**Apuntes programación clase II**

Document llama a un archivo, getElementById obtiene un elemento por su ID, ese identificador se busca en el html. Es importante respetar las mayus y minus. Su tarea es conseguirnos la dirección de memoria donde está el elemento que nos interesa.

document.getElementById(“elIDdeldatoquenecesito”).value

.value es necesario para que traiga el valor.

devuelve null cuando tiene que devolver una dirección de memoria pero no la encuentra porque no existe.

ver el video tipo 19:26 tiempos de qué?

un operador es un símbolo que realiza una operación.

operadores binarios cuando un operador tiene dos operando por ejemplo la suma

el unario tiene un solo operando

hay varios tipos de datos

string

number (int o float)

el resto me sirve para saber si un numero es par o impar

si divido por 2 y da 0 es par

también pueden ser múltiplos

a % se le llama modulo y sirve para saber el resto

**Clase III**

 /\* Machetito:

    Qué datos voy a precisar?

    De dónde los obtengo?

    Qué hago con esos datos?

    Qué hago con el producto del análisis de esos datos?

    1\_ Identificar datos y reservar memoria para ellos (declarar la variable)

    2\_ Conseguir el dato

    3\_ Calcular el resultado

    4\_ Informo el resultado

    (Esto es pseudocódigo)\*/

a\*b+c

Poner paréntesis es redundante porque \* / y % tienen prioridad así que los va a resolver primero

para modificar la cantidad de decimales variable.toFixed (2) (para dos decimales)

cuando quiero editar muchas cosas a la vez toco alt+lugar donde quiero que aparezca otro cursor

**Clase IV**

El condicional if permite ejecutar un código dependiendo de si la condición es verdadera o falsa.

if (condición)

{se ejecuta este código si la condición es verdadera}

else

{se ejecuta este código si la condición es falsa}

Se ejecuta un bloque o el otro, no ambos.

Operadores aritméticos: (+ - \* / %)

un operador aritmético tiene un operando a cada lado y se obtiene un resultado de tipo number

un operador relacional relaciona un operando con otro:

== (igual)

!= (distinto)

< (menor)

> (mayor)

<= (menor igual)

>= (mayor igual)

Tanto los operadores relacionales como los aritméticos son operadores binarios porque ambos operan con un operando a la izq y otro a der

En el caso de los operadores relacionales los operandos pueden ser number y number o boolean y boolean o string y string y el resultado siempre va a ser booleano

Si bien es válido hacer dos veces if para dos condiciones distintas (ej, mayor o igual a 18 y menor a 18) le estaríamos diciendo al interprete que lea ambos bloques cuando podría leer uno solo, no es productivo. El if tiene dos resultados posibles, verdadero o falso, si se le pide una función y está resulta ser falsa no hace falta preguntarle si la otra es verdadera porque se sobreentiende, para eso se usa else.

El ejercicio 3 también lo puedo resolver planteando la condición al revés (es lo mismo)

Los operadores lógicos tienen dos operadores booleanos y el resultado es booleano.

Tablas de verdad: (ver en la guía)

el operador “or” (unión) me dice si al menos una de las condiciones es verdadera (siempre me va a dar true a menos que ambos operandos sean false) se representa como || (pipe)

el operador “and” (intersección) me dice si se cumplen ambas condiciones (siempre me va a dar false a menos que ambos operandos sean true) se representa como && (ampersand)

or y and son binarias

existe un operador lógico unario porque solo tiene un operando

una variable booleana tiene dos posibles valores: true y false

el operador not me da como resultado el valor opuesto a la variable booleana que estoy analizando. Se representa como !

ejemplo: si a es true, not a va a ser false y viceversa. (a = true, !a =false)

No hace falta poner nada entre paréntesis porque los operadores relacionales tienen prioridad sobre los lógicos ya que estos necesitan de una variable booleana para operar.

**Clase VI**

no es conveniente que los códigos lleven símbolos no anglosajones (como la ñ) porque corresponden a ascii extendido y puede generar errores en el código.

**Clase VIII**

Ya vimos funciones secuenciales donde el código se ejecuta secuencialmente línea por línea, funciones condicionales donde una condición define qué línea de código se va a ejecutar y ahora vamos a ver las iterativas (o repetitivas) donde en algún momento se me genera un bucle por una condición.

El while es igual al if pero tiene estructura repetitiva, es decir, mientras la condición del while siga siendo verdadera, se va a repetir el bloque de código dentro del while.

**Clase IX**

Al contador lo llamo contador porque le sumo una constante mientras que al acumulador le sumo una variable.

Contador incrementa o decrementa en valores fijos (constantes), acumulador incrementa o decrementa en valores variables.

+= sirve para no tener que volver a escribir la variable y que se acumule.

Ej.: numero = numero + x == numero += x

El do while se utiliza para no tener que inicializar una variable para que entre en la condición. Lo que hace es evaluar la condición al final, así en lugar de ver si me deja entrar al while, evalúa si me deja salir dependiendo de la condición. Así el código se tiene que ejecutar por lo menos una vez.

En el caso del ejercicio 8 de while, vamos a utilizar un flag. Un flag es una bandera que me avisa si pasé o no pasé por algún lugar del código.

En este caso el flag nos va a servir para saber si se ingresó un negativo o no.

Vimos 3 estructuras: while, do while y for. ¿Cuándo usar cada una?

El do while lo voy a usar cuando quiero que se entre por lo menos una vez al bucle (ya que me permite leer la condición al final y ahí evaluar si vuelvo a entrar o no)

Cuando conozco el número de repeticiones que necesito voy a usar for, de lo contrario uso while.