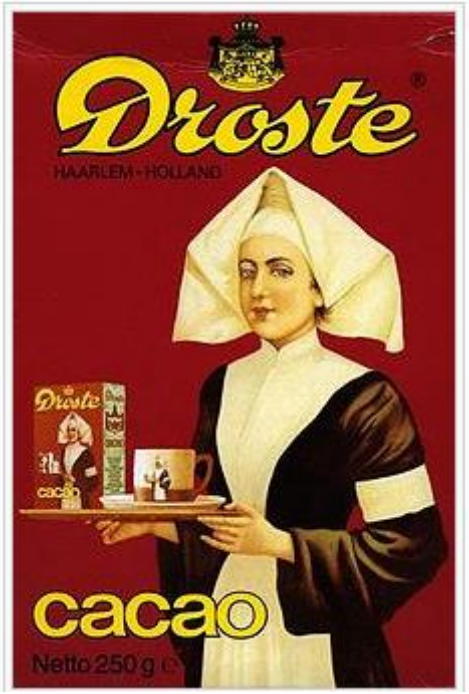
PP
Recursão

Recursão

Recursão no nosso dia-a-dia

- Exemplos:
  - Imagem recursiva – efeito Droste (imagem dentro de outra)

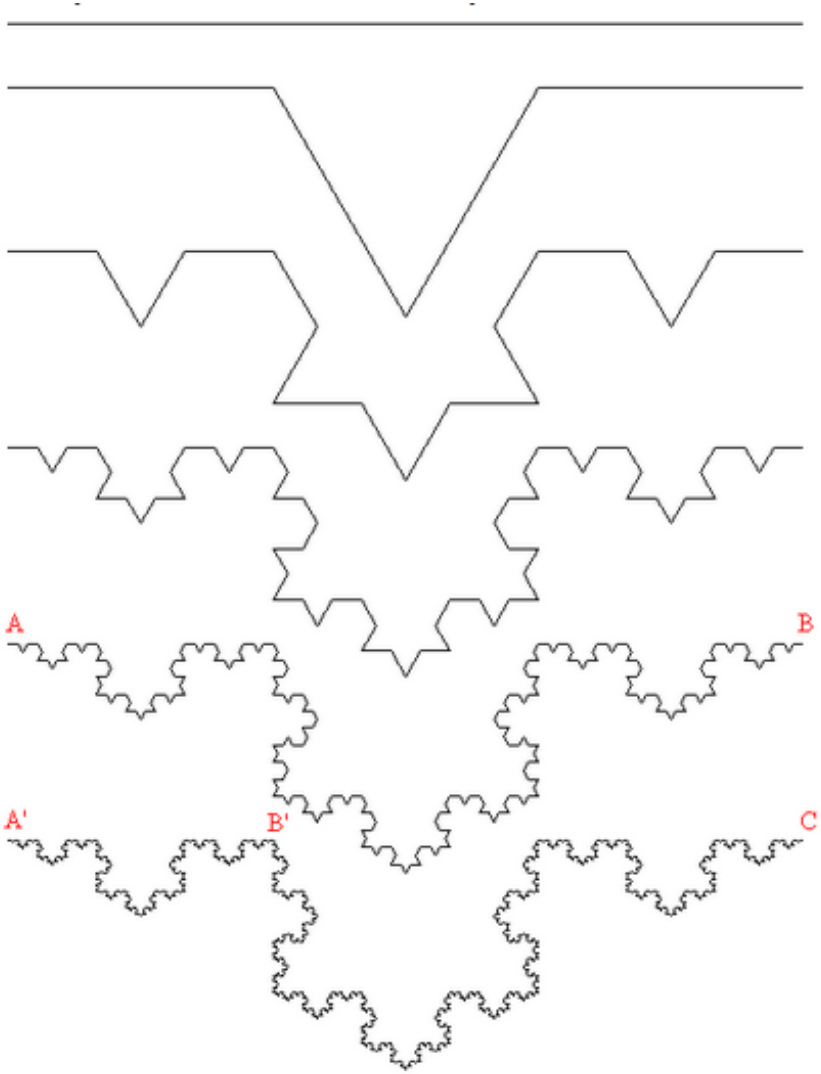
    Como efeito causado por um espelho na frente de outro espelho
  - Definição de número natural
  - Acrônimos recursivos
  - Fractais
  - Triângulo de Sierpinski
  - Couve-flor



Recursão no nosso dia-a-dia

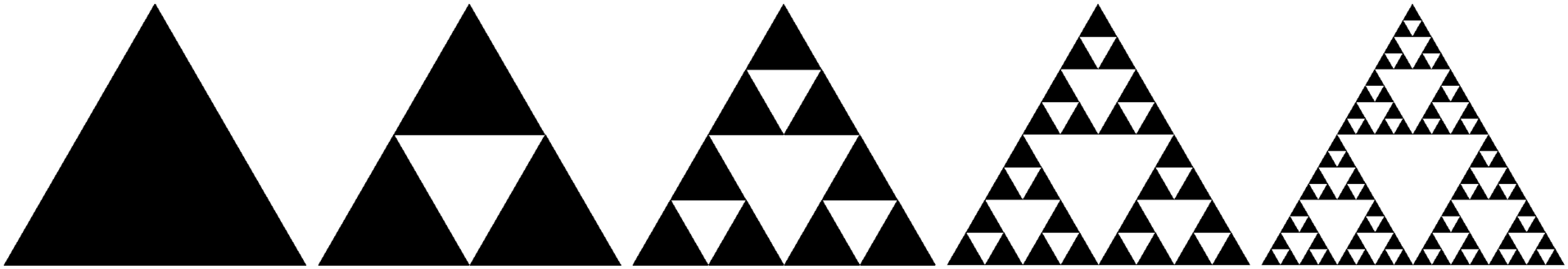
- Exemplos:
  - Definição de número natural:
    - 0 é um número natural.
    - Se n é um número natural, então n+1 (sucessor de um número natural) também é um número natural.
  - Acrônimos recursivos
    - **GNU** – **G**nu is **N**ot **U**nix
    - **PHP** – **P**HP: **H**ypertext **P**reprocessor
      originalmente: Personal Home Page
    - **BING** – **B**ing **I**s **N**ot **G**oogle
  - Fractais:
    - Curva de Koch

Curva de Koch



Recursão no nosso dia-a-dia

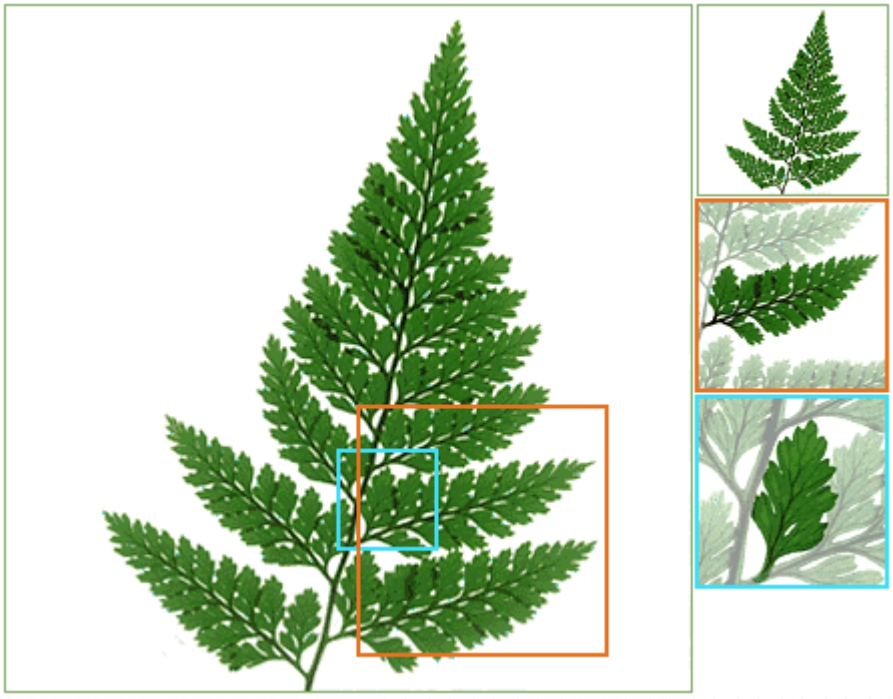
- Triângulo de Sierpinski



- Couve-flor

Recursão no nosso dia-a-dia

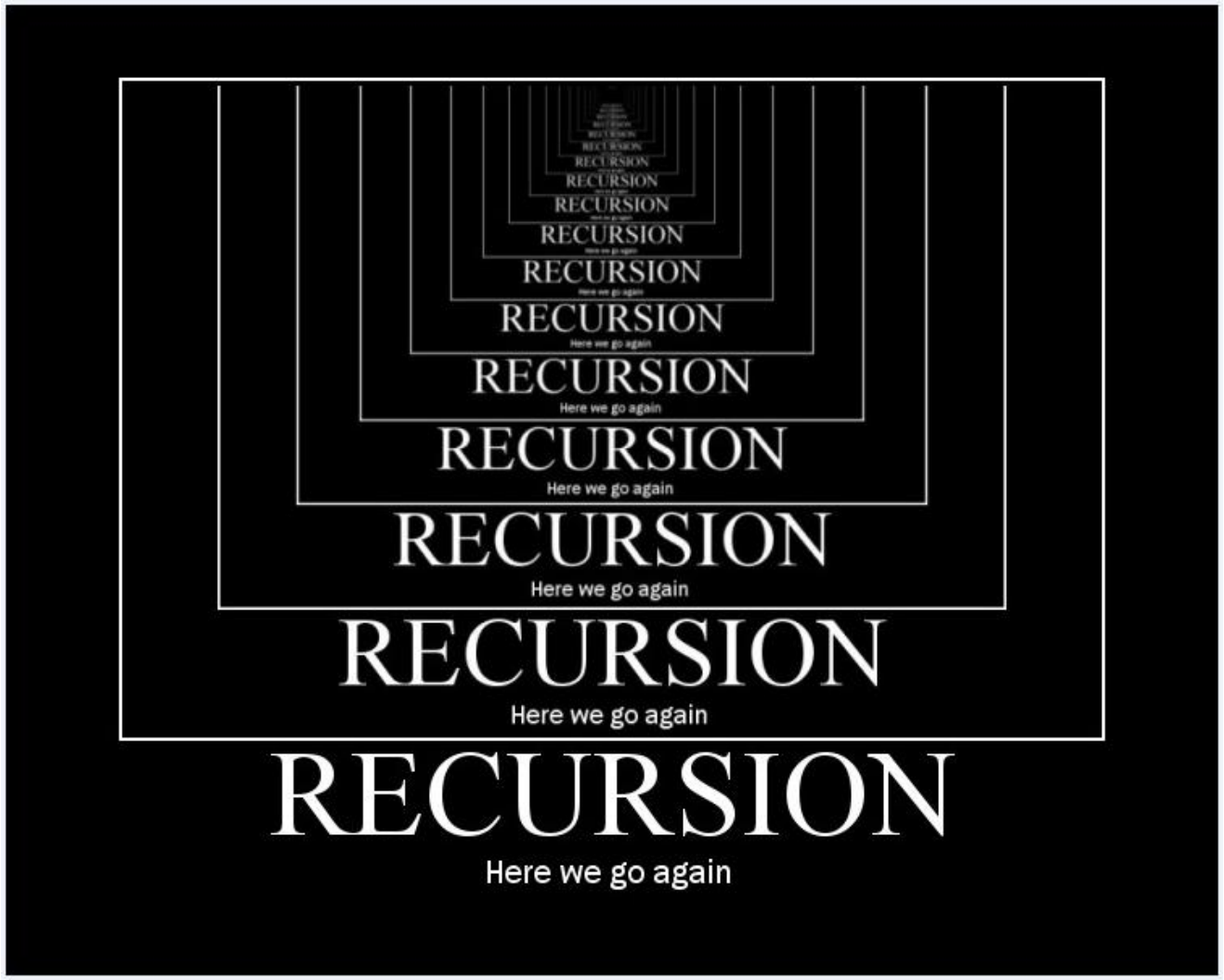
- Samambaia



Recursão

- Técnica poderosa da matemática
- Permite definir um elemento em função de um caso “mais simples” dele mesmo
- Algoritmo recursivo
  - Algoritmo que chama a si mesmo, de forma direta ou indireta
  - Apropriado quando o problema a ser resolvido pode ser definido em termos recursivos
  - Exemplo: cálculo de potência ou fatorial

Recursão



Exemplo: Potência

- Potência positiva *(exp >= 0)* de um número a
  - $a^{exp} = 1$ *se* $exp=0$
  - $a^{exp} = a * a^{exp-1}$ *se* $exp > 0$
- Exemplos:
  - a = 3, exp = 0
    $3^0 = 1$
  - a = 3, exp = 1
    $3^1 = 3 * 3^0 = 3 * 1 = 3$
  - a = 3, exp = 2
    $3^2 = 3 * 3^1 = 3 * 3 * 3^0 = 3 * 3 * 1 = 9$
  - a = 3, exp = 3
    $3^3 = 3 * 3^2 = 3 * 3 * 3^1 = 3 * 3 * 3 * 3^0 = 3 * 3 * 3 * 1 = 27$
  - a = 3, exp = 4
    $3^4 = 3 * 3^3 = 3 * 3 * 3^2 = 3 * 3 * 3 * 3^1 = 3 * 3 * 3 * 3 * 3^0 =$
    $= 3 * 3 * 3 * 3 * 1 = 81$

Exemplo: Potência

- Potência positiva *(exp >= 0)* de um número a
  - $a^{exp} = 1$ *se exp=0* *(condição de parada ou caso básico)*
  - $a^{exp} = a * a^{exp-1}$ *se exp > 0* *(parte recursiva)*
- Características:
  - Condição de parada ou caso básico:
    - Algum evento que encerra a autochamada consecutiva. No caso da potência, isso ocorre quando a função é chamada com parâmetro 0 (zero).
    - Um algoritmo recursivo precisa garantir que esta condição exista e que ela seja alcançada em algum momento, para evitar um loop infinito.
  - Mudança de estado
    - A cada chamada, deve haver uma diferença entre a chamada sendo executada e a próxima , de forma que a execução deva convergir para o caso básico.
    - No caso da potência, o expoente é decrementado a cada chamada.

Exemplo: Potência

```
static int Pot(int a, int exp)
{
    if(exp==0)
        return 1;
    else
        return a * Pot(a,exp-1);
}
static void Main (string[] args)
{
    Console.WriteLine(Pot(3,4));
}
```

Simulação de Pot (3,4)

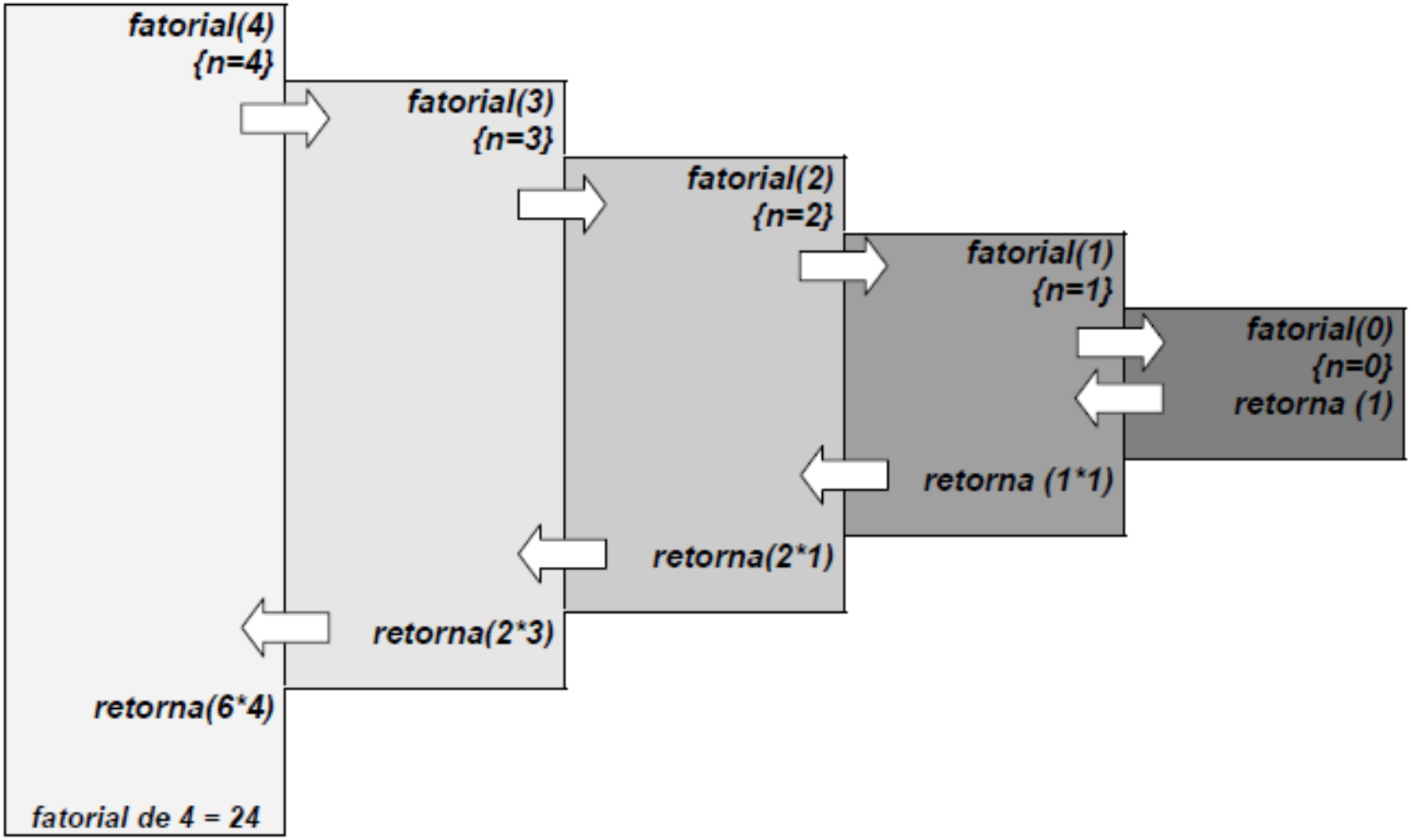
```
Pot(3,4)
 Pot é chamada com a=3, exp=4
 |return 3*Pot(3,3)
 |          Pot é chamada com a=3, exp=3
 |          |return 3*Pot(3,2)
 |          |          Pot é chamada com a=3, exp=2
 |          |          | return 3*Pot(3,1)
 |          |          |           Pot é chamada com a=3, exp=1
 |          |          |           |return 3*Pot(3,0)
 |          |          |           |          Pot  a=3, exp=0
 |          |          |           |          |return 1
 |          |          |           |return 3*1 = 3
 |          |          |return 3*3 = 9
 |          |return 3*9 = 27
 |return 3*27 = 81
```

Exemplo: Fatorial

- Fatorial – definição:
  - *0! = 1* *(parte básica)*
  - *n! = n * (n-1)!* *(parte recursiva)*
- Algoritmo:

```
static int Fatorial(int n){
   if (n==0)
        return 1;
   else
        return n*Fatorial(n-1);
}
static void Main(string[] args) {
   Console.WriteLine(Fatorial(4));
}
```

Simulação: Fatorial(4)



Recursão x Iteração

- Os problemas resolvidos por algoritmos recursivos podem também ser resolvidos de forma iterativa (através de um loop ou laço)
- Exemplo: versão iterativa para o Fatorial

```
static int FatIterativo(int n){
    int resultado=1;
    while (n >= 1) {
          resultado = resultado * n;
          n--;
    }
    return resultado;
}
```

Recursão x Iteração

- Tanto iteração como recursão se baseiam em uma instrução de controle:
  - Iteração utiliza uma instrução de repetição (por exemplo: for, while ou do…while).
  - Recursão utiliza uma instrução de seleção (por exemplo: if, if…else ou switch)
- Ambas envolvem repetição:
  - A iteração utiliza explicitamente uma instrução de repetição.
  - A recursão alcança a repetição por meio de chamadas sucessivas do próprio método.

Recursão x Iteração

- Tanto uma como outra envolvem um teste de terminação:
  - A iteração termina quando a condição de continuação do loop falha.
  - A recursão termina quando um caso básico é alcançado.
- Loop infinito pode ocorrer em ambos:
  - Um loop infinito ocorre com iteração se o teste de continuação do loop nunca se tornar falso.
  - A recursão infinita ocorre se o passo de recursão não reduzir o problema sempre em uma maneira que convirja para o caso básico, ou se o caso básico não for testado.

Recursão x Iteração

- Eficiência
  - A versão iterativa normalmente é mais eficiente do que a recursiva
  - A recursão envolve várias chamadas consecutivas ao próprio algoritmo, acarretando num maior consumo de tempo e memória do que a versão iterativa
- Clareza
  - Muitas vezes, a versão recursiva apresenta uma maior clareza do que o correspondente iterativo

Exemplo: Algoritmos recursivos de impressão de um vetor

```
static void ImprimeVet (int[] v, int n){
      if (n == v.Length)
          return;
       else {
           Console.WriteLine(v[n]);  // 1 - imprime o elemento corrente
           ImprimeVet(v, n+1);        // 2 - chama para imprimir o resto
      }
}
// imprime o vetor invertido
static void ImprimeVetInv (int[] v, int n){
      if (n < 0)
          return;
       else {
           Console.WriteLine(v[n]);  // 3 - imprime o elemento corrente
           ImprimeVetInv(v, n-1);     // 4 - chama para imprimir o resto
      }
}
static void Main(string[] args){
      int[] vetor = {10,15,20,27,35};
     ImprimeVet(vetor, 0);  // imprime vetor no sentido normal
      ImprimeVetInv(vetor, vetor.Length-1); // imprime vetor invertido
}
```

Exemplo: Algoritmos recursivos de impressão de um vetor

- No algoritmo ImprimeVet anterior (o 1º), se invertermos as linhas 1 e 2, o algoritmo passará a imprimir o vetor de forma invertida, pois:
  - primeiro chamará o método recursivamente para imprimir o restante,
  - e depois imprime o elemento corrente.
- No 2º algoritmo também, se invertermos as linhas 3 e 4, o algoritmo deixará de imprimir o vetor de forma invertida

Recursão indireta: Exemplo

- Recursão indireta
  - Quando um algoritmo A chama um B, que por sua vez chama novamente o algoritmo A
  - Exemplo:
    - algoritmo que verifica se um número é par e algoritmo que verifica se um número é ímpar
    - Definição tradicional de par e ímpar:
      - Número é par quando for divisível por 2
      - Número é ímpar quando não for divisível por 2
    - Definição recursiva de par e ímpar:
      - Número n>1 é par se n-1 for ímpar; 0 é par, 1 é ímpar
      - Número n>0 é ímpar se n-1 é par; 1 é ímpar, 0 é par

Recursão indireta: exemplo

```
Método Impar(int n)
  /* devolve true se n é ímpar
            e false caso contrário */
  Se n = 0
   então devolve false
  senão se n = 1
            então devolve true
            senão devolve par(n-1)
```

Recursão indireta: exemplo

```
Método Par(int n)
  /* devolve true se n é par
            e false caso contrário */
  Se n = 0
   então devolve true
  senão se n = 1
            então devolve false
            senão devolve impar(n-1)
```