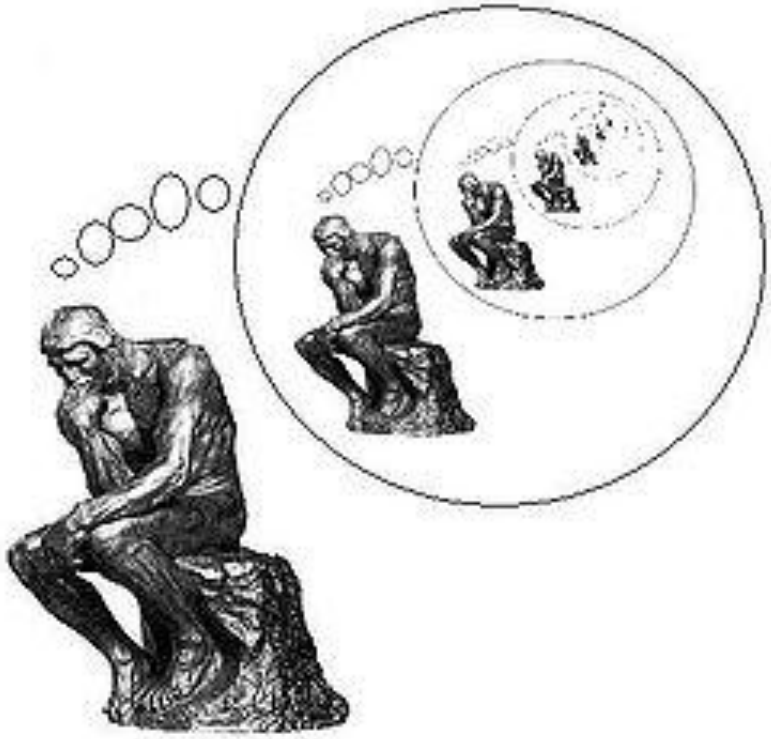
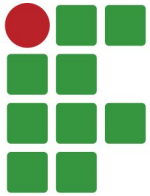
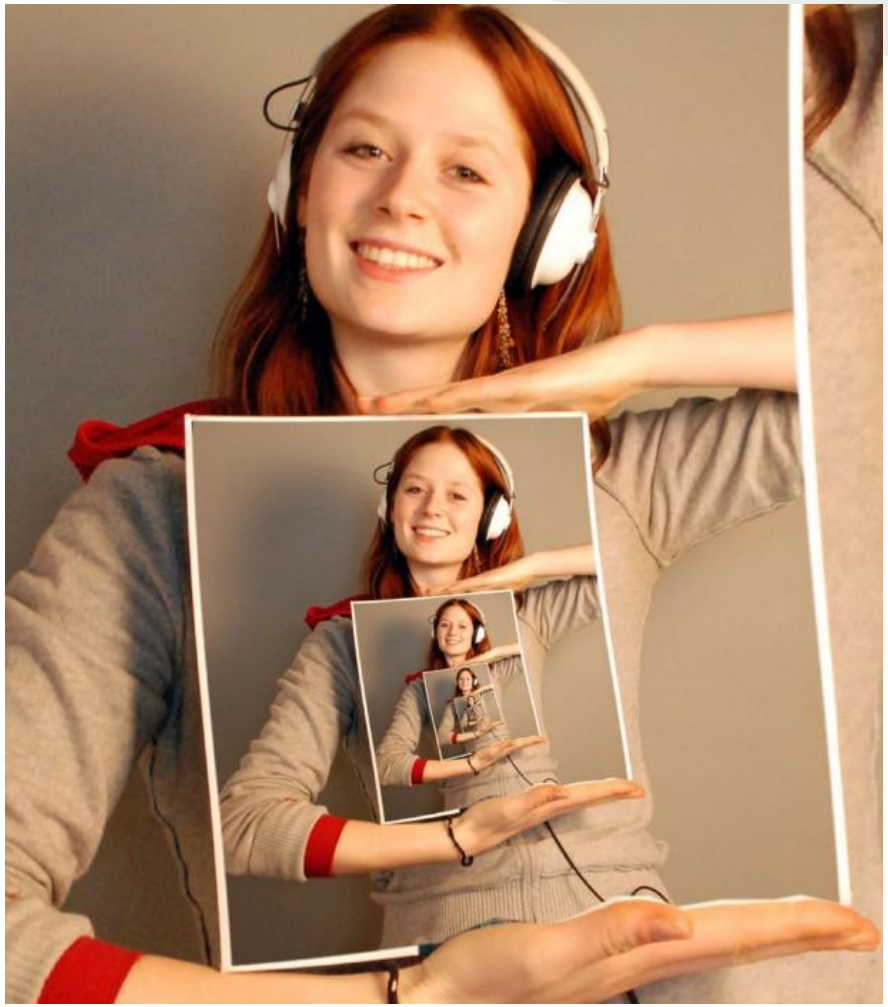
Recursividade

- “Para entender a recursividade, nós devemos primeiro entender a recursividade” (Anônimo)



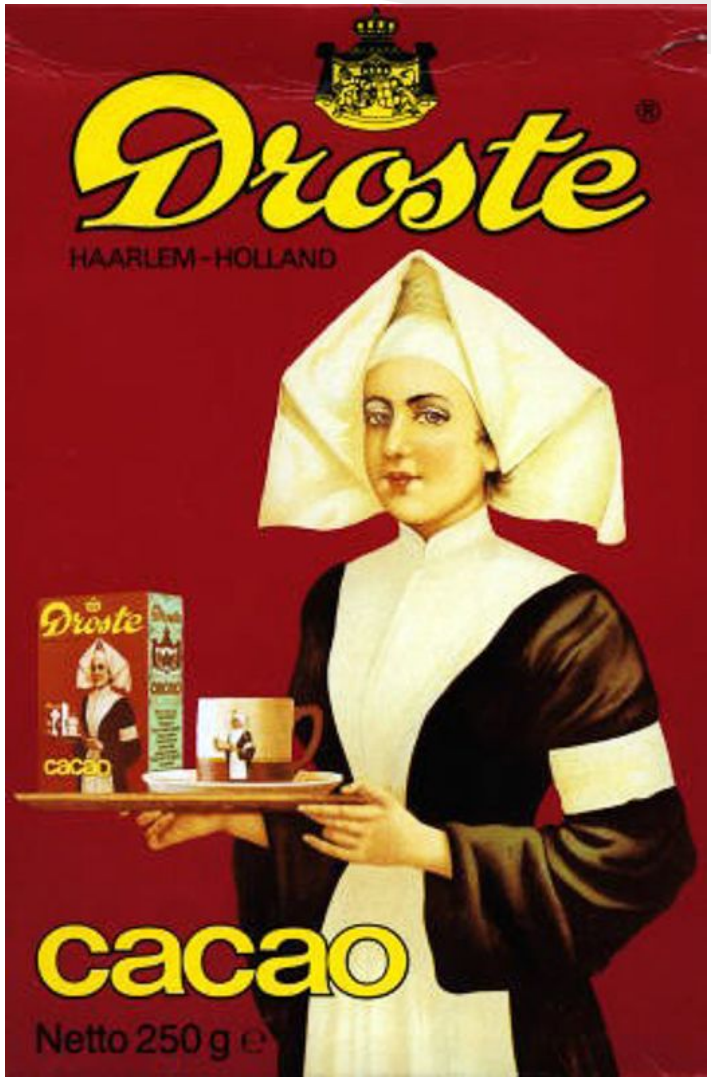


Recursividade



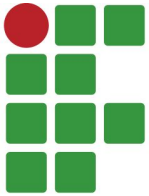
Efeito Droste

Recursividade



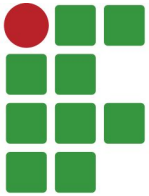
Recursividade

- Fundamental em Matemática e Ciência da Computação
  - Um programa recursivo é um programa que chama a si mesmo.
- Exemplos
  - Potência, função fatorial, árvore.
- Conceito poderoso
  - Define conjuntos infinitos com comandos finitos.



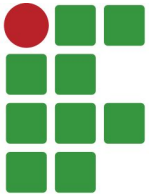
Recursividade

- A recursividade é uma estratégia que pode ser utilizada sempre que o cálculo de uma função para o valor n, pode ser descrito a partir do cálculo desta mesma função para o termo anterior (n-1).
- Exemplo: Potência ***p*** para número ***n***



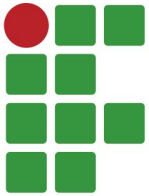
Recursividade

- Definição: dentro do corpo de uma função, chamar novamente a própria função
- **recursão direta**: a função A chama a própria função A
- **recursão indireta**: a função A chama uma função B que, por sua vez, chama A



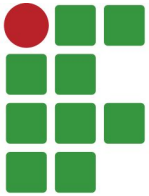
Condição de Parada

- Nenhum programa nem função pode ser exclusivamente definido por si;
- Um programa seria um loop infinito;
- Uma função teria definição circular;
- **Condição de parada**
- Permite que o procedimento pare de se executar (solução básica)

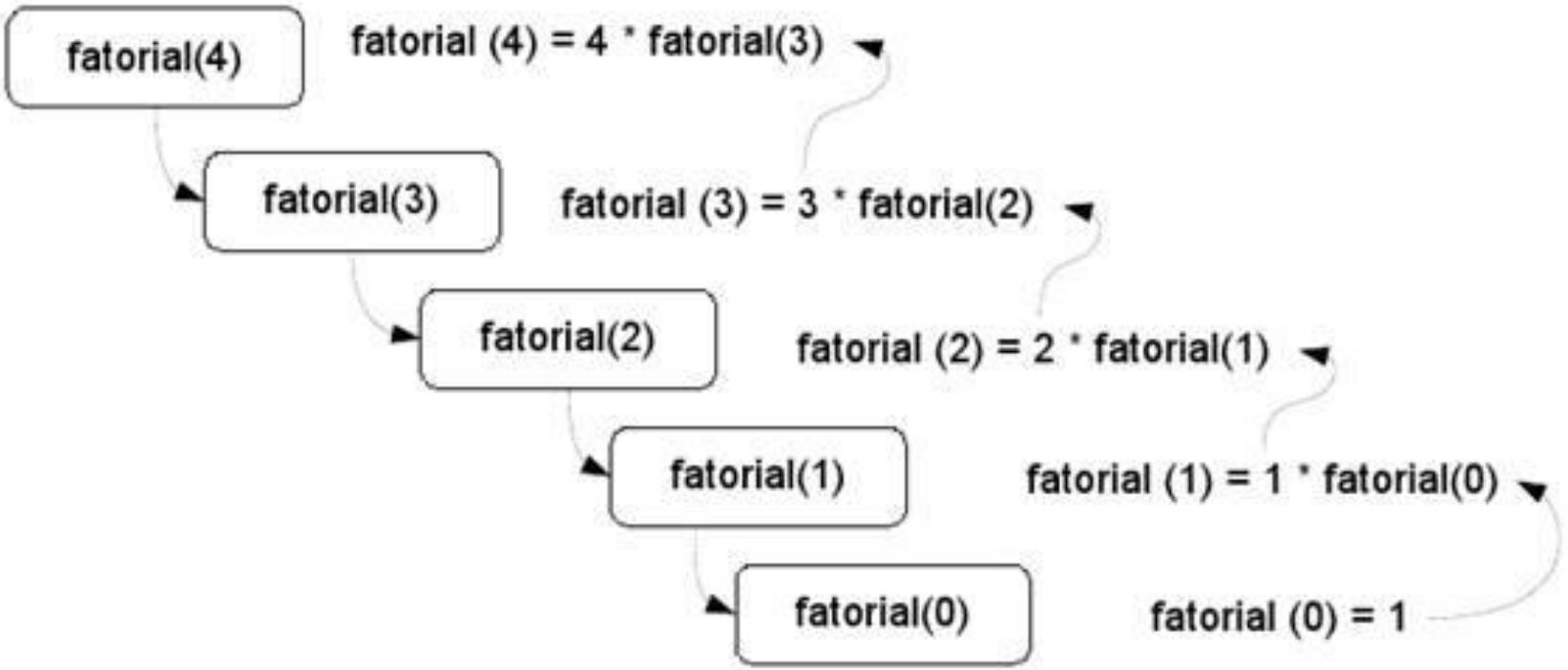


Funcionamento na Memória

- Para cada chamada de uma função, recursiva ou não, os parâmetros e as variáveis locais são empilhados na pilha de execução.
- Internamente, quando qualquer chamada de função é feita dentro de um programa, é criado um Registro de Ativação na Pilha de Execução do programa
- O registro de ativação armazena os parâmetros e variáveis locais da função bem como o “ponto de retorno” no programa ou subprograma que chamou essa função.
- Ao final da execução dessa função, o registro é desempilhado e a execução volta ao subprograma que chamou a função.

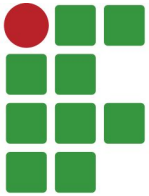


Funcionamento na Memória



Quando usar recursividade?

- Problema naturalmente recursivo e a versão recursiva não gera ineficiência evidente, se comparada à versão interativa;
- O algoritmo se torna compacto sem perda de clareza;
- É possível prever que o número de chamadas (consequentemente alocação na pilha) não vai provocar interrupções no processo.
- Exemplos:
  - Dividir para Conquistar (Ex. Quicksort)
  - Caminhamento em Árvores (pesquisa, backtracking)
  - Torre de Hanoi



Quando não usar recursividade?

- Solução recursiva causa ineficiência;
- Existe uma única chamada do procedimento;
- O uso de recursão acarreta número maior de cálculos que a versão interativa;
- Não é possível prever o número de chamadas que podem causar estouro de pilha (*stack overflow*).