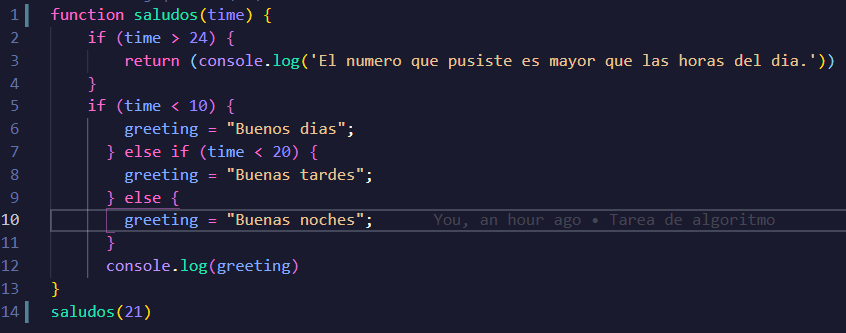
Análisis

**Función del algoritmo**: Darte un saludo adecuado con la hora que agregues.

**Nombre del algoritmo**: saludos

**Lenguaje**: JavaScript.

**Tiempo de la función**: O(1)

**Explicación**: Este algoritmo es de orden O(1) porque

Es constante, no importa el número que le pasemos

el siempre va hacer la misma condición y tiempo va ser el mismo al ejecutarse.

**Tamaño del archivo**: 1 KB.

**Cuanto recurso consume a al sistema**: Muy poco.

**Que se necesita para ejecutar el algoritmo**: Que agregues una variable de tipo numero.

**Como se ejecutar el algoritmo**: Agregas un número a la función, él va comprobar que no sea mayor que 24(pues el día solo tiene 24 horas), después de ahí va comprobar que si es menor de las 10, menor de 20 o mayor de 20, así sabrá que tipo de saludo darte.

**Cuantas líneas de código tiene**: 14 líneas.

**Resultado**: 21 = Buenas noches

Diseño -Diagrama de flujo.

Fin: la función te va devolver un string con tu saludo: buenas noches

Va comprobar que cantidad tiene el número que agregaste para darte el saludo de acuerdo a la hora.

Va comprobar que no sea mayor de 24.

La Función sort

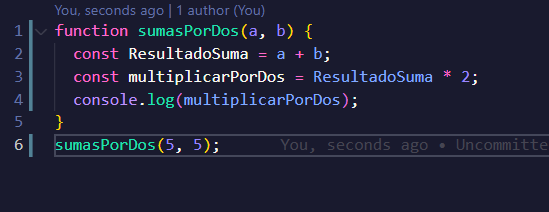
Signo de resta = -

Iniciar la function:

Saludos(21)

Análisis

**Función del algoritmo**: Sumar 2 números y multiplicarlos por 2.

**Nombre del algoritmo**: sumarPorDos

**Lenguaje**: JavaScript.

**Tiempo de la función**: O(log A+B)

**Explicación**:

Es logarítmico, porque el tiempo va depender

de los valores que le pasemos (si A + B = AB por 2 tiempo de ejecución va durar más porque la multiplicación es más grande.)

**Tamaño del archivo**: 1 KB.

**Cuanto recurso consume a al sistema**: Muy poco.

**Que se necesita para ejecutar el algoritmo**: 2 números y llamar la función con sus 2 valor.

**Como se ejecutar el algoritmo**: Primero el algoritmo toma los dos valor que le agregaste los suma y ahí mismo en otra variable los multiplica por 2 y te lo devuelve por consola.

**Cuantas líneas de código tiene**: 6 líneas.

**Resultado**: 20

Diseño-Diagrama de flujo.

El signo de suma = +

Y el de Multiplicacion = \*

MultiplicarPorDos = ResultadoSuma \* 2

Fin:

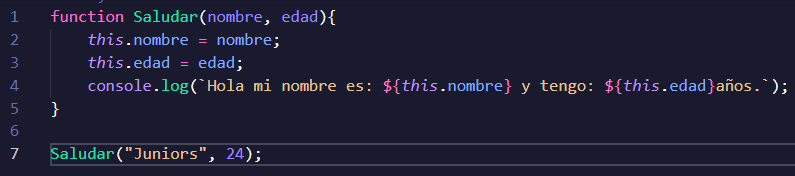
El resultado de la las dos variables: MultiplicarPorDos y ResultadoSuma

ResultadoSuma = a + b

Iniciar la function:

sumarPorDos(5,5)

Análisis

**Función del algoritmo**: Cuando agregas tu nombre y tu edad la función crear un oración con esos valores.

**Nombre del algoritmo**: Saludar

**Lenguaje**: JavaScript.

**Tiempo de la función**: O(1)

**Explicación**: Este algoritmo es de orden O(1) porque

Es constante, no importa los dos valores que le pasemos

el siempre va hacer la oración al mismo tiempo, ya que el valor de las variables no afecta la creación de la oración.

**Tamaño del archivo**: 1 KB.

**Cuanto recurso consume a al sistema**: Muy poco.

**Que se necesita para ejecutar el algoritmo**: Que agregues dos variables una tipo String y otra tipo Number

**Como se ejecutar el algoritmo**: Agregas las dos varibles y te devuelve la oración con esos valores

**Cuantas líneas de código tiene**: 7 líneas.

**Resultado**: Hola mi nombre es: Juniors y tengo: 24años.

Diseño -Diagrama de flujo.

Fin: te devuelve una oración con eso valores agregados

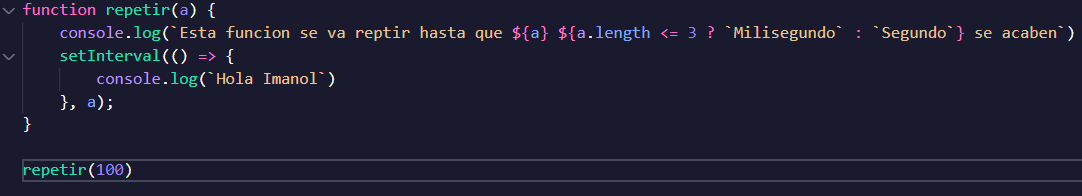
“This.”

Iniciar la función:

Saludar(“junior”, 24)

Análisis

**Función del algoritmo**: Se va a ejecutar una función hasta que el tiempo se acabe

**Nombre del algoritmo**: repetir

**Lenguaje**: JavaScript.

**Tiempo de la función**: O(A)

Es de tipo lineal, porque va depende del número que le pasemos que va ser la cantidad de tiempo que se va ejecutar dicha función más grande el numero más tiempo va durar ejecutando la funciona varias veces.

**Tamaño del archivo**: 1 KB.

**Cuanto recurso consume a al sistema**: Muy poco.

**Que se necesita para ejecutar el algoritmo**: Que agregues una variable de tipo número para ser el tiempo

**Como se ejecutar el algoritmo**: Agregas un número a la función, la función serInterval va tomar ese número y va repetir lo que agregues dentro y lo va repetir hasta que se acabe el tiempo.

**Cuantas líneas de código tiene**: 8 líneas.

**Resultado**: Esta función duro 100 Segundo repitiendo la misma función.

Diseño -Diagrama de flujo.

La función setInterval va comprobar el número y va repetir hasta que se acabe el tiempo

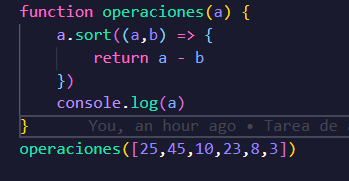
Fin: Vas a ver la función repetida varias veces

La Función setInterval

Iniciar la function:

repetir(100)

Análisis

**Función del algoritmo**: Ordena una array de números

**Nombre del algoritmo**: operaciones

**Lenguaje**: JavaScript.

**Tiempo de la función**: O(A)

Es de tipo lineal, porque va depende del tamaño del array

más grande el array mayor tiempo va durar en organizarlo correctamente este.

**Tamaño del archivo**: 1 KB.

**Cuanto recurso consume a al sistema**: Muy poco.

**Que se necesita para ejecutar el algoritmo**: Una array de numeros no importa la cantida.

**Como se ejecutar el algoritmo**: Primero el algoritmo toma el array de numeros y los procesa por la función sort que toma dos variables a y b, va comprobando cual es mayor a o b, hasta que se acabe la cantidad de numeros que hay en array.

**Cuantas líneas de código tiene**: 7 líneas.

**Resultado**: [ 3, 8, 10, 23, 25, 45 ]

Diseño -Diagrama de flujo.

Sort compruébala si a es mayor que b

Fin: la función sort devuelve un array con los numeros organizando de menor a mayor

Entran a la function sort

La Función sort

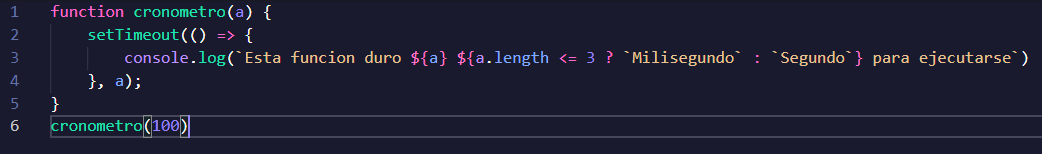
Signo de resta = -

Iniciar la function:

Operaciones ([25,45,10,23,8,3])

Análisis

**Función del algoritmo**: Activar una función después del tiempo que tú le pongas

**Nombre del algoritmo**: cronometro

**Lenguaje**: JavaScript.

**Tiempo de la función**: O(A)

Es de tipo lineal, porque va depende del tamaño del array

más grande el array mayor tiempo va durar en organizarlo correctamente este.

**Tamaño del archivo**: 1 KB.

**Cuanto recurso consume a al sistema**: Muy poco.

**Que se necesita para ejecutar el algoritmo**: Que agregues una variable de tipo número para ser el tiempo

**Como se ejecutar el algoritmo**: Agregas un número a la función, la función setTimeount va tomar ese número y va durar por la cantidad de ese número para activar la función.

**Cuantas líneas de código tiene**: 6 líneas.

**Resultado**: Esta función duro 100 Segundo para ejecutarse

Diseño -Diagrama de flujo.

Fin: se va activar la función después que pase el tiempo agregados.

La función setTimeout va comprobar el número que pusiste

La Función setTimeout

Iniciar la function:

Cronometro(100)