BITWISE OPERATORS IN C LANGUAGE

In the arithmetic-logic unit (which is within the CPU), mathematical operations like: addition, subtraction, multiplication and division are done in bit-level. To perform bit level operations in C programming, bitwise operators are used.

Operators	Meaning of operators
&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
~	Bitwise complement
	Shift left
>>	Shift right

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. In C Programming, the bitwise AND operator is denoted by &. Let us suppose the bitwise AND operation of two integers 12 and 25.

```
12 = 00001100 (In Binary)
25 = 00011001 (In Binary)

Bit Operation of 12 and 25
  00001100
& 00011001

  --------
  00001000  = 8 (In decimal)
```

Example:

```c
#include <stdio.h>

int main() {

    int a = 12, b = 25;
    printf("Output = %d", a & b);

    return 0;
}
```

Output:

```
Output = 8
```

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:

```c
#include <stdio.h>

int main() {

    int a = 12, b = 25;
    printf("Output = %d", a | b);

    return 0;
}
```

Output:

```
Output = 29
```

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:

```c
#include <stdio.h>

int main() {

    int a = 12, b = 25;
    printf("Output = %d", a ^ b);

    return 0;
}
```

Output:

```
Output = 21
```

Bitwise complement operator is a 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)
```

Example:

```c
#include <stdio.h>

int main() {

    printf("Output = %d\n", ~35);
    printf("Output = %d\n", ~-12);

    return 0;
}
```

Output:

```
Output = -36
Output = 11
```

RIGHT AND SHIFT OPERATORS

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 shifts all bits towards left by a certain number of specified bits. The bit positions that have been vacated by the left shift operator are filled with 0. The symbol of the left shift operator is <<.

```
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:

```c
#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;
}
```

Output:

```
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
```