Algoritmos e Programação

AULA 21: Recursividade

UNIVERSIDADE FEDERAL DE PELOTAS
CENTRO DE DESENVOLVIMENTO TECNOLÓGICO
CIÊNCIA DA COMPUTAÇÃO

Recursividade

- A **recursão** é o processo pelo qual passa um certo procedimento quando um de seus passos envolve a repetição completa deste mesmo procedimento.

- Um procedimento que se utiliza da recursão é dito **recursivo**.

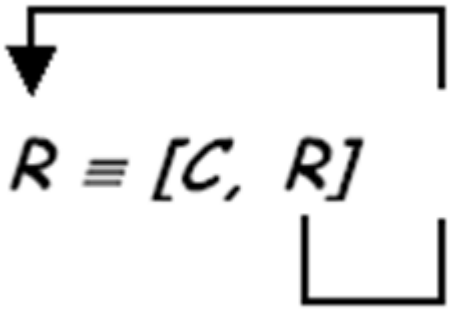
Recursividade

- Um sub-algoritmo é dito recursivo quando contém, em seu corpo, uma chamada a si mesmo (este tipo de recursividade é chamada de recursividade direta).

- São utilizados quando é possível decompor o problema a ser resolvido em problemas menores, semelhantes ao problema inicial.

Recursão

- Um sub-algoritmo recursivo R pode ser expresso como uma composição:
  - Formada por um conjunto de comandos C (que não contém chamadas a R),
  - uma chamada (recursiva) à R.



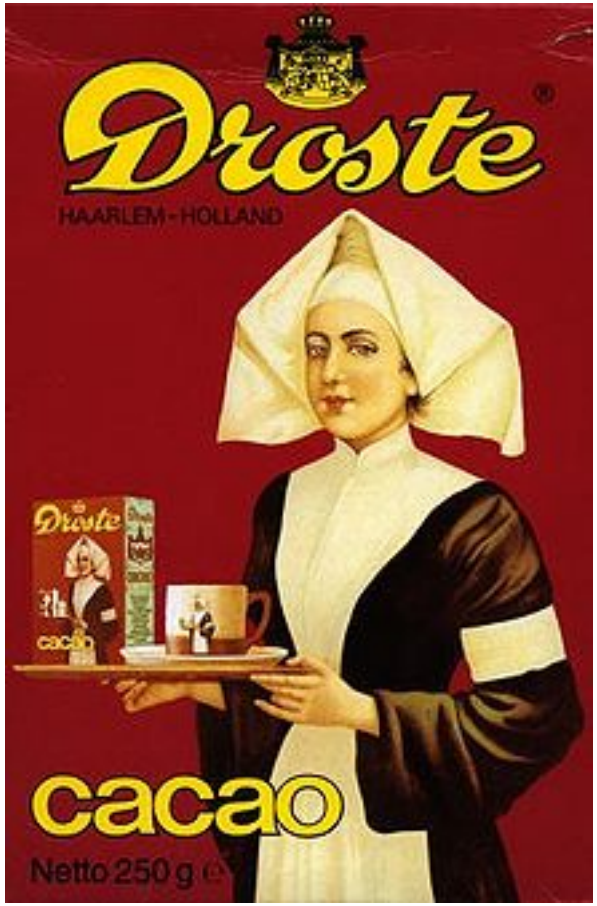
Problemas com estrutura recursiva

- Para resolver um problema com estrutura recursiva podemos aplicar o seguinte método:
  - se a instância em questão é pequena,
    - resolva-a diretamente
      - Caso base
      - use força bruta se necessário;
  - senão,
    - reduza-a a uma instância menor do mesmo problema,
    - aplique o método à instância menor e
    - volte à instância original.

Quando aplicar?

- A recursão é uma técnica apropriada se o problema a ser resolvido tem as seguintes características:
  1) a resolução dos casos maiores do problema envolve a resolução de um ou mais casos menores;
  2) os menores casos possíveis do problema podem ser resolvidos diretamente;
  3) a solução iterativa do problema (usando enquanto, para ou repita) é complexa.

Forma visual – Efeito Droste

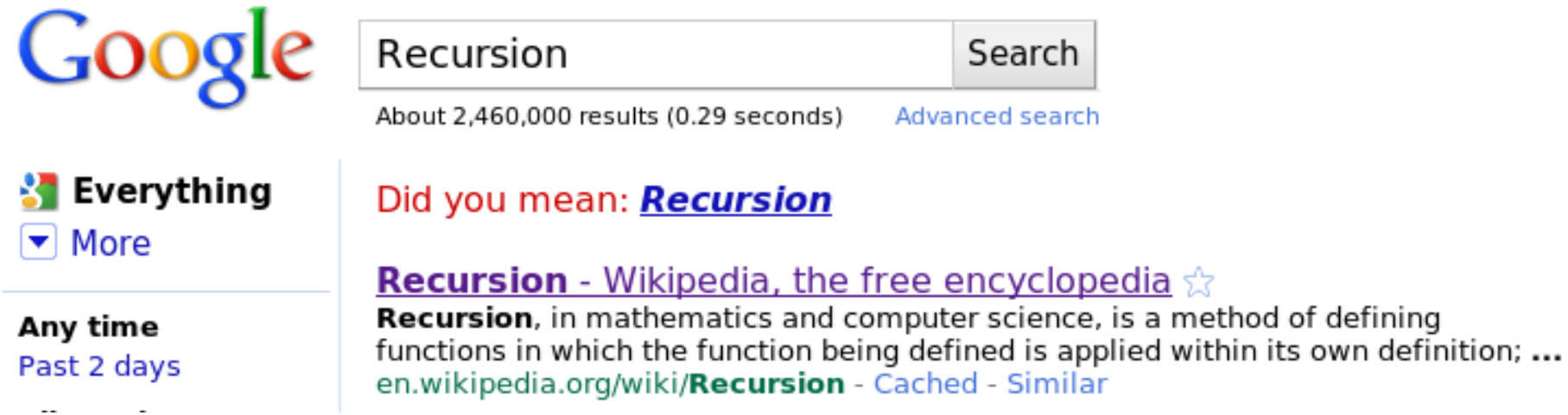


Recursão na matemática

- Triângulo de Sierpinski: é uma figura geométrica obtida através de um processo recursivo.

Humor recursivo

- Definição de: recursão
  - Se você ainda não entendeu; ver recursão



Exemplo 1 – Sub-rotina recursiva

```
sub-rotina LerValor
    Escreva "Digite um valor positivo", \n
    Leia Valor
    Se Valor < 0
          então
                Escreva "Valor Inválido, \n
                LerValor
    fim_se
fim_sub-rotina
```

Exemplo 2 - Função fatorial recursiva

- Fatorial:

  0! = 1

  1! = 1

  2! = 2 x 1!

  3! = 3 x 2!

  ⋮

  n! = n x (n-1)!

Exemplo 2 - Função fatorial recursiva

- Fatorial:

  0! = 1

  1! = 1

  2! = 2 x 1!

  3! = 3 x 2!

  ⋮

  n! = n x (n-1)!

$$fat(n) = \begin{cases} 1 & se \quad n = 0 \quad ou \quad n = 1 \\ n \times fat(n-1) & se \quad n \geq 2 \end{cases}$$

Exemplo 2 - Função fatorial recursiva

- Definição Recursiva

$$fat(n) = \begin{cases} 1 & se \quad n = 0 \quad ou \quad n = 1 \\ n \times fat(n-1) & se \quad n \geq 2 \end{cases}$$

Passo Base

Passo Recursivo

Exemplo 2 - Função fatorial recursiva

$$fat(n) = \begin{cases} 1 & se \quad n = 0 \quad ou \quad n = 1 \\ n \times fat(n-1) & se \quad n \geq 2 \end{cases}$$

- Algoritmo:

```
função numérico fatorial(n numérico)
  se n=0 OU n=1
    então Retorna 1
    senão Retorna n * fatorial(n--1)
  fim_se
fim_função
```

Exemplo 2 - Função fatorial recursiva

- Execução da chamada de fatorial(3)

  fatorial(3)

Exemplo 2 - Função fatorial recursiva

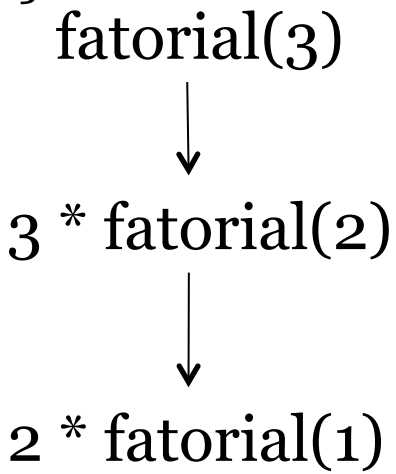
- Execução da chamada fatorial(3)

fatorial(3)

3 * fatorial(2)

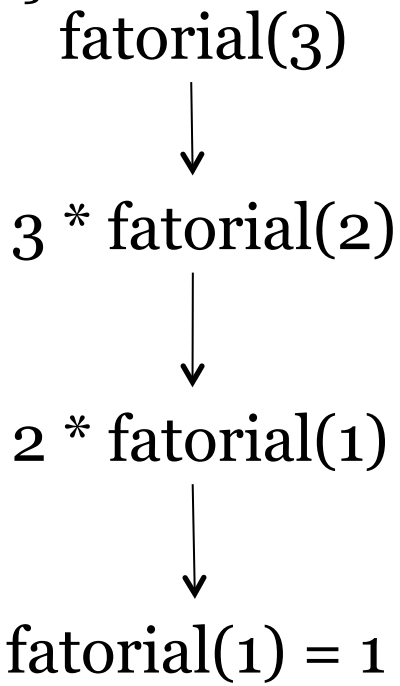
Exemplo 2 - Função fatorial recursiva

- Execução da chamada fatorial(3)



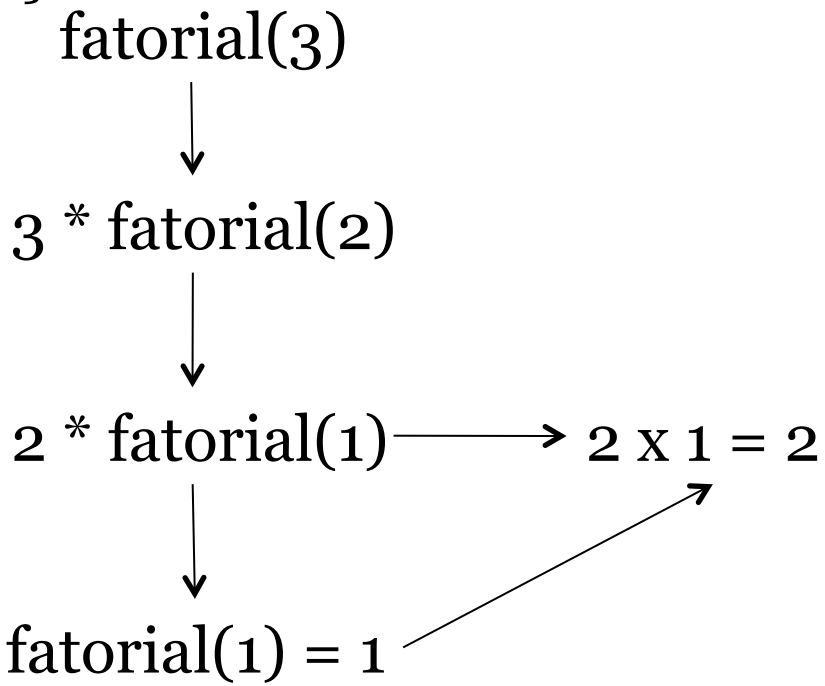
Exemplo 2 - Função fatorial recursiva

- Execução da chamada fatorial(3)



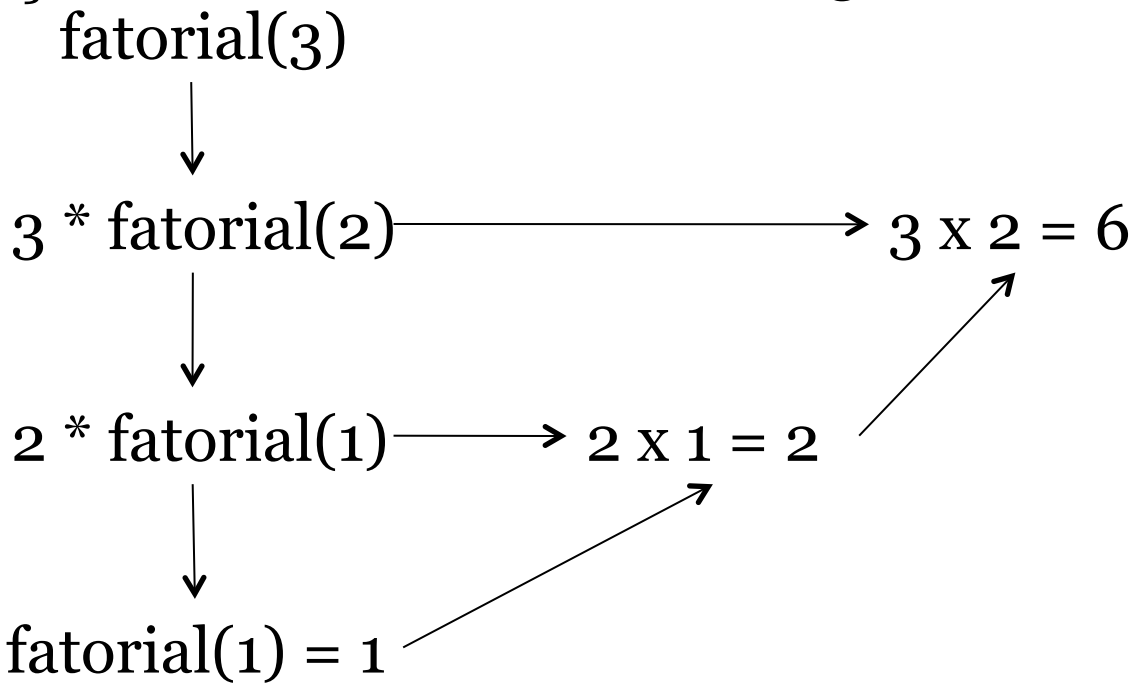
Exemplo 2 - Função fatorial recursiva

- Execução da chamada fatorial(3)



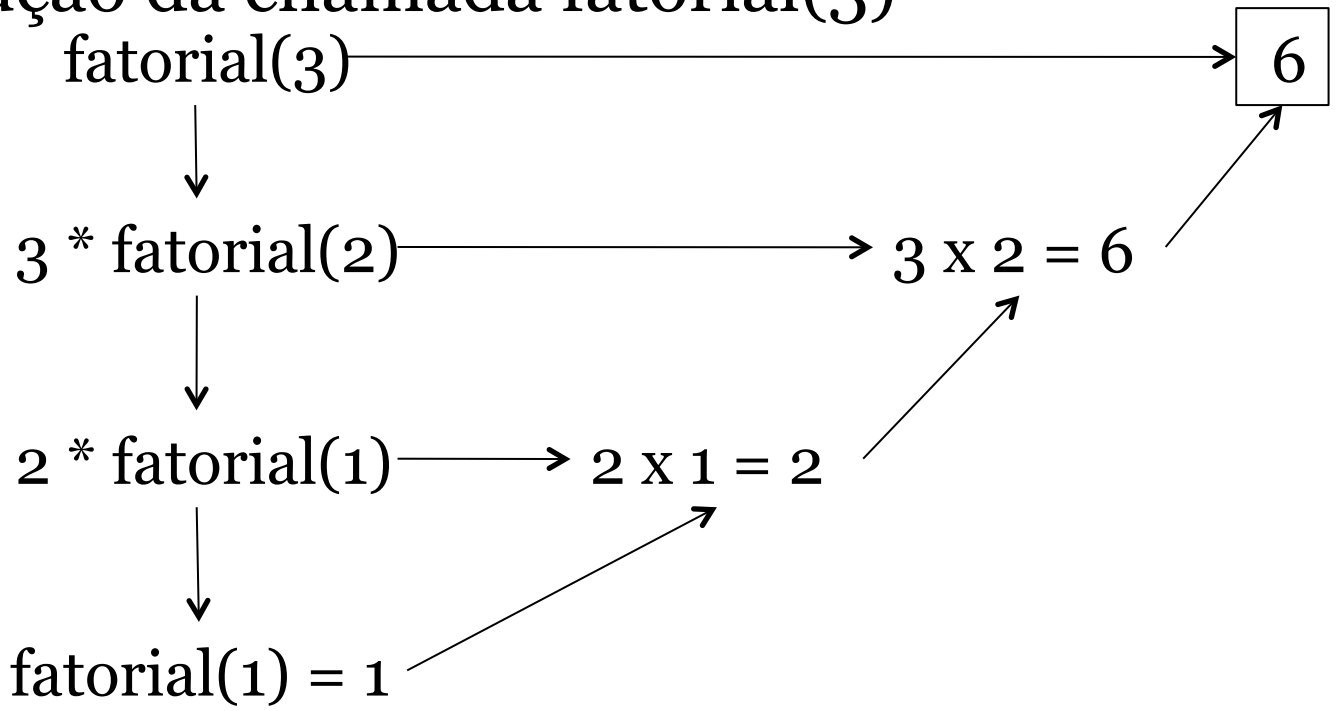
Exemplo 2 - Função fatorial recursiva

- Execução da chamada fatorial(3)



Exemplo 2 - Função fatorial recursiva

- Execução da chamada fatorial(3)



Exemplo 3 -Função soma ímpares recursiva

- Soma dos números ímpares de 1 até n

Exemplo 3 -Função soma ímpares recursiva

- Soma dos números ímpares de 1 até n

  SomaI(1) = 1

  SomaI(2) = SomaI(1)

  SomaI(3) = 3 + SomaI(1)

  SomaI(4) = SomaI(3)

  SomaI(5)=5 + SomaI(3)

Exemplo 3 -Função soma ímpares recursiva

- Soma dos números ímpares de 1 até n

  SomaI(1) = 1

  SomaI(2) = SomaI(1)

  SomaI(3) = 3 + SomaI(1)

  SomaI(4) = SomaI(3)

  SomaI(5)=5 + SomaI(3)

- Definição recursiva

$$somaI(n) = \begin{cases} 1 & se \quad n = 1 \\ somaI(n-1) & se \quad n \text{ é } par \\ n + somaI(n-2) & se \quad n \text{ é ímpar} \end{cases}$$

Exemplo 3 -Função soma ímpares recursiva

$$somaI(n) = \begin{cases} 1 & se \quad n = 1 \\ somaI(n-1) & se \quad n \text{ é } par \\ n + somaI(n-2) & se \quad n \text{ é } ímpar \end{cases}$$

- Algoritmo:

```
função numérico somaI(n numérico)
  se n=1
     então Retorna 1
     senão se Resto(n,2) = 0
                então Retorna somaI(n--1)
                senão Retorna n + somaI(n-2)
           fim_se
  fim_se
fim_função
```

Exemplo 4

- Faça uma sub-rotina para escrever números de n até 1:

Exemplo 4 - Solução

- Escrever números de n até 1:

$$esc(n) = \begin{cases} Escreva \quad 1 & se\ n = 1 \\ Escreva \quad n & se\ n \geq 2 \\ esc(n-1) & \end{cases}$$

Exemplo 4 - Solução

- Escrever números de n até 1:

$$esc(n) = \begin{cases} Escreva \quad 1 & se\ n = 1 \\ Escreva \quad n & se\ n \geq 2 \\ esc(n-1) & \end{cases}$$

- Algoritmo:

```
Sub-rotina esc(n numérico)
  se n=1
    então Escreva 1
    senão Escreva n
          esc(n--1)
  fim_se
fim_sub-rotina
```

Exercício 1

- Faça uma sub-rotina para escrever números de 1 até n:

Exercício 1 - Solução

- Escrever números de 1 até n:

$$esc(n,i) = \begin{cases} Escreva \quad i \\ esc(n,i+1) \qquad\qquad se\ n > i \end{cases}$$

Exercício 1 - Solução

- Escrever números de 1 até n:

$$esc(n,i) = \begin{cases} Escreva \quad i \\ esc(n,i+1) \qquad\qquad se\ n > i \end{cases}$$

- Algoritmo:

```
Sub-rotina esc(n,i numérico)
  Escreva i
  se n>i
    então esc(n, i+1)
  fim_se
fim_sub-rotina
```

Exercício 1 - Solução

- Escrever números de 1 até n:

$$esc(n,i) = \begin{cases} Escreva \quad i \\ esc(n,i+1) \qquad\qquad se\ n > i \end{cases}$$

- Algoritmo:

```
Sub-rotina esc(n,i numérico)
  Escreva i
  se n>i
    então esc(n, i+1)
  fim_se
fim_sub-rotina
```

- Chamada:

```
esc(x, 1)
```

Exercício 2

- Faça um algoritmo que leia e defina o n-ésimo termo da série de fibonacci.
- Utilizar uma função para definir o termo recursivamente.

n	fibonacci(n)
0	0
1	1
2	1
3	2
4	3
5	5
6	8
⋮	⋮

Exercício 2 - Solução

```
Algoritmo
        Declare Num, Termo Numérico
        função numérico fibonacci(n numérico)
                Se n=0 OU n=1
                        então Retorna n
                        senão Retorna fibonacci(n-1)+fibonacci(n-2)
                fim_se
        fim_função

        Escreva “Digite  termo  que deseja calcular”, \n
        Leia Num
        Termo:= fibonacci(Num)
        Escreva “Fibonacci”, Termo
fim_algoritmo
```

Exercício 3

- Faça um algoritmo que calcule 2^n (n deve ser um valor maior ou igual a zero a ser lido). Calcular a potência utilizando uma função recursiva.

Exercício 3 - Solução

```
Algoritmo
        Declare n, Result Numérico
        função numérico Pot(x numérico)
                Se x=0
                        então Retorna 1
                        senão Retorna  2 * Pot(x-1)
                fim_se
        fim_função
        Repita
                Escreva “Digite  o valor de n”, \n
                Leia n
        até n>=0
        result:= Pot(n)
        Escreva “Potência de 2 elevado a”, n, “é:”, result
fim_algoritmo
```

Exercício 4

- Considere uma sequência de números onde cada termo é dado pela combinação dos 4 termos anteriores
  - $A_n = A_{n-4} + 2^* A_{n-3} + 3^* A_{n-2} + 4^* A_{n-1}$
- e os 4 primeiros termos são por definição:
  - $A_1=1 \quad A_2=2 \quad A_3=3 \quad A_4=4$
- Escreva uma função recursiva que receba um número n e retorne o termo A_n

Exercício 4 - Solução

```
Algoritmo
        Declare n, Result Numérico
        função numérico seq(x numérico)
                Se x<=4
                        então Retorna x
                        senão Retorna seq(x-4)+2*seq(x-3)+3*seq(x-2)+4*seq(x-1)
                fim_se
        fim_função
        Repita
                Escreva “Digite  termo  que deseja calcular”, \n
                Leia n
        até n>=1
        result:= seq(n)
        Escreva “Termo:”, result
fim_algoritmo
```

Exercício 5

- Faça um algoritmo que verifique se dois números lidos são amigos.
- Dois números são amigos entre si, se a soma dos divisores de cada um deles é igual ao outro.
  - Exemplo, 220 e 284:
  - Divisores de 220: 1, 2 , 4, 5, 10, 11, 20, 22, 44, 55, 110
    - Soma: 284
  - Divisores de 284: 1, 2, 4, 71, 142
    - Soma 220

Exercício 5 - Solução

```
Algoritmo
   Declare a,b numérico
   função numérico soma(num,cont numérico)
            Se cont=1
                        então Retorna 1
                        senão
                                    Se RESTO(num,cont)=0
                                                then Retorna cont + soma(num,cont-1)
                                                senão Retorna soma(num,cont-1)
                                    fim_se
                        fim_se
   fim_função
   Escreva “Digite dois valores para testar se são amigos”, \n
   Leia a,b
   Se a=soma(b, b-1) E b=soma(a, a-1)
            então Escreva 'São amigos'
            senão Escreva 'Não são amigos'
   fim_se
fim_algoritmo
```

Exercício 6

- Faça um algoritmo que leia um vetor de 10 elementos numéricos, através de uma sub-rotina e que mostre o resultado do somatório dos elementos deste vetor, calculado por uma função recursiva.

Exercício 6 - Solução

```
Algoritmo
        Declare v()10, Result Numérico

        Sub-rotina LeVet(CR a() numérico; t numérico)
                Declare i numérico
                Para i de 1 até t faça
                        Escreva “Digite um valor”,\n
                        Leia a(i)
                fim_para
        fim_sub-rotina

        função numérico SomaVetor (a (), t numérico)
                Se t=1
                        então Retorna a(t)
                        senão Retorna a(t) + SomaVetor(a, t-1)
                fim_se
        fim_função

        LerVet(v, 10)
        Result := SomaVetor(v,10)
        Escreva “Soma dos valores do vetor:”, Result, \n
fim_algoritmo
```