Bitwise Operators

11 January 2022 17:39

① Bitwise AND : (&)

Eg: 2 & 3 = 010 & 011 = 010

Both bits must be 1 to get 1.

x	y	AND
0	0	0
0	1	0
1	0	0
1	1	1

AND gives the small number as output

Eg: 5 & 7 :

```
  101
& 111
-----
  101   = (5)
```

② Bitwise OR : (|)

Eg: 2 | 5 :

```
  010
  101
-----
  111
```

OR gives the big number as output

If at least one of the bits is 1 then the output will be 1.

x	y	OR
0	0	0
0	1	1
1	0	1
1	1	1

Eg: 3 | 6 :

```
  011
  110
-----
  111
```

③ Bitwise NOT : (~)

Eg: (consider 4 byte representation)

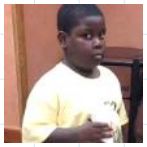
a = 2 = 000...0010 (30 bits)

~a = ~2 = 1 11...1101

1 ↓ Signed Bit

Two's complement : 000...0010 + 1

x	NOT
0	1
1	0



000....0011 = -3

④ Bitwise XOR : (^)

x	y	XOR
0	0	0
0	1	1
1	0	1
1	1	0

If different bits → 1
If same bits → 0

Eg: 5 ^ 7 = 101
^ 111
010 = 2

```
1  #include<iostream>
2  using namespace std;
3
4
5  int main() {
6
7      int a = 4;
8      int b = 6;
9
10     cout<<" a&b " << (a&b) << endl;
11     cout<<" a|b " << (a|b) << endl;
12     cout<<" ~a " << ~a << endl;
13     cout<<" a^b " << (a^b) << endl;
14
15 }
```

```
BitwiseOperators && "/Users/lovebabbar/"BitwiseOperators
 a&b 4
 a|b 6
 ~a -5
 a^b 2
lovebabbar@192 ~ %
```

⑤ Left Shift : (<<)

Eg: 5 << 1 → Left shift 5, 1 time
= 101 << 1 → 00....0101 << 1
= 00....1010 = 10 = 5×2^1

Eg: 3 << 2 → 00....011 << 2
= 00....1100 = 12 = 3×2^2

In most cases we multiply with power of 2.
But in some cases, this isn't true.

Eg: 0100...00 << 1 = 1000...00
Positive → Negative ??

So << is ok for smaller numbers.

⑥ Right Shift : (>>)

Eg: 15 >> 1 = 000....1111 >> 1
= 000....0111 = 7

Eg: 5 >> 2 = 000....0101 >> 2
000....0001 = (1)

Padding in << and >> for POSITIVE numbers is done with 0.

For NEGATIVE numbers, padding is compiler dependent.

```
cout<< (17>>1)<<endl;
cout<< (17>>2) <<endl;
cout<< (19<<1) <<endl;
cout<< (21<<2) <<endl;
```

```
8
4
38
84
lovebabbar@192 ~ %
```

(7) Increment / Decrement :

→ We can write $i = i+1$ as $i++$ or $++i$.

$i++$ is called Post - Increment
$++i$ is called Pre - Increment.

→ We can write $i = i-1$ as $i--$ or $--i$.

$i--$ is called Post - decrement
$--i$ is called Pre - decrement.

→ $a = a+b$ is same as $a += b$.

→ $a = a-b$ is same as $a -= b$.

Post - Increment : The value gets used first and then increments.

Eg: int i = 3, a = 2;

```
int sum = a + (i++);
sum = 2 + 3;
sum = 5;
```
Now i is 4.

Pre - Increment : The value gets incremented first and then gets used.

Eg: int i = 3, a = 2;

```
int sum = a + (++i);     i has become 4
sum = 2 + 4 ;            first.
sum = 6;
```

Post - Decrement : The value gets used first and then rements.

Eg: int i = 3, a = 2;

```
int sum = a + (i--);
sum = 2 + 3 ;
sum = 5;
  Now i is 2.
```

Pre - Decrement : The value gets remented first and then gets used.

Eg: int i = 3, a = 2;

```
int sum = a + (--i);     i has become 2
sum = 2 + 2 ;            first.
sum = 4 ;
```

```
20    int i = 7;
21
22    cout<< (++i) <<endl;
23    // 8
24    cout<< (i++) <<endl;
25    // 8 , i = 9
26    cout<< (i--) <<endl;
27    //9 , i = 8
28    cout<< (--i) <<endl;
29    // 7, i =7
```

```
8
8
9
7
lovebabbar@192 ~ %
```

Q1

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5          int a, b = 1;
6          a = 10;
7          if (++a)
8             cout << b;
9          else
10             cout << ++b;
11 }
```

Answer : 1

Q2

```
#include <iostream>
using namespace std;
int main()
{
    int a = 1;
    int b = 2;

    if(a-- > 0 && ++b > 2 ){
    cout << "Stage1 - Inside If ";
    } else{
    cout << "Stage2 - Inside else ";
    }
    cout << a << " " << b << endl;
}
```

Answer : Stage1 - Inside If 0 3

Q3

```
#include <iostream>
using namespace std;
int main()
{
    int a = 1;
    int b = 2;

    if(a-- > 0 || ++b > 2 ){
    cout << "Stage1 - Inside If ";
    } else{
    cout << "Stage2 - Inside else ";
    }
    cout << a << " " << b << endl;
}
```

Answer : Stage1 - Inside If 0 2

Hint: Only one of the conditions must be true for || so it won't check ++b > 2.

Q3

```
#include <iostream>
using namespace std;
int main()
{
        int number = 3;
        cout << (25 * (++number) );
}
```

Answer : 100

Q4

```
#include <iostream>
using namespace std;
int main()
{
    int a = 1;
    int b = a++;
    int c = ++a;
    cout << b ;
    cout << c;
}
```

Answer : 13

Example Code :

```cpp
#include<iostream>
using namespace std;

int main() {

    int n ;
    cout<<" enter the value of n" <<endl;
    cin >> n;

    cout<<"printing count from 1 to n" << endl;

    for(int i = 1; i<=n; i++) {
        cout<< i << endl;
    }


}
```

```
& "/Users/lovebabbar/"forLoop
 enter the value of n
5
printing count from 1 to n
1
2
3
4
5
lovebabbar@192 ~ %
```

Structure : `for ( int i = 0 ; i < n ; i++ )`

- `int i = 0` — initialization
- `i < n` — cond. to check before every loop/iteration
- `i++` — updation

```
{
  ~~~~
  ~~~~
}
```

Not Necessary to specify.

Example : for(;;;) = ?

```cpp
#include<iostream>
using namespace std;

int main() {

    int n ;
    cout<<" enter the value of n" <<endl;
    cin >> n;

    cout<<"printing count from 1 to n" << endl;
    int i = 1;
    for(; i<=n; ) {
        cout<< i << endl;
        i++;
    }


}
```

→

```
lovebabbar@192 ~ % cd "/Users/lovebabbar/" && g++ forLoop.cpp -o forLoop &
& "/Users/lovebabbar/"forLoop
 enter the value of n
5
printing count from 1 to n
1
2
3
4
5
lovebabbar@192 ~ %
```

You can declare all the 3 parts outside the paranthesis ().

WRONG :

```cpp
int i = 1;
for(;;; ) {
    if(i<=n) {
        cout<< i << endl;
    }

    i++;
}
```

→ 

The for loop doesn't know when to stop.

Solution : break ;

↳ Gets you out of the current loop.

```cpp
int n ;
cout<<" enter the value of n" <<endl;
cin >> n;

cout<<"printing count from 1 to n" << endl;
int i = 1;
for(; ; ) {
    if(i<=n) {
        cout<< i << endl;
    }
    else{
        break;
    }

    i++;
}
```

```
 enter the value of n
5
printing count from 1 to n
1
2
3
4
5
lovebabbar@192 ~ %
```

Example: Using multiple inits., conditions and updations.

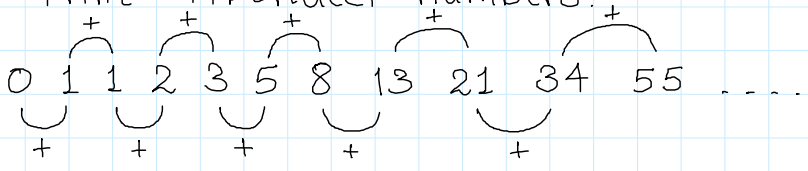
```cpp
for(int a = 0 , b =1, c = 2; a>=0 && b>=1 && c>=2; a--,b--, c-- ){
    cout<<a <<" "<< b << " "<<c << endl;
}
```

Question: Print 1 to n.

```cpp
#include<iostream>
using namespace std;

int main() {

    int n ;
    cout<<" enter the value of n" <<endl;
    cin >> n;

    int sum = 0;

    for(int i=1; i<=n; i++ ) {
        //sum = sum + i;
        sum += i;
    }

    cout<< sum <<endl;

}
```

```
lovebabbar@192 ~ % cd "/Users/lovebabbar/" && g++ forSum.cpp -o forSum &&
"/Users/lovebabbar/"forSum
 enter the value of n
5
15
lovebabbar@192 ~ %
```

Question: Print Fibonacci numbers.



Solution :

```cpp
#include<iostream>
using namespace std;

int main() {

    int n = 10;

    int a = 0;
    int b = 1;
    cout<<a <<" " <<b<<" ";
    for(int i = 1; i<=n; i++ ) {

        int nextNumber = a+b;
        cout<<nextNumber<<" ";

        a = b;
        b = nextNumber;
    }


}
```

```
lovebabbar@192 ~ % cd "/Users/lovebabbar/" && g++ forFib.cpp -o forFib &&
"/Users/lovebabbar/"forFib
0 1 1 2 3 5 8 13 21 34 55 89
lovebabbar@192 ~ %
```

Question: Print if prime. (Logic already covered)

Ex: I/P - 7
O/P - Yes

Solution:

```cpp
#include<iostream>
using namespace std;

int main() {

    int n ;
    cout<<" enter the value of n" <<endl;
    cin >> n;

    bool isPrime = 1 ;

    for(int i = 2; i<n; i++) {

        //rem = 0, Not a Prime
        if(n%i == 0) {
            //cout<<" Not a Prime Number" << endl;
            isPrime = 0;
            break;
        }
    }

    if(isPrime == 0) {
        cout<<" Not a Prime Number"<<endl;
    }
    else
    {
        cout<<"is a Prime Number "<<endl;
    }


}
```

```
lovebabbar@192 ~ % cd "/Users/lovebabbar/" && g++ forPrime.cpp -o forPrime
 && "/Users/lovebabbar/"forPrime
 enter the value of n
101
is a Prime Number
```

break ka brother continue

continue: Used to skip an iteration of the current loop.

It skips the remaining block of code for that iteration.

code example:

```cpp
#include<iostream>
using namespace std;

int main() {

    for(int i=0; i<5; i++) {

        cout<< " HI " << endl;
        cout<< " Hey " << endl;
        continue;

        cout<< "Reply toh karde " <<endl;

    }


}
```

```
lovebabbar@192 ~ % cd "/Users/lovebabbar/" && g++ continue.cpp -o continue
 && "/Users/lovebabbar/"continue
 HI
 Hey
 HI
 Hey
 HI
 Hey
 HI
 Hey
 HI
 Hey
lovebabbar@192 ~ %
```

Output Questions:

Q1.

```cpp
#include<iostream>
using namespace std;
int main() {

    for(int i = 0; i<=5; i++) {
        cout<< i << " ";
        i++;
    }
```

Output: 0 2 4

Q1.

```cpp
#include<iostream>
using namespace std;
int main() {

    for(int i = 0; i<=5; i++) {
        cout<< i << " ";
        i++;
    }

}
```

Output : 0 2 4

Q2.

```cpp
#include<iostream>
using namespace std;
int main() {

    for(int i = 0; i<=5; i--) {
        cout<< i << " ";
        i++;
    }

}
```

Output : 0 0 0 0 0 ..... (infinite times)

Q3.

```cpp
#include<iostream>
using namespace std;
int main() {

    for(int i = 0; i<=15; i += 2 ) {
        cout<< i << " ";

        if( i&1 ){
            continue;
        }

        i++;
    }
}
```

Output : 0 3 5 7 9 11 13 15

Q4

```cpp
#include<iostream>
using namespace std;
int main() {

    for(int i=0;i<5;i++) {
        for(int j=i;j<=5;j++) {
            cout<<i << " " << j <<endl;
        }
    }
}
```

Output :

```
0 0
0 1
0 2
0 3
0 4
0 5
1 1
1 2
1 3
1 4
1 5
2 2
2 3
2 4
2 5
3 3
3 4
3 5
4 4
4 5
```

Q5

```
#include<iostream>
using namespace std;
int main() {

    for(int i=0;i<5;i++) {
        for(int j=i;j<=5;j++) {
            if(i+j == 10) {
                break;
            }
            cout<<i << " " << j <<endl;
        }
    }
}
```

Output:

```
0 0
0 1
0 2
0 3
0 4
0 5
1 1
1 2
1 3
1 4
1 5
2 2
2 3
2 4
2 5
3 3
3 4
3 5
4 4
4 5
```

Scope: The life-time of a variable. Where does the variable exist and after what line of code will it get destroyed. In short, its accessibility.

Example:

```cpp
#include<iostream>
using namespace std;


int main() {

    int a = 3;
    cout<< a << endl;

    if(true) {
        cout<< a <<endl;
    }


}
```

a is accessible throughout the main() function, after it is declared.

Example:

```cpp
#include<iostream>
using namespace std;


int main() {

    int a = 3;
    cout<< a << endl;

    if(true) {
        int a = 5;
        cout << a <<endl;
    }


}
```

This is 'local' to int main().

This is 'local' to if block, and is only accessible within the if block.

This will print out 3
5

Example:

```cpp
#include<iostream>
using namespace std;


int main() {

    int a = 3;
    cout<< a << endl;

    if(true) {
        int b = 5;
        cout << b <<endl;
    }

    cout<< b << endl;

}
```

??

→ 

Example:

```cpp
int i = 8;
cout << b << endl;

for(; i<8; i++) {
    cout<<" HI " << endl;
}
```

→ Kuch Nahi (Empty)

```
for(; i<8; i++) {
    cout<<" HI " << endl;
}
```

The i outside the for loop gets used.

Precedence Table:

(No need to mug up)

Precedence order	Operator	Associativity
1	() [] →	Left to right
2	++ -- - (unary) ! ~ * & sizeof	Right to left
3	* / %	Left to right
4	+ -	Left to right
5	<< >>	Left to right
6	< <= > >=	Left to right
7	== !=	Left to right
8	& (bitwise AND)	Left to right
9	^ (bitwise XOR)	Left to right
10	| (bitwise OR)	Left to right
11	&& (logical AND)	Left to right
12	|| (logical OR)	Left to right
13	? : (conditional)	Right to right
14	= += -= *= /= %= (assignment operators)	Right to left
15	, (comma Operator)	Left to right

Just like BODMAS prioritizes Brackets, we can also prioritize calculations using Brackets.

Q1. Subtract the product and sum of the digits of an integer.

Soln :

```cpp
class Solution {
public:
    int subtractProductAndSum(int n) {
        int product = 1, sum = 0;
        while(n) {
            product = product * (n % 10); // Can also use *=
            sum += (n % 10);
            n /= 10; // same as n = n / 10;
        }
        return product - sum;
    }
};
```

Success Details >

Runtime: 0 ms, faster than 100.00% of C++ online submissions for Subtract the Product and Sum of Digits of an Integer.

Memory Usage: 5.9 MB, less than 23.16% of C++ online submissions for Subtract the Product and Sum of Digits of an Integer.

Q2. Count number of 1 bits.

Soln :

```cpp
class Solution {
public:
    int hammingWeight(uint32_t n) {
        int ans = 0;
        while(n) {
            if(n & 1) {
                ans++;
            }
            n = n >> 1;
        }
        return ans;
    }
};
```

Q3. Reverse integer.

Soln :

```cpp
class Solution {
public:
    int reverse(int x) {
        int rev = 0;
        while (x != 0) {
            int pop = x % 10;
            x /= 10;
            if (rev > INT_MAX/10 || (rev == INT_MAX / 10 && pop > 7)) return 0;
            if (rev < INT_MIN/10 || (rev == INT_MIN / 10 && pop < -8)) return 0;
            rev = rev * 10 + pop;
        }
        return rev;
    }
};
```

Q4. Complement of Base 10

Soln :

```cpp
class Solution {
public:
    int bitwiseComplement(int n) {
        if(n == 0) return 1;

        int ans = 0, fac = 1;

        while(n != 0){
            int bit = n % 2 == 0;
            ans += fac * bit;
            fac *= 2;
            n /= 2;
        }
        return ans;
    }
};
```

Q5 Number Compliment.

Soln :

```cpp
class Solution {
public:
    int findComplement(int num) {
        int msb = (int)(log2(num));
        if(num ==0) {
            return 1;
        }

        int sum = 0;
        for(int i=msb; i>=0;i--) {
            if(num &(1<<i)){
                continue;
            }
            else {
                sum+=pow(2,i);
            }
        }
        return sum;
    }
};
```

Q6. Binary to Decimal.

Soln :

```cpp
#include <iostream>
using namespace std;

int main()
{
    int num = 1010001;
    int dec_value = 0;
    int base = 1;
    int temp = num;

    while (temp) {
        int last_digit = temp % 10;
        temp = temp / 10;

        dec_value += last_digit * base;

        base = base * 2;
    }
    cout << dec_value << endl;
    return 0;
}
```