

### The result of bitwise XOR operator is 1 if the corresponding bits of two operands are opposite. It is denoted by ^.

### 12 = 00001100 (In Binary)

### 25 = 00011001 (In Binary)

### Bitwise XOR Operation of 12 and 25

### 00001100

### ^ 00011001

### \_\_\_\_\_\_\_\_

### 00010101 = 21 (In decimal)

### **Example #3: Bitwise XOR**

### #include <stdio.h>

### int main()

### {

### int a = 12, b = 25;

### printf("Output = %d", a^b);

### return 0;

### }

### **Output**

### Output = 21

### **Bitwise complement operator ~**

### Bitwise compliment operator is an unary operator (works on only one operand). It changes 1 to 0 and 0 to 1. It is denoted by ~.

### 35 = 00100011 (In Binary)

### Bitwise complement Operation of 35

### ~ 00100011

### \_\_\_\_\_\_\_\_

### 11011100 = 220 (In decimal)

### **Shift Operators in C programming**

### There are two shift operators in C programming:

### Right shift operator

### Left shift operator.

### **Right Shift Operator**

### Right shift operator shifts all bits towards right by certain number of specified bits. It is denoted by >>.

### 212 = 11010100 (In binary)

### 212>>2 = 00110101 (In binary) [Right shift by two bits]

### 212>>7 = 00000001 (In binary)

### 212>>8 = 00000000

### 212>>0 = 11010100 (No Shift)

### **Left Shift Operator**

### Left shift operator shifts all bits towards left by certain number of specified bits. It is denoted by <<.

### 212 = 11010100 (In binary)

### 212<<1 = 110101000 (In binary) [Left shift by one bit]

### 212<<0 =11010100 (Shift by 0)

### 212<<4 = 110101000000 (In binary) =3392(In decimal)

### **Example #5: Shift Operators**

### #include <stdio.h>

### int main()

### {

### int num=212, i;

### for (i=0; i<=2; ++i)

### printf("Right shift by %d: %d\n", i, num>>i);

### printf("\n");

### for (i=0; i<=2; ++i)

### printf("Left shift by %d: %d\n", i, num<<i);

### 

### return 0;

### }

### Right Shift by 0: 212

### Right Shift by 1: 106

### Right Shift by 2: 53

### Left Shift by 0: 212

### Left Shift by 1: 424

### Left Shift by 2: 848

Ternary operations: -

The ternary operator take three arguments:

1. The first is a comparison argument
2. The second is the result upon a true comparison
3. The third is the result upon a false comparison

It helps to think of the ternary operator as a shorthand way or writing an if-else statement. For example: -

int a = 10, b = 20, c;

if (a < b) {

c = a;

}

else {

c = b;

}

printf("%d", c)

The above code takes more than 5 lines to complete. The same result can be obtained by using a ternary operator but with less lines. The syntax of a ternary operation is given by: -

**condition ? value\_if\_true : value\_if\_false**

The statement evaluates to value\_if\_true if condition is met, and value\_if\_false otherwise.

Here’s the above example rewritten to use the ternary operator:

int a = 10, b = 20, c;

c = (a < b) ? a : b;

printf("%d", c);

Output of the example above should be:

10