



### 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

Eg:  $5 \& 7$  : 
$$\begin{array}{r} 101 \\ \& 111 \\ \hline 101 \end{array} = 5$$

### ② Bitwise OR : (|)

Eg:  $2 | 5$  : 
$$\begin{array}{r} 010 \\ | 101 \\ \hline 111 \end{array}$$

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

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

Eg:  $3 | 6$  : 
$$\begin{array}{r} 011 \\ | 110 \\ \hline 111 \end{array}$$

### ③ Bitwise NOT : (~)

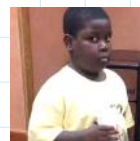
Eg: (Consider 4 byte representation)

$a = 2 = \overbrace{000 \dots 0010}^{30 \text{ bits}}$   
 $\sim a = \sim 2 = \textcircled{1}11 \dots 1101$

↓  
Signed Bit

Two's complement : 
$$\begin{array}{r} 000 \dots 0010 \\ + \quad \quad \quad 1 \\ \hline \end{array}$$

x	NOT
0	1
1	0



$$000\dots0011 = -3$$

#### ④ Bitwise XOR : (^)

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

If different bits  $\rightarrow 1$

If same bits  $\rightarrow 0$

$$\text{Eg: } 5 \wedge 7 = \begin{array}{r} 101 \\ \wedge 111 \\ \hline 010 \end{array} = 2$$

```
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 }
```

```
lovebabbar@192 ~ % cd - /Users/lovebabbar/ && g++ BitwiseOperators.cpp -o
BitwiseOperators && "/Users/lovebabbar/"BitwiseOperators
a&b 4
a|b 6
~a -5
a^b 2
lovebabbar@192 ~ %
```

#### ⑤ Left Shift : (<<)

$$\begin{aligned} \text{Eg: } 5 << 1 &\rightarrow \text{left shift 5, 1 time} \\ &= 101 << 1 \rightarrow 00\dots0101 << 1 \\ &= 00\dots1010 = 10 = 5 \times 2^1 \end{aligned}$$

$$\begin{aligned} \text{Eg: } 3 << 2 &\rightarrow 00\dots011 << 2 \\ &= 00\dots1100 = 12 = 3 \times 2^2 \end{aligned}$$

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

$$\text{Eg: } 0100\dots00 << 1 = 1000\dots00$$

Positive  $\rightarrow$  Negative ??

So << is ok for smaller numbers.

#### ⑥ Right Shift : (>>)

$$\begin{aligned} \text{Eg: } 15 >> 1 &= 000\dots1111 >> 1 \\ &= 000\dots0111 = 7 \end{aligned}$$

Eg:  $5 \gg 2 = 000\dots0101 \gg 2$   
 $000\dots0001 = 1$

Padding in  $\ll$  and  $\gg$  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 0 0  
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 decrements.

Eg: `int i = 3, a = 2;`

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

**Pre-Decrement :** The value gets decremented 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 I
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

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int a = 1;
6     int b = 2;
7
8     if(a-- > 0 && ++b > 2 ){
9         cout << "Stage1 - Inside If ";
10    } else{
11        cout << "Stage2 - Inside else ";
12    }
13    cout << a << " " << b << endl;
14 }
```

Answer: Stage1 - Inside If 0 3

Q3

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int a = 1;
6     int b = 2;
7
8     if(a-- > 0 || ++b > 2 ){
9         cout << "Stage1 - Inside If ";
10    } else{
11        cout << "Stage2 - Inside else ";
12    }
13    cout << a << " " << b << endl;
14 }
```

Answer : Stage1 - Inside If 0 2

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

Q3

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int number = 3;
6     cout << (25 * (++number) );
7 }
```

Answer : 100

Q4

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

Answer : 13



## Example Code :

```
1 #include<iostream>
2 using namespace std;
3
4 int main() {
5
6     int n;
7     cout<<" enter the value of n" <<endl;
8     cin >> n;
9
10    cout<<"printing count from 1 to n" << endl;
11
12    for(int i = 1; i<=n; i++) {
13        cout<< i << endl;
14    }
15
16
17
18 }
```

```
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 ~ %
```

Structure :  $\text{for}(\underbrace{\text{int } i = 0}_{\text{initialization}}; \underbrace{i < n}_{\text{cond. to check before every loop/iteration}}; \underbrace{i++}_{\text{updtation}})$

{

~~~~~

}

Not Necessary to specify.

Example :  $\text{for}(:;:) = ?$

```
1 #include<iostream>
2 using namespace std;
3
4 int main() {
5
6     int n;
7     cout<<" enter the value of n" <<endl;
8     cin >> n;
9
10    cout<<"printing count from 1 to n" << endl;
11    int i = 1;
12    for(; i<=n; ) {
13        cout<< i << endl;
14        i++;
15    }
16
17
18
19 }
```

```
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 :

```
11 int i = 1;
12 for(;; ) {
13     if(i<=n) {
14         cout<< i << endl;
15     }
16
17     i++;
18 }
```



The for loop doesn't know when to stop.

Solution : `break;`

↳ Gets you out of the current loop.

```

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.

```

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.

```

1 #include<iostream>
2 using namespace std;
3
4 int main() {
5
6     int n ;
7     cout<<" enter the value of n" <<endl;
8     cin >> n;
9
10    int sum = 0;
11
12    for(int i=1; i<=n; i++ ) {
13        //sum = sum + i;
14        sum += i;
15    }
16
17    cout<< sum <<endl;
18
19 }

```

```

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

```

question: Print Fibonacci numbers.

0 1 1 2 3 5 8 13 21 34 55 ...

Diagram showing the addition of previous two numbers to get the next number in the sequence:

```

      +   +   +   +   +
    0 1 1 2 3 5 8 13 21 34 55
      +   +   +   +   +

```

Solution :

```

1 #include<iostream>
2 using namespace std;
3
4 int main() {
5
6     int n = 10;
7
8     int a = 0;
9     int b = 1;
10    cout<<a <<" " <<b<<" ";
11    for(int i = 1; i<=n; i++ ) {
12
13        int nextNumber = a+b;
14        cout<<nextNumber<<" ";
15
16        a = b;
17        b = nextNumber;
18    }
19
20
21
22
23
24 }

```

```

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:

```
1 #include<iostream>
2 using namespace std;
3
4 int main() {
5
6     int n ;
7     cout<<" enter the value of n" <<endl;
8     cin >> n;
9
10    bool isPrime = 1 ;
11
12    for(int i = 2; i<n; i++) {
13
14        //rem = 0, Not a Prime
15        if(n%i == 0) {
16            //cout<<" Not a Prime Number" << endl;
17            isPrime = 0;
18            break;
19        }
20    }
21
22    if(isPrime == 0) {
23        cout<<" Not a Prime Number"<<endl;
24    }
25    else
26    {
27        cout<<"is a Prime Number "<<endl;
28    }
29
30 }
31 }
```

```
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:

```
1 #include<iostream>
2 using namespace std;
3
4 int main() {
5
6     for(int i=0; i<5; i++) {
7
8         cout<<" HI " << endl;
9         cout<<" Hey " << endl;
10        continue;
11
12        cout<<"Reply toh karde " <<endl;
13    }
14
15 }
16
17 }
```

```
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.

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

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

Output: 0 2 4



Q1.

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

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

}
```

Output : 0 2 4

Q2.

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

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

}
```

Output: 00000.....  
(infinite times)

Q3.

```
#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

```
#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 :

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

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

Example :

```
1 #include<iostream>
2 using namespace std;
3
4
5 int main() {
6
7     int a = 3;
8     cout<< a << endl;
9
10    if(true) {
11        int a = 5;
12        cout << a << endl;
13    }
14
15
16 }
```

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 :

```
1 #include<iostream>
2 using namespace std;
3
4
5 int main() {
6
7     int a = 3;
8     cout<< a << endl;
9
10    if(true) {
11        int b = 5;
12        cout << b << endl;
13    }
14
15    cout<< b << endl;
16
17
18 }
```



Example :

```
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 left |
| 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:

```
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:

```
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:

```
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:

```
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:

```
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:

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int num = 1010001;
7      int dec_value = 0;
8      int base = 1;
9      int temp = num;
10
11     while (temp) {
12         int last_digit = temp % 10;
13         temp = temp / 10;
14
15         dec_value += last_digit * base;
16
17         base = base * 2;
18     }
19     cout << dec_value << endl;
20     return 0;
21 }
```