1. Bitwise operators in c

# These operators are used to perform bit operations. Decimal values are converted into binary values which are the sequence of bits and bitwise operator wok on these bits

# Bitwise operators in c language are & (bitwise AND), / (bitwise OR), ~ (bitwise NOT), ^ (bitwise XOR),<<(left shift)and>>(right shift)

**Bitwise operators and their name in c language**

## &-bitwise AND

## /- bitwise OR

## ~-bitwise NOT

## ^- bitwise XOR

## <<- left shift

## >>- right shift

Example: 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 1.

12=00001100 (in binary)

25=00011001 (in binary)

Bitwise OR operation of 12 and 25

00001100

00011001

00011101 = 29 (in decimal)

Programme:

#Include<stdio.h>

Int main ()

{

Int a= 12, b= 25 ;

Print f (“output = %d” , a/b);

Return 0;

}

## Output

Output = 29

1. Ternary operators in c

# 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 . here’s a simple decision -making example using if and else:

Int a= 10, b= 20, c ;

If(a<b) {

C= a ;

}

Else {

C= b ;

}

Print f (“%d” , c);

## This example takes more than 10 lines, but that isn’t necessary . you can write the above program in just 3 lines of code using a ternary operator.

**Syntax**

**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;

Print f (“%d”,c);

Output of the example above should be:

10

C is set equal to a , because the condition a <b was true .

Remember that the arguments value-if-true and value-if-false must be of the same type , and they must be simple expression rather then full statements

1. Arithmetic operator

#include <stdio.h>

Main () {

Int a= 21 ;

Int b= 10 ;

Int c ;

C= a+ b ;

Print f ( “ sum of 21 and 10 is %d/n” , c) ;

C = a-b ;

Print f (“ difference of 21 and 10 is %d/n “ , c);

C= a\*b ;

Print f (“ multiple of 21 and 10 is %d/n “ ,c );

C= a%b ;

Print f (“ division of 21 and 10 is %d/n” , c);

}

Return 0 ;

Output:

Sum of 21 and 10 is 31

Difference of 21 and 10 is 11

Multiple of 21 and 10 is 210

Division of 21 and 10 is 2

# These operators are used to perform bit operations’