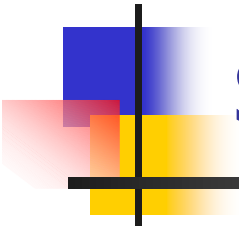


SIN211 Algoritmos e Estruturas de Dados

Prof. João Batista Ribeiro

joao42lbatista@gmail.com

UFV

Universidade Federal de Viçosa

Slides baseados no material da Prof.ª Rachel Reis

Recursividade - Revisão

- É o processo de resolução de um problema, reduzindo-o em um ou mais subproblemas com as seguintes características:
  - São idênticos aos problemas originais;
  - São mais simples de resolver.

- Uma vez realizada a primeira subdivisão, a mesma técnica de decomposição é usada para dividir cada subproblema.

- Eventualmente, os subproblemas tornam-se tão simples que é possível resolvê-los sem efetuar novas subdivisões.

Recursividade - Revisão

- Recursão é usado desde arte (em figuras, telas, etc), em matemática e em programação

- Exemplo de uma definição recursiva sobre ancestralidade:

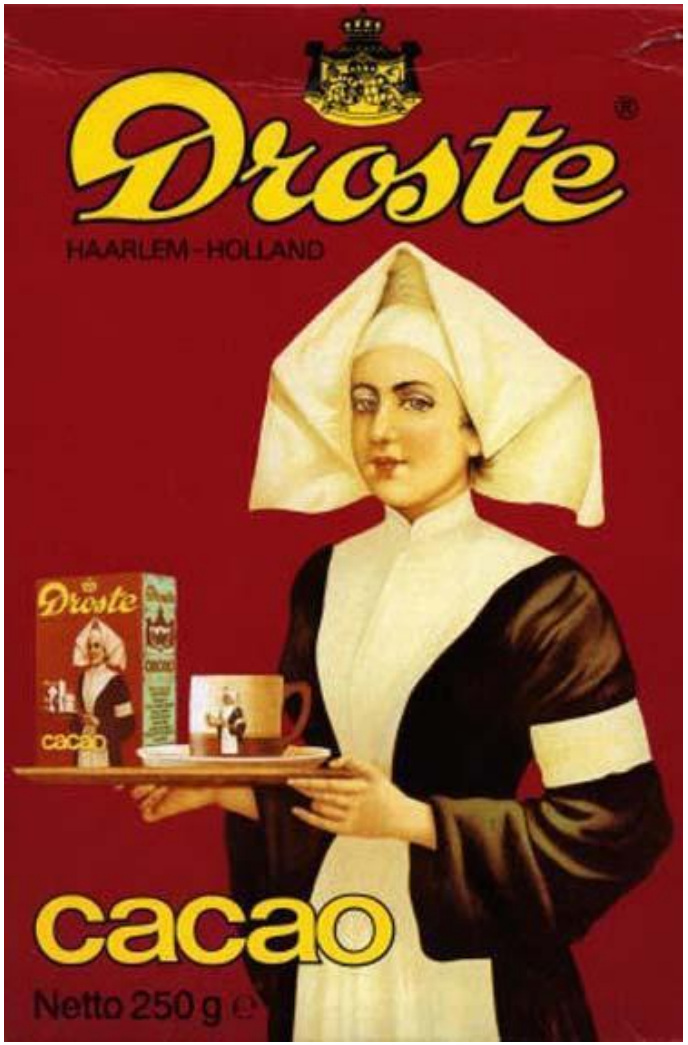
  - (caso base) Os pais de uma pessoa são seus antepassados

  - (passo recursivo) Os pais de qualquer antepassado de uma pessoa são também antepassados desta pessoa

Recursividade - Revisão

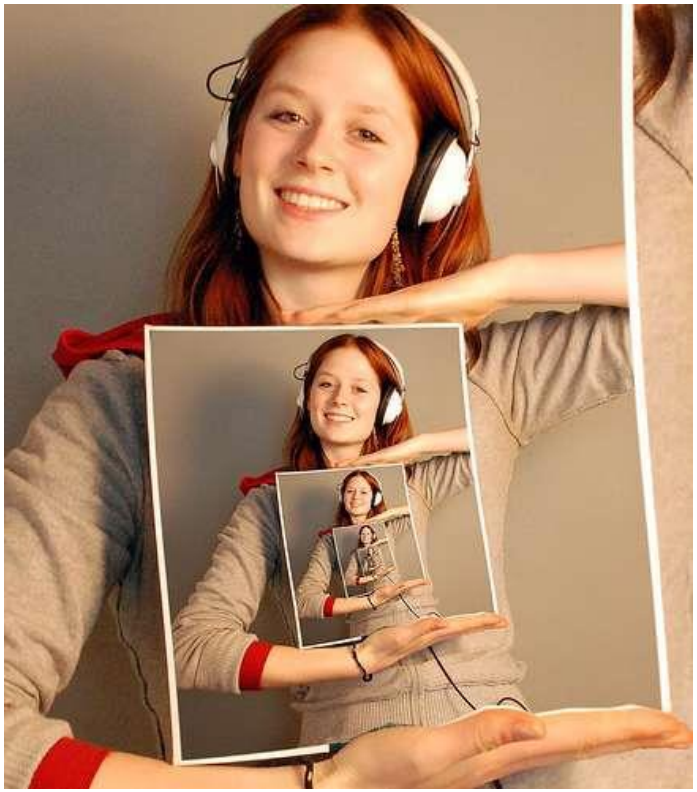
- Em figuras, é usado quando a figura contem ela mesma. Isto gera um efeito chamado de efeito “Droste”

- O nome veio de um produto holandês (cacau em pó), cuja embalagem possui figura recursiva



Recursividade - Revisão

- Este tipo de efeito é frequentemente usado em fotos e álbuns como o Ummagumma (Pink Floyd)



Recursividade - Revisão

- Também pode ser usado para obter um sonho recursivo

Recursividade - Revisão

- No Brasil também temos um produto com figura recursiva

Recursividade - Revisão

- Em termos de programação uma função é dita recursiva quando ela chama a si mesma.

Recursividade

- Exemplo 1 (cálculo do fatorial)

  Como se calcula o fatorial do número 4?

$$4! = 4 * 3 * 2 * 1$$

Recursividade

- Exemplo 1 (cálculo do fatorial)

Dado que:

0! = 1
1! = 1
2! = 2 * 1
3! = 3 * 2 * 1
4! = 4 * 3 * 2 * 1

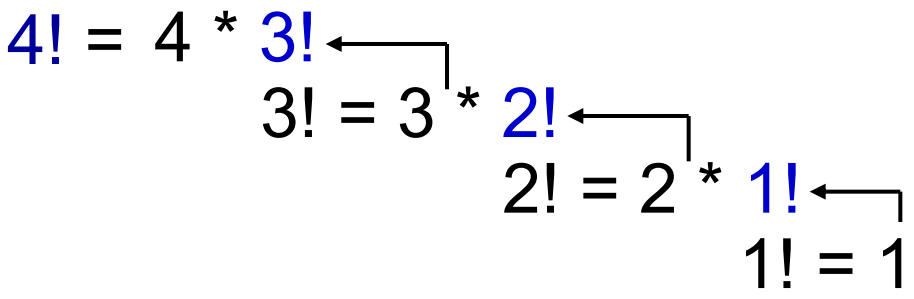
1! = 1
2! = 2 * 1!
3! = 3 * 2 !
4! = 4 * 3 !

Recursividade

- Exemplo 1 (cálculo do fatorial)

```
int fatorial (int n) {
   if (n <= 1)     // caso base
      return (1);
   else
      return (n * fatorial (n-1));  // passo recursivo
}
```

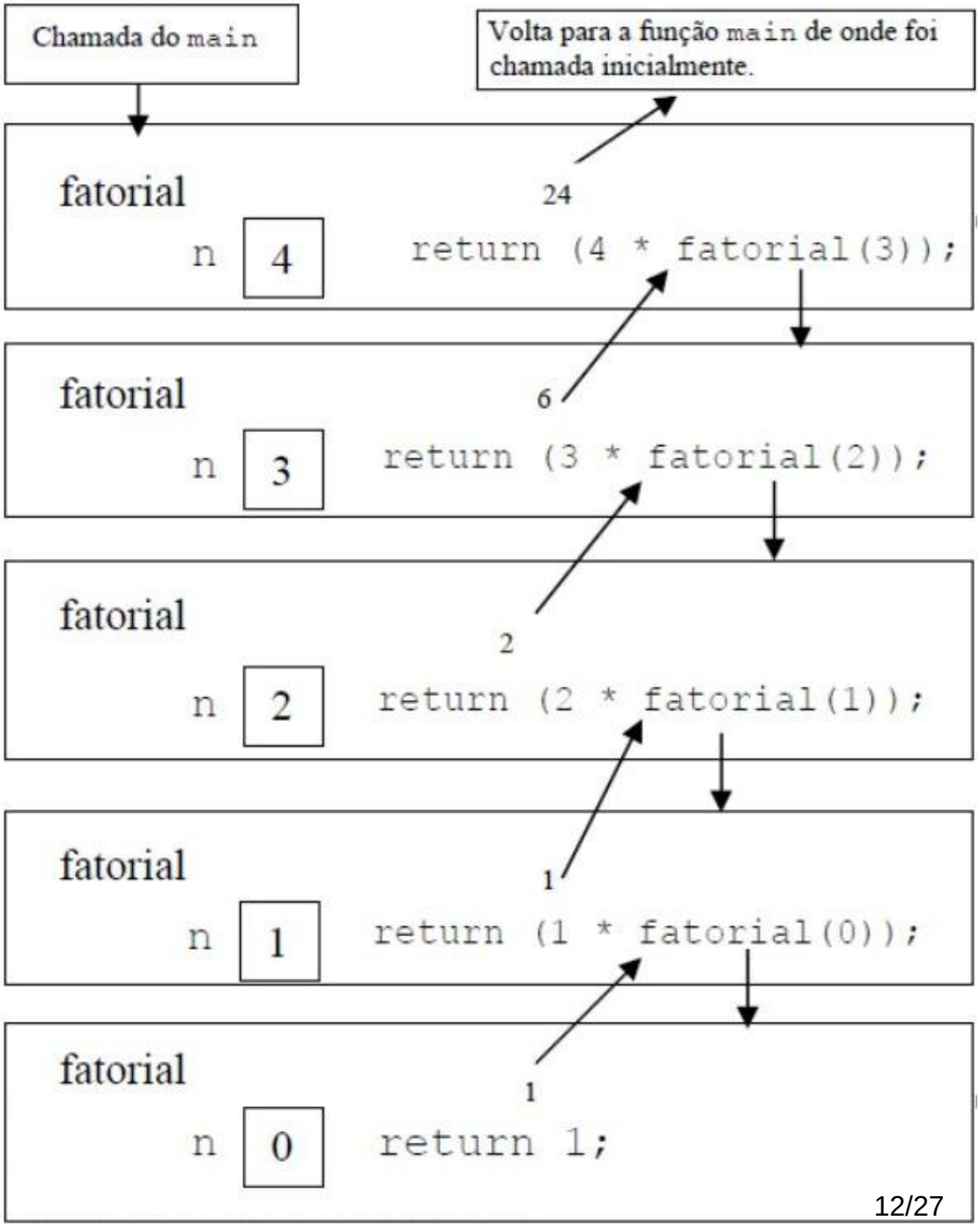
Calcule o fatorial de 4:



Exemplo (cálculo do fatorial)

Calcule o fatorial de 4:

4! = 4 * 3!
3! = 3 * 2!
2! = 2 * 1!
1! = 1



Recursividade

- Exemplo 1 (cálculo do fatorial)

- Representação matemática:

$$n! \begin{cases} 1 & se \quad n \leq 1 \quad e \\ n.(n-1)! & se \quad n > 1 \end{cases}$$

```
int fatorial (int n) {
   if (n <= 1)     // caso base
      return (1);
   else
      return (n * fatorial (n-1));  // passo recursivo
}
```

Recursividade

- Exemplo 1 (cálculo do fatorial)

RECURSIVO

```
int fatorial(int n) {
      if (n <= 1) {
            return 1;
      } else {
           return n * fatorial(n-1);
      }
}
```

ITERATIVO

```
int fatorial(int n) {
       int i, temp = 1;
       for (i=n; i>1; i--) {
             temp = temp * i;
       }
       return temp;
}
```

Recursividade

- A solução iterativa é, em geral, mais rápida do que a recursiva.
- A solução recursiva pode também ocupar muito mais espaço na memória, pois precisa armazenar em uma pilha cada resultado antes de resolvê-lo.
- Em alguns problemas a solução recursiva é muito mais prática, e em algumas linguagens de programação, só é possível fazer repetição por meio de recursividade. Por exemplo, a linguagem LISP, do paradigma funcional.

Recursividade

- Uma função recursiva deve obrigatoriamente ter um **critério de parada**.

- A parada da recursividade se dá pelo caso base (que não possui recursão).

Recursividade

- Exemplo 2: (cálculo do somatório)

Qual o somatório de [2, 5]?

somatorio(2,5) = 2 + 3 + 4 + 5

Recursividade

- Exemplo 2 (cálculo do somatório)

```
int somatorio (int m, int n) {
   if (n == m)  // caso base
      return (m);
   else
      return (m + somatorio (m+1, n)); // passo recursivo
}
```

Dado S(m,n), onde n > m, calcule S(2 , 5):

S(2, 5) =2 + S(3, 5)

S(3, 5) = 3 + S(4, 5)

S(4,5) = 4 + S(5,5)

S(5,5) = 5

Recursividade

- Exemplo 2 (cálculo do somatório)

- Representação matemática:

$$\sum_{k=m}^{n} = \begin{cases} m & \text{se } n = m \text{ e} \\ m + \sum_{k=m+1}^{n} & \text{se } n > m. \end{cases}$$

```
int somatorio (int m, int n) {
   if (n == m)   // caso base
      return (m);
   else
      return (m + somatorio (m+1, n));  // passo recursivo
}
```

Recursividade

Dado o código recursivo abaixo:

```
int somatorio (int m, int n) {
      if (n == m) // caso base
            return (m);
      else
            return (m + somatorio (m+1, n));  // passo recursivo
}
```

Como seria o código iterativo?

Recursividade

- Exemplo 3: (cálculo da potência)

Qual a potência de 3^4

potencia(3,4) = 3 * 3 * 3 * 3

Dado que:

$3^0 = 1$
$3^1 = 3$
$3^2 = 3 * 3$
$3^3 = 3 * 3 * 3$
$3^4 = 3 * 3 * 3 * 3$

$3^0 = 1$
$3^1 = 3 * 3^0$
$3^2 = 3 * 3^1$
$3^3 = 3 * 3^2$
$3^4 = 3 * 3^3$

Recursividade

- Exemplo 3 (cálculo da potência)

```
int potencia (int x, int n) {
   if (n == 1)     // caso base
        return (n);
   else if(n > 0)
        return (x * potencia (x, n-1));// passo recursivo
}
```

Dado P(x, n) calcule a potência de P(3,4):

P(3, 4) = 3 * P(3, 3)

P(3, 3) = 3 * P(3, 2)

P(3,2) = 3 * P(3,1)

P(3,1) = 3

Recursividade

- Exemplo 3 (cálculo da potência)

- Representação matemática:

$$x^n = \begin{cases} 1/x^n & \text{se } n < 0, \\ 1 & \text{se } n = 0 \text{ e} \\ x \times x^{n-1} & \text{se } n > 0. \end{cases}$$

```
int potencia (int x, int n) {
   if (n == 0)     // caso base
         return (1);
   else if(n > 0)
         return (x * potencia (x, n-1)); // passo recursivo
}
```

Recursividade

Dado o código recursivo abaixo:

```
int potencia (int x, int n) {
   if (n == 0) // caso base
        return (1);
   else if(n > 0)
        return (x * potencia (x, n-1)); // passo recursivo
}
```

Como seria o código iterativo?

Recursividade

Outro exemplo clássico de recursividade:

A série de Fibonacci:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...

Qual a propriedade desta sequência?

Recursividade

A série de Fibonacci:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...

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

Qual seu código iterativo? E o recursivo?

Exercícios

Desenvolva um programa em Linguagem C que permita fazer as seguintes operações recursivas sobre uma lista simplesmente encadeada:

a) calcular o comprimento da lista

b) somar todos os elementos da lista

c) multiplicar todos os elementos da lista