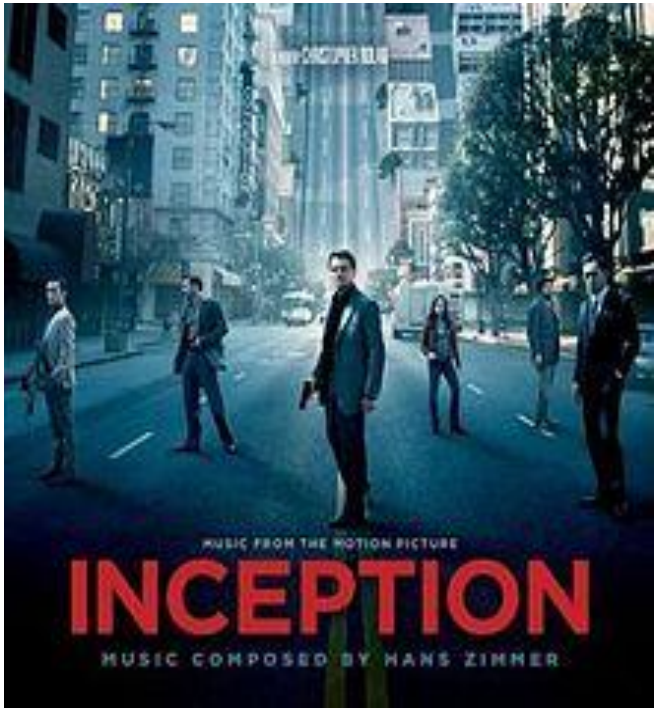
Introdução a Técnicas de Programação

Aula 11
Recursão

Conceito de recursão

Filme “A Origem”

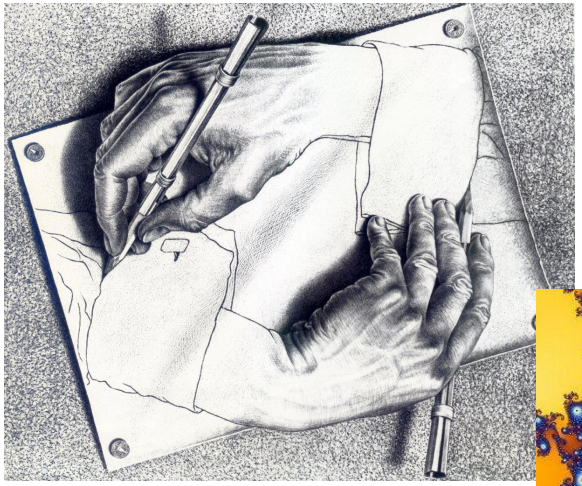
- Realidade → mundo dos sonhos
- Mundo dos sonhos é “real” pra quem sonha



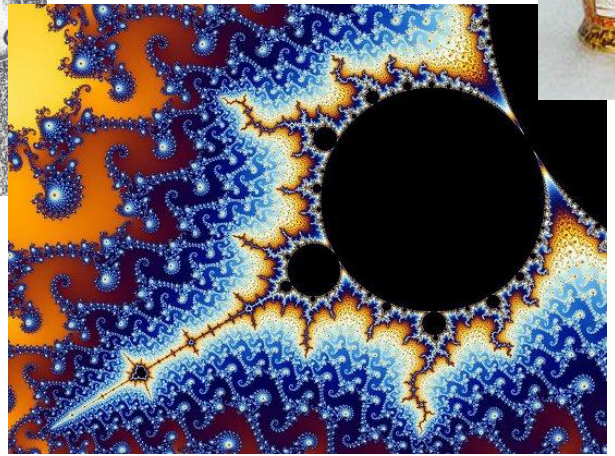
Recursão ocorre quando algo é definido a partir de sua própria definição

“*Recursividade é algo recursivo*” 😁

Recursividade em várias áreas

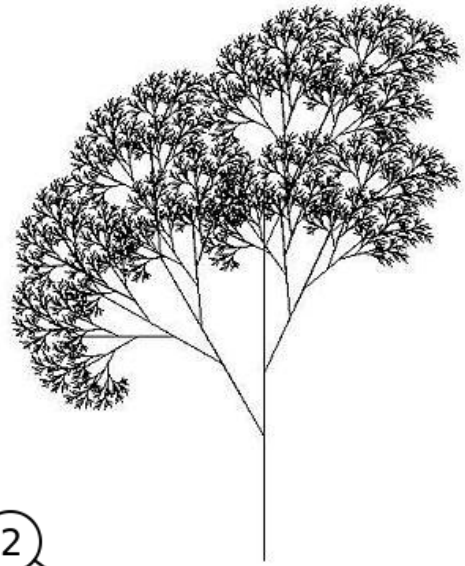


M.C.Escher



$$\mathbb{N} = \begin{cases} 0 \\ n+1 & \text{se } n \in \mathbb{N} \end{cases}$$

Recursividade em computação

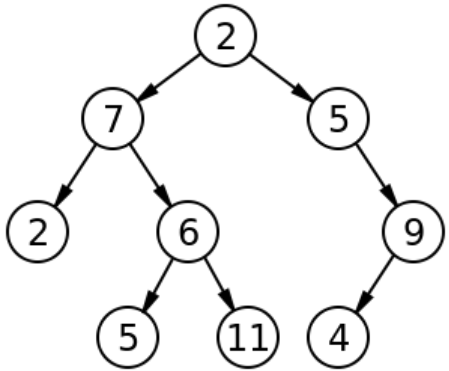


Técnica de resolução de problemas

- Redução de um problema maior em um menor
- As vezes fica mais fácil resolver problemas menores

Áreas

- Computação gráfica (subdivisões espaciais)
- Algoritmos (ordenação)
- Estruturas de dados (árvores de busca)
- Compiladores (árvores sintáticas)
- Jogos (geração procedural de conteúdo)
- ...

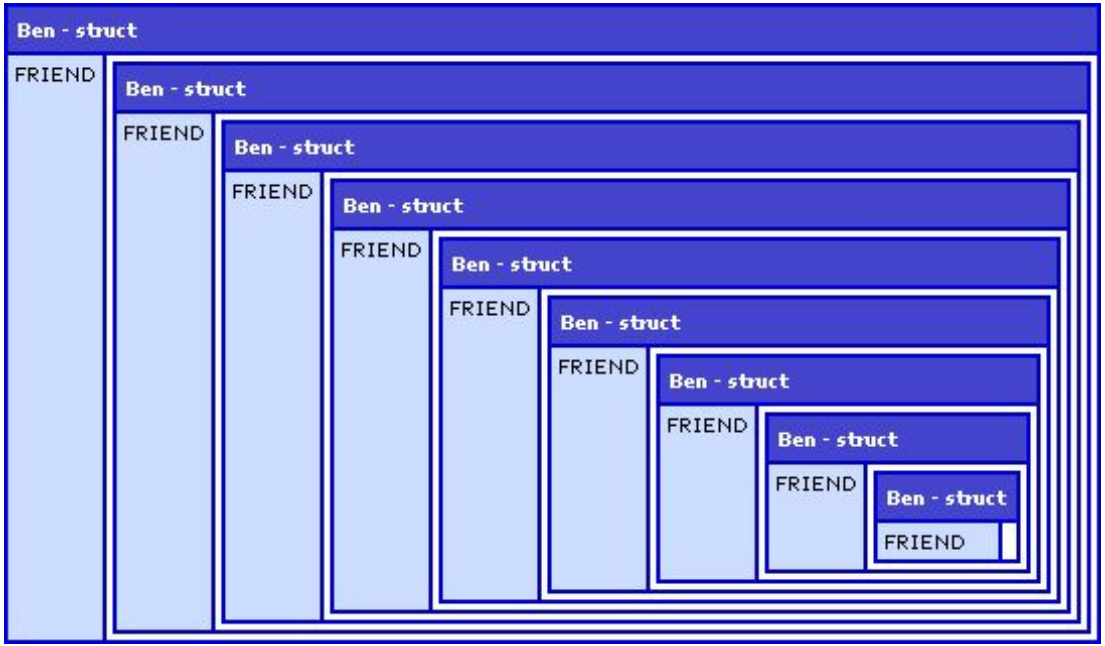


Forma recursiva

Elementos essenciais

1. Caso base
2. Caso recursivo

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



Resolução de problemas usando recursão

Ótima ferramenta para **quebrar problemas complexos** em mais simples
Ao tentar resolver um problema complexo, reduz o problema e aplica a mesma estratégia no problema mais simples.

Para ilustrar (algo simples)... calcular o somatório de uma sequência de inteiros

Soma de $\{ s_1, s_2, s_3, ..., s_n \} = s_1 +$ **Soma** de $\{ s_2, s_3, ..., s_n \}$

Soma de $\{ s_2, s_3, ..., s_n \} = s_2 +$ **Soma** de $\{ s_3, ..., s_n \}$

...

Soma de $\{ s_n \} = s_n$

Função recursiva

1. Caso base

   Define quando a recursão deve parar.

   Se não houver, haverá loop infinito!

   No exemplo, a recursão para quando n = 0

2. Caso recursivo

   Define quando o caso pode ser resolvido usando casos menores.

   As chamadas recursivas devem conduzir ao caso base, senão haverá loop infinito!

   No exemplo, há chamada recursiva quando n > 0

Fatorial

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

```
int fat(int n) {
  if (n == 0) {
    return 1;
  }
  return n * fat(n-1);
}
```

`fat(5) = 120`

Solução recursiva vs. solução iterativa

Alguns problemas são **naturalmente recursivos**.
Nesses casos, uma solução recursiva é mais simples de implementar.
Exemplo: Série de Fibonacci

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

versão iterativa

```
int fib(int n) {
  int f = 0;
  int i, aa = 0, a = 1;
  for (i = 0; i < n; i++) {
    f = a + aa;
    aa = a;
    a = f;
  }
  return f;
}
```

versão recursiva

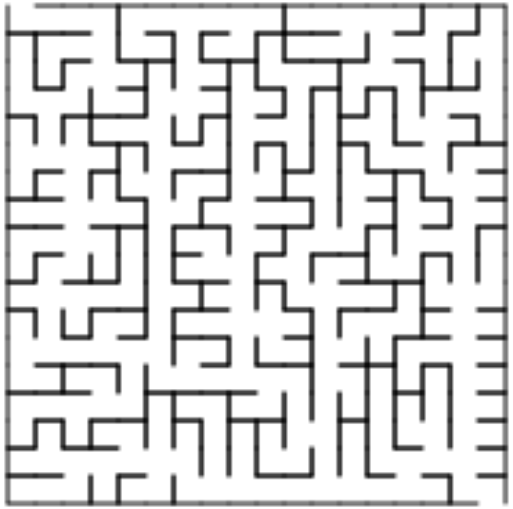
```
int fib(int n) {
  if (n <= 1) {
    return n;
  }
  return fib(n-1) + fib(n-2);
}
```

Explorando as possibilidades

A recursão pode ser usada como procedimento para explorar as possibilidades de um problema.

Exemplo: achar a saída de um labirinto

```
algoritmo sair(posição)
  se não está na saída
    se há caminho à esquerda não percorrido
      sair(posição à esquerda)
    senão se há caminho em frente não percorrido
      sair(posição à frente)
    senão se há caminho à direita não percorrido
      sair(posição à direita)
```

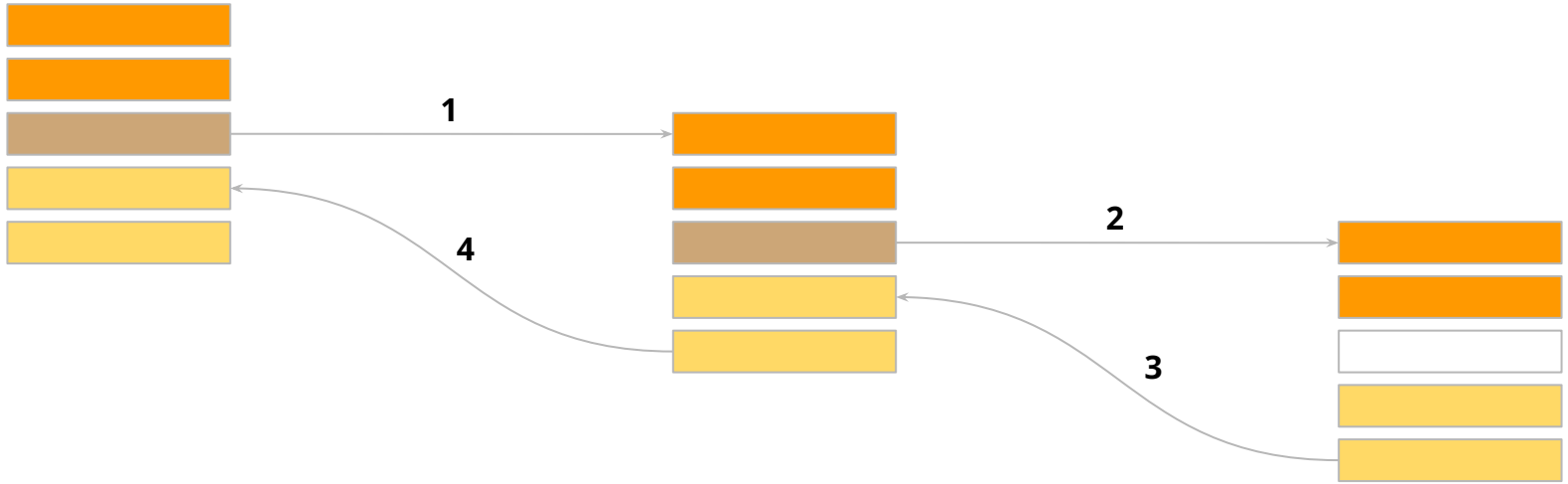


Processamento da recursão

Pode haver processamento:

- Antes da chamada recursiva
- Depois da chamada recursiva

Dependendo de quando a chamada é feita o resultado é diferente



Exemplo de processamento pré e pós-chamada

Problema

Implemente um programa que converte um número de base decimal para base binária. Exs: 15 → 1111, 10 → 1010

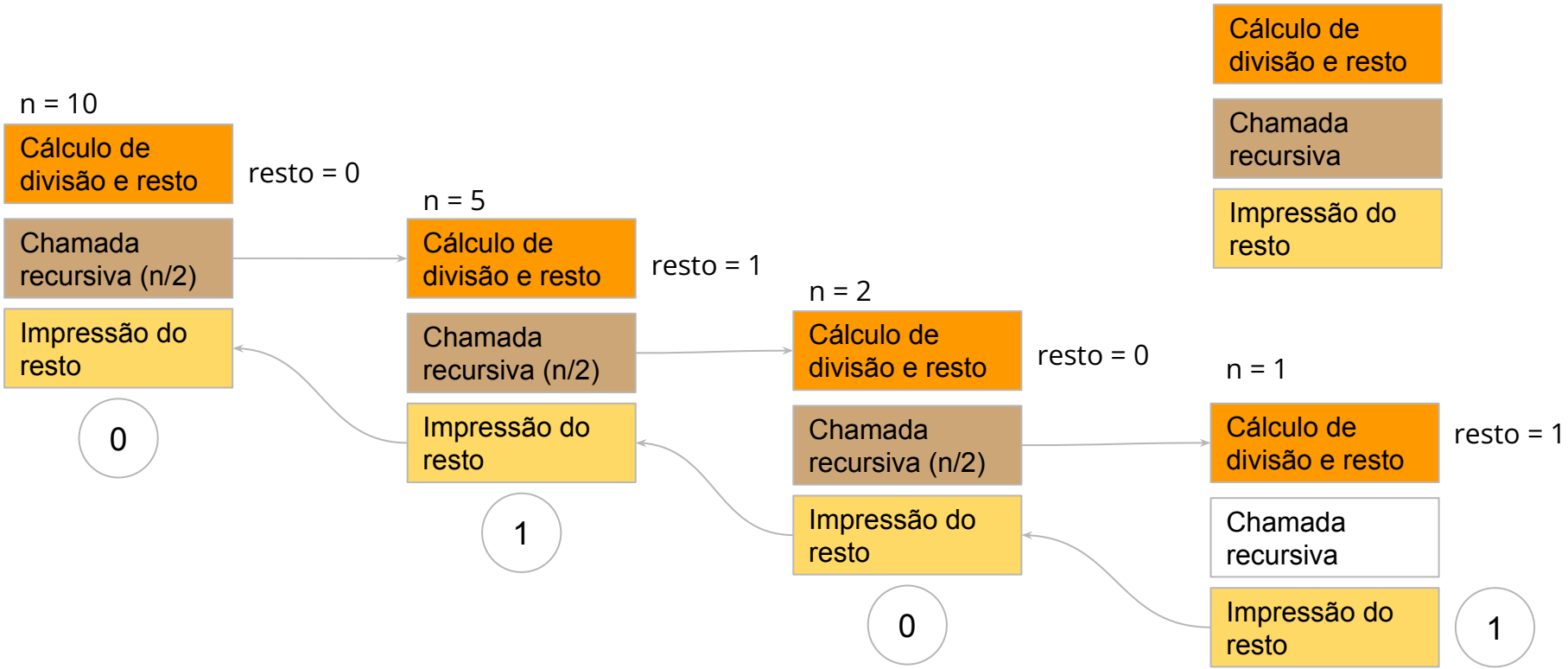
Ideia de solução: dividir o valor por 2, pegar o resto e aplicar a mesma solução para a parte inteira do resultado da divisão

Ex: para valor 10

1. 10 / 2 = 5 (resta **0**)
2. 5 / 2 = 2 (resta **1**)
3. 2 / 2 = 1 (resta **0**)
4. 1 / 2 = 0 (resta **1**)

Resultado é a **sequência de restos na ordem invertida**: 1010

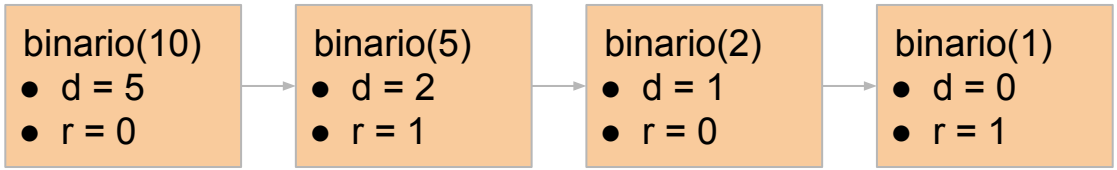
Exemplo de processamento pré e pós-chamada



Solução em código

?

As chamadas são “empilhadas” na memória
Para cada chamada um novo escopo é criado



Recursão indireta (ou múltipla)

Um recursão pode ser realizada usando mais de uma função
Por exemplo:

- Um número natural é par se ele for 0 ou se seu antecessor for ímpar;
- Um número natural é ímpar se ele não for 0 e seu antecessor for par.

```
int par(int n) {
  return (n == 0 || impar(n-1));
}
```

```
int impar(int n) {
  return (n != 0 && par(n-1));
}
```

Práticas

Escreva funções recursivas para:

1. Calcular a potência de um número a partir da seguinte relação:

$$b^n = b.\, b^{n-1}$$

2. Calcular o somatório de um array de inteiros a partir da seguinte relação:

$$\sum_{i=1}^{n} v_i = v_1 + \sum_{i=2}^{n}$$

3. Achar o maior valor de uma sequência de inteiros
4. Verificar se um dado número se encontra entre os números de uma sequência ordenada de números inteiros.
5. Contar quantos caracteres tem uma string

Problema - Maze runner

Dado um labirinto, escreva um programa para verificar se ele há saída a partir de uma posição inicial. O labirinto é representado por uma matriz de 0 e 1, onde 0 indica uma posição livre (onde é possível se mover) e 1 indica uma barreira, e só é possível mover nas 4 direções (cima, baixo, direita e esquerda).

Entrada
A primeira linha da entrada contém dois valores inteiros L e C representando as dimensões do labirinto (nº de linhas e nº de colunas, respectivamente). As L linhas seguintes contém C valores 0 ou 1, que indicam cada posição do labirinto, se encontra-se livre (0) ou se há uma barreira (1). As linhas seguintes contêm as coordenadas na posição inicial e a saída do labirinto.

Saida
Seu programa deve imprimir “sim” se houver saída ou “não” caso contrário.

Problema - Maze runner

Exemplos de entrada e saída

Entrada	Saída
4 4 1 0 1 0 1 0 0 0 1 1 0 1 0 0 0 0 0 1 3 0	sim

Entrada	Saída
6 6 1 0 1 0 0 0 1 0 0 0 1 1 1 1 0 1 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 1 0 1 1 0 1 5 0	não

Problema - Flood it

Flood it (https://unixpapa.com/floodit/) é um jogo em que precisamos preencher passo-a-passo uma matriz com uma única cor. A cada passo, escolhemos uma cor para preencher a partir da célula superior esquerda. Escreva um programa que realiza um passo do jogo. As cores na matriz do jogo são representadas por valores de 0 a 5.

Entrada
A primeira linha contém a dimensão N de uma matriz quadrada. As N linhas seguintes contêm N valores entre 0 e 5, que indicam as cores de cada célula da matriz. A linha seguinte contém um valor entre 0 e 5 que irá preencher a matriz a partir da célula superior esquerda.

Saída
O programa deve imprimir o resultado da matriz do jogo após o preenchimento da cor passada.

Problema - Flood it

Exemplos de entrada e saída

Entrada	Saída
4 1 1 1 0 1 5 1 4 1 1 2 1 4 5 1 3 4	4 4 4 0 4 5 4 4 4 4 2 1 4 5 1 3

Entrada	Saída
4 4 4 4 3 4 5 4 4 4 4 4 4 4 5 4 3 5	5 5 5 3 5 5 5 5 5 5 5 5 5 5 5 3

Entrada	Saída
4 5 5 5 3 5 5 5 5 5 5 5 5 5 5 5 3 3	3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3