

Bitwise operator

AND

OR

NOT

XOR

AND-

The truth table for an AND gate, which performs a logical conjunction operation, is as follows:

Input A	Input B	Output (A AND B)
0	0	0
0	1	0
1	0	0
1	1	1

OR (|)-

The OR gate performs a logical disjunction, outputting 1 if at least one input is 1.

Input A	Input B	Output (A OR B)
0	0	0
0	1	1
1	0	1
1	1	1

NOT (~)-

The NOT gate is a logical inverter that flips the input value.

Input	Output (NOT Input)
0	1
1	0

Take 2's complement with the sign

XOR(Exclusive OR)(^)

The XOR (Exclusive OR) gate outputs 1 only if the inputs are different.

Input A	Input B	Output (A XOR B)
0	0	0
0	1	1
1	0	1
1	1	0

Left Shift and Right shift operator-

Left Shift

$5 \ll 1$

(5 ko ek baar left shift kr do)

(means we are shifting all the bits by 1 place left)

(Ultimately we are multiplying the number by 2)(In majority of cases)

($n \times 2$)

Right Shift-

$15 \gg 1$

$5 \gg 2$

(shift the digits by right side)

How to shorten while loop

$i = i + 1$ can be written as $i++$ (post increment) , $++i$ (pre increment)

$i += 1$

$i = i - 1$ can be written as $i--$ (post decrement) , $--i$ (pre decrement)

$i -= 1$

Pre/Post/Increment/Decrement-

```
Int i=4;
```

```
Int a=i++;
```

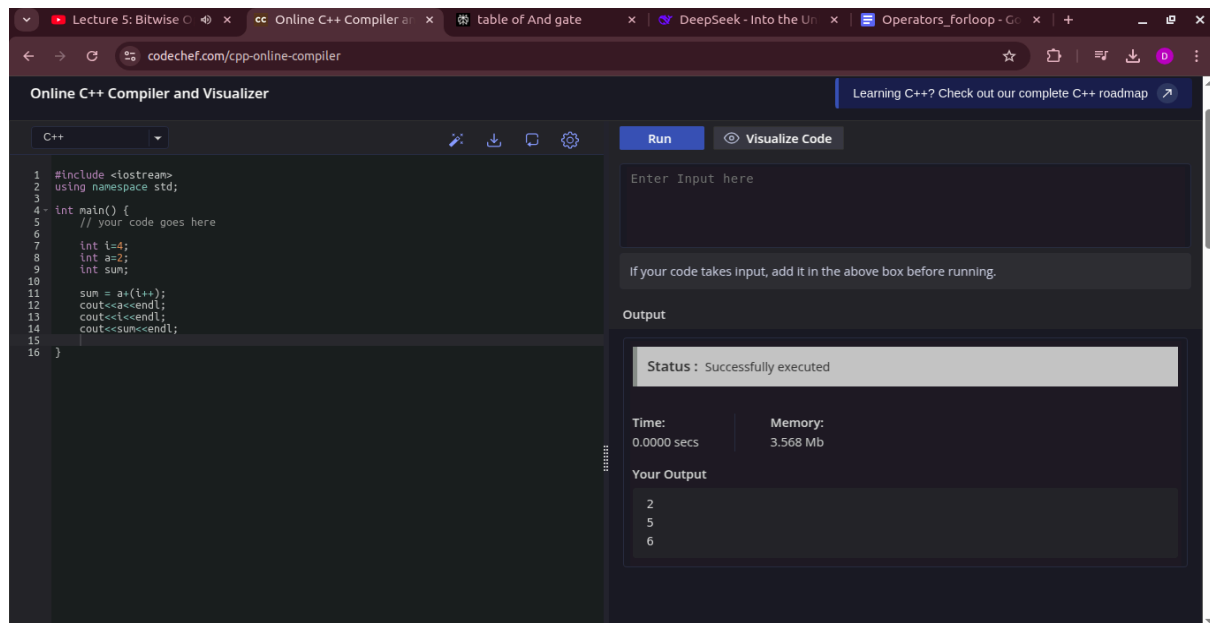
```
cout<<a<<i<<endl;
```

4 5

Because a is just used here to increment i , it holds the old value of i

But i has increased by 1

Phle use kro fir value badha do (Post increment)



```
int i=4;
    int a=++i;
    int sum;

    sum = a+(++i);
    cout<<a<<endl;
    cout<<i<<endl;
    cout<<sum<<endl;
```

5
6
11

The screenshot shows the CodeChef online C++ compiler interface. The code in the editor is as follows:

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     // your code goes here
6
7     int i=4;
8     int a=2;
9     int sum;
10
11     sum = a+(i++);
12     cout<<a<<endl;
13     cout<<i<<endl;
14     cout<<sum<<endl;
15 }
16
```

The output section shows the following results:

Status : Successfully executed

Time: 0.0000 secs Memory: 3.568 Mb

Your Output

```
2
5
6
```

Post increment
Phle use kr lo, fir ghata do

The screenshot shows the CodeChef online C++ compiler interface. The code in the editor is as follows:

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

The output section shows the following results:

Status : Successfully executed

Time: 0.0000 secs Memory: 3.48 Mb

Your Output

```
4
3
```

Pre Decrement-
Phle decrement karo fir use karo

For loop

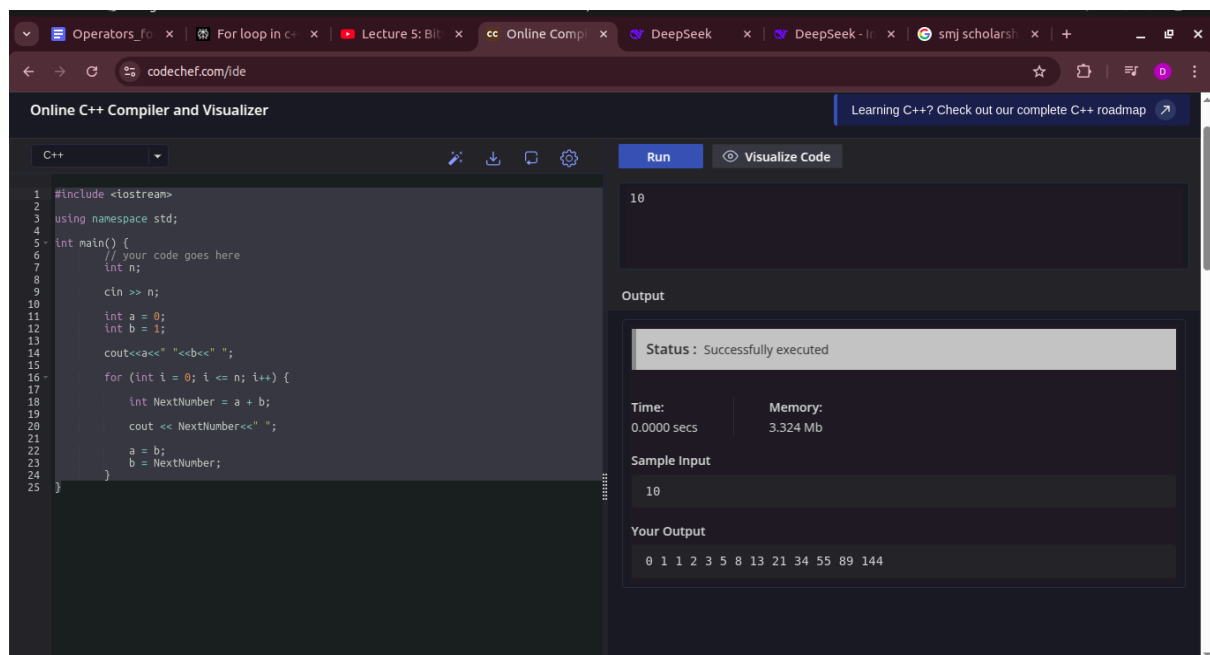
The basic syntax of a for loop in C++ is:

```
for (initialization; condition; update) {  
    // code block to be executed  
}
```

We can do multiple operations in 1 for loop

Means we can initialize 2 variables, give 2 conditions and update 2 variables

Fibonacci series-



The screenshot shows a web browser with multiple tabs, including 'codechef.com/ide'. The active page is an 'Online C++ Compiler and Visualizer'. The code editor contains the following C++ code:

```
1 #include <iostream>  
2  
3 using namespace std;  
4  
5 int main() {  
6     // your code goes here  
7     int n;  
8  
9     cin >> n;  
10  
11     int a = 0;  
12     int b = 1;  
13  
14     cout << a << " " << b << " ";  
15  
16     for (int i = 0; i <= n; i++) {  
17         int NextNumber = a + b;  
18         cout << NextNumber << " ";  
19  
20         a = b;  
21         b = NextNumber;  
22     }  
23  
24  
25 }
```

The output section shows the number '10' entered in the input field. The status is 'Successfully executed'. The execution time is '0.0000 secs' and memory is '3.324 Mb'. The sample input is '10'. The output shows the sequence: '0 1 1 2 3 5 8 13 21 34 55 89 144'.

Prime Number or not

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     // your code goes here
6     int n;
7
8     cin >> n;
9
10    bool isPrime = 1;
11
12    for(int i=2; i<n; i++){
13
14        //remainder 0, not a prime number
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";
24    }
25    else{
26        cout<<"Is a prime number";
27    }
28 }
29
30
31
32 }
```

10

Output

Status: Successfully executed

Time: 0.0000 secs Memory: 3.524 Mb

Sample Input

10

Your Output

Not a prime number

Continue keyword

loop control statement that skips the rest of the code in the current iteration and moves directly to the next iteration of the loop.

if(i & 1)

- **What it does:** Checks if the least significant bit (LSB) of *i* is 1.
- **Purpose:** Tests if *i* is odd.
 - If *i* is odd (1, 3, 5, ...), *i* & 1 evaluates to 1 (true).
 - If *i* is even (0, 2, 4, ...), *i* & 1 evaluates to 0 (false).

Scope of variables-

1. Local (Block) Scope

A variable declared inside a block (`{ }`) is accessible only within that block and its nested blocks.

```
1 #include <iostream>
2 using namespace std;
3
4 int main(){
5     int x = 10; // Local to myFunction
6     if (x > 5) {
7         int y = 20; // Local to this if-block
8         cout << x << " * " << y; // Valid (10 20)
9     }
10    // cout << y; // Error: y is not accessible here
11 }
12
```

Enter Input here

If your code takes input, add it in the above box before running.

Output

Status : Successfully executed

Time: 0.0000 secs Memory: 3.568 Mb

Your Output

10 20

We cannot define 2 variable in single block

If we have another block inside a block then we can use both the variables

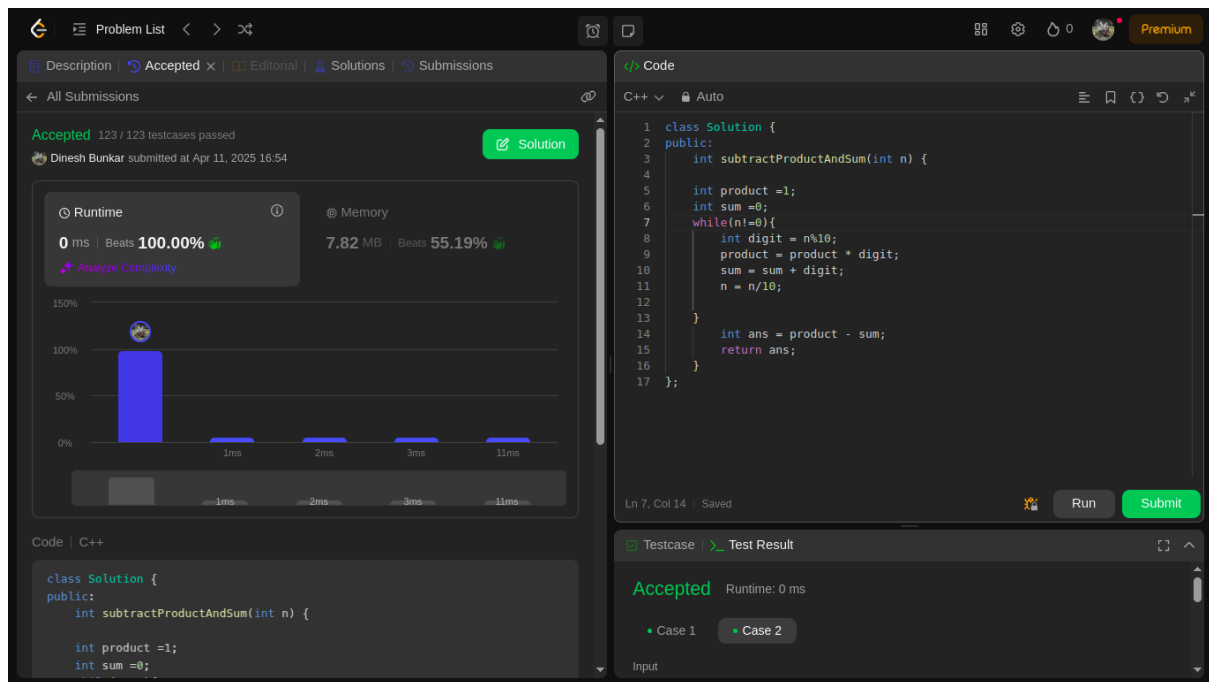
Operator Precedence-

It determines the order which operators are evaluated in an expression

It is quite similar to the bodmas rule

Leetcode question

Subtract the product and sum of digits of an integer



No of 1 bits
Solving it by using right shift operator

$n \gg 1$ right shifts by 1 digit

- For example, if n is 8 (binary 1000), $n \gg 1$ becomes 4 (binary 0100).

Reverse integer

Integer= 123

Reverse integer = 321

$$123 = 3 \cdot 10^0 + 2 \cdot 10^1 + 1 \cdot 10^2$$

Output should be 321=

$$3 \cdot 10^2 + 2 \cdot 10^1 + 1 \cdot 10^0$$

Binary and Decimal number system

1. Decimal to Binary

- Divide the number by 2.
- Write down the **remainder**.
- Repeat with the **quotient** until it's 0.
- **Read the remainders in reverse.**

Example:

Convert 13 to binary

$$13 \div 2 = 6 \text{ remainder } 1$$

$$6 \div 2 = 3 \text{ remainder } 0$$

$$3 \div 2 = 1 \text{ remainder } 1$$

$$1 \div 2 = 0 \text{ remainder } 1$$

→ **Binary: 1101**

2. Binary to Decimal

- Multiply each bit by 2^{position} and sum them.

Storing a number

For eg number is 123

Digits are 1,2,3

We have to save this in same flow

The formula we will use is

$ans=0$, to initialize it with 0

$ans = (ans * 10) + digit;$

$ans = (0 * 10) + 1 = 1$

$ans = (1 * 10) + 2 = 12$

$ans = (12 * 10) + 3 = 123$

So by this way we can save the number in same flow

In reverse order

The formula is

$ans=0$, to initialize it with 0

$ans = (digit * 10^i) + ans$

$ans = (1 * 10^0) + 0$

$ans = 1$

$ans = (2 * 10^1) + 1 = 21$

$ans = (3 * 10^2) + 21 = 321$

Decimal to binary

```
1 #include <iostream>
2 #include <math.h>
3 using namespace std;
4
5 int main() {
6     // your code goes here
7     int n;
8     cin >> n;
9
10    int ans = 0;
11    int i = 0;
12
13
14    while(n != 0) {
15        int bit = n & 1;
16
17        ans = (bit * pow(10, i)) + ans;
18
19
20        n = n >> 1;
21        i = i + 1;
22    }
23
24
25    cout << "Ans is" << ans << endl;
26
27
28 }
29
```

08

Output

Status: Successfully executed

Time: 0.0000 secs Memory: 3.712 Mb

Sample Input

08

Your Output

Ans is1000

Binary of a negative number=

Ignore negative

Convert to binary

Find 2's complement

(1's complement +1)