**BITWISE OPERATOR**

An **operator** is a symbol that operates on a value to perform specific mathematical or logical computations. They form the foundation of any programming language. In C++, we have built-in operators to provide the required functionality.

An operator operates the **operands**. For example,

int c = a + b;

Here, ‘+’ is the addition operator. ‘a’ and ‘b’ are the operands that are being ‘added’.

**Operators in C++ can be classified into 6 types:**

1. Arithmetic Operators
2. Relational Operators
3. Logical Operators
4. **Bitwise Operators**
5. Assignment Operators
6. Ternary or Conditional Operators

**Bitwise Operators**

These operators are used to perform bit-level operations on the operands. The operators are first converted to bit-level and then the calculation is performed on the operands. The mathematical operations such as addition, subtraction, multiplication, etc. can be performed at the bit-level for faster processing.

| **Name** | **Symbol** | **Description** | **Example** |
| --- | --- | --- | --- |
| Binary AND | & | Copies a bit to the evaluated result if it exists in both operands | int a = 2, b = 3;  (a & b); //returns 2 |
| Binary OR | | | Copies a bit to the evaluated result if it exists in any of the operand | int a = 2, b = 3;  (a | b); //returns 3 |
| Binary XOR | ^ | Copies the bit to the evaluated result if it is present in either of the operands but not both | int a = 2, b = 3;  (a ^ b); //returns 1 |
| Left Shift | << | Shifts the value to left by the number of bits specified by the right operand. | int a = 2, b = 3;  (a << 1); //returns 4 |
| Right Shift | >> | Shifts the value to right by the number of bits specified by the right operand. | int a = 2, b = 3;  (a >> 1); //returns 1 |
| One’s Complement | ~ | Changes binary digits 1 to 0 and 0 to 1 | int b = 3;  (~b); //returns -4 |

***Note:****Only****char****and****int****data types  can be used with Bitwise Operators.*

**Example:**

C++

// CPP Program to demonstrate the Relational Operators

#include <iostream>

using namespace std;

int main()

{

int a = 6, b = 4;

// Equal to operator

cout << "a == b is " << (a == b) << endl;

// Greater than operator

cout << "a > b is " << (a > b) << endl;

// Greater than or Equal to operator

cout << "a >= b is " << (a >= b) << endl;

// Lesser than operator

cout << "a < b is " << (a < b) << endl;

// Lesser than or Equal to operator

cout << "a <= b is " << (a <= b) << endl;

// true

cout << "a != b is " << (a != b) << endl;

return 0;

}

**Output**

a == b is 0

a > b is 1

a >= b is 1

a < b is 0

a <= b is 0

a != b is 1

NOTE -

**The left shift and right shift operators should not be used for negative numbers**.

**The bitwise OR of two numbers is just the sum of those two numbers if there is no carry involved, otherwise you just add their bitwise AND.**

C++

#include <iostream>

using namespace std;

// Function to return the only odd

// occurring element

int findOdd(int arr[], int n)

{

int res = 0, i;

for (i = 0; i < n; i++)

res ^= arr[i];

return res;

}

// Driver Method

int main(void)

{

int arr[] = { 12, 12, 14, 90, 14, 14, 14 };

int n = sizeof(arr) / sizeof(arr[0]);

cout << "The odd occurring element is "<< findOdd(arr, n);

return 0;

}

**Output**

The odd occurring element is 90