**PROGRAMACIÓN IMPERATIVA**

**C1A:**

**Pensamiento computacional:** abordar un problema complejo, comprenderlo y plantear una solución. Proceso de resolución de problemas que se vale de la organización lógica y el análisis de datos. Posibilita el uso de una computadora u otro recurso para resolverlos.

Etapas:

1. Descomposición del problema.
2. Reconocimiento de patrones: series que se repiten
3. Abstracción: simplificar un problema complejo, depurando los datos más relevantes.
4. Algoritmo: definición de los pasos necesarios para la resolución del problema.

Si cambian las variables, debemos plantear un nuevo algoritmo.

**C4A**

Las **variables** son espacios de memoria en la computadora donde podemos almacenar distintos tipos de datos.

**Tipos de variables:**

* Var: son variables que están disponibles en el entorno global, se puede acceder desde cualquier parte del código. No obstante, primero debe ser declarada y luego realizar el console.log. Es la única que se puede volver a re-declarar, pisando su valor anterior.
* Let: son variables que están disponibles sólo en el entorno local, es decir en el bloque de código en el que se hayan declarado. No se pueden volver a declarar (tira error)
* Const: son variables que al igual que let sólo están disponibles en el ambiente o entorno en el que se hayan declarado. No se pueden volver a declarar (tira error)

**Restricciones en el nombre:**

* Sólo puede estar formado por letras, números y los símbolos pesos $ y guion bajo (\_)
* No pueden empezar con un número
* No deberían contener ñ o caracteres con acentos.

JavaScript es un lenguaje sensible case, es decir hace diferencia entre MAYÚSCULAS y minúsculas.

TIPOS DE DATOS:

* Number
* Booleanos
* String
* Undefined (valor sin definir)
* Null (valor nulo)
* NaN (not a number)

**C5A**

**OPERADORES MATEMÁTICOS:** los operadores nos permiten manipular el valor de las variables, realizar operaciones y comparar sus valores.

**OPERADORES ARITMÉTICOS:** siempre reciben un resultado numérico.

* - \* / %

++ (incremento)

— (decremento)

**OPERADORES DE COMPARACIÓN: siempre devuelven un booleano**

>=

<=

<

>

**Comparación simple**

5 == “5”; True: porque compara solo los valores

5!=”5”;False

**Comparación estricta**

5===”5”;False: porque compara los valores y el tipo de dato. Uno es numérico y el otro es string.

5===5; True

5!==”5”; True

**OPERADORES LÓGICOS: siempre devuelven un booleano**

Y: &&

O: ||

Negación: !

**OPERADOR DE CONCATENACIÓN:**

Concatenar: 2 + “ manzanas“ == “2 manzanas”

**SCOPE: es el alcance o ámbito de una variable**

* Local: la variable se declara dentro de la función y tiene scope local. No se puede acceder a ella por fuera de esa función.
* Global: la variable se declara por fuera de la función y tiene scope global. Podemos acceder a ella desde cualquier parte de nuestro código.

**FUNCIONES:** es un bloque código que puede recibir argumentos, realizar una tarea específica y retornarnos un valor. Las funciones pueden ser reutilizadas y sólo se ejecutan al invocarlas.

Se utiliza la palabra reserva **function**

**Diferencia entre parámetros y argumentos**

**Parametros: al declarar la función**

function **sumar**(**num1, num2)** {

return num1 + num2;

}

**Argumentos: al invocar la función**

sumar(**num1, num2**)

**FUNCIONES DECLARADAS:** son funciones que se declaran usando la estructura básica. Pueden recibir un nombre escrito a continuación de la palabra function, a través del cual podremos invocarla.

Funciones nombradas:

* function suma(a, b){acción y el return}

**FUNCIONES EXPRESADAS:** son aquella que se asignan como valor de una variable. En este caso, la función en sí no tiene nombre, es una función anónima.

let resultadoDeSuma = function (a,b) {

Return a + b}

console.log(resultadoDeSuma(2,3))

**FALSY**

* False
* 0
* “”
* Null
* Undefined
* NaN

**TRUTHY**

* “0”
* “false”
* [] (arreglo vacío)
* {} (objeto vacío)
* function(){} (función vacía)

Las reglas son:

* False, cero y cadenas vacías son todas equivalentes. (Dan como resultado TRUE en comparación simple) Ej: false === 0 && false === "" && 0 === “") esto es TRUE
* Null y Undefined son equivalentes a ellos mismos y sólo entre ellos.
* NaN no es equivalente a nada, incluido otro NaN

**CONDICIONALES: nos permiten evaluar condiciones y realizar diferentes acciones según el resultado de esas evaluaciones.**

If (condición) {

Acción que se ejecuta si la condición es verdadera

}

**Con bloque else:**

If (condición) {

Acción que se ejecuta si la condición es verdadera

} else {

Acción que se ejecuta si la condición es falsa

}

**Con bloque else if:**

If (condición) {

Acción que se ejecuta si la condición es verdadera

} else if (otra condición) {

Acción a ejecutar si la otra condición es verdadera

} else {

Acción que se ejecuta si la condición es falsa

}

**VERSIONES ABREVIADAS DE LOS CONDICIONALES**

**IF TERNARIO**

Estructura:

Condición ? Acción si es True : Acción si es False

Llueve ? “Sí, llueve” : “ “

Llueve && “Sí, llueve”

Llueve || “Está soleado”

**SWITCH**

let materia = "historia"

switch (materia) {

case "lengua":

console.log("asdhjkasdk");

break;

case "matemática":

console.log(horarioMatematica);

break;

case ("historia" || "naturales"):

console.log("3")

break;

default:

console.log("nada");

}