RECURSIVIDAD

INTERMEDIATE

Es un mecanismo de programación, que resulta muy útil en aquellas situaciones en las que la solución a un problema puede dividirse fácilmente en varias tareas del mismo tipo, pero mas simples.

La recursión o recursividad es utilizada para resolver problemas que contienen subproblemas mas pequeños.

Es el acto de una función que se llama a si misma para resolver un problema.

Reto 1

Implemente la función *potencia*, la cual recibe dos parámetros **a** y **b**, la función debe retornar el valor de a elevado a la potencia de b.

Nota

No esta permitido el uso de funciones de la librería Math.

Existen dos formas de pensamiento.

Pensamiento iterativo

Entregar una solución a través del uso de estructuras repetitivas o loops, **for**, **while**, **do-while**.

Pensamiento recursivo

Entregar la solución al problema en términos de la misma definición de la función.

Solución mediante el pensamiento iterativo

```
'use strict'
     function potencia(a, b) {
         let resultado = 1;
         for (let indice = 1; indice <= b; indice++) {
             resultado *= a;
         return resultado;
12
     console.log(potencia(2, 3));
13
```

Solución mediante el pensamiento recursivo

```
'use strict'
function potencia(a, b) {
    if (b == 1) {
        return a;
    } else {
        return a * potencia(a, b - 1);
console.log(potencia(2, 3));
```

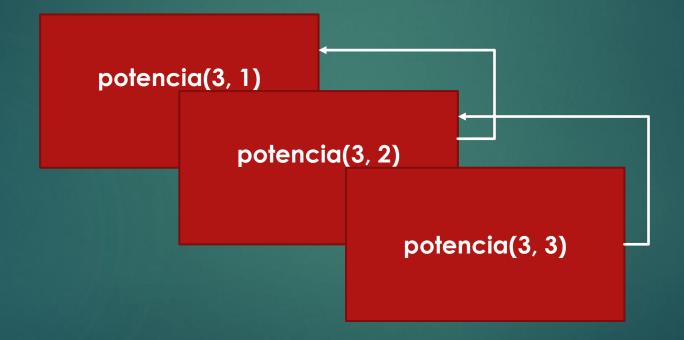
Pila de llamados a la función recursiva.

potencia(3, 3) =
$$3 \times potencia(3, 2) = 3 \times 9 = 27$$

potencia(3, 2) =
$$3 \times potencia(3, 1) = 3 \times 3 = 9$$

Potencia
$$(3, 1) = 3$$

Pila de llamados a la función recursiva.



A cada uno de estos llamados de la función se denomina profundidad de la función y el ámbito en que se ejecutan se denomina contexto de la función recursiva.

Por cada reto siguientes utilice la solución iterativa y la solución recursiva.

Reto 2

Implemente la función *intercambio*, la cual recibe dos parámetros **m** y **n**, de valores enteros la función debe retornar los valores intercambiados.

Reto 3

Implemente la función sumaTodo, la cual recibe un parámetros **n**, de valor enteros. La función debe retornar la suma de los valores hasta **n**.

```
sumaTodo(1) = 1
sumaTodo(2) = 1 + 2 = 3
sumaTodo(3) = 1 + 2 + 3 = 6
```

Reto 4

Implemente la función sumaTodo, la cual recibe un parámetros **n**, de valor enteros. La función debe retornar la suma de los valores hasta **n**.

```
sumaTodo(1) = 1
sumaTodo(2) = 1 + 2 = 3
sumaTodo(3) = 1 + 2 + 3 = 6
```

Reto 5

Diseñar una función que me permita obtener el valor absoluto de n.

Reto 6

Diseñar una función **factorial(n)**, que me permita obtener el valor factorial de **n**.

```
n! = n * (n - 1) * (n - 2) * ...*1

1! = 1

2! = 2 * 1 = 2

3! = 3 * 2 * 1 = 6

4! = 4 * 3 * 2 * 1 = 24

5! = 5 * 4 * 3 * 2 * 1 = 120
```

Reto 7

Diseñar una función **fibonacci(n)**, que me permita obtener el valor **n** de la serie de fibonacci.

La secuencia de sucesión de Fibonacci tiene la fórmula Fn = Fn-1 + Fn-2. En otras palabras, el siguiente número es una suma de los dos anteriores.

La serie debe comenzar con los números 0 y 1.

Happy coding... ©