

# Técnicas de Programación

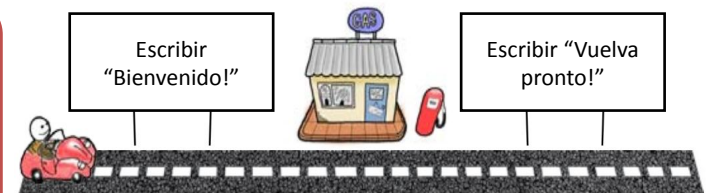
## Carrera Programador full-stack

*Selección (Repaso)*

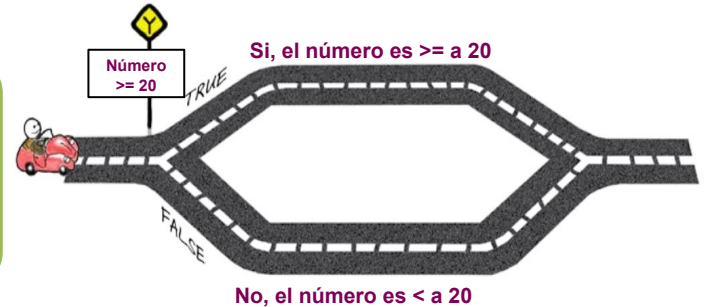
# Estructura de Control

## Selección

Secuenciales



Selección o de Decisión



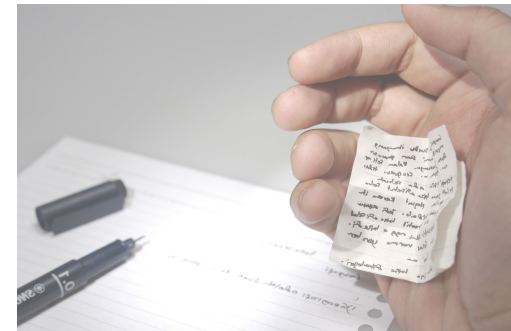
Repetitivas

# Estructura de Control

## *Selección Simple y Múltiple*

### Selección Simple

```
if (condición) {  
    <instrucciones>  
} else {  
    <instrucciones>  
}
```



### Alternativa Múltiple

```
switch <variable>{  
    case <número1>:  
        <instrucciones>;  
        break;  
    case <número2>:  
        <...>  
    default: <instrucciones>  
}
```

# Prueba de Escritorio

- Técnica utilizada para validar la resolución de problemas con algoritmos, de uso frecuente en el ámbito informático
- Sirve para validar utilizando datos reales como ejemplo, un algoritmo definido y así comprobar si se obtiene el resultado deseado
- Ejemplo, recuerde el ejercicio de verificar si un número es mayor a 20. Se podría verificar con un número mayor a 20, un número igual a 20 y un número menor que 20

# Prueba de Escritorio

## *Ejercicio - Mayor a 20*

Código	Datos Entrada	Respuesta Deseada
<pre>let nroDeseado : number = rls.questionInt("Escriba el número que desea verificar si es mayor o no a 20: ");  if (nroDeseado &gt; 20) {     console.log('El número es mayor a 20: ' + nroDeseado ); } else {     console.log('El número es menor o igual a 20: ' + nroDeseado ); }</pre>	nroDeseado = 20	El número es menor o igual a 20: 20
	nroDeseado = 3	El número es menor o igual a 20: 3
	nroDeseado = 45	El número es mayor a 20: 45

# Técnicas de Programación

## Carrera Programador full-stack

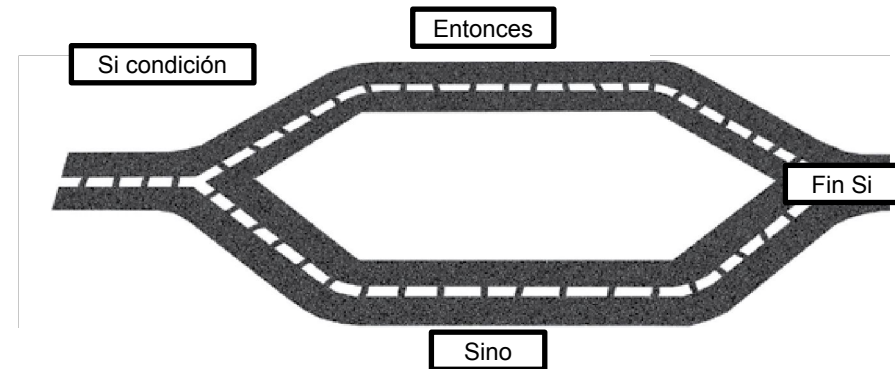
*Repetición (Conceptos)*

# Estructuras de Control

Secuenciales

Selectivas o De  
Decisión

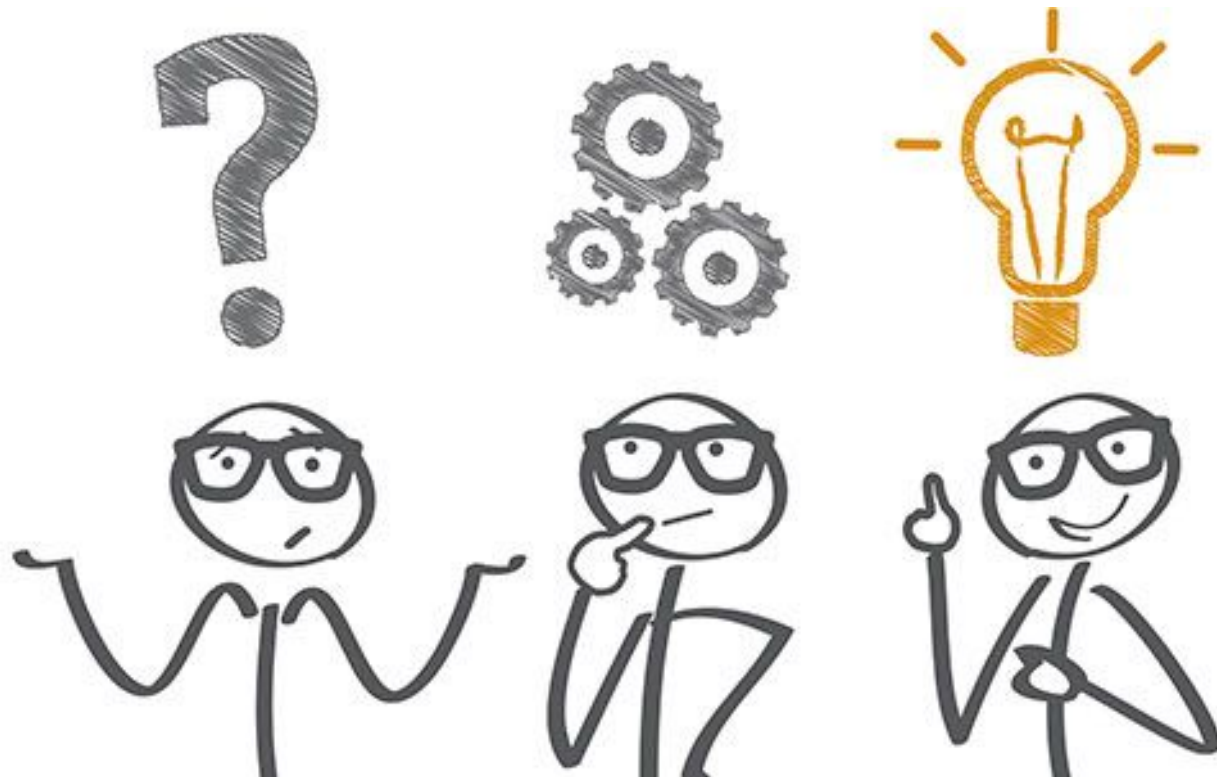
Repetitivas



# Estructuras de Control

*Decir 5 Veces “Hola”*

- Escriba un programa que salude 5 veces por pantalla de forma secuencial





# Estructuras de Control

*Decir 5 Veces "Hola"*

```
console.log("Hola");  
console.log("Hola");  
console.log("Hola");  
console.log("Hola");  
console.log("Hola");
```

¿Es práctico repetir  
las instrucciones  
explícitamente?

# Estructuras de Control

*Calcular el Promedio de 10 Notas*

- Escriba un programa que solicite 10 números al usuario y calcule el promedio de las mismas. Luego, muestre el resultado por pantalla.

EVALUACIÓN ASIGNATURA	EVALUACIÓN				CALIFICACIÓN FINAL
	SEPT.	OCT.	NOV.	DIC.	
GEOM. ANALÍTICA Y FUN.	10	10	10	10	10
TALLER DE LECT. Y RED. III	9	10	10	10	10
HISTORIA DE MÉXICO I	10	10	10	10	10
ORI. VOCACIONAL	10	10	10	10	10
FÍSICA I	10	10	10	10	10
INFORMÁTICA	10	10	10	10	10
INGLÉS III	9	10	10	10	10
CONTABILIDAD BÁSICA	10	10	10	10	10
ADMINISTRACIÓN I	10	10	10	10	10
QUÍMICA	10	10	10	10	10
EDUCACIÓN FÍSICA	10	10	10	10	10
FORMACIÓN EN VALORES	10	10	10	10	10
EDUCACIÓN EN LA FE	9	9	9	9	9
HÁBITOS PERSONALES	10	10	10	10	10
INASISTENCIAS HORA/CLASE					
FIRMA DEL PADRE O TUTOR	OCT. - NOV.				
SEPT.				DIC.	

$$\bar{x} = \frac{\sum x}{n}$$

# Estructuras de Control

## *Calcular el Promedio de 10 Notas*

```
import * as rls from 'readline-sync';
```

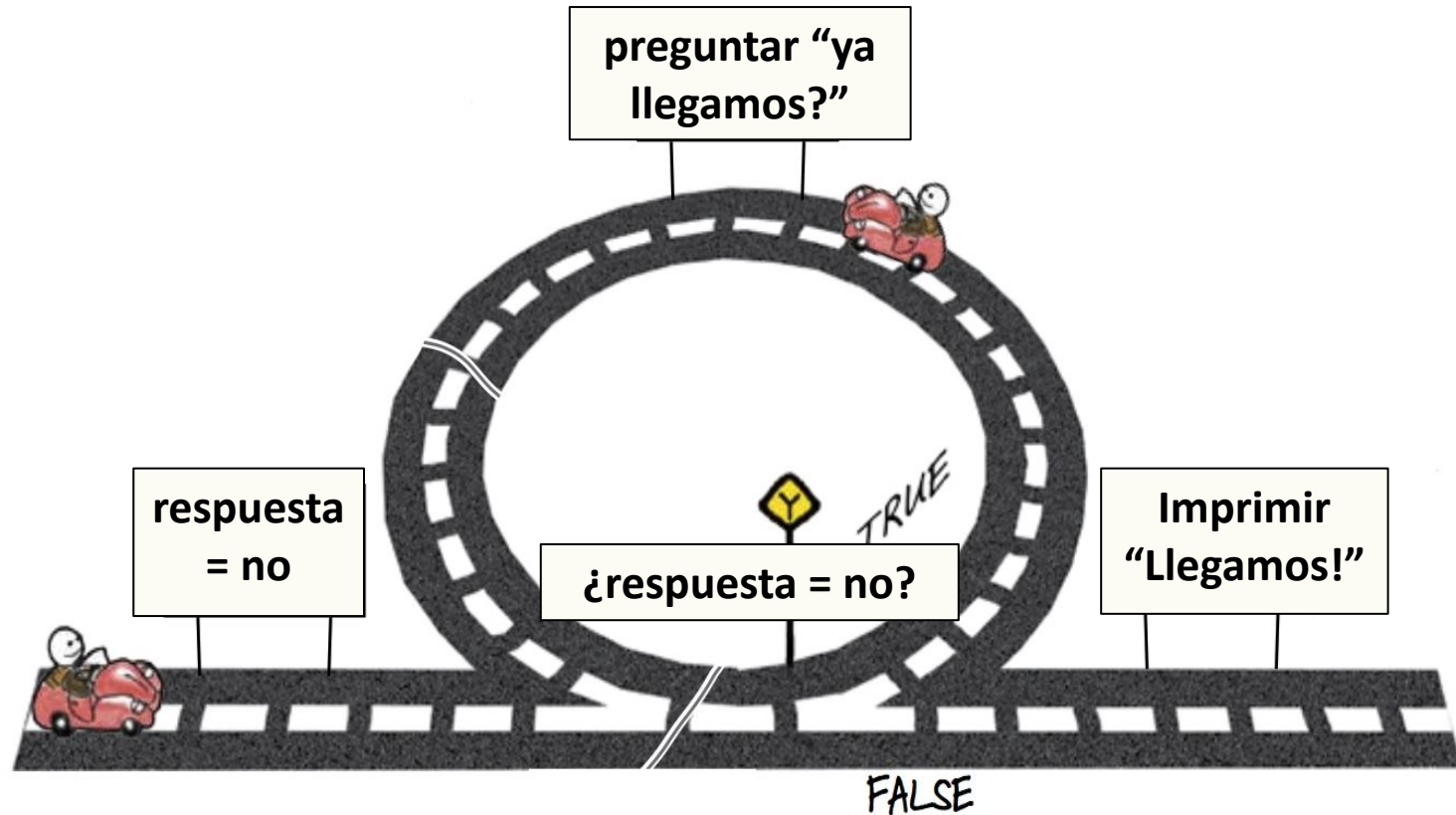
```
let nota1 : number = rls.questionInt("Ingrese la nota 1: ");  
let nota2 : number = rls.questionInt("Ingrese la nota 2: ");  
let nota3 : number = rls.questionInt("Ingrese la nota 3: ");  
let nota4 : number = rls.questionInt("Ingrese la nota 4: ");  
let nota5 : number = rls.questionInt("Ingrese la nota 5: ");  
let nota6 : number = rls.questionInt("Ingrese la nota 6: ");  
let nota7 : number = rls.questionInt("Ingrese la nota 7: ");  
let nota8 : number = rls.questionInt("Ingrese la nota 8: ");  
let nota9 : number = rls.questionInt("Ingrese la nota 9: ");  
let nota10 : number = rls.questionInt("Ingrese la nota 10: ");  
let total : number = nota1+nota2+nota3+nota4+nota5+nota6+nota7+nota8+nota9+nota10;  
let promedio : number = total/10;  
console.log("El promedio de las notas es: " + promedio);
```

¿Puede hacerse  
**más corto** este  
código?

Es bastante  
tedioso...

# Estructuras de Control

## *Repetición*

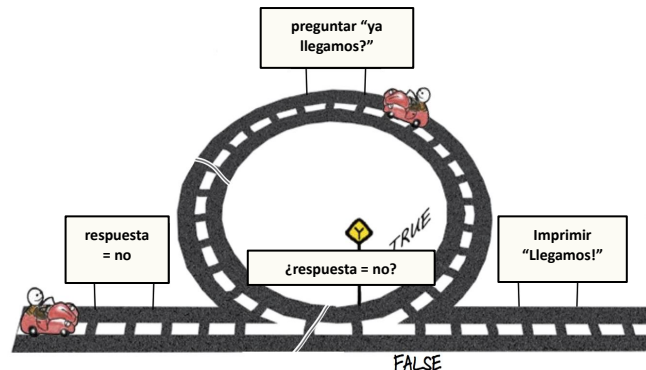


# Comparativa code.org



Recuerden lo que hacíamos en code.org al poner un bloque **mientras**. Lo que está dentro del bloque es lo que va entre llaves.

```
while (hayCaminoAdelante) {  
    console.log("Avanzar");  
}
```



# Estructuras de Control

## *Instrucción **While***

- La instrucción **while** ejecuta una secuencia de instrucciones mientras una condición sea verdadera
  - También llamados iteraciones o “loops” en Inglés
  - Sirven para ejecutar código varias veces
  - La condición se verifica al principio
  - La cantidad de veces ejecutado depende de una condición (puede que no se ejecute ninguna vez)



# Estructuras de Control

## *Instrucción **While***

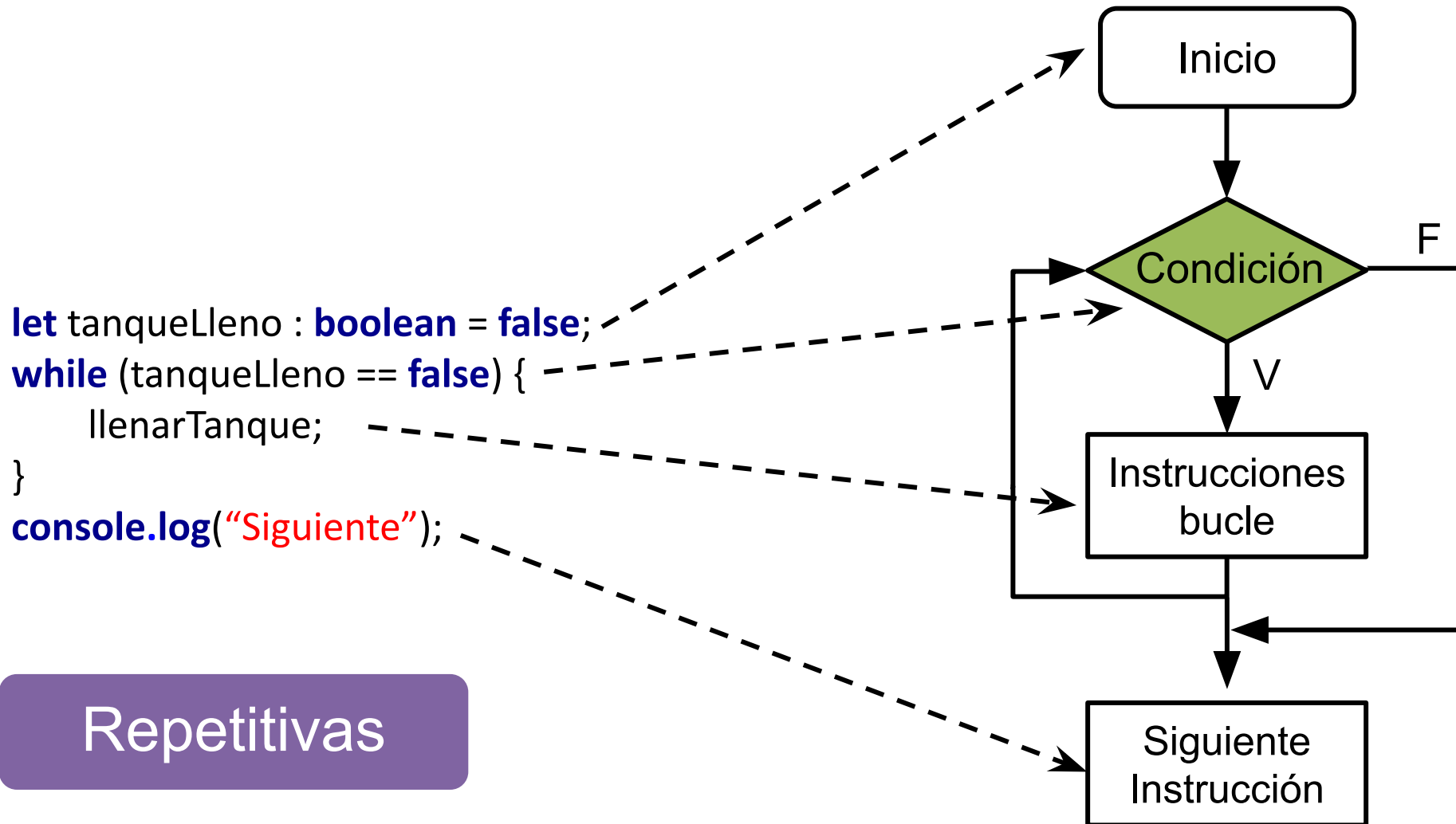
- La instrucción **while** ejecuta una secuencia de instrucciones mientras una condición sea verdadera



```
while (<condición>) {  
    <instrucciones>  
}
```

# Estructuras de Control

## *Instrucción Mientras*





# Estructuras de Control

*Decir 5 Veces "Hola"*

```
let cantHolas : number = 1;  
while (cantHolas <= 5) {  
    console.log("Hola");  
    cantHolas = cantHolas + 1;  
}
```

Usamos la variable  
*cantHolas* como  
contador

Se repite hasta que  
*cantHolas* sea  
mayor que 5

En cada iteración se  
incrementa en 1

# Estructuras de Control

## *Calcular el Promedio de 10 Notas*

```
import * as rls from 'readline-sync';

let nota, suma, promedio, contador : number;
contador = 1;
suma = 0;
while (contador <= 10) {
    nota = rls.questionInt("Ingrese una nota: ");
    suma = suma + nota;
    contador = contador + 1;
}
promedio = suma / 10;
console.log("El promedio de las notas es: " + promedio);
```

Usamos la variable *suma* como acumulador de las notas

La variable *suma* se inicializa en cero

En cada iteración se suma la nota ingresada por el usuario en la variable

# Estructuras de Control

## *Simular la Espera de un Colectivo*

- Cuando llegamos a la parada, miramos si el colectivo arribó a la parada
- Siempre tenemos que esperar antes de que llegue



# Estructuras de Control

## *Simular la Espera de un Colectivo*



```
import * as rls from 'readline-sync';

let llegadaColectivo : string;
console.log("Esperando el colectivo");
llegadaColectivo = rls.question("Llegó el colectivo? (S/N): ");
while (llegadaColectivo=="N") {
    console.log("Esperando el colectivo");
    llegadaColectivo = rls.question("Llegó el colectivo? (S/N): ");
}
console.log("Llegó el colectivo");
```

# Comparativa code.org



Recuerden lo que hacíamos en code.org al poner un bloque **contar**. Lo que está dentro del bloque es lo que va entre llaves.

```
for (let contador : number = 50; contador <= 150; contador+=50 ) {  
    console.log("Dibujar una casa. Longitud:",contador);  
}
```



# Estructuras de Control

## *Instrucción For*


- La instrucción **for** ejecuta una secuencia de instrucciones utilizando contadores con principio, incrementos y final
  - Son útiles cuando hay que hacer un conteo (fijo)
  - El valor inicial del conteo, el valor final de corte y los se definen en una sola instrucción
  - La declaración de la variable debe realizarse antes



# Estructuras de Control

## *Instrucción For*

- La instrucción **for** ejecuta una secuencia de instrucciones utilizando contadores con principio, incrementos y final




```
for (num=inicial; <condición> ; incremento) {  
    <instrucciones>  
}
```

# Estructuras de Control

## *Instrucción For*

- **num**: la variable que se va a usar de contador
- **inicial**: el valor de *num* desde el cual se comenzará a iterar
- **condición**: La sentencia que define si se debe seguir iterando o si ya se llegó al final del recorrido.
- **incremento**: la modificación que se debe realizar a *num* en cada iteración



```
for (num=inicial; <condición> ; incremento) {  
    <instrucciones>  
}
```



# Estructuras de Control

## *Instrucción For*

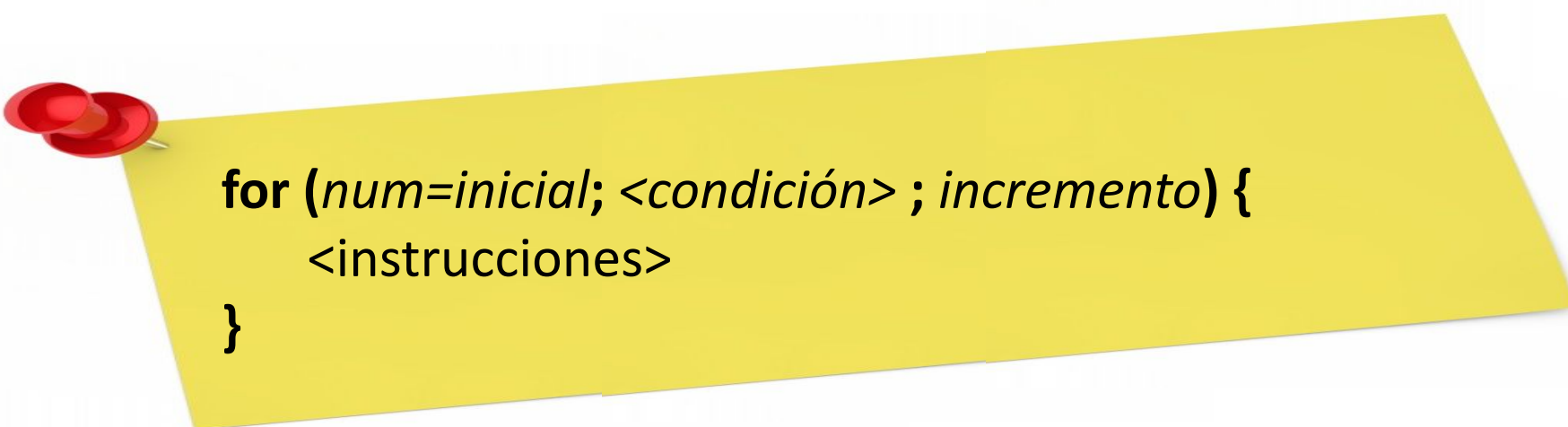
```
for (let rueda : number = 1; rueda < 4; rueda++ ) {  
    console.log("Inflar Rueda");  
}
```

Repetitivas

# Estructuras de Control

*Calcular el Promedio de 10 Notas*

Retomar el ejercicio del promedio de 10 notas y resuélvalo utilizando la estructura de control **for**



```
for (num=inicial; <condición> ; incremento) {  
    <instrucciones>  
}
```

# Estructuras de Control

## *Calcular el Promedio de 10 Notas*

```
import * as rls from 'readline-sync';

let nota, suma, promedio, contador : number;

suma=0;

for (contador=1; contador<11; contador++) {
    nota = rls.questionInt("Ingrese la nota " + contador + ": ");
    suma +=nota;
}

promedio = suma/10;

console.log("El promedio de las notas es: " + promedio);
```

Si necesitamos repetir un conjunto de instrucciones por un número predeterminado de veces, la instrucción **for** es muy útil

Nos evitamos inicializar e incrementar el contador, ya que es parte del Para

# Estructuras de Control

## *Calcular el Promedio de 10 Notas*

```
import * as rls from 'readline-sync';
```

Esta variable se declara aquí y se inicializará dentro del **For**

```
let nota, suma, promedio, contador : number;
```

```
suma=0;
```

```
for (let contador : number =1; contador<11; contador++) {
```

```
    nota = rls.questionInt("Ingrese la nota " + contador + ": ");
```

```
    suma +=nota;
```

```
}
```

```
promedio = suma/10;
```

```
console.log("El promedio de las notas es: " + promedio);
```

Tener en cuenta que el contador va desde 1 (inclusive) hasta 10 (inclusive)

# Técnicas de Programación

## Carrera Programador full-stack

*Repetición (Ejercicios)*

# Estructuras de Control

## *Eureka*

- Escribir un algoritmo que nos pida una clave y verifique que sea la correcta
- Tenga en cuenta que la clave es la palabra “eureka”
- Solo tenemos 3 intentos para acertar, si fallamos los 3 intentos el sistema mostrará un mensaje indicándonos que hemos agotado todas las oportunidades
- Si acertamos la clave, saldremos directamente del programa



# Estructuras de Control

## *Múltiplos*

- Cree un algoritmo que visualice los números que son múltiplos de 2 o de 3 que hay entre 1 y 100
- Tener en cuenta que hay números que son múltiplos de 2 y de 3 al mismo tiempo
- En dichos casos, solamente indique el número una vez



# Estructuras de Control

## *Par/Impar*

- Realizar un algoritmo que dado un número entero ingresado por el usuario, visualice en pantalla si es par o impar
- En el caso de ingresar un cero, se debe volver a pedir el número por teclado (hasta que se ingrese un número mayor que cero)





# Estructuras de Control

## *Suma entre Números*

- Escriba un programa que pida al usuario dos números enteros, y luego retorne la suma de todos los números que están entre ellos
- Por ejemplo, si los números son 2 y 7, debe entregar como resultado  $2 + 3 + 4 + 5 + 6 + 7 = 27$

Ingrese num: 2  
Ingrese num: 7  
La suma es 27



# Estructuras de Control

## *Tablas de Multiplicación*

- Diseñar un algoritmo que muestre por pantalla la tabla de multiplicación del número ingresado por el usuario
- Para definir hasta qué número desea que muestre la tabla de multiplicación, el usuario también deberá ingresar dicho valor

Ingrese el número: 9  
 Ingrese hasta qué número: 4  
 $9 \times 1 = 9$   
 $9 \times 2 = 18$   
 $9 \times 3 = 27$   
 $9 \times 4 = 36$

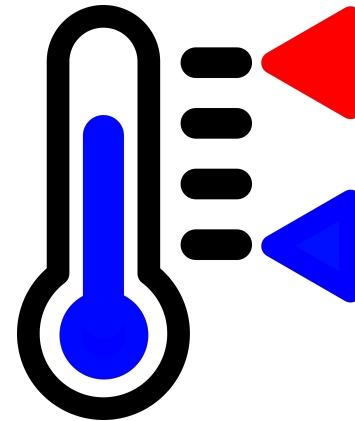
X	1	2	3	4	5	6	7	8	9	10	11	12
1	1	2	3	4	5	6	7	8	9	10	11	12
2	2	4	6	8	10	12	14	16	18	20	22	24
3	3	6	9	12	15	18	21	24	27	30	33	36
4	4	8	12	16	20	24	28	32	36	40	44	48
5	5	10	15	20	25	30	35	40	45	50	55	60
6	6	12	18	24	30	36	42	48	54	60	66	72
7	7	14	21	28	35	42	49	56	63	70	77	84
8	8	16	24	32	40	48	56	64	72	80	88	96
9	9	18	27	36	45	54	63	72	81	90	99	108
10	10	20	30	40	50	60	70	80	90	100	110	120
11	11	22	33	44	55	66	77	88	99	110	121	132
12	12	24	36	48	60	72	84	96	108	120	132	144

# Estructuras de Control

## *Encontrar el Número Máximo*

- Leer valores hasta que se introduzca un cero (0)
- El usuario puede introducir valores positivos y negativos
- Encontrar el máximo de los elementos que se introdujeron
- Analizar cómo cambia el programa para hallar el mínimo

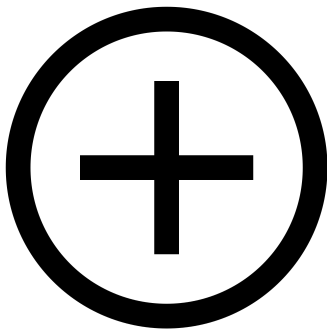
Ingrese número: 9  
Ingrese número: 7  
Ingrese número: -1  
Ingrese número: 1  
Ingrese número: 0  
El máximo es 9



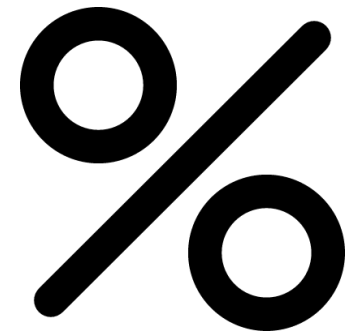
# Estructuras de Control

## *Cantidad y Distribución de Positivos*

- Leer valores del usuario hasta que introduzca un 0
- El usuario puede introducir valores numéricos, tanto positivos como negativos
- Contar la cantidad de valores introducidos que sean mayores a 0 y el porcentaje de positivos respecto del total



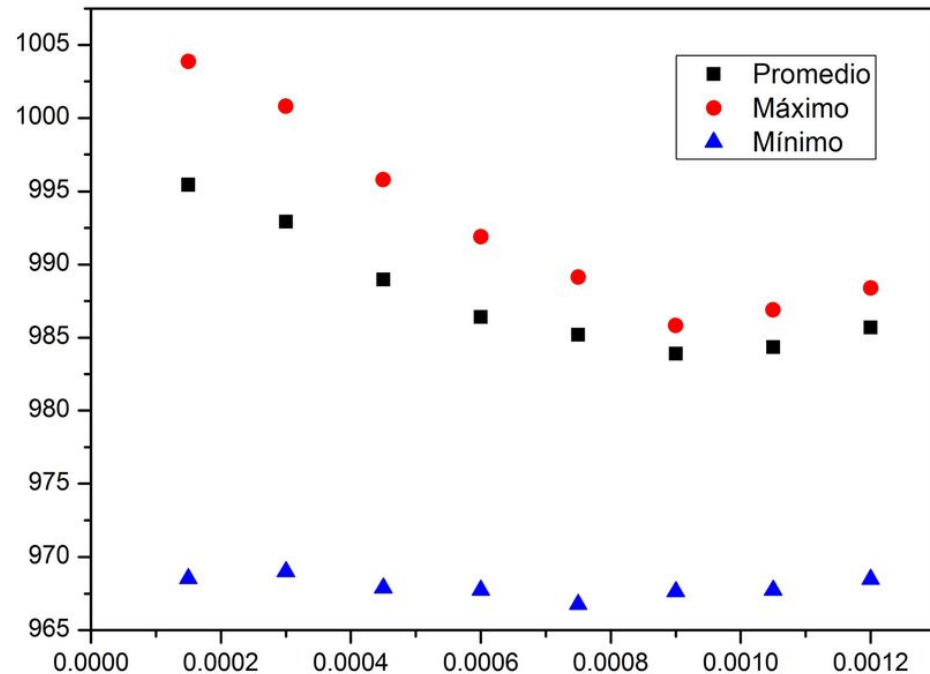
Ingrese número: 9  
Ingrese número: 7  
Ingrese número: -1  
Ingrese número: 1  
Ingrese número: 0  
3 positivos, 75% del total



# Estructuras de Control

## *Promedio-Máximo-Mínimo*

- Diseñar un algoritmo que lea números enteros hasta teclear 0
- Determinar y mostrar el máximo, el mínimo y la media de todos los números ingresados
- Pensar cuidadosamente cómo debemos inicializar las variables



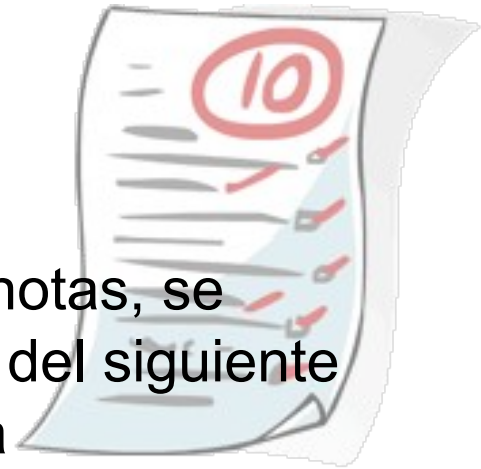
# Estructuras de Control

## *Calificaciones*

Calcular las calificaciones de un grupo de alumnos, donde la nota final de cada alumno se calcula según el siguiente criterio:

- la parte práctica vale el 10%
- la parte de problemas vale el 50%
- la parte teórica el 40%

Se debe ingresar el nombre del alumno y sus tres notas, se escribirá el resultado y se volverá a pedir los datos del siguiente alumno hasta que el nombre sea una cadena vacía



Las notas deben estar entre 0 y 10 (si no lo están, no imprimirá las notas, mostrará un mensaje de error y continuará con otro alumno)

# Estructuras de Control

## *Dados*

- Al tirar un dado tenemos  $1/6$  de probabilidades de sacar 6
- Si tiramos dos dados tenemos  $1/36$  probabilidades de sacar doble 6
- Al aumentar el número de dados la probabilidad de sacar todos 6 es cada vez menor
- Escriba un programa que calcule la probabilidad de sacar todos los dados 6 siendo que tiramos N dados (dato ingresado por el usuario)

