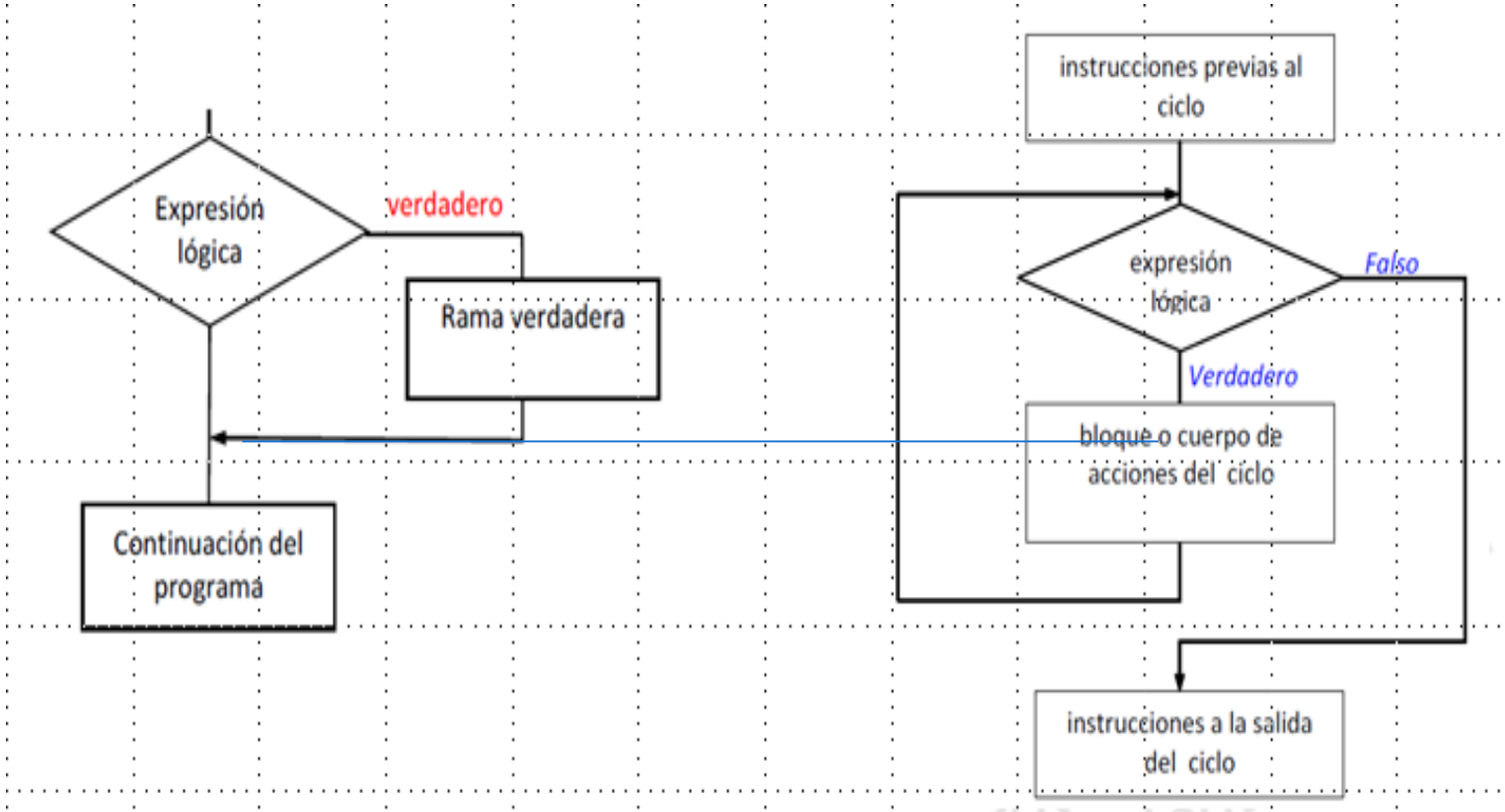
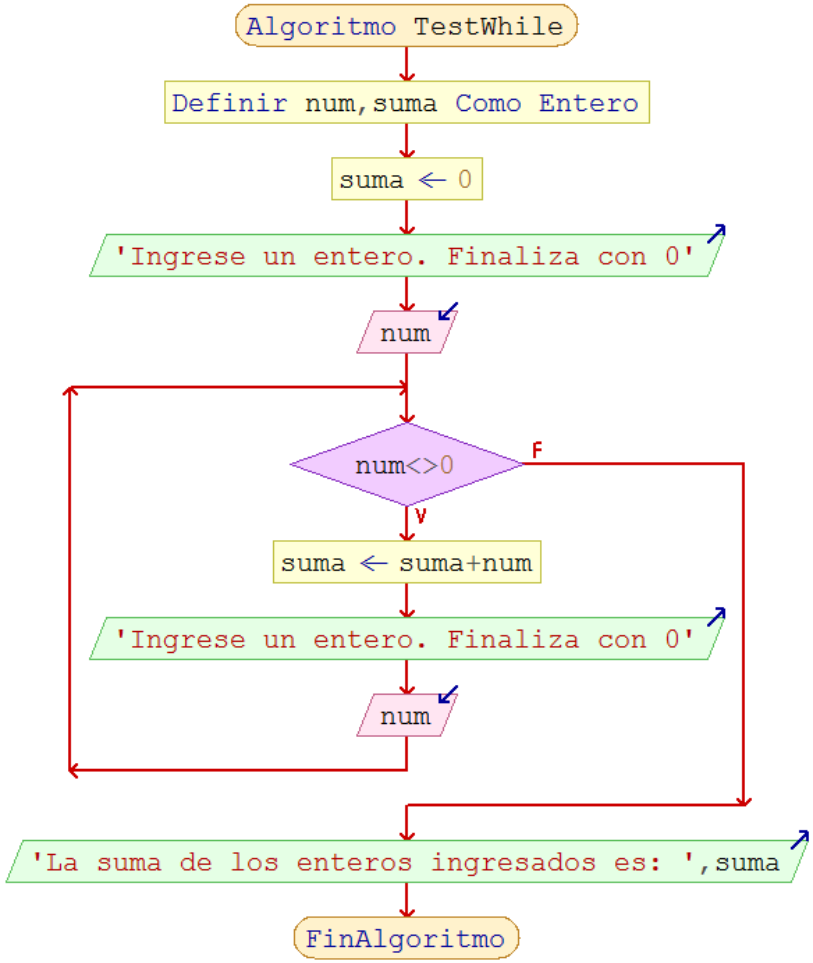
Ciclo while (Mientras)



En **JavaScript**:

```
while(condicion){
}
```

Ciclo while (Mientras)



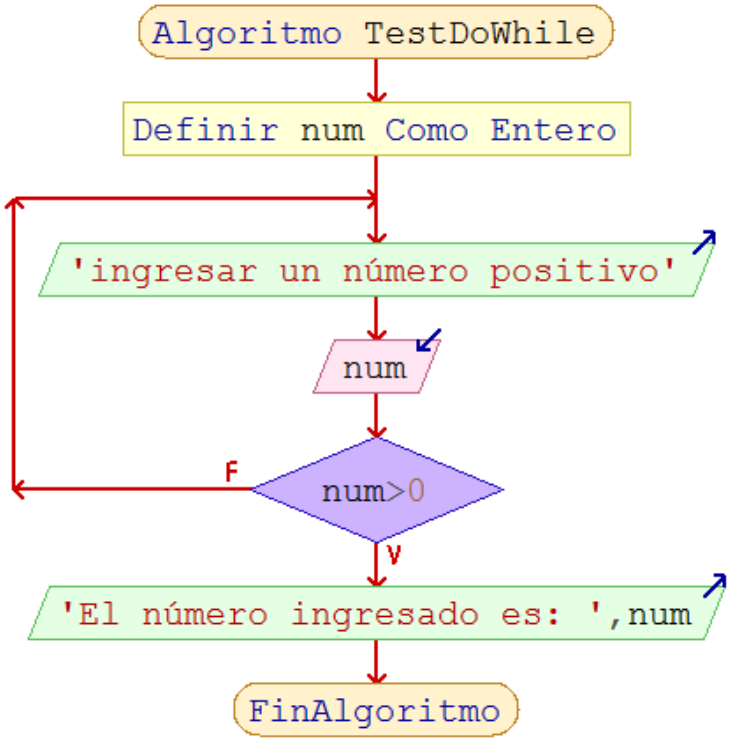
En Pseudocódigo:

```
Algoritmo TestWhile
        Definir num,suma Como Entero
        suma <- 0
        Escribir 'Ingrese un entero. Finaliza con 0'
        Leer num
        Mientras num<>0 Hacer
                suma <- suma+num
                Escribir 'Ingrese un entero.
                Finaliza con 0'
                Leer num
        FinMientras
        Escribir 'La suma de los enteros ingresados
        es: ',suma
FinAlgoritmo
```

Ciclo while (Mientras)

- Se lo conoce como **ciclo 0-N** porque puede no ejecutarse (si la condición es falsa inicialmente) o ejecutarse **N veces** mientras la condición es verdadera.
- Se utiliza cuando **no conocemos de antemano** cuántas veces se repite el bloque de acciones que forma el ciclo.
- Tener presente que para la carga de datos, el ciclo es un **ciclo de doble lectura**
- Cuidado con los **ciclos infinitos**!

Ciclo do-while (hacer-mientras)




```
Algoritmo TestDoWhile
        Definir num Como Entero
        Repetir
                Escribir 'ingresar un
                número positivo'
                Leer num
        Hasta Que num>0
        Escribir 'El número ingresado es:
',num
FinAlgoritmo
```

Ciclo do-while (hacer-mientras)

- Se lo conoce como **ciclo 1-N** porque puede se ejecuta al menor 1 vez o bien puede ejecutarse **N veces** mientras la condición es verdadera.

- Se utiliza cuando **no conocemos de antemano** cuántas veces se repite el bloque de acciones que forma el ciclo.

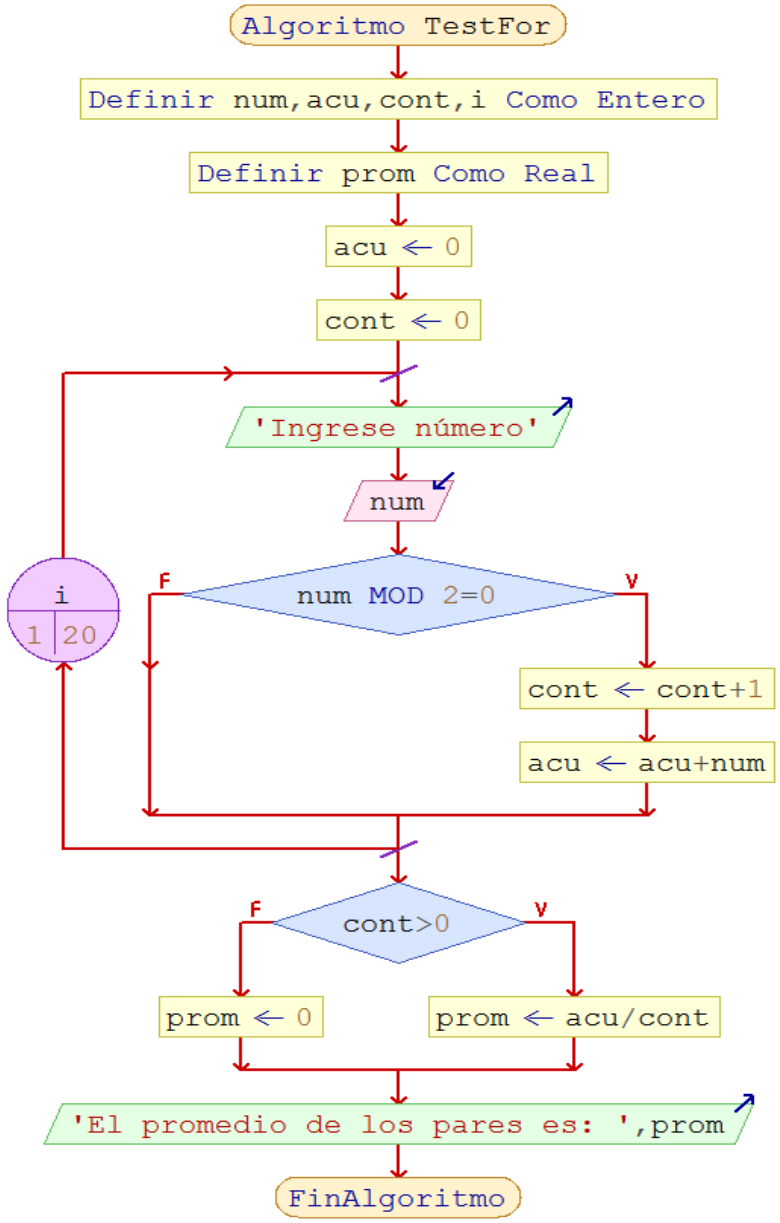
- Lo vamos a utilizar **principalmente para**:
  - **validar datos de entrada**
  - mostrar un **menú** de opciones

En **JavaScript**:

```
do{
} while(condicion);
```

Ciclo for (hasta)

```
Algoritmo TestFor
        Definir num,acu,cont,i Como Entero
        Definir prom Como Real
        acu <- 0
        cont <- 0
        Para i<-1 Hasta 20 Hacer
                Escribir 'Ingrese número'
                Leer num
                Si num MOD 2=0 Entonces
                        cont <- cont+1
                        acu <- acu+num
                FinSi
        FinPara
        Si cont>0 Entonces
                prom <- acu/cont
        SiNo
                prom <- 0
        FinSi
        Escribir 'El promedio de los pares es:
        ',prom
FinAlgoritmo
```



Ciclo for(para)

- Ciclo **N**: si la condición de corte está exclusivamente vinculada a validar que la variable de control llegue a un valor límite, el ciclo se ejecuta **exactamente N veces**.

- El ciclo for comúnmente se usa **cuando se conoce de antemano la cantidad exacta de repeticiones** que deben realizarse.

En **JavaScript**:

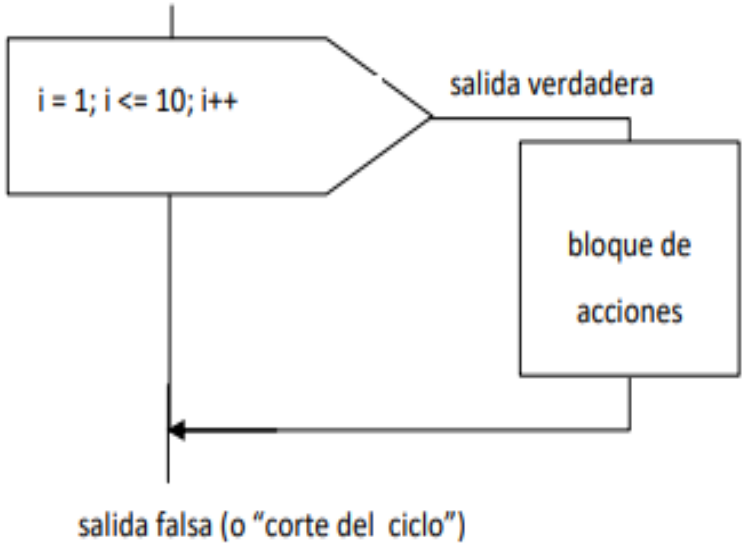
```
for(let i=0; i<10; i++){
//cuerpo del ciclo
}
```

3.4 Ciclo for(para)

Sintaxis:

```
          1      2      4
for ( i=1; i<=10; i++)
{
     3  bloque de acciones
}
```

Diagrama



(1) Inicialización: indica el valor inicial de la/s variables de control del ciclo.
(2) Condición de corte: permite controlar la ejecución del ciclo.
(4) Incremento: se indica la forma en que cambiará el valor de cada variable de control del ciclo.

3.4 Ciclo for(para)

Se inicia fuera

a se incrementa de 2

Condición **compuesta**

Sin cuerpo!

1)
```
let c = 0;
for ( ;  c != 20; c++) ----------;
```

2)
```
let a;
for (a = 1; a <= 100; a += 2) ---------;
```

3)
```
for (a = 0;  a < 10 && b != 20; a++)
{
      --------;
      b = //leer valor de la variable
}
```

4)
```
let i;
for (i = 1; i<=10; i++);
```

Casos especiales

Ciclo for (hasta)

Vamos a practicar

Caso práctico 04

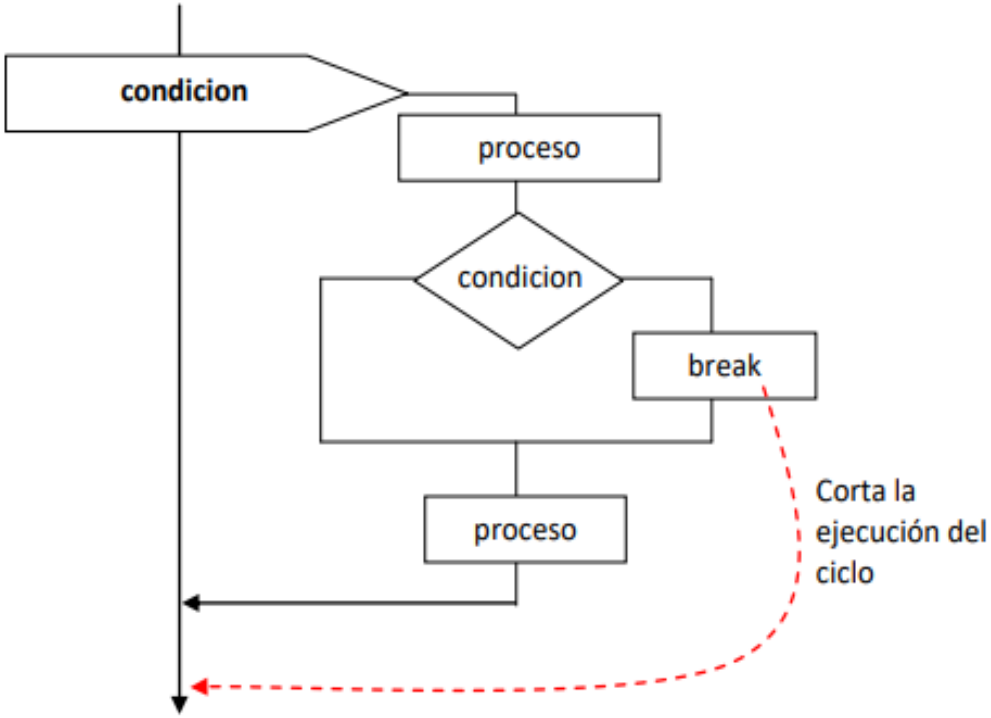
- Se pretende desarrollar un script que permita procesar los **n** datos de temperaturas recibidos desde una central meteorológica (siendo **n** un valor entero generado aleatoriamente en un rango [1:100]). Luego de informar por consola:

Amplitud térmica (temp. max – temp. mínima)

Temperatura promedio

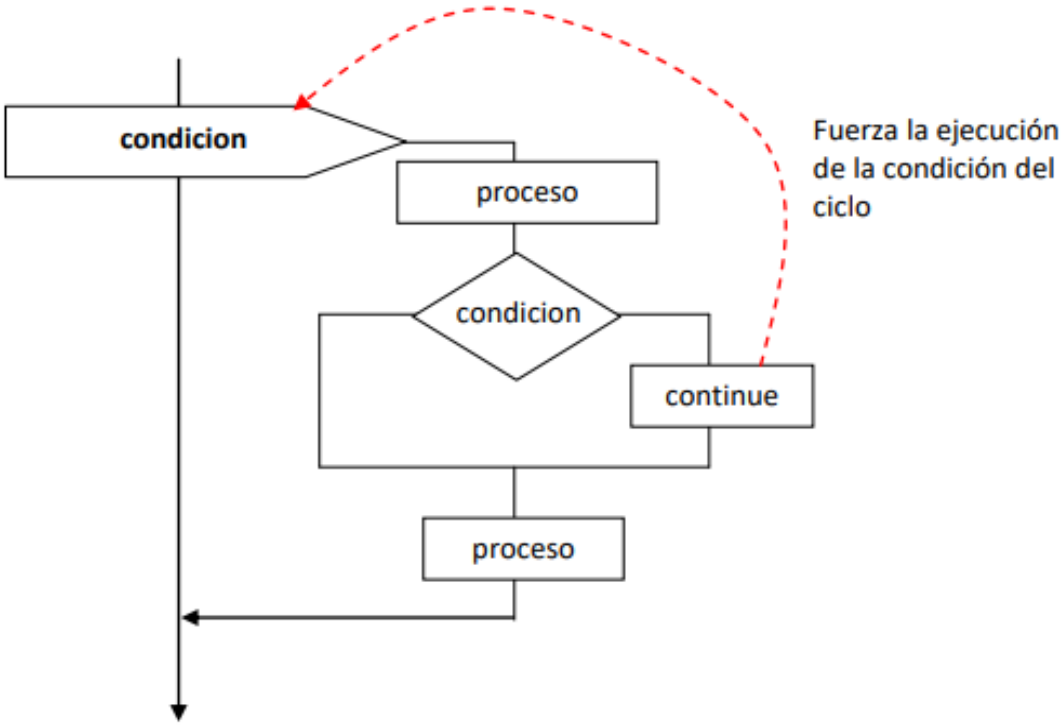
Conociendo un valor **histórico de 43 grados**, determinar si al menos una lectura recibida supera este valor de referencia

3.5 Elementos de control: break y continue



El bloque de acciones de un ciclo (while o for) en Java se puede incluir una instrucción **break** para **cortar el ciclo de inmediato sin retornar a la cabecera** para evaluar la expresión lógica de control.

3.5 Elementos de control: break y continue



De forma similar, pero a la inversa, un ciclo cualquiera puede incluir una instrucción **continue** para **forzar una repetición del ciclo sin terminar de ejecutar las instrucciones** que queden por debajo de la invocación a **continue**.