For loops

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

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

```
{
   ____
   ____
}
```

Example : `for (;;;)` = ?

```
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++;
    }
```

Output

```
Enter the value of n
5
printing count from 1 to n
1
2
3
4
5
```

NOTE : You can declare all the 3 part outside the parenthesis

WRONG

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

else {
    break;
}

NOTE: The for loop doesn't know when to stop

Solution: break gets you out of the current loop.

Example: Using multiple inits, condition and updations

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

Question: Print 1 to n.

```
int n;
cout<<"Enter the value of n"<<endl;
cin>>n;
int sum=0;
for (int i=1; i<=n; i++) {
    sum+=i;
}
cout<<sum<<endl;
```

output

```
Enter the value of n
5
15
```

Question: Print Fibonacci numbers

0 1 1 2 3 5 8 13 21 ----

code:

```
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;
}
```

output 0 1 1 2 3 5 8 13 21 34 55 89

Question: Print if Prime

```
int n;
cout<<"enter the value of n "<<endl;
cin >>n;
bool isPrime =1;
for (int i=2 ; i<n ; i++) {
    if (n%i ==0) {
        isPrime = 0;
        break;
    }
}
if (isPrime ==0) {
    cout << " Not a prime number "<<endl;
}
else {
    cout <<" is a prime number "<<endl;
}
```

Output

```
Enter the value of n
101
is a prime Number
```

continue:
- Used to skip an iteration of the current loop.
- It skips the remaining block of code for that iteration

Ex

```
for (int i=0 ; i<5; i++)
{
  cout << " HI " << endl ;
  cout << " Hey " << endl;
  continue
  cout << "Reply to keyde" <<endl;
}
```

Output

```
HI
Hey
HI
Hey
HI
Hey
HI
Hey
HI
Hey
```

Output Question

①
```
for (int i=0; i<=5; i++)
{ cout << i << " ";
  i++;
}
```

Output

```
0 2 4
```

②
```
for (int i=0; i<=5; i--)
{
  cout << i << " ";
  i++;
}
```

Output

```
0 0 0 0 0 ... --
(infinite times)
```

③
```
for (int i=0; i<=15; i+=2)
{ cout<<i<<" ";
  if (i%2) {
    continue;
  }
  i++;
}
```

Output

```
0 3 5 7 9 11 13 15
```

(4)

```
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
```

(5)

```
for (int i=0; i<5; i++)
{
    for (int j=i; j<=5; j++)
    {
        if (i+j == 6) {
            break;
        }
        cout <<i <<" " <<j <<endl;
    }
}
```

Scope: The life-time of a variable, where does the variable exists and after what kind of code will it get destroyed In short, its accessibility

```
int main()
{
    int a = 3;
    cout << a << endl;
    if (true){
        cout << a << endl;
    }
}
```

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

Operator Precedence

Precedence table:

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

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