

Programação I

Recursividade (ficha 6)

v2.0

1. Implemente a versão iterativa e a versão recursiva da função `factorial(n)`.
2. Implemente a função `fibonacci(n)`, definida como:

$$fibonacci(n) = \begin{cases} 1 & \text{se } n = 0 \text{ ou } n = 1 \\ fibonacci(n-1) + fibonacci(n-2) & \text{se } n > 1 \end{cases}$$

3. Implemente a função recursiva `soma(n)` que devolve a soma dos primeiros `n` inteiros.

```
soma(3)
6
```

4. Implemente a função recursiva `multiplo(n,i)` que calcula o `i`-ésimo múltiplo de `n`. Por exemplo:

```
multiplo(3,2)
6
```

5. O máximo divisor comum, `mdc(m,n)` pode ser calculado de forma recursiva utilizando o algoritmo de Dijkstra. Assumindo que `m,n>0`, temos:

$$mdc(m,n) = \begin{cases} m & \text{se } n = m \\ mdc(m-n,n) & \text{se } m > n \\ mdc(m,n-m) & \text{se } m < n \end{cases}$$

6. Construa a função Ackermann, `A(m,n)` de forma recursiva. `A(m,n)` é definida como:

$$A(m,n) = \begin{cases} n+1 & \text{se } m = 0 \\ A(m-1,1) & \text{se } m > 0 \text{ e } n = 0 \\ A(m-1, A(m,n-1)) & \text{se } m > 0 \text{ e } n > 0 \end{cases}$$

```
>>> A(3,4)
125
```

7. Implemente a função recursiva `pascal(lin,col)` que calcula o número da `lin`-ésima linha, `col`-ésima coluna do triângulo de Pascal. Este número pode ser definido como:

$$pascal(lin,col) = \begin{cases} 1 & \text{se } col = 0 \text{ ou } col = lin \\ pascal(lin-1,col-1) + pascal(lin-1,col) & \text{caso contrario} \end{cases}$$

8. Usando a função `pascal(lin,col)`, implemente a função `trianguloPascal(n)` que imprime as primeiras `n` linhas do triângulo de Pascal. Por exemplo:

```
trianguloPascal(6)
```

```
1
```

```
1 1
```

```
1 2 1
```

```
1 3 3 1
```

```
1 4 6 4 1
```

```
1 5 10 10 5 1
```