Recursão

- Recursão é um conceito na área da programação que basicamente significa uma função que chama a sí mesma x vezes. Mas para isso, é necessário definirmos um caso base, ou seja, um caso de parada para evitar um loop em nossas chamadas de função.

- PADRÃO 1: descobrir qual o n-ésimo elemento de uma sequência infinita.

```
// [EXEMPLO] Observe a sequência aritmética a seguir e crie um
// programa para encontrar o valor do n-ésimo elemento:
// {2,7,12,17,22,...}

// Função recursiva
const nEsimoElemento = (valor) => {
    if (valor == 1) return 2;
    else {
        // Chamamos a própria função e decrementamos o valor em -1 x vezes.
        return nEsimoElemento(valor - 1) + 5;
    }
};

// Retorna o elemento 22 que é o quinto elemento da sequência.
console.log(nEsimoElemento(5))
```

- [EXEMPLO: Lista 04, Q2] N-ésimo termo da sequência {0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...}

```
// Sequência de fibonacci

const fibonacci = (n) => {
    // Caso base
    if (n == 0) return 0;
    else if (n == 1) return 1;
    else {
        // Fazemos uma dupla recursividade e somamos os resultados.
        return fibonacci(n- 1) + fibonacci(n - 2);
    }
};

// Resultado é 5.
console.log(`Fibonacci de 5 é: ${fibonacci(5)}`);
```

- PADRÃO 2: implementar uma operação que é formada por uma repetição de operações mais primitivas.

```
// Cálculo da potência de um número natural de forma recursiva.

const potencia = (b) => (e) => {
    // Caso base
    if (e == 0) return 1;
    else {
        // Multiplicamos a base vezes o resultado da chamada recursiva n vezes.
        return b * potencia(b)(e - 1);
    }
}

// Variáveis e atribuições.
const base = 2;
const expoente = 10;

// Resultado = 1024.
console.log(`Base: ${base}, Expoente: ${expoente}, resultado: ${potencia(base)(expoente)
```