Mapeamento de elementos

Considere novamente mais uma vez o exemplo genérico de uso de listas em Javascript.

```
lista = [5,8,10,3,11]
const soma = (l) => l[0]+l[1]+l[2]+l[3]+l[4]

console.log(soma(lista)) //5+8+10+3+11=37
```

Parte da inconveniência que fora discutida fica resolvida, mas uma questão ainda permanece: **como lidar propriamente com a manipulação dos valores dentro de uma função?** Observe que na definição da função de `soma` acima, fizemos referência a 5 elementos da lista. Mas como sabemos disso? E se o usuário fornecer uma lista com 10 elementos? E com 500? Nossa função de `soma` deveria estar **preparada para lidar DINAMICAMENTE** com essa questão.

[EXEMPLO] Função para aplicar um desconto de 10% em uma lista de produtos de supermercado.

Algoritmo em pseudocódigo

"DE CIMA PARA BAIXO"

resultado $\rightarrow desconto10(produtos)$

Subproblemas são gerados...
$produtos = ?$
$desconto10 = ?$

... e resolvidos:
$produtos$ é pré-definido ou fornecido pelo usuário como uma lista de valores.
$produtos = [valor1, valor2, ...]$
$desconto10 \rightarrow desconto(10) \rightarrow (produtos)$

Novo subproblema é gerado...
$desconto(desc)(lista) = ?$

... e resolvido:
$desconto(desc)(lista) =$ *para cada valor da lista:*

> *(i) multiplique esse valor por $desc/100$;*
> *(ii) subtraia esse resultado do valor original gerando um novo valor;*
> *(iii) considere esse novo valor.*

Ao propor algoritmos para solução de problema com coleções de dados, iremos frequentemente nos deparar com essa sentença: **"para cada valor da lista... faça alguma coisa."**.

Quando a solução planejada consistir de algum tipo de **transformação de cada valor original para um novo valor**, gerando uma nova lista de valores modificados, temos um caso de **MAPEAMENTO**.

(ex.) Se
`produtos = [100, 80, 60, 120, 90]` ,
um desconto de 10% **mapeia** essa lista anterior para:
`novoprodutos = [90, 72, 54, 108, 81]`
Ou seja, `novoprodutos[i] = produtos[i]-produtos[i]*(10/100)`

Função `map`

Em Javascript, fazemos esse mapeamento com uso de uma função pré-definida `map(f)` onde `f` é a função definida que representa a operação de transformação a ser aplicada a cada elemento.

[EXEMPLO] Programa para aplicar um desconto de 10% em uma lista de produtos.

```
const desconto = (desc) => (lista) => lista.map((x)=>x-x*(desc)/100)
const desconto10 = desconto(10)
const listaProdutosR$ = [10.60,8.50,5.55,6.40,41.00,23.05,19.90,15.90,22.10,2.75]
const resultado = desconto10(listaProdutosR$)
console.log(resultado)
```

[EXEMPLO] Programa para calcular o triplo de cada elemento de uma lista.

```
const valores = [3,4,-2,0,1,40]
const triplo = valores.map((x) => 3*x)
console.log(triplo)
```

[EXEMPLO] Programa para extrair a inicial de cada nome de uma lista.

```
const nomes = ['Ana', 'Bia', 'Gui', 'Lia', 'Rafa']

const primeiraLetra = (texto) => texto[0]
const iniciais = nomes.map(primeiraLetra)
console.log(iniciais)
```

[EXEMPLO] Programa para adicionar um sobrenome a cada nome de uma lista.

```
1 const nomes = ['Ana', 'Bia', 'Gui', 'Lia', 'Rafa']
2
3 const addSobrenome = (sobrenome) => (nome) => `${nome} ${sobrenome}`
4 const nomeCompleto = nomes.map(addSobrenome("Costa"))
5 console.log(nomeCompleto)
```