Conceptos básicos de lógica

En tu proceso de aprendizaje, ten muy presente los siguientes conceptos para aplicarlos en los ejercicios de lógica de programación o desarrollo.

CONCEPTO	EJEMPLO

Condicionales: Son estructuras de control que nos permiten tomar decisiones dentro de nuestro código. Es decir, nos permiten ejecutar diferentes bloques de código dependiendo de si se cumple o no una determinada condición.	`let edad = 18;`
	`if (edad >= 18) {`
	`  console.log("Eres mayor de edad.");`
	`} else {`
	`  console.log("Eres menor de edad.");`
	`}`
	
	`let edad = 20;`
	`if (edad < 18) {`
	`  console.log("Eres menor de edad.");`
	`}`
	`else if (edad >= 18 && edad < 65) {`
	`  console.log("Eres un adulto.");`
	`}`
	`else {`
	`  console.log("Eres una persona mayor.");`
	`}`

Condicional anidado: es una estructura de control donde un bloque `if-else` se encuentra dentro de otro bloque `if-else`.	`let edad = 20;`
	`if (edad >= 18) {`
	`  if (edad >= 65) {`
	`  console.log("Eres una persona mayor.");`
	`  } else {`
	`  console.log("Eres un adulto.");`
	`  }`
	
	`} else {`
	`  console.log("Eres menor de edad.");`
	`}`

Ciclo for: Es una estructura de control que permite repetir un bloque de código un número determinado de veces. Es especialmente útil cuando se sabe cuántas veces se desea ejecutar el ciclo.	`for (let i = 0; i < 5; i++) { console.log(i); }`

Ciclo While: Es una estructura de control que repite un bloque de código **mientras** una condición sea verdadera. Se utiliza cuando no se conoce de antemano cuántas veces se ejecutará el ciclo, pero se sabe que debe repetirse mientras una condición continúe cumpliéndose.	`let i = 0;`
	`while (i < 5) {`
	`  console.log(i);`
	`  i++;`
	`}`

Ciclo Do-While: Es una estructura de control que ejecuta un bloque de código **al menos una vez**, y luego sigue repitiendo el ciclo mientras una condición sea verdadera. La diferencia principal con el ciclo `while` es que el `do-while` evalúa la condición **después** de ejecutar el bloque de código, garantizando que se ejecute al menos una vez.	`let i = 0;`
	`do {`
	`  console.log(i);`
	`  i++;`
	`} while (i < 5);`

Ciclos anidados: son estructuras donde un ciclo se encuentra dentro de otro ciclo. Esto es útil para recorrer estructuras de datos bidimensionales o realizar operaciones repetitivas con múltiples niveles.	`for (let i = 1; i <= 3; i++) {`
	`  console.log("Fila " + i + ":");`
	`  for (let j = 1; j <= 3; j++) {`
	`  console.log("  Columna " + j);`
	`  }`
	`  console.log("------------------");`
	`}`

`break:` Se utiliza para salir de un ciclo de forma anticipada cuando se cumple una condición.	`for (let i = 0; i < 10; i++) {`
	`  if (i === 5) {`
	`  // Si 'i' es 5, usamos 'break' para salir del ciclo`
	`  break;`
	`  }`
	`  // Si no se cumple la condición, imprimimos el valor de 'i'`
	`  console.log("Número:", i);`
	`}`

`continue:` se utiliza para saltar la iteración actual del ciclo y pasar a la siguiente.	`for (let i = 0; i < 10; i++) {`
	`  if (i === 5) {`
	`  // Si 'i' es 5, usamos 'continue' para saltar esta iteración y no imprimir el 5`

	continue;
	}
	// Imprimimos el valor de 'i' solo si no es 5
	console.log("Número:", i);
	}

Condicional Sencillo (if):

Un condicional sencillo se utiliza para ejecutar una acción si una condición es verdadera. Si la condición es falsa, el código dentro del bloque if no se ejecuta.

Sintaxis:

```
if (condición) {

 // Código que se ejecuta si la condición es verdadera

}
```

Ejemplo

```
let edad = 18;

if (edad >= 18) {

 console.log("Eres mayor de edad");

}
```

En este ejemplo, si la variable `edad` es mayor o igual a 18, el mensaje "Eres mayor de edad" se imprimirá.

Condicional Compuesto (`if-else`, `else if`):

Un condicional compuesto permite ejecutar diferentes bloques de código dependiendo de varias condiciones.

Sintaxis:

```
if (condición1) {

 // Código que se ejecuta si la condición1 es verdadera
```

```
} else if (condición2) {

 // Código que se ejecuta si la condición2 es verdadera

} else {

 // Código que se ejecuta si ninguna de las condiciones anteriores es verdadera

}
```

Ejemplo:

```
let temperatura = 25;



if (temperatura > 30) {

 console.log("Hace mucho calor");

} else if (temperatura < 10) {

 console.log("Hace mucho frío");

} else {

 console.log("El clima es agradable");

}
```

En este ejemplo, dependiendo de la temperatura, se imprimirá un mensaje diferente.

Problema de ejemplo: Categoría en torneo de videojuegos

Escribe un programa que reciba la edad de una persona y determine su categoría en un torneo de videojuegos:

- Si tiene entre 12 y 15 años, pertenece a la categoría Junior.
- Si tiene entre 16 y 18 años, pertenece a la categoría Adolescente.
- Si tiene más de 18 años, pertenece a la categoría Adulto.
- Si tiene menos de 12 años, no puede participar.

Solución:

```
let edad = 17;



```

```
if (edad >= 12 && edad <= 15) {

 console.log("Categoría: Junior");

} else if (edad >= 16 && edad <= 18) {

 console.log("Categoría: Adolescente");

} else if (edad > 18) {

 console.log("Categoría: Adulto");

} else {

 console.log("No puede participar");

}
```

Problema de ejemplo: Clasificación de Peso Corporal

Se desea escribir un programa que clasifique a una persona según su índice de masa corporal (IMC). El IMC se calcula dividiendo el peso en kilogramos entre la altura en metros al cuadrado.

La clasificación es la siguiente:

- Si el IMC es menor a 18.5, la persona está **bajo de peso**.
- Si el IMC está entre 18.5 y 24.9, la persona tiene **peso normal**.
- Si el IMC está entre 25.0 y 29.9, la persona tiene **sobrepeso**.
- Si el IMC es mayor o igual a 30.0:
    - Si está entre 30.0 y 34.9, se considera **obesidad grado 1**.
    - Si está entre 35.0 y 39.9, se considera **obesidad grado 2**.
    - Si es 40.0 o más, se considera **obesidad grado 3**.

Solución con Condicionales Anidados:

```
let peso = 75;  // Peso en kilogramos
let altura = 1.75;  // Altura en metros

// Cálculo del IMC
let imc = peso / (altura * altura);
console.log("IMC: " + imc.toFixed(2));  // Mostrar el IMC con 2 decimales

// Clasificación del IMC usando condicionales anidados
if (imc < 18.5) {
 console.log("Clasificación: Bajo de peso");
} else if (imc >= 18.5 && imc <= 24.9) {
```

```
 console.log("Clasificación: Peso normal");
} else if (imc >= 25.0 && imc <= 29.9) {
 console.log("Clasificación: Sobrepeso");
} else {  // IMC mayor o igual a 30.0
 if (imc >= 30.0 && imc <= 34.9) {
   console.log("Clasificación: Obesidad grado 1");
 } else if (imc >= 35.0 && imc <= 39.9) {
   console.log("Clasificación: Obesidad grado 2");
 } else {
   console.log("Clasificación: Obesidad grado 3");
 }
}
```

Problema de ejemplo: Suma de los Primeros N Números

Un programa que pida al usuario un número N y calcule la suma de todos los números del 1 al N utilizando un ciclo `for`

```
let N = 5;
let suma = 0;

for (let i = 1; i <= N; i++) {
 suma += i;
}

console.log("La suma de los primeros " + N + " números es: " + suma);
```

Problema de ejemplo: Tablas de Multiplicar del 1, 2, 3, 4 y del 5, del 1 al 10:

```
for (let i = 1; i <= 5; i++) {
 console.log("Tabla del " + i + ":");
 for (let j = 1; j <= 10; j++) {
   console.log(i + " x " + j + " = " + (i * j));
 }
 console.log("");
}
```

Problema de ejemplo: Adivinar un Número

Un programa que permita al usuario adivinar un número secreto entre 1 y 10. El programa debe seguir solicitando al usuario que adivine hasta que ingrese el número correcto. Utiliza un ciclo `while`.

```
let numeroSecreto = 7;
let intento = parseInt(prompt("Adivina el número entre 1 y 10:"));
```

```
while (intento !== numeroSecreto) {
 intento = parseInt(prompt("Incorrecto. Intenta nuevamente:"));
}

console.log("¡Felicidades! Adivinaste el número secreto.");
```

Problema de ejemplo: Matriz de Asteriscos

Un programa que imprima un cuadrado de asteriscos de tamaño n x n, donde n es un número introducido por el usuario. Utiliza un ciclo `while` anidado para generar el patrón.

```
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

```
let n = 5;
let i = 0;

while (i < n) {
 let j = 0;
 let fila = "";
  while (j < n) {
   fila += "* ";  // Agregar un asterisco y un espacio
   j++;
 }
  console.log(fila);  // Imprimir la fila completa
 i++;
}
```

Problema de ejemplo: Validación de Entrada de Usuario

Un programa que solicite al usuario ingresar un número entre 1 y 10. El programa debe seguir solicitando el número hasta que el usuario ingrese un valor válido en ese rango.

```
let numero;

do {
 numero = parseInt(prompt("Introduce un número entre 1 y 10:"));
} while (numero < 1 || numero > 10);
```

```
console.log("Número válido ingresado: " + numero);
```

Contador

Un **contador** es una variable que se utiliza para llevar el conteo de la cantidad de veces que se ha ejecutado un ciclo o la cantidad de ocurrencias de un evento. Generalmente, se incrementa o decrementa en cada iteración del ciclo.

Ejemplo:

```
let contador = 0;

for (let i = 0; i < 10; i++) {

   contador++;  // Incrementa el contador en cada iteración

}

console.log("El ciclo se ejecutó " + contador + " veces.");
```

Acumulador

Un **acumulador** es una variable que almacena la suma o combinación de valores a lo largo de un ciclo. A diferencia del contador, no sólo cuenta, sino que "acumula" un valor en cada iteración, que suele ser una suma, producto u otra operación matemática.

Ejemplo:

```
let acumulador = 0;

for (let i = 1; i <= 5; i++) {

   acumulador += i;  // Acumula la suma de los números

}

console.log("La suma de los números del 1 al 5 es: " + acumulador);
```

Bandera

Una bandera o flag es una variable booleana que se utiliza para señalar si un evento específico ha

ocurrido o si una condición se ha cumplido dentro del ciclo. Es muy útil para controlar el flujo de ejecución.

```
let encontrado = false;
let numeros = [1, 2, 3, 4, 5];

for (let i = 0; i < numeros.length; i++) {

   if (numeros[i] === 3) {

       encontrado = true;

       break;

   }

}

if (encontrado) {

   console.log("El número 3 fue encontrado.");

} else {

   console.log("El número 3 no fue encontrado.");

}
```

Lectura Y Recorrido De Cadenas

Lectura y longitud

Para leer cadenas lo podemos hacer similar a como lo hacemos para leer elementos de un arreglo. En una cadena, cada caracter que la conforma se puede leer mediante un índice. Así, la cadena “Colombia” está compuesta por el caracter **“C”** el cual tiene índice 0, el caracter **“o”** el cual tiene índice 1, el caracter **“l”** el cual tiene índice 2 y así sucesivamente. Las cadenas al igual que los arreglos también tienen método **.length** el cual nos devuelve la longitud de la cadena, es decir, el número de caracteres que la componen. Por ejemplo, dada la variable **let mensaje = “hola”**, usando **mensaje.length** obtendríamos 4, ya que el número de caracteres que componen la cadena es 4.

Ejemplo:

Dada la cadena “Somos programadores”, lea e imprima el caracter de índice 11, lea e imprima el caracter de índice 15, también imprima la longitud de la cadena.

```
let mensaje = "Somos programadores";
console.log(mensaje[11]);//imprime a
console.log(mensaje[15]);//imprime o
console.log(mensaje.length);//imprime 19
```

Recorrido

Para recorrer una cadena se puede usar ciclo for, de forma similar a como se recorre un arreglo unidimensional.

Ejemplo:

Recorrer la cadena “Somos programadores”.

```
let mensaje = "Somos programadores";
for (let index = 0; index < mensaje.length; index++) {
 console.log(mensaje[index]);//imprimirá cada caracter de la cadena
}
```

La salida del anterior código será:

```
S
o
m
o
s

p
r
o
g
r
a
m
a
d
o
r
e
s
```

Inmutabilidad

Las cadenas son inmutables, diferente a un arreglo, estas no se pueden modificar. Por tanto, la operación de asignación no existe. Por ejemplo, dada la variable **let mensaje = "hola"** tratemos de modificar uno de sus caracteres igual que lo haríamos con un arreglo. Tratemos de cambiar el caracter **"o"** por **"z"**

```
let mensaje = "hola";
mensaje[1] = "z";
console.log(mensaje);//imprimirá hola
```

acá notamos que a pesar de tratar de asignar el caracter "z" a la cadena, esta no cambia, se mantiene inmutable.

Taller

1.Dada la cadena "Javascript es un buen lenguaje":

a) Leer e imprimir los elementos de índices 3, 6, 8
b) Recorra la cadena usando ciclo for

2.Dada la cadena "Somos VankVersity":

c) Recorra la cadena usando ciclo for y muestre si la cadena contiene la letra "V"
d) Recorra la cadena usando ciclo for y muestre cuántas veces está la letra "o" . Como sugerencia use una variable contadora para contar el número de veces que se encuentra la letra "o".

3.Dada la cadena "Javascript es genial" recorra la cadena usando ciclo for y muestre por cuantas palabras está compuesta la cadena. Como sugerencia se puede basar en el modelo del ejercicio 2 literal d.