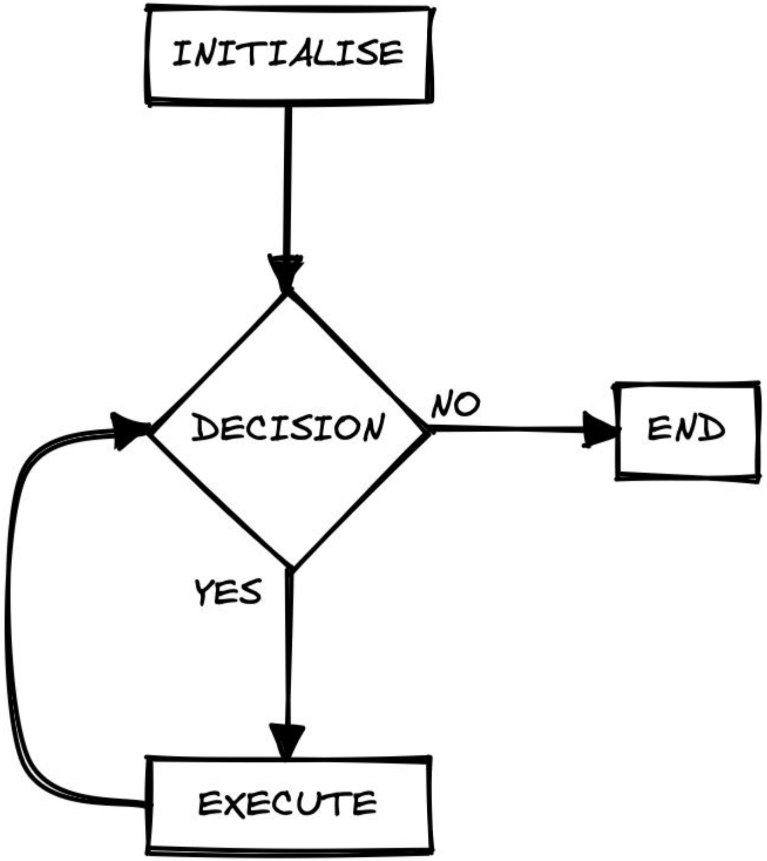
skill academy

Iterative Algorithms

By Pranjal Nadhani

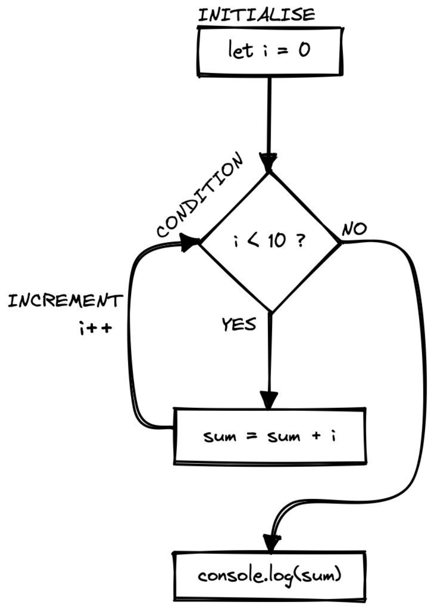


Concept

Iterative algorithms make use of iterations or loops which is about repeating a certain set of instructions until a condition is met.

In JavaScript, this can be achieved using a for loop, a while loop, or a do-while loop. Nevertheless, the keywords and the syntax may differ in various other programming languages.

A loop has 3 parts - the initialisation, the condition, and the increment/decrement operation. Almost all kinds of loops can come down to include at least 2 of these 3 steps.



Example

Loops are used to shorten the code for repetitive operations. Hence, the example shown on the right side can demonstrate how a program that contains 12 instructions can be shortened to just 4 lines of code.

```
let sum = 0;
sum += 1;
sum += 2;
sum += 3;
sum += 4;
sum += 5;
sum += 6;
sum += 7;
sum += 8;
sum += 9;
sum += 10;
console.log(sum);
```

IS EQUIVALENT TO

```
let sum = 0;
for (let i = 1; i <= 10; i++) {
 sum += i;
}
console.log(sum);
```

for and while loop

A **for loop** combines all 3 steps of an iterative algorithm - initialisation, decision and increment/decrement operations in a simple, concise syntax. This is good for the kind of algorithms where all these steps are required.

A while loop, on the other hand, gives the flexibility of writing programs that don't necessarily use all the 3 steps mentioned above. It only requires a condition as a mandatory parameter and we are free to implement the other steps in our own way.

Example 1 shows a for loop which is used to find the factors of 10 from 1 to 10.

Example 2 shows traversal of nodes in a Linked List.

```
const x = 10;
for (let i = 1; i < x; i++) {
 if (x % i === 0) {
   console.log(x, "is divisible by", i);
 }
}
```

```
const temp = head;
while (temp.link !== null) {
 console.log(temp.data, "-> ");
 temp = temp.link
}
```