1. **BITWISE OPERATORS**:-
   * 1. **Bitwise AND operator**-

It is represented by a single ampersand sign (&). The result of the bitwise AND operation is 1 if both the bits have the value as 1; otherwise, the result is always 0.

Let us consider that we have 2 variables op1 and op2 with values as follows:

Op1 = 0000 1101

Op2 = 0001 1001

The result of the AND operation on variables op1 and op2 will be

Result = 0000 1001

**Example**-

#include <stdio.h>

int main()

{

int a = 12, b = 25;

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

return 0;

}

**Output**- 8

* + 1. **Bitwise OR operator**-

It is represented by a single vertical bar sign (|).The result of the bitwise OR operation is 1 if at least one of the expression has the value as 1; otherwise, the result is always 0.

Let us consider that we have 2 variables op1 and op2 with values as follows:

Op1 = 0000 1101

Op2 = 0001 1001

The result of the OR operation on variables op1 and op2 will be

Result = 0001 1101

**Example**-

#include <stdio.h>

int main()

{

int a = 12, b = 25;

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

return 0;

}

**Output**- 29

* + 1. **Bitwise XOR operator**-

It is represented by a symbol (^).The result of the bitwise Exclusive-OR operation is 1 if only one of the expression has the value as 1; otherwise, the result is always 0.

Let us consider that we have 2 variables op1 and op2 with values as follows:

Op1 = 0000 1101

Op2 = 0001 1001

The result of the OR operation on variables op1 and op2 will be

Result = 0001 0100

**Example**-

#include <stdio.h>

int main()

{

int a = 12, b = 25;

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

return 0;

}

**Output**- 21

* + 1. **Bitwise Complement operator**-

The bitwise complement is also called as one's complement operator since it always takes only one value or an operand. It is a unary operator. When we perform complement on any bits, all the 1's become 0's and vice versa. If we have an integer expression that contains 0000 1111 then after performing bitwise complement operation the value will become 1111 0000. Bitwise complement operator is denoted by symbol tilde (~).

**Example**-

#include <stdio.h>

int main()

{

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

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

return 0;

}

**Output**- -36

11

* + 1. **Bitwise SHIFT operators**-

1. **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)

1. **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**-

#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

1. **TERNARY OPERATORS**-

The conditional operator is kind of similar to the if-else statement as it does follow the same algorithm as of if-else statement but the conditional operator takes less space and helps to write the if-else statements in the shortest way possible.

**Syntax**:

The conditional operator is of the form

variable = Expression1 ? Expression2 : Expression3

**Example**-

#include <stdio.h>

int main()

{

// variable declaration

int n1 = 5, n2 = 10, max;

// Largest among n1 and n2

max = (n1 > n2) ? n1 : n2;

// Print the largest number

printf("Largest number between"

" %d and %d is %d. ",

n1, n2, max);

return 0;

}

**Output**-

Largest number between 5 and 10 is 10.