Bitwise Operators in C/C++

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

& (bitwise AND) Takes two numbers as operands and does AND on every bit of two numbers. The result of AND is 1 only if both bits are 1.

| (bitwise OR) Takes two numbers as operands and does OR on every bit of two numbers. The result of OR is 1 any of the two bits is 1.

^ (bitwise XOR) Takes two numbers as operands and does XOR on every bit of two numbers. The result of XOR is 1 if the two bits are different.

<< (left shift) Takes two numbers, left shifts the bits of the first operand, the second operand decides the number of places to shift.

>> (right shift) Takes two numbers, right shifts the bits of the first operand, the second operand decides the number of places to shift.

~ (bitwise NOT) Takes one number and inverts all bits of it

Following is example C program.

```
/* C Program to demonstrate use of bitwise operators */
#include<stdio.h>
int main()
{
    unsigned char a = 5, b = 9; // a = 5(00000101), b = 9(00001001)
    printf("a = %d, b = %d\n", a, b);
    printf("a&b = %d\n", a&b); // The result is 00000001
    printf("a|b = %d\n", a|b);  // The result is 00001101
    printf("a^b = %d\n", a^b); // The result is 00001100
    printf("~a = %d\n", a = ~a);   // The result is 11111010
    printf("b<<1 = %d\n", b<<1);  // The result is 00010010
    printf("b>>1 = %d\n", b>>1);  // The result is 00000100
    return 0;
}
```

Output:

```
a = 5, b = 9
a&b = 1
a|b = 13
a^b = 12
~a = 250
b<<1 = 18
b>>1 = 4
```

Following are interesting facts about bitwise operators.

1) The left shift and right shift operators should not be used for negative numbers If any of the operands is a negative number, it results in undefined behaviour. For example results of both -1 << 1 and 1 << -1 is undefined. Also, if the number is shifted more than the size of integer, the behaviour is undefined. For example, 1 << 33 is undefined if integers are stored using 32 bits. See this for more details.

2) The bitwise XOR operator is the most useful operator from technical interview perspective. It is used in many problems. A simple example could be "Given a set of numbers where all elements occur even number of times except one number, find the odd occurring number" This problem can be efficiently solved by just doing XOR of all numbers.

```
// 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;
}

int main(void) {
   int arr[] = {12, 12, 14, 90, 14, 14, 14};
   int n = sizeof(arr)/sizeof(arr[0]);
   printf ("The odd occurring element is %d ", findOdd(arr, n));
   return 0;
}
// Output: The odd occurring element is 90
```

Run on IDE

The following are many other interesting problems which can be used using XOR operator.
Find the Missing Number, swap two numbers without using a temporary variable, A Memory Efficient Doubly Linked List, and Find the two non-repeating elements. There are many more (See this, this, this, this, this and this)

3) The bitwise operators should not be used in place of logical operators.
The result of logical operators (&&, || and !) is either 0 or 1, but bitwise operators return an integer value. Also, the logical operators consider any non-zero operand as 1. For example, consider the following program, the results of & and && are different for same operands.

```
int main()
{
   int x = 2, y = 5;
   (x & y)? printf("True ") : printf("False ");
   (x && y)? printf("True ") : printf("False ");
   return 0;
}
// Output: False True
```

Run on IDE

4) The left-shift and right-shift operators are equivalent to multiplication and division by 2 respectively.

As mentioned in point 1, it works only if numbers are positive.

```
int main()
{
   int x = 19;
   printf ("x << 1 = %d\n", x << 1);
   printf ("x >> 1 = %d\n", x >> 1);
   return 0;
}
// Output: 38 9
```

Run on IDE

5) The & operator can be used to quickly check if a number is odd or even

The value of expression (x & 1) would be non-zero only if x is odd, otherwise the value would be zero.

```
int main()
{
   int x = 19;
   (x & 1)? printf("Odd"): printf("Even");
   return 0;
}
// Output: Odd
```

Run on IDE

6) The ~ operator should be used carefully

The result of ~ operator on a small number can be a big number if the result is stored in an unsigned variable. And result may be negative number if result is stored in signed variable (assuming that the negative numbers are stored in 2's complement form where leftmost bit is the sign bit)

```
// Note that the output of the following program is compiler dependent
int main()
{
   unsigned int x = 1;
   printf("Signed Result %d \n", ~x);
   printf("Unsigned Result %ud \n", ~x);
   return 0;
}
/* Output:
Signed Result -2
Unsigned Result 4294967294d */
```

Run on IDE

Important Links :

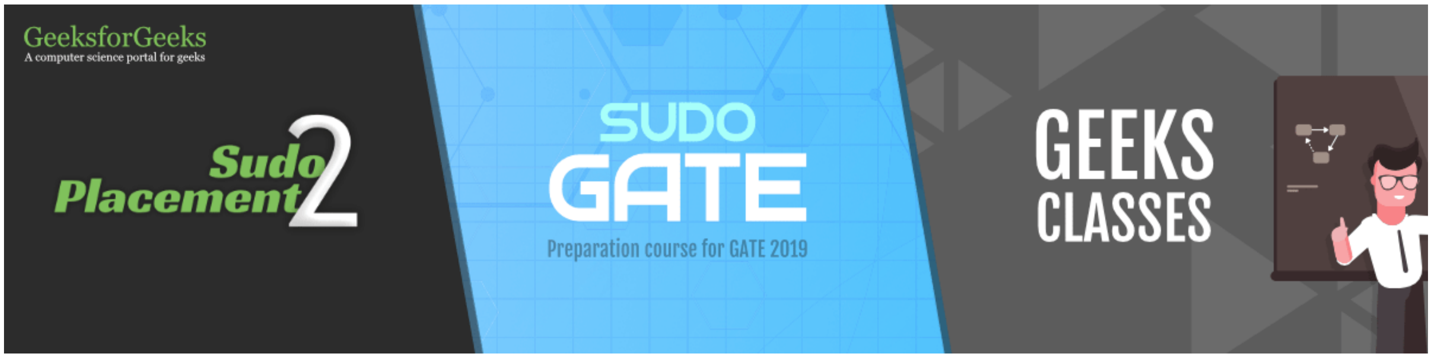
1. Bits manipulation (Important tactics)
2. Bitwise Hacks for Competitive Programming
3. Bit Tricks for Competitive Programming

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Improved By : Shubham Dhiman 1, prakash_

Practice Tags : Bit Magic C CPP

Article Tags : Bit Magic C C++ C-Operators cpp-operator XOR


GeeksforGeeks
A computer science portal for geeks
Sudo Placement 2
SUDO GATE
Preparation course for GATE 2019
GEEKS CLASSES

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content. Login to Improve this Article

Recommended Posts:

Bits manipulation (Important tactics)
Bitwise Hacks for Competitive Programming
Print a long int in C using putchar() only
How to swap two numbers without using a temporary variable?
Operators in C / C++
Remove duplicates from a string in O(1) extra space
Calculate the total fine to be collected
Sum of bitwise OR of all subarrays
Value in a given range with maximum XOR
Check if a number can be expressed as 2^x + 2^y

(Login to Rate)

2.7 Average Difficulty : **2.7/5.0**
Based on **115** vote(s)

Basic Easy Medium Hard Expert

Add to TODO List
Mark as DONE

Feedback Add Notes Improve Article

▲

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Share this post!

0 Comments GeeksforGeeks

1 Login

Recommend Tweet Share

Sort by Newest

Start the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Be the first to comment.

Subscribe Add Disqus to your siteAdd DisqusAdd Disqus' Privacy PolicyPrivacy PolicyPrivacy

A computer science portal for geeks

710-B, Advant Navis Business Park,
Sector-142, Noida, Uttar Pradesh - 201305
feedback@geeksforgeeks.org

COMPANY

About Us
Careers
Privacy Policy
Contact Us

LEARN

Algorithms
Data Structures
Languages
CS Subjects
Video Tutorials

PRACTICE

Company-wise
Topic-wise
Contests
Subjective Questions

CONTRIBUTE

Write an Article
Write Interview Experience
Internships
Videos

▲