Ciclos / bucles / estructuras repetitivas

Introducción

Se trata de repetir código varias veces. Puede ser una sola línea o varias (bloque de código). Esto evita que repitamos código que podemos simplificar. Supongamos que queremos imprimir diez veces un número y que en cada repetición sea el mismo número (que no cambie).

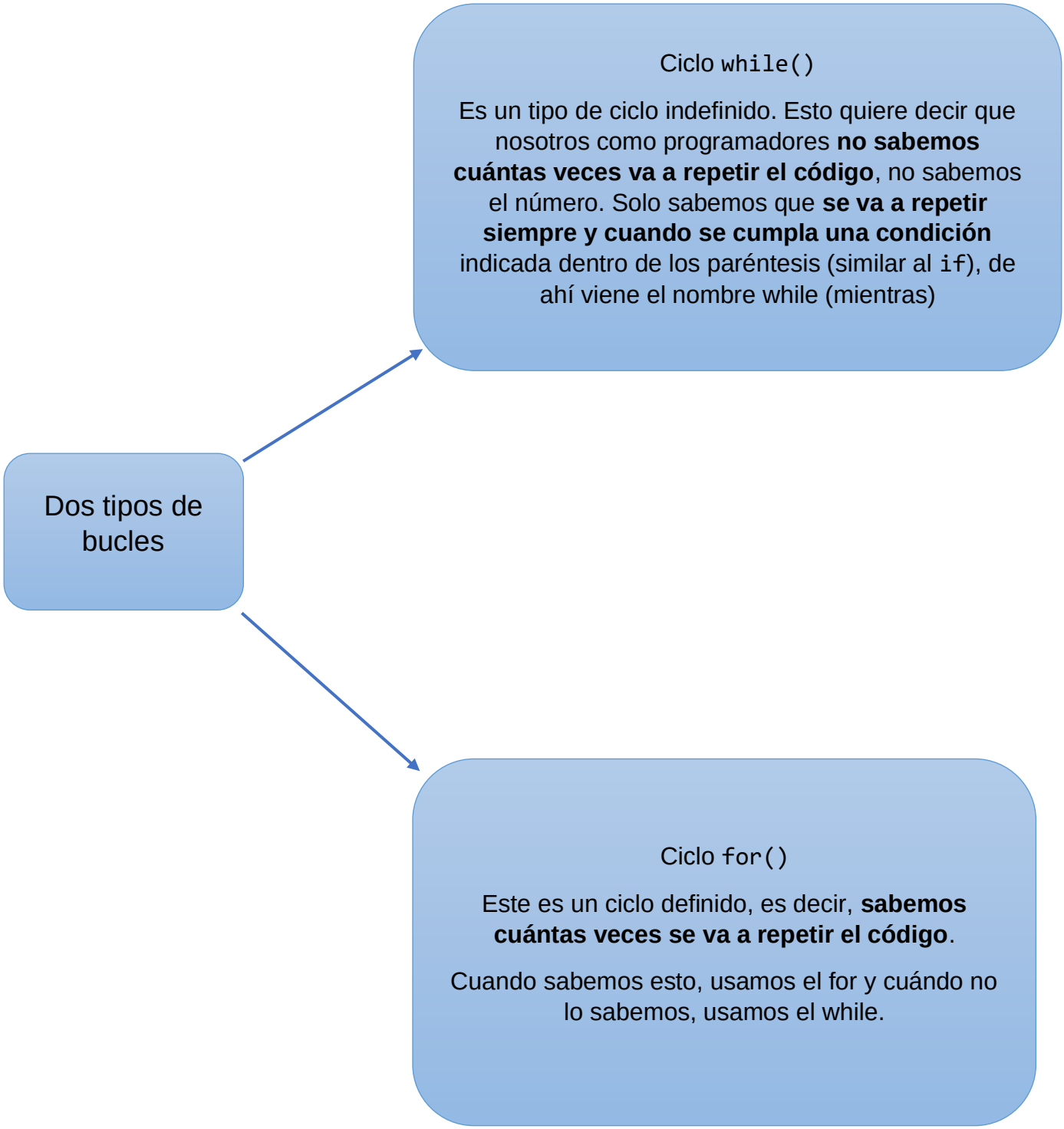
```
1   console.log(5);
2   console.log(5);
3   console.log(5);
4   console.log(5);
5   console.log(5);
6   console.log(5);
7   console.log(5);
8   console.log(5);
9   console.log(5);
10  console.log(5);
```

Normalmente necesitaríamos 10 líneas de `console.log()`

```
console.log(5);
```

pero eso podemos simplificarlo a uno solo. Es decir, usar un solo `console.log()` y que el programa lo vaya repitiendo 10 veces ella sola.

Con esto nos ahorramos muchas líneas de código y simplificamos todo. Esto nos ayuda a nosotros como programadores para leer más fácil el código y aparte de que el archivo pesa menos por tener menos líneas y menos caracteres.


Dos tipos de bucles
Ciclo while()
Es un tipo de ciclo indefinido. Esto quiere decir que nosotros como programadores no sabemos cuántas veces va a repetir el código, no sabemos el número. Solo sabemos que se va a repetir siempre y cuando se cumpla una condición indicada dentro de los paréntesis (similar al if), de ahí viene el nombre while (mientras)
Ciclo for()
Este es un ciclo definido, es decir, sabemos cuántas veces se va a repetir el código.
Cuando sabemos esto, usamos el for y cuándo no lo sabemos, usamos el while.

Ciclo while

Como bien se dijo antes, el ciclo while se usa cuando no sabemos cuántas veces se va a repetir el código. No sabemos si se va a repetir dos veces, o mil veces. En esos casos usamos el ciclo while.

El ciclo while va a repetir código siempre y cuando se cumpla una condición. Cuando se deja de cumplir, el programa se sale del ciclo y deja de repetir el código para ejecutar lo que sigue abajo del while.

Estructura (similar al if)

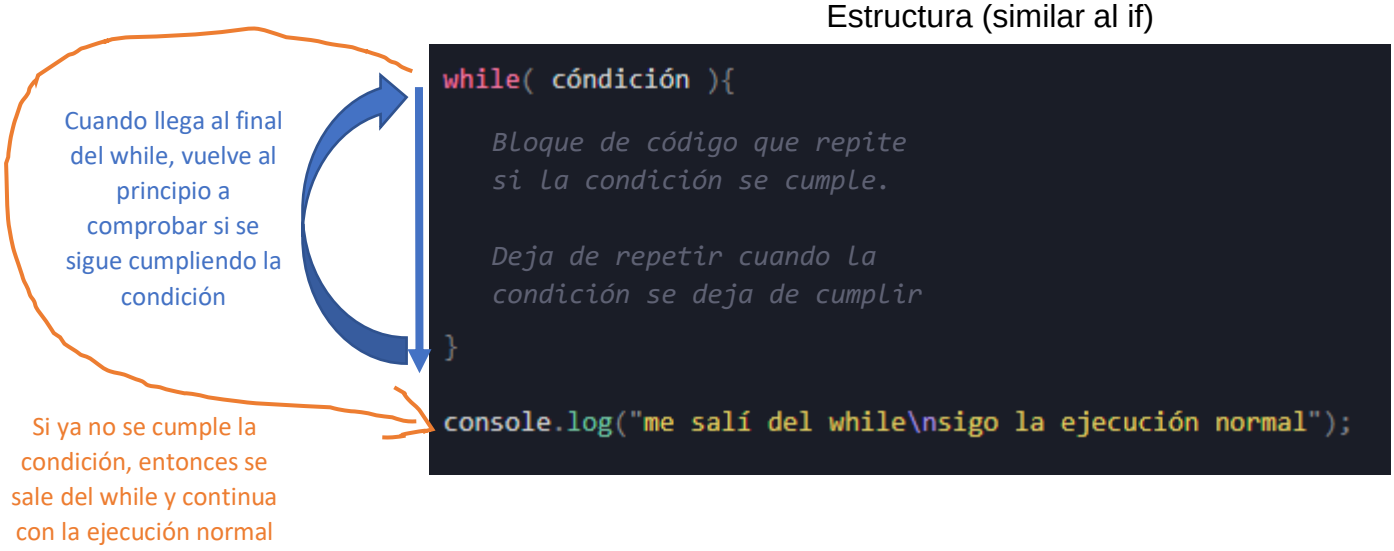
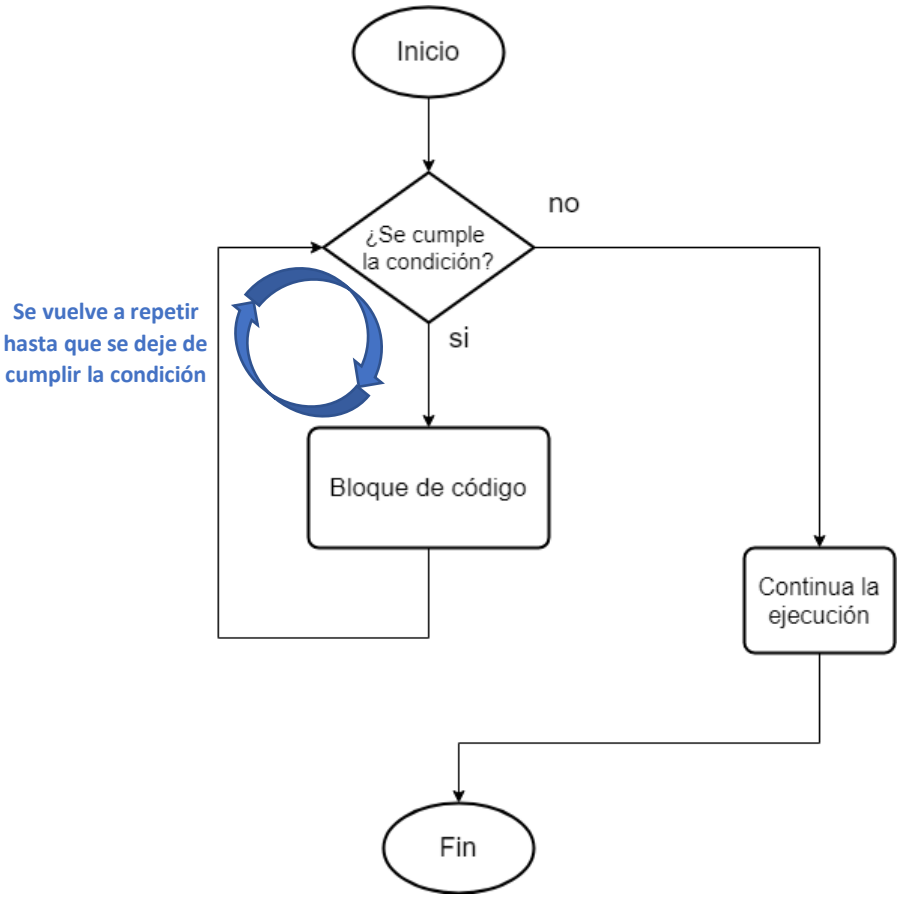


Diagrama de while



Ejemplo

```
let num = 5;

while( num <= 10 ){

    document.write( num + "<br>");

    num++;
}

console.log("me salí del while\nsigo la ejecución normal");
```

En este ejemplo, creo una variable y la inicializo en 5

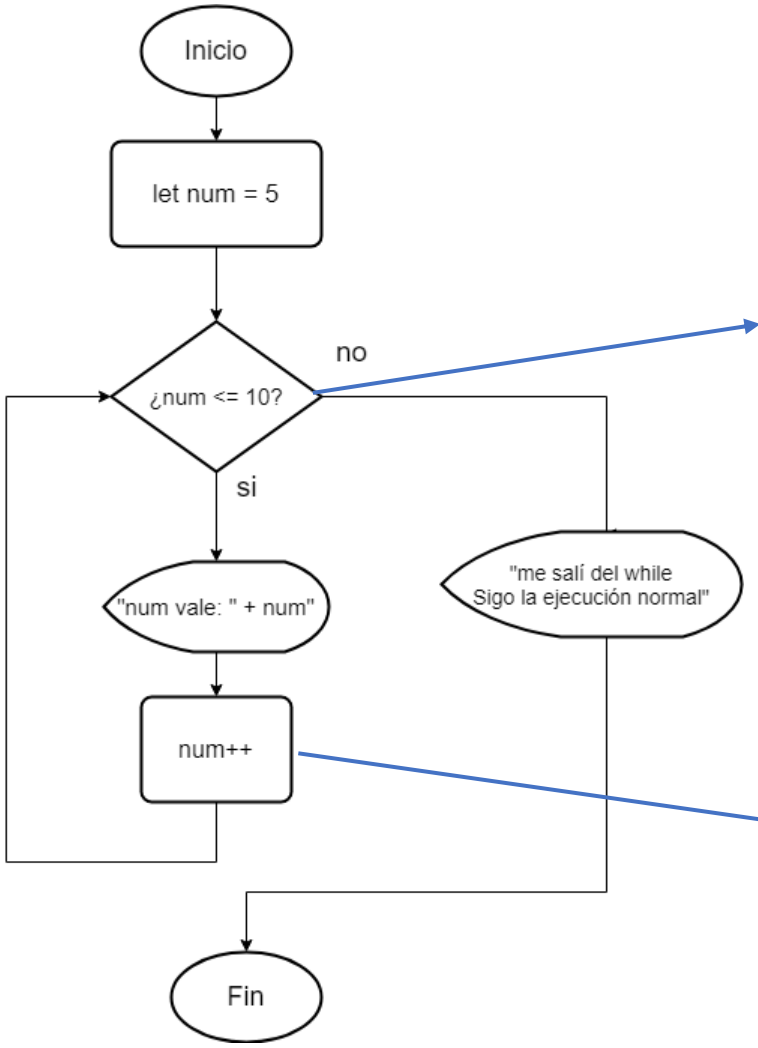
Siempre que sea menor o igual a 10, se va a repetir el código. Es decir, se va a repetir hasta que la variable valga 11

Siempre tengo que modificar la variable para que cambie en las siguientes repeticiones. Esto es para no generar un bucle infinito.

En este caso uso un operador de incremento, el cual incremente en uno el valor de la variable (es decir, le suma un uno).

En el ejemplo se muestran los valores de una variable, hasta llegar a 10.

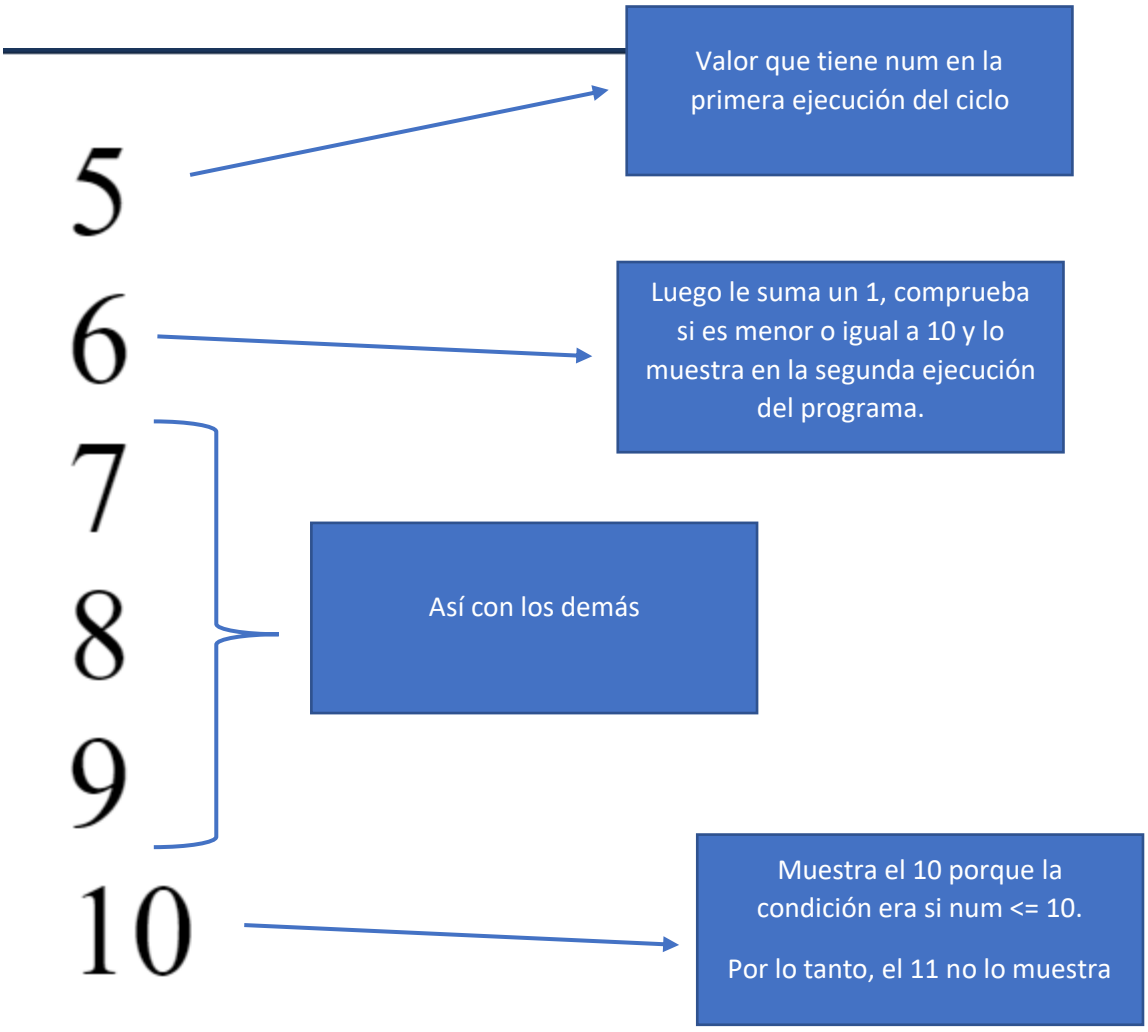
Diagrama del ejemplo



Va a repetir el código hasta que num sea mayor a 10 (ósea, 11).

Incrementa el valor de num para que no sea siempre el mismo (y así evitar un bucle infinito).

Resultado


5
6
7
8
9
10
Valor que tiene num en la primera ejecución del ciclo
Luego le suma un 1, comprueba si es menor o igual a 10 y lo muestra en la segunda ejecución del programa.
Así con los demás
Muestra el 10 porque la condición era si num <= 10.
Por lo tanto, el 11 no lo muestra

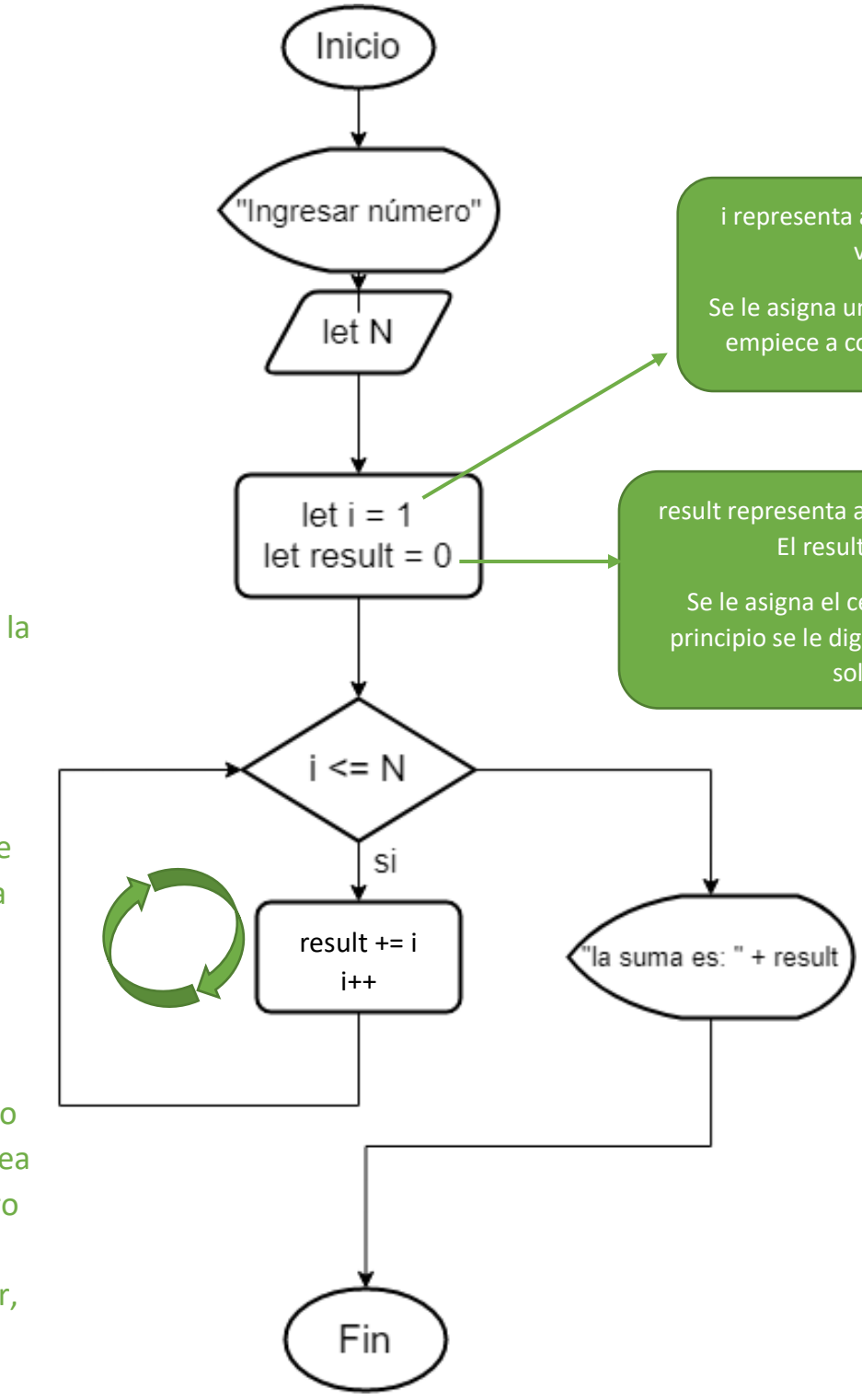
Ejercicio: Hacer un programa que solicite al usuario un número N y muestre el valor de la suma de los números que van del 1 al N (1+2+3+4+5+…+N)

Ejemplo:

-Si El usuario ingresa 6 (N = 6), se hace 1+2+3+4+5+6

-Si el usuario ingresa 8 (N = 8), se hace 1+2+3+4+5+6+7+8

Diagrama



i representa a todos los números que van de 1 a N.

Se le asigna un 1 porque queremos que empiece a contar desde ese número

result representa a la suma de esos números. El resultado de la suma.

Se le asigna el cero para que ya desde el principio se le diga al programa que guarda solo números

Se va guardando la suma en la variable result

Luego a la i se le suma un 1 para incrementar el valor

Va a repetir todo eso hasta que i sea mayor al número que ingresó el usuario (es decir, hasta que i sea mayor a N)

Ciclo for

El ciclo for se comporta de una manera similar al while, también repite código, pero la diferencia es que, con el ciclo for, nosotros sabemos el número de veces que va a repetir el código.

Dependiendo del problema vamos a decidir entre uno u otro. Incluso hay veces en los que no sabemos cual usar y la única manera de decidir es probar uno y si no funciona como queremos, usar el otro.

Veamos un ejemplo:

> Pensemos en un contador de números, que básicamente va a ir contando números del 1 al 10 por ejemplo. En este caso, nosotros sabemos que el código se va a ejecutar 10 veces, por lo tanto, vamos a usar el for (también se puede usar el while, pero en estos casos en los que sabemos la cantidad, el for le gana).

Ahora pensemos otro ejemplo:

> Digamos que en un boliche se le tiene la entrada prohibida para aquel que se llama "Kevin Díaz" y el patovica de la entrada tiene que ir comprobando el nombre de cada persona que va ingresando. En este caso no se puede usar un for ya que no sabemos cuantas personas hay en la entrada, no sabemos si más tarde viene más gente. En resumen, **no sabemos cuántas veces se va a repetir la verificación**, por lo tanto, usaremos el while (literalmente, el while significa *mientras*, es decir, vamos a decirle al patovica "mientras que la persona no se llame Kevin Díaz, dejala pasar").

Estructura del for

El for se escribe igual que las estructuras anteriores (while, if) con la particularidad de que dentro de los paréntesis hay tres partes separadas por puntos y comas.

Se trata de la inicialización de una variable llamada i.

i viene de iterador, significa que va a ir contando desde el número que nosotros le asignemos, hasta otro número indicado en la condición.

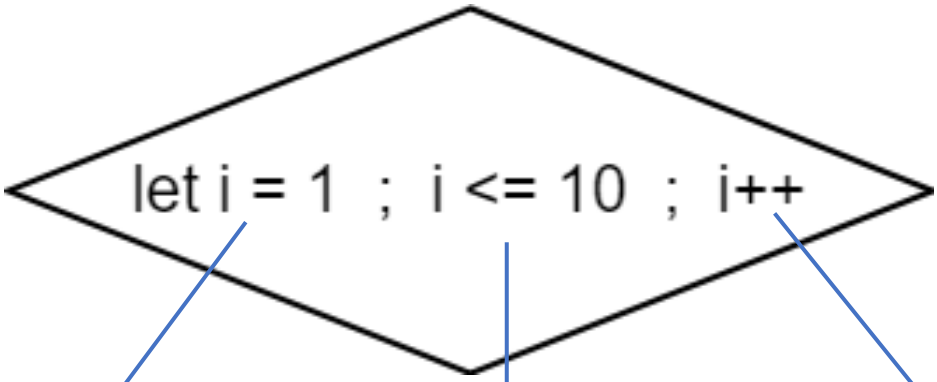
Indica hasta que número va a contar. Le dice al programa cuando terminar de contar.

Siempre usa la variable i para comprobar si es mayor o igual al número límite

Es la actualización. Recordemos que la variable debe ser actualizada con un i++ para que no tenga el mismo valor y no genere un bucle infinito.

```
for( inicialización ; condición ; operador_de_aumento ){
    /*
    Bloque de código
    que va a repetir
    */
}
```

Este sería un ejemplo de cómo se rellenan los paréntesis



Básicamente le digo al programa que comience a contar desde 1

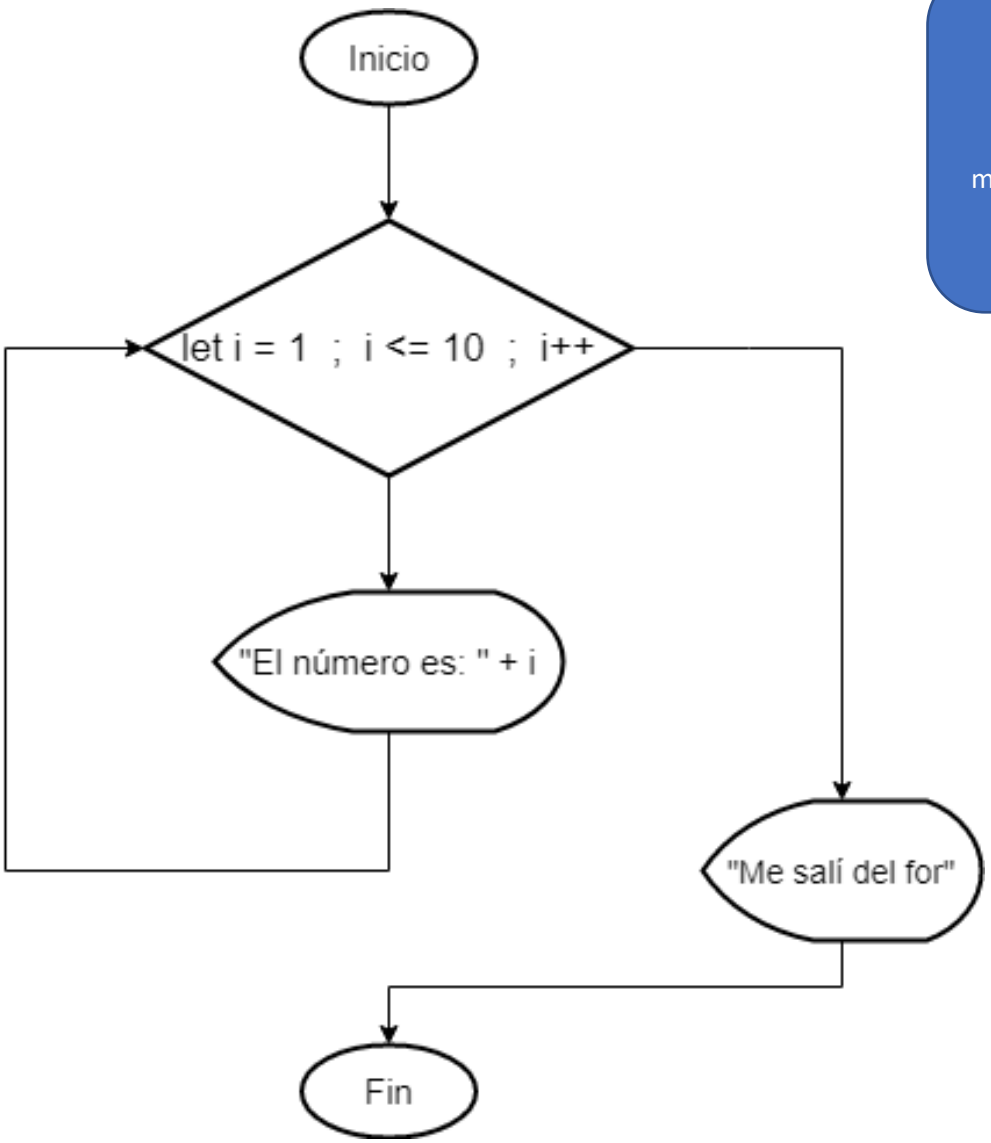
En la segunda parte le digo que termine de contar cuando la variable i sea mayor a 10

En la tercera parte le digo que al final de cada cuenta, vaya sumándole un 1 a la variable, para que vaya incrementando su valor hasta llegara 10

Ya con estas dos partes juntas, el programa sabe que va a contar desde 1 hasta 10

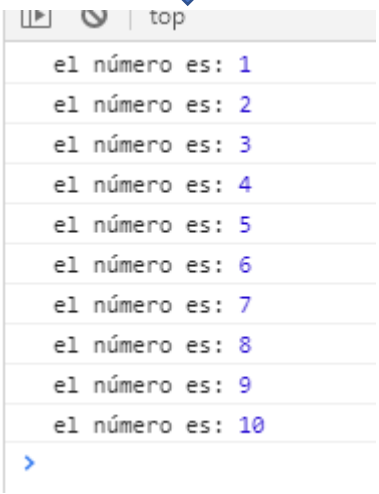
Ejemplo

Veamos un diagrama del ejemplo del contador que cuenta número del 1 al 10.



En la primera repetición va a mostrar 1, porque i = 1.

En la segunda repetición va a mostrar 2, porque se le incrementó el valor. Ahora i = 2



Código del ejemplo

Comienza a contar desde 1

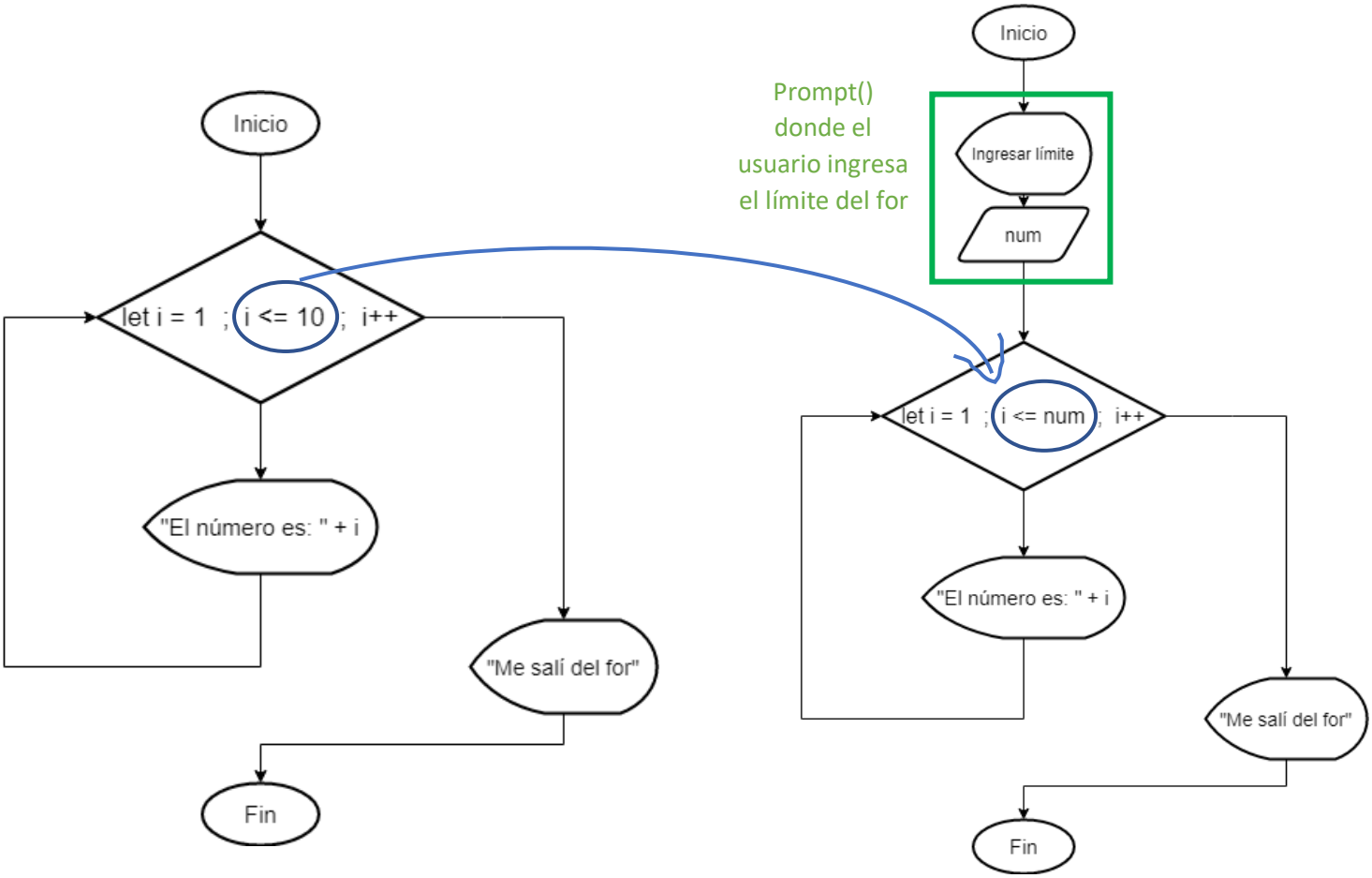
Y termina de contar cuando llegue a 11 (ya que 11 es mayor a 10)

Al final del for (en este caso, después del console.log) incrementa de labor la variable i.

La tercera parte es la que hace el conteo

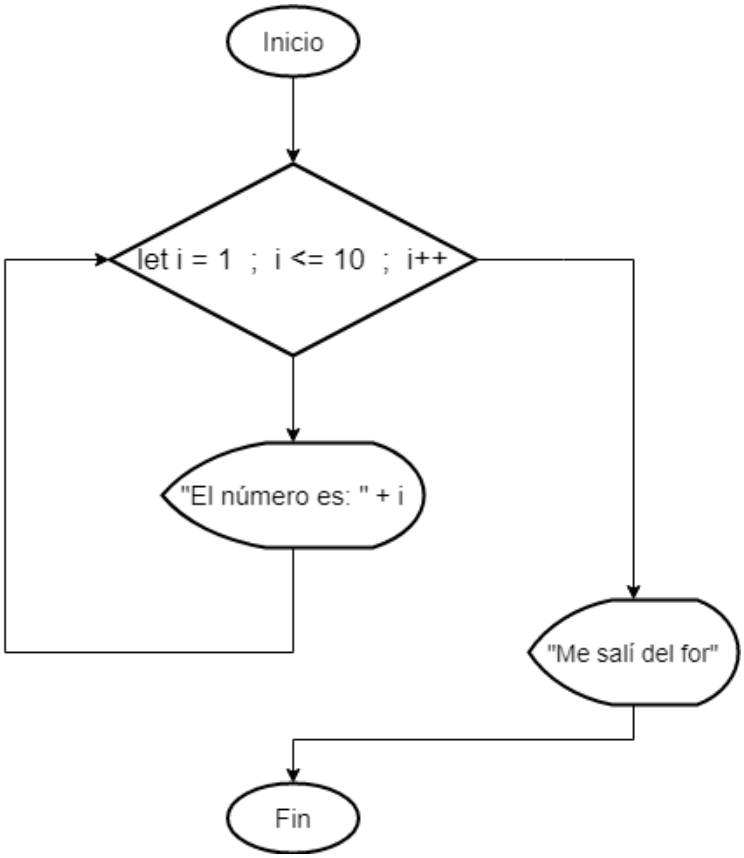
```
for(let i = 1 ; i <= 10 ; i++){
    console.log("el número es:", i);
}
```

En el caso en el que queramos que el usuario nos diga hasta donde quiere el conteo de número. Hacemos que lo escriba por teclado y ese número que ingresa el usuario lo guardamos en una variable (variable que puede ser llamada num, por ejemplo) y cambiamos la condición del for poniendole la variable límite que ingresó el usuario.

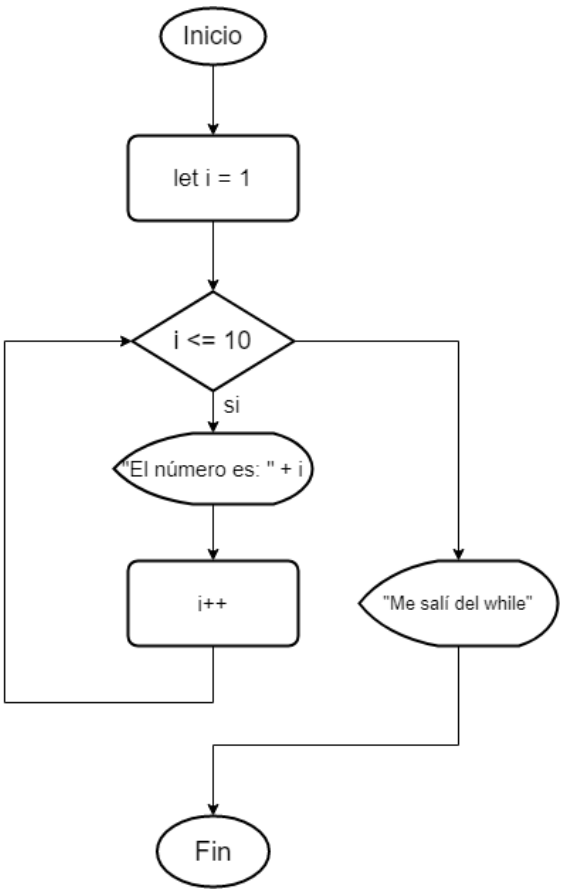


Como pudimos dije anteriormente, se puede implementar un while como si fuera un for. De hecho, el ejemplo del contador se puede hacer con un while.

Con for



Con while



Como podemos ver, el while es un poco más largo que el for. Por eso se suele decir que el for le gana en estos ejemplos.

Podemos rescatar que, en el while, primero creamos la variable i y al finalizar cada repetición del while debemos incrementar la variable i. En el for hacemos la creación de la variable y el incremento dentro de los paréntesis.

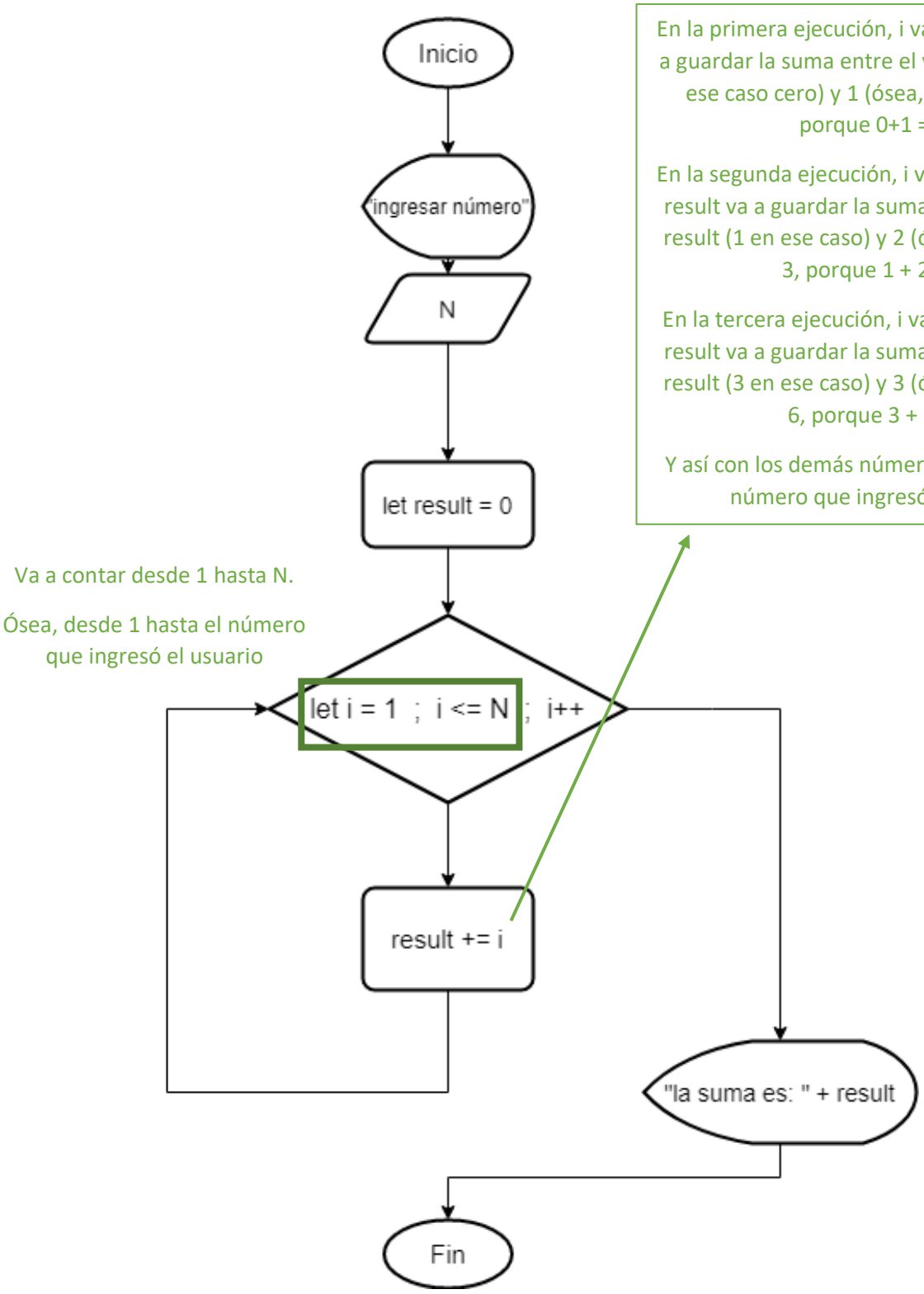
Ejercicio: Hacer un programa que solicite al usuario un número N y muestre el valor de la suma de los números que van del 1 al N (1+2+3+4+5+…+N)

Ejemplo:

-Si El usuario ingresa 6 (N = 6), se hace 1+2+3+4+5+6

-Si el usuario ingresa 8 (N = 8), se hace 1+2+3+4+5+6+7+8

Vamos a hacer el mismo ejercicio del while, pero esta vez con un for.



En la primera ejecución, i vale 1 y en result va a guardar la suma entre el valor de result (en ese caso cero) y 1 (ósea, va a guardar 1, porque 0+1 = 1).

En la segunda ejecución, i vale 2, entonces en result va a guardar la suma entre el valor de result (1 en ese caso) y 2 (ósea, va a guardar 3, porque 1 + 2 = 3).

En la tercera ejecución, i vale 3, entonces en result va a guardar la suma entre el valor de result (3 en ese caso) y 3 (ósea, va a guardar 6, porque 3 + 3= 6)

Y así con los demás números, hasta llegar al número que ingresó el usuario

Va a contar desde 1 hasta N.

Ósea, desde 1 hasta el número que ingresó el usuario