Introdução à Programação

Recursão

Recursão

decoração
Substantivo feminino.
1.Ato ou efeito de **decorar**

decorar
Verbo transitivo direto.
1.Guarnecer com adorno(s); dispor formas e cores em; ornamentar, embelezar;
2.Realçar, avivar, adornar;

Fonte: Dicionário Aurélio

Tópicos da Aula

- Hoje, aprenderemos a usar uma técnica de programação chamada de recursão
  - Conceito de Recursão
  - Definições Recursivas
  - Casos Base e Geral
  - Programando Recursivamente
  - Recursão X Laço

Recursão

- Uma definição **recursiva** de um conceito consiste em utilizar o próprio conceito na definição
  - Ex: fat(n) = n * fat(n-1)
- Definições recursivas em linguagem natural não são, geralmente, muito úteis
- Contudo, em outras situações, uma definição recursiva pode ser a mais apropriada para explicar um conceito
- **Recursão** é uma técnica de programação que pode fornecer soluções elegantes para determinados problemas

Mas, antes, deve-se aprender a pensar recursivamente!

Definições Recursivas: *Lista*

- Considere a seguinte lista de números

24, 88, 40, 37

- Podemos definir uma lista de números como:

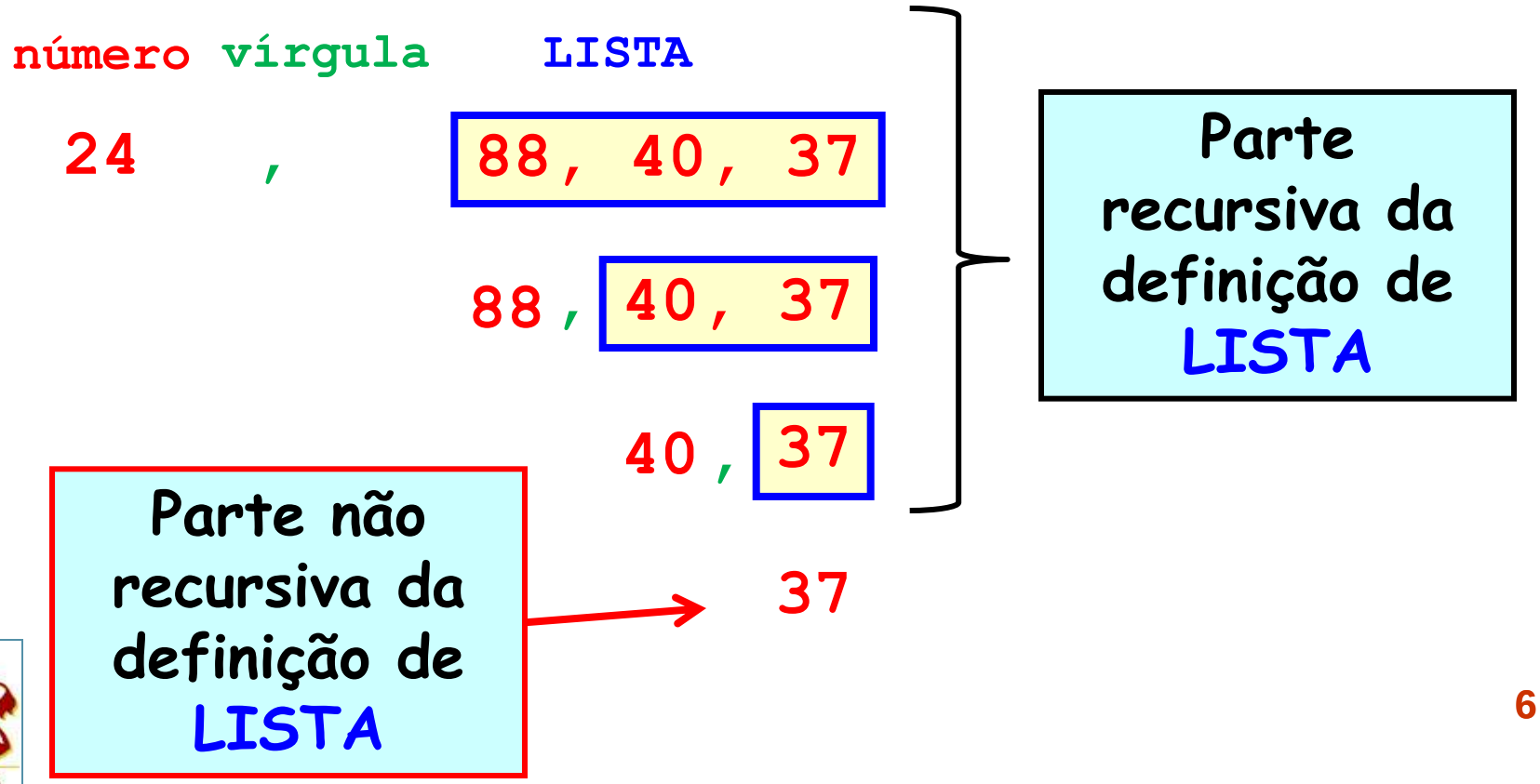
Uma **LISTA** é um: **número**
ou um: **número** **vírgula** **LISTA**

Resumindo: uma LISTA é definida ou como um único número, ou como um número seguido de uma vírgula seguida de uma LISTA

Utilizamos LISTA para sua própria definição

Definições Recursivas: *Lista*

A parte recursiva da definição de **LISTA** é utilizada diversas vezes, até que se possa utilizar a parte não recursiva



Definições Recursivas: *Fatorial*

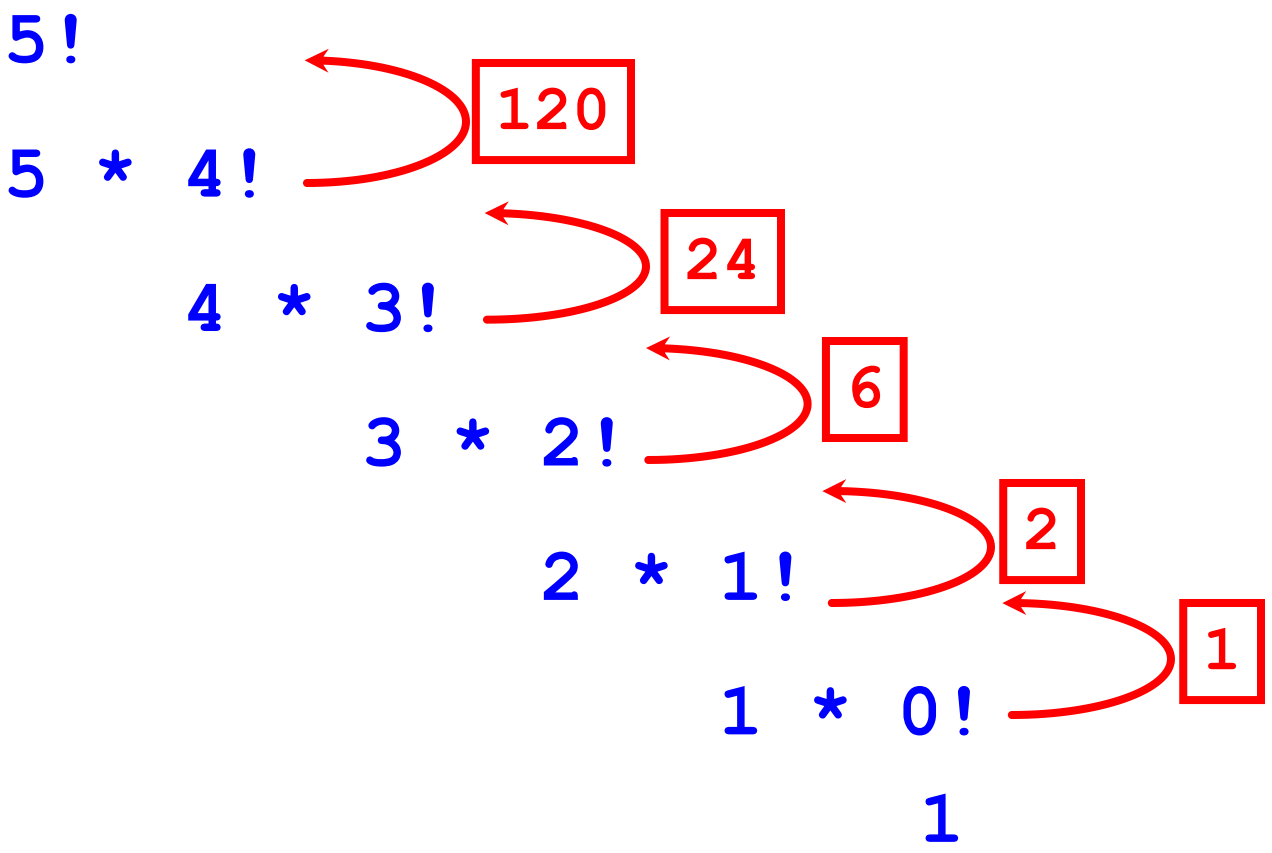
- **Fatorial** de um número N (N!), que é inteiro e positivo, é definido como a multiplicação de todos os inteiros compreendidos entre 1 e N (inclusive)

- Podemos definir fatorial recursivamente:

```
0! = 1
N! = N x (N-1)!
```

Mais uma vez, utilizamos fatorial para sua própria definição

Definições Recursivas: *Fatorial*



Caso Base e Caso Geral

- Toda definição recursiva deve ter uma parte recursiva e outra não recursiva
- Se não houver a parte não recursiva, o caminho recursivo nunca termina
  - Gera uma recursão infinita
  - Similar a um laço infinito
- Denomina-se de **caso base**, a parte não recursiva da definição, enquanto o **caso geral** é a parte recursiva

Programação Recursiva

- Uma função em C pode chamar ela mesma
  - Função recursiva
- O código de uma função recursiva **deve** tratar o caso base e o caso geral
- Cada chamada da função cria novos parâmetros e variáveis locais
  - Cria-se uma nova instância da função
- Como em qualquer chamada de função, assim que a função termina sua execução, o controle retorna para a função que a chamou (que neste caso, é outra instância da mesma função)

Exemplo de Função Recursiva: *Fatorial*

```
int fatorial(int n) {
      int resultado;

      if (n == 0)
            resultado = 1;

      else
            resultado = n * fatorial(n - 1);

      return resultado;
}
```

Caso Base

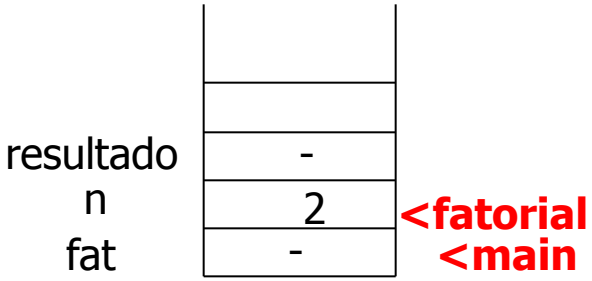
Caso Geral

Chamando Função Recursiva: *Fatorial*

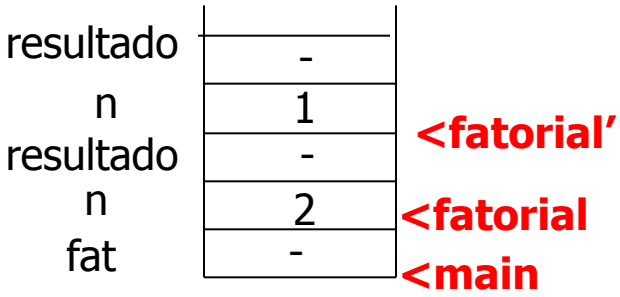
```
int main() {
      int fat = fatorial(2);
      printf("Fatorial de 2 é %d",fat);

      return 0;
}
```

Pilha de Execução de Funções Recursivas

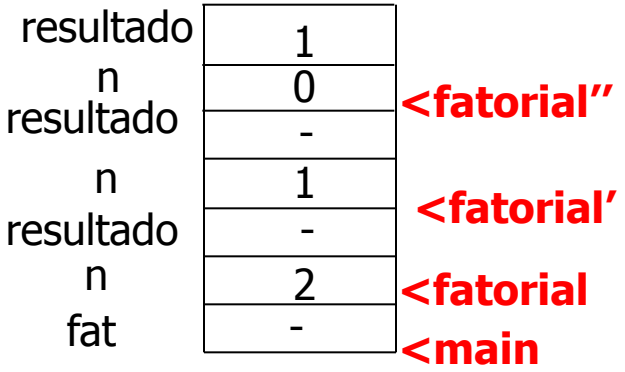
1 – Chamada da função: cópia do argumento



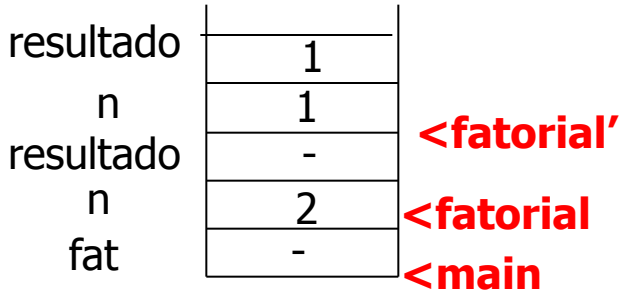
2 – Chamada recursiva: cópia do argumento



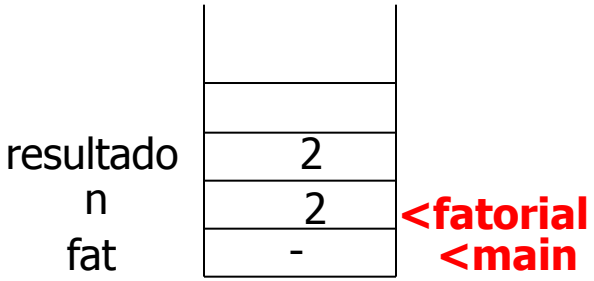
3 – Chamada recursiva: cálculo da expressão



4 – Retorno da 3a chamada: cálculo da expressão

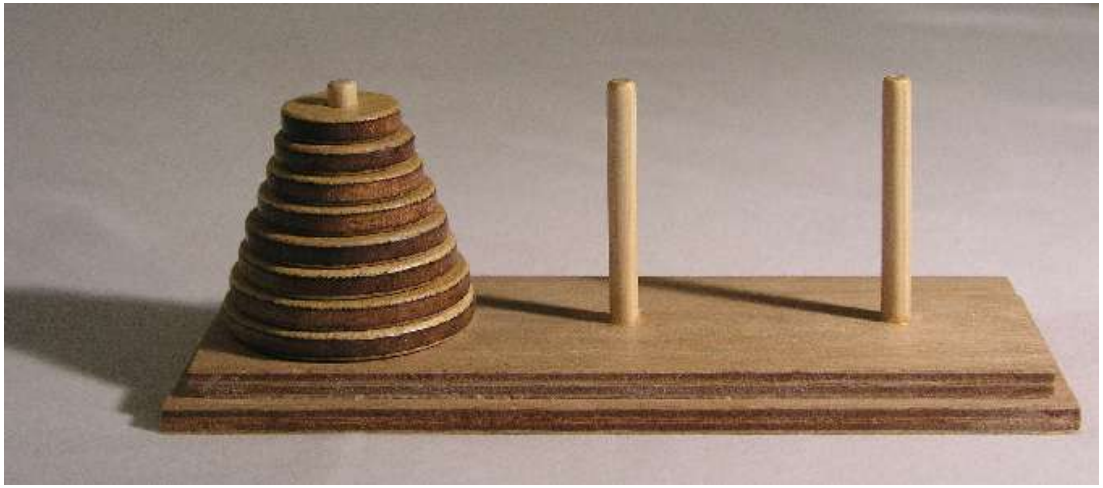


5 – Retorno da 2a chamada: cálculo da expressão



5 – Retorno da 1a chamada

Usando Recursividade para resolver problemas

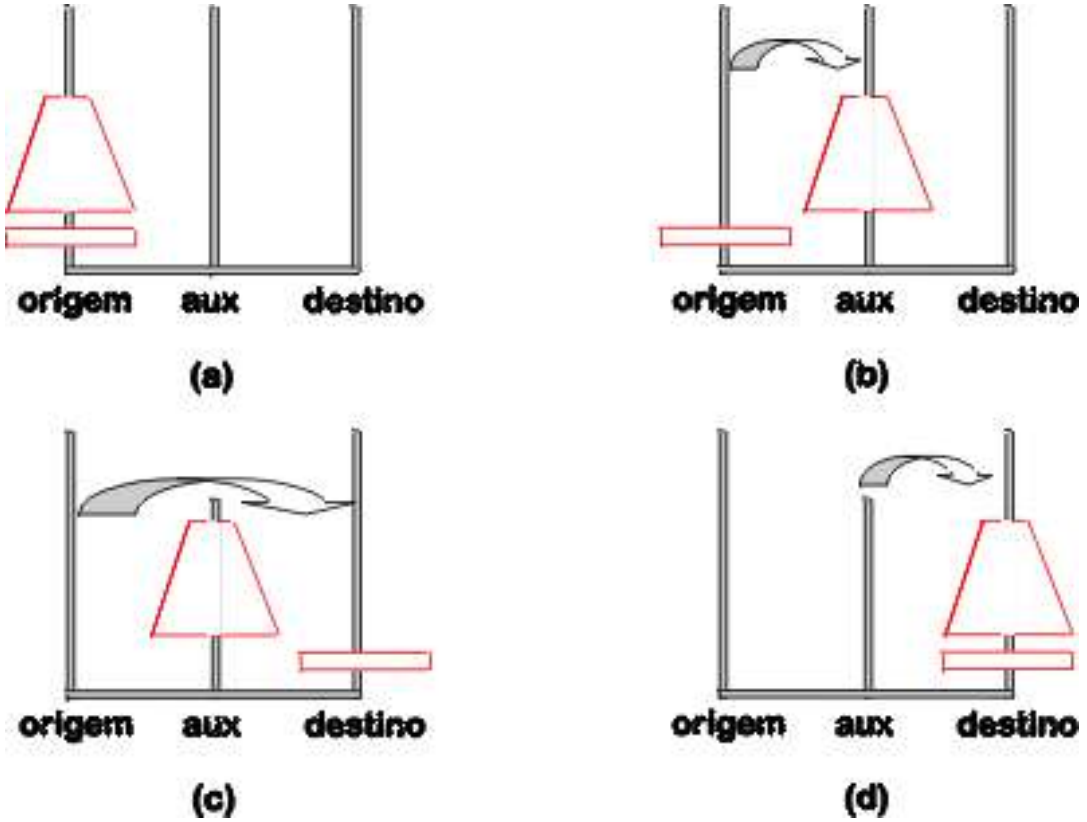


Problema: Torre de Hanoi

Deve-se transportar todos os discos da primeira estaca até a última obedecendo as seguintes regras:

1. Só pode ser deslocado um disco por vez (o do topo de uma estaca)
2. Em nenhum momento um disco maior pode estar sobre um menor

Resolvendo Recursividade o Problema da Torre de Hanoi



Solução: Torre de Hanoi

1. Mover **n – 1** discos da estaca Origem para a Temporária
2. Mover o disco **n** da estaca Origem para a Destino
3. Mover **n – 1** discos da estaca Temporária para a Destino

Torre de Hanoi

Caso Base

```
void mover(int n, char orig, char temp, char dest) {
 if (n == 1)
   printf("Mova o disco 1 da estaca %c para a estaca
            %c\n",orig,dest);
 else {
   mover (n-1,orig,dest,temp);
   printf("Mova o disco %d da estaca %c para a estaca
            %c\n",n,orig,dest);
   mover(n-1,temp,orig,dest);
 }
}
```

Caso Geral

Pontos Chave de uma Solução Recursiva

- Definir o problema em termos recursivos
  - Definir o caso geral
- Determinar o caso base
- Definir uma solução de modo que a cada chamada recursiva, podemos nos aproximar do caso base

Recursão x Laço

- Não é porque existe uma solução recursiva, que sempre devemos usá-la
- Soluções iterativas (com laço) são geralmente mais eficientes
  - Soluções recursivas geralmente demandam mais poder de processamento e memória
- Porém, soluções utilizando recursão podem ser mais elegantes e mais compreensíveis que soluções iterativas equivalentes
- Programador deve analisar caso a caso qual é a melhor técnica a aplicar para a resolução do problema

Resumindo ...

- Recursão
  - Conceito
  - Exemplos de definições recursivas
  - Casos Base e Geral
  - Programando Recursivamente
  - Recursão X Laço