Fundamentos da Programação

Recursividade

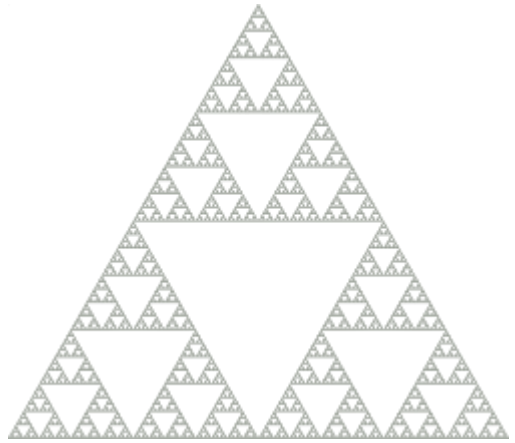
P.PORTO
ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

Conteúdo

- Recursividade

Recursividade

- É uma função que se invoca a si própria.
  - Processo cuja realização envolve a repetição desse mesmo processo.
- A execução consiste em ir resolvendo **sub-problemas** sucessivamente mais simples até se atingir o caso mais simples de todos, cujo resultado é obtido de imediato.



Recursividade

- Em todas as funções recursivas existe:
  - Caso(s) básico(s) cujo resultado é conhecido à partida e que não precisa de recursividade para ser resolvido.
    - Este caso é também conhecido por critério de paragem.
  - Um caso genérico em que se tenta recursivamente resolver um sub-problema do problema inicial.
- O cálculo de um fatorial é um exemplo típico da utilização de funções recursivas:

$$n! = \begin{cases} 1 & \text{se } n = 0 \\ n \cdot (n-1)! & \text{se } n > 0 \end{cases}$$

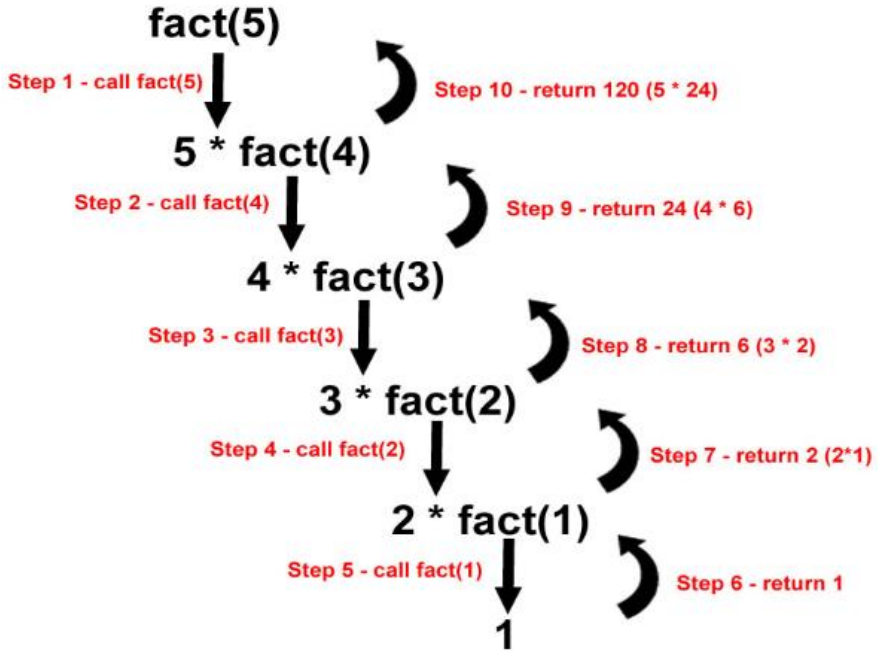
Factorial

- Em todas as funções recursivas existe:
  - **Caso(s) básico(s)** cujo resultado é conhecido à partida e que não precisa de recursividade para ser resolvido.
    - Este caso é também conhecido por **critério de paragem**.
  - Um **caso genérico** em que se tenta recursivamente resolver um sub-problema do problema inicial.
- O cálculo de um fatorial é um exemplo típico da utilização de funções recursivas:

$$n! = \begin{cases} 1 & \text{se } n = 0 \\ n \cdot (n-1)! & \text{se } n > 0 \end{cases}$$

Factorial

$$n! = \begin{cases} 1 & \text{se } n = 0 \\ n \cdot (n-1)! & \text{se } n > 0 \end{cases}$$




```
…
int factorial(int valor) {
    if (valor == 0) {
        return 1;
    }
    return valor * factorial(valor - 1);
}
…
```

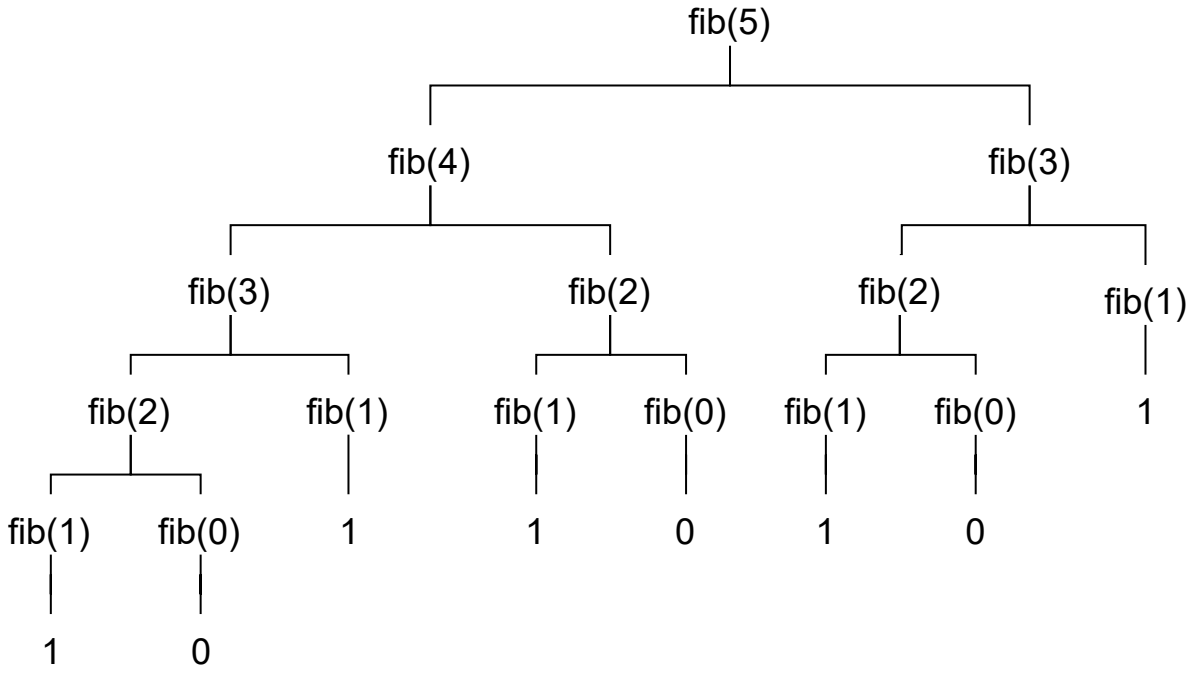
http://codesvalley.blogspot.com/2014/10/factorial-number-using-recursion.html

Recursividade

- A escrita de funções recursivas permite encontrar boas soluções para certos tipos de problemas e, devido ao menor número de instruções escritas, torna mais legível o código.
  - No entanto, em alguns casos, pode provocar perda de desempenho.

Fibonacci

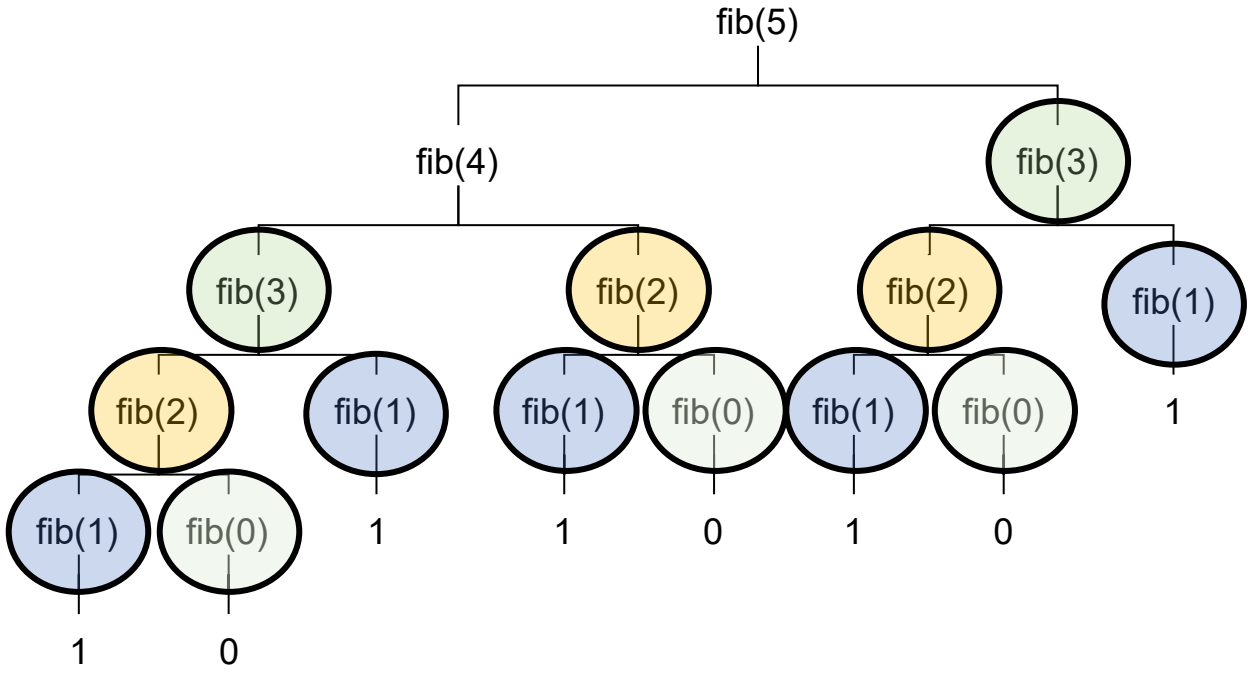
$$fib(n) = \begin{cases} 0 & se\ n = 0 \\ 1 & se\ n = 1 \\ fib(n-1) + fib(n-2) & se\ n > 1 \end{cases}$$



https://www.vadimbulavin.com/recursion-in-swift/

Fibonacci

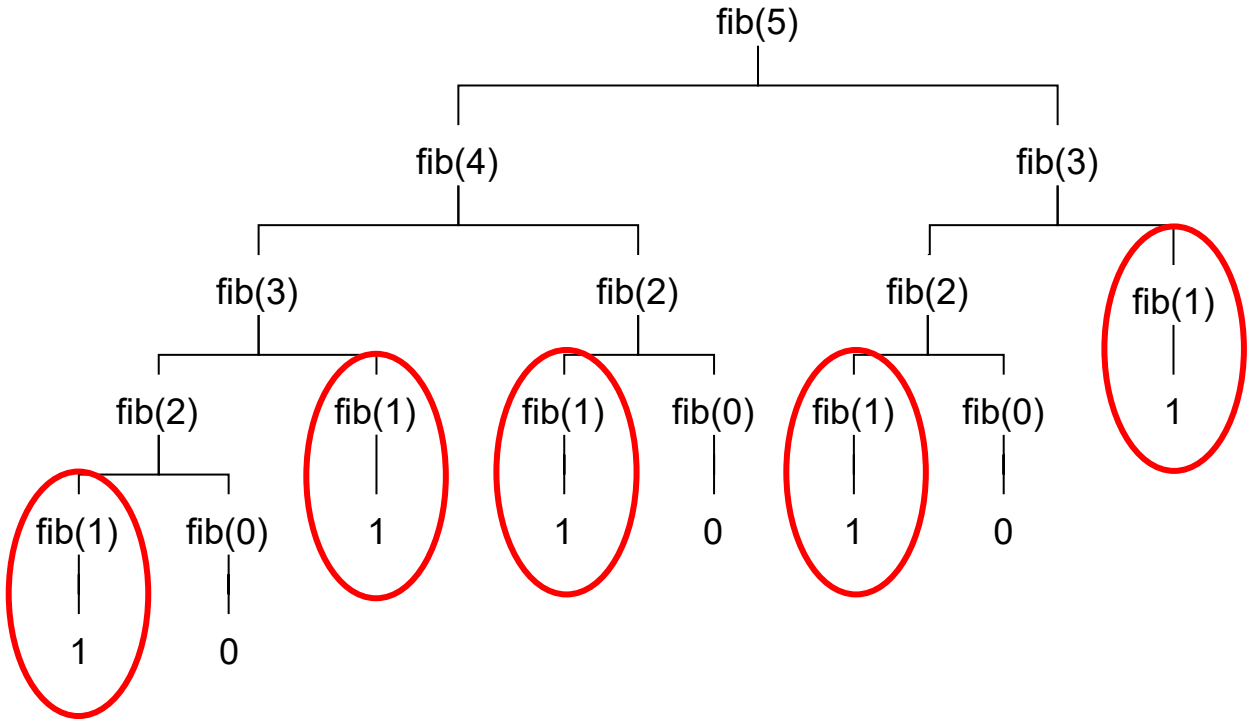
$$fib(n) = \begin{cases} 0 & se\ n = 0 \\ 1 & se\ n = 1 \\ fib(n-1) + fib(n-2) & se\ n > 1 \end{cases}$$



https://www.vadimbulavin.com/recursion-in-swift/

Fibonacci

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



https://www.vadimbulavin.com/recursion-in-swift/

Leitura recomendada

- (Capítulo 9) Damas, L. Linguagem C; FCA – Editora de Informática, Lda, 1999; ISBN 9789727221561.