Estrutura de Dados

Prof. Marcelo Zorzan
Prof[a]. Melissa Zanatta

Recursividade

- 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 e usada para dividir cada subproblema.

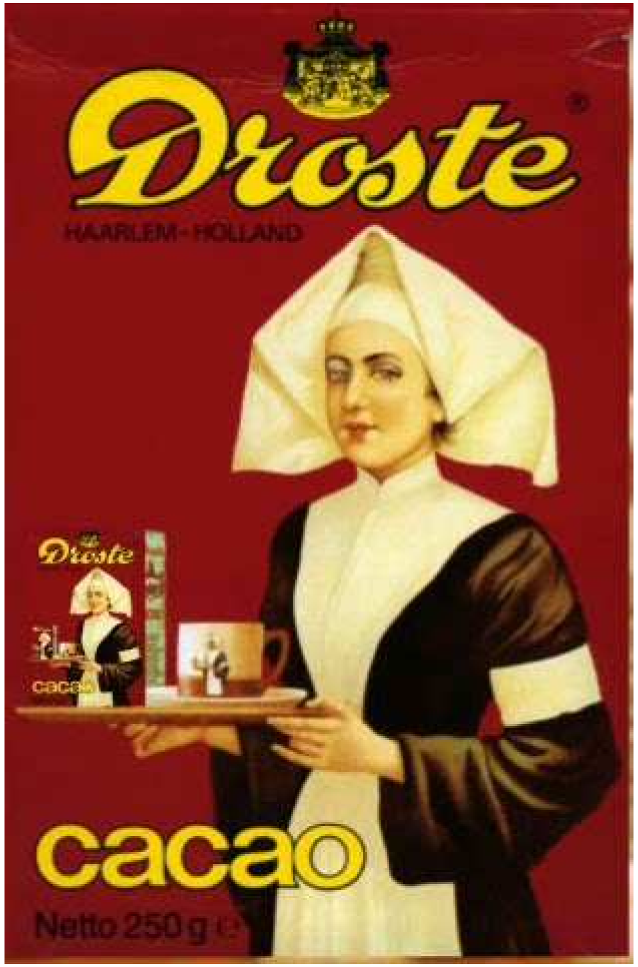
- Eventualmente, os subproblemas tornam-se tão simples que é possível resolvê-los sem efetuar novas subdivisões.

Recursividade

- Podemos aplicar o conceito de recursão:
  - arte (em figuras, telas, etc.),
  - matemática
  - programação

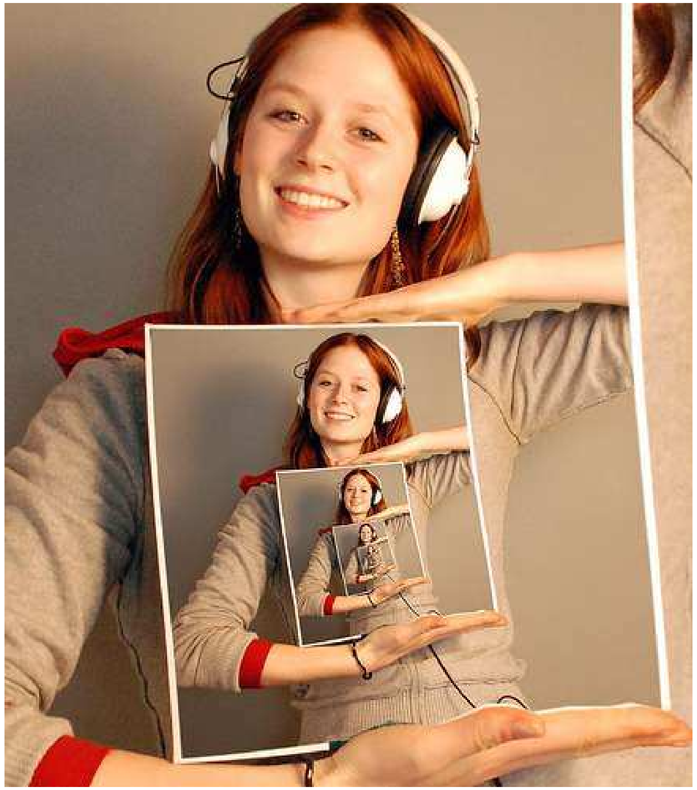
Recursividade

- Em figuras, é usado quando a figura contém 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



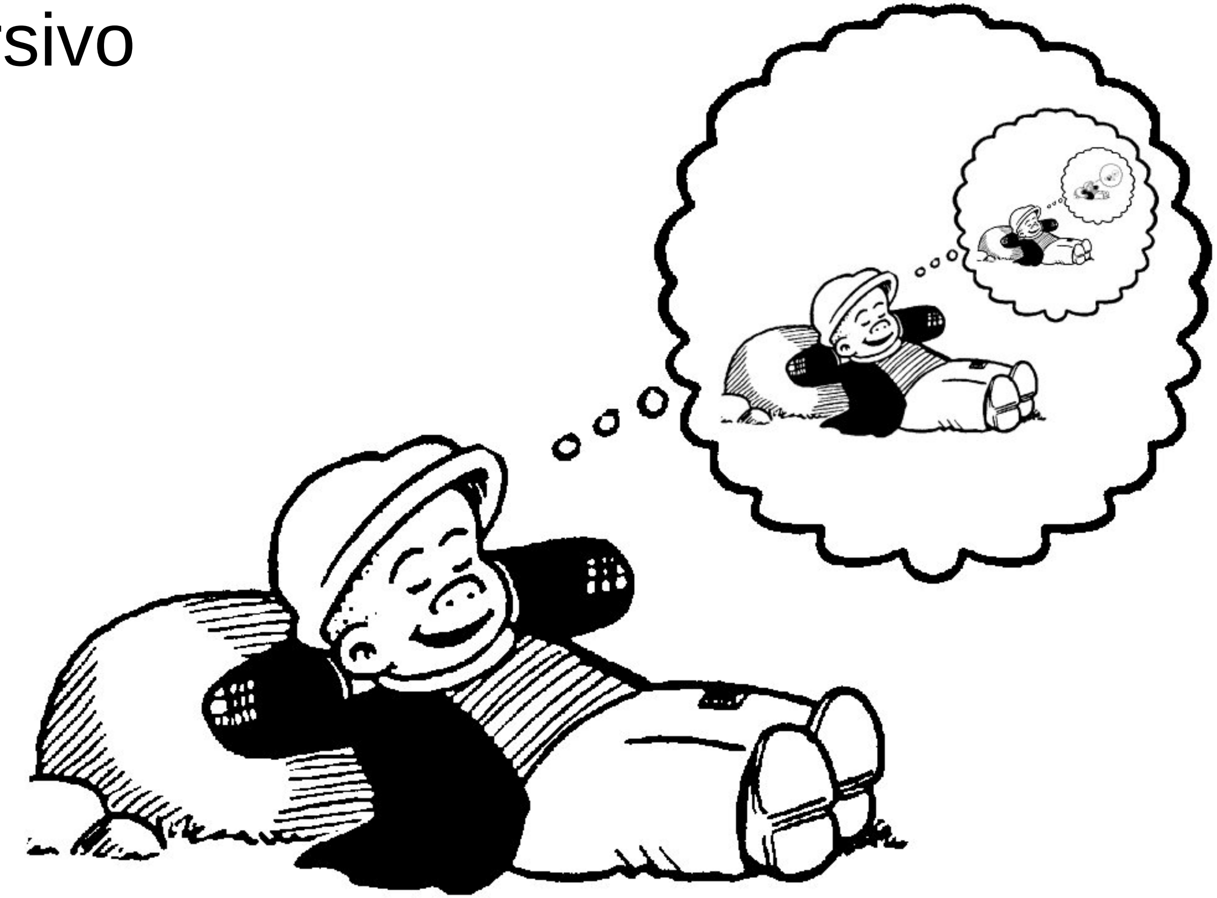
Recursividade

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



Recursividade

- Também pode ser usado para obter um sonho recursivo



Recursividade

- No Brasil também temos um produto com figura recursiva

Recursividade

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

Recursividade

- Exemplo 1 (cálculo do fatorial)

Como calcular 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:

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 – Programação

- 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

- Exemplo 3: (cálculo da potência)

- Qual o a potência de 3^4

$$pot(3,4) = 3 * 3 * 3 * 3$$

- Dado que:

$$\left.\begin{array}{l} 3^0 = 1 \\ 3^1 = 3 \\ 3^2 = 3 * 3 \\ 3^3 = 3 * 3 * 3 \\ 3^4 = 3 * 3 * 3 * 3 \end{array}\right\} \quad \begin{array}{l} 3^0 = 1 \\ 3^1 = 3 * 3^0 \\ 3^2 = 3 * 3^1 \\ 3^3 = 3 * 3^2 \\ 3^4 = 3 * 3^3 \end{array}$$

Recursividade

- Exemplo 3 (cálculo da potência)

```
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
}
```

- 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 * P(3,0)

P(3,0) = 1

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
}
```

Exercício

1) Desenvolva uma função recursiva para imprimir os elementos de uma lista simplesmente encadeada de trás para frente.