Pontifícia Universidade Católica
do Rio de Janeiro

INF 1007 – Programação II

Aula 05 – Recursividade

Edirlei Soares de Lima
<elima@inf.puc-rio.br>

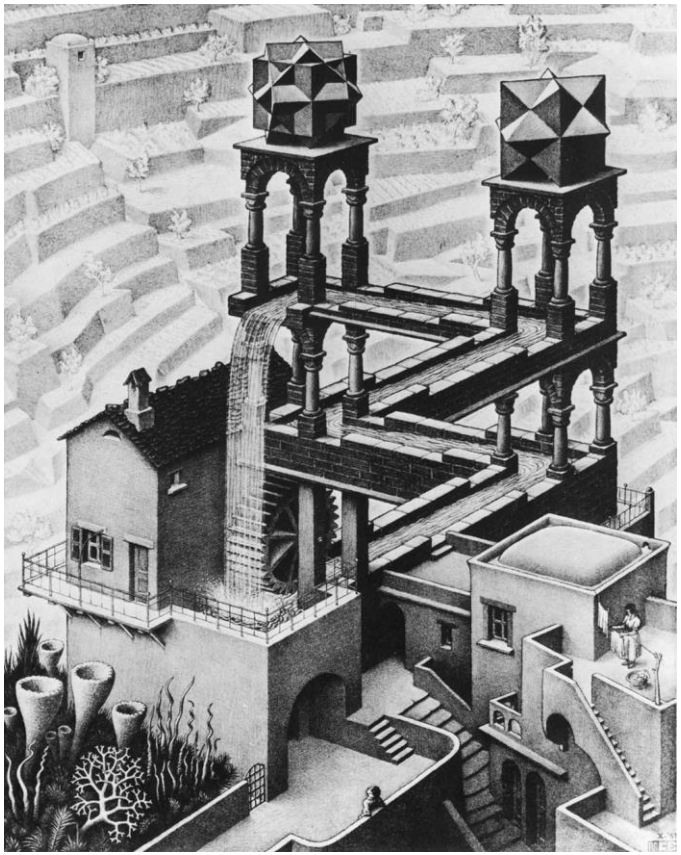
visionlab
visualization laboratory

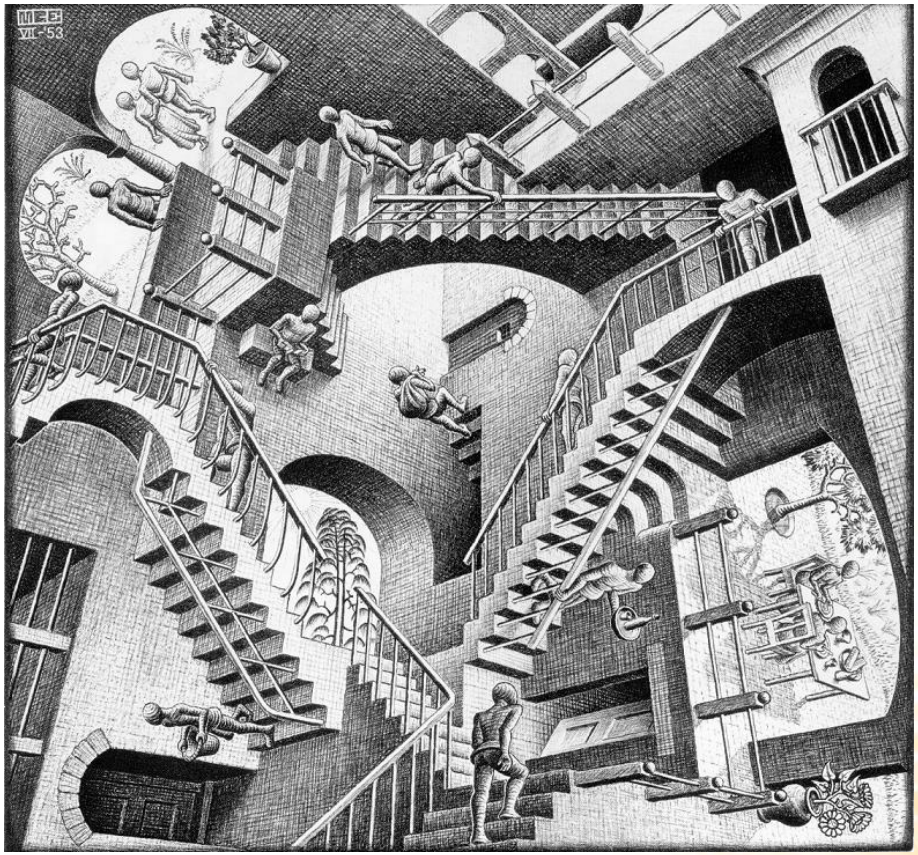
Introdução

- As seguintes sentenças são Verdadeiras ou Falsas?

1. *Alguém diz: “Estou mentido agora!”*;
2. *Alguém diz: “Esta sentença é falsa!”*;
3. *Um homem natural de Creta, em praça pública, anuncia: “Todo cretense é mentiroso!”*;

Introdução





Definições Recursivas

- Em uma definição recursiva um item é definido em termos de si mesmo, ou seja, **o item que está sendo definido aparece como parte da definição**;

- Em todas as funções recursivas existe:
  - **Caso base** (um ou mais) cujo resultado é imediatamente conhecido;
  - **Passo recursivo** em que se tenta resolver um sub-problema do problema inicial.

Definições Recursivas

- Função fatorial: $fat(n) = n \times (n - 1) \times \ldots \times 1$

- A função fatorial pode ser definida recursivamente como:

$$fat(n) = \begin{cases} 1, se\ n = 0 \\ n \times fat(n - 1), se\ n > 0 \end{cases}$$

Caso Base

Passo Recursivo

Chamada Recursiva

- O **caso base** é uma situação trivial da função, onde calcular o valor da função é imediato e direto.

Definições Recursivas

- Função fatorial: $fat(n) = n \times (n - 1) \times \ldots \times 1$

- A função fatorial pode ser definida recursivamente como:

$$fat(n) = \begin{cases} 1, se\ n = 0 \\ n \times fat(n - 1), se\ n > 0 \end{cases}$$

Caso Base

Passo Recursivo

Chamada Recursiva

- O **passo recursivo** é a aplicação recursiva da função em uma versão de menor porte do problema.

Definições Recursivas

- **Solução Conceitual:**
  - Problema de Ordem-n:
    $fat(n)$: $\mathrm{N} \rightarrow \mathrm{N}$ ← A função *fat* recebe um número natural *n* e retorna um número natural que é o fatorial de *n*
  - Precondição:
    $n \in \mathrm{N}$ ← *n* só pode ser um número natural
  - Caso(s) Base:
    $n = 0 \rightarrow fat(n) = 1$ ← Se *n* é zero, a função *fat* retorna 1
  - Chamada Recursiva:
    $fat(n-1)$ ← valor de *fat* para *n*-1 é necessário
  - Passo Recursivo:
    $fat(n) = n * fat(n-1)$ ← A função *fat* retorna o valor de *n* multiplicado pelo valor de *fat* para *n*-1

Funções Recursivas

- Na programação, uma função recursiva é aquela que faz uma chamada para si mesma;

- Essa chamada pode ser:
  - **Direta:** uma função A chama a ela própria
  - **Indireta:** função A chama uma função B que, por sua vez, chama A

```
void func_rec(int n)
{
   ...
   func_rec(n-1);
   ...
}
```

Recursao Direta

Funções Recursivas

- Função recursiva para cálculo de fatorial:

$$fat(n) = \begin{cases} 1, se\ n = 0 \\ n \times fat(n-1), se\ n > 0 \end{cases}$$

Caso Base

Passo Recursivo

Chamada Recursiva

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

Caso Base

Passo Recursivo

Funções Recursivas

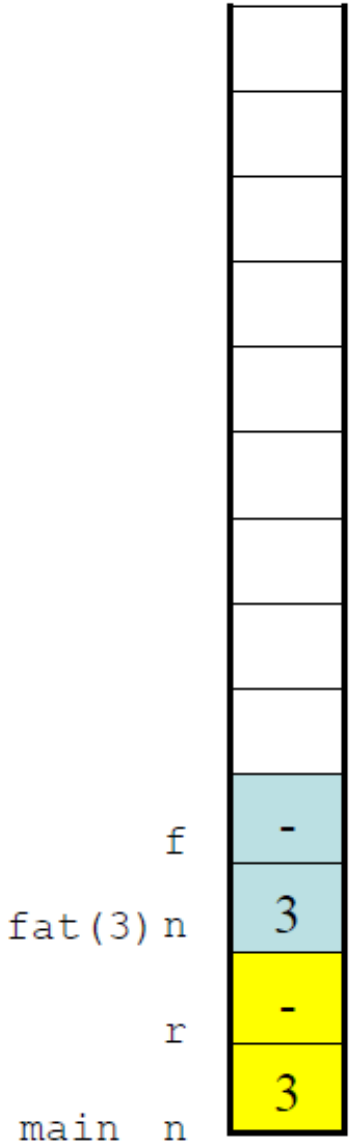
- **Comportamento de uma função recursiva:**
  - Quando uma função é chamada recursivamente, cria-se um ambiente local para cada chamada;
  - As variáveis locais de chamadas recursivas são independentes entre si, como se estivéssemos chamando funções diferentes;

Funções Recursivas

```
#include <stdio.h>

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

int main(void)
{
  int n = 3, r;
  r = fat(n);
  printf("Fatorial de %d = %d \n", n, r);
  return 0;
}
```

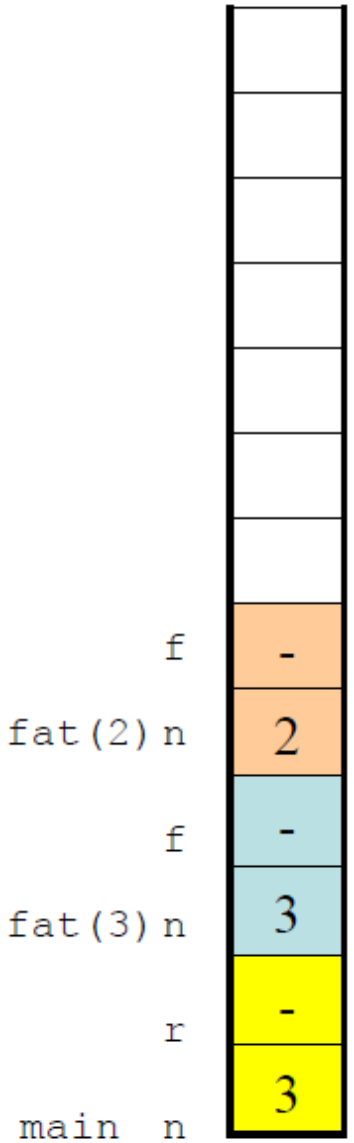


Funções Recursivas

```
#include <stdio.h>

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

int main(void)
{
  int n = 3, r;
  r = fat(n);
  printf("Fatorial de %d = %d \n", n, r);
  return 0;
}
```

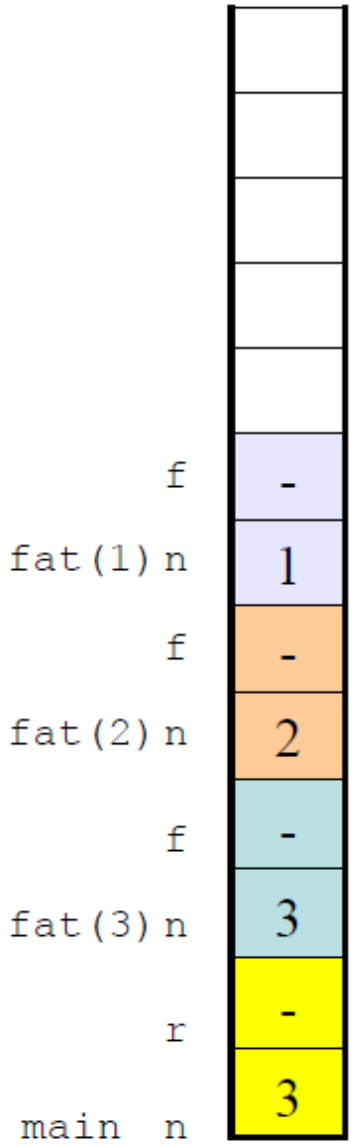


Funções Recursivas

```
#include <stdio.h>

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

int main(void)
{
  int n = 3, r;
  r = fat(n);
  printf("Fatorial de %d = %d \n", n, r);
  return 0;
}
```

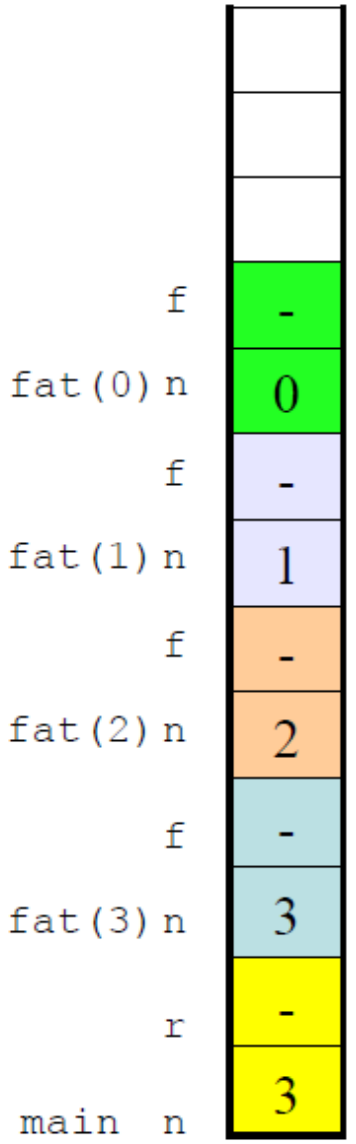


Funções Recursivas

```
#include <stdio.h>

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

int main(void)
{
  int n = 3, r;
  r = fat(n);
  printf("Fatorial de %d = %d \n", n, r);
  return 0;
}
```

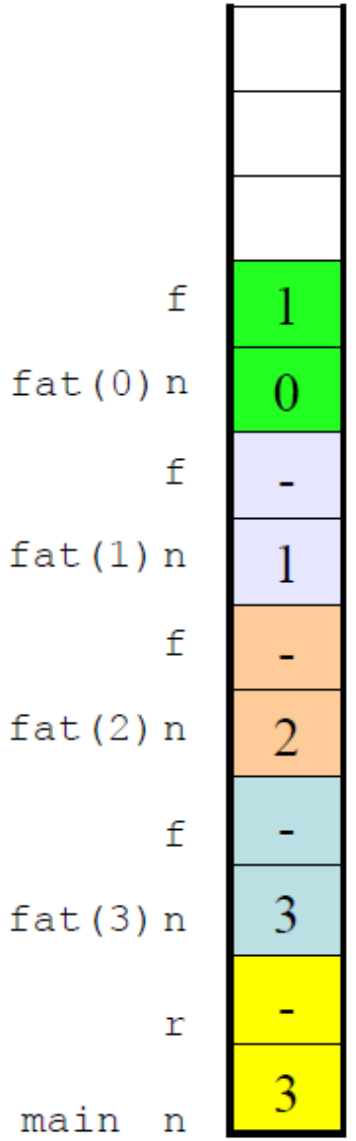


Funções Recursivas

```
#include <stdio.h>

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

int main(void)
{
  int n = 3, r;
  r = fat(n);
  printf("Fatorial de %d = %d \n", n, r);
  return 0;
}
```

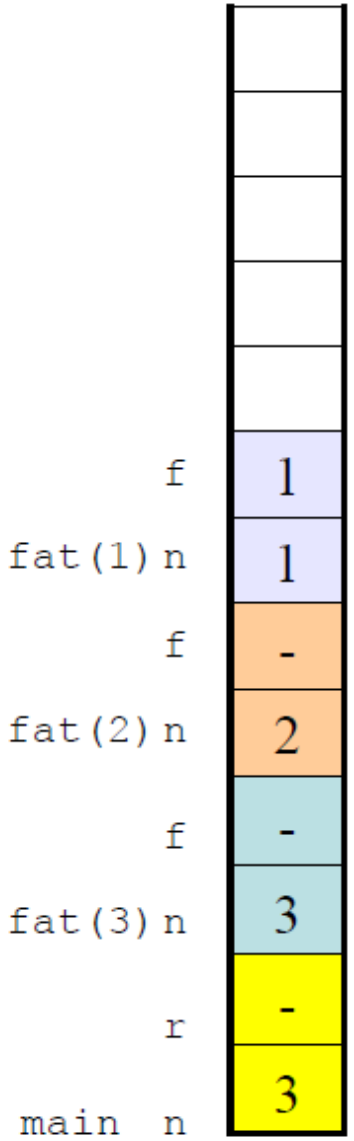


Funções Recursivas

```
#include <stdio.h>

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

int main(void)
{
  int n = 3, r;
  r = fat(n);
  printf("Fatorial de %d = %d \n", n, r);
  return 0;
}
```

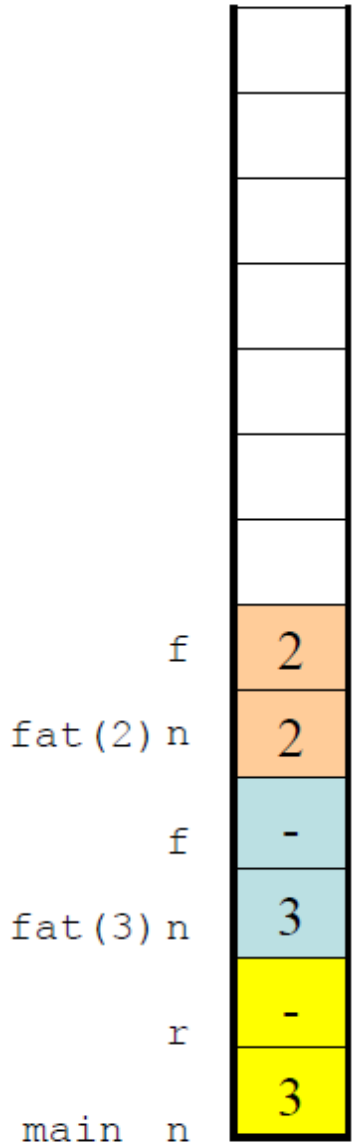


Funções Recursivas

```
#include <stdio.h>

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

int main(void)
{
  int n = 3, r;
  r = fat(n);
  printf("Fatorial de %d = %d \n", n, r);
  return 0;
}
```

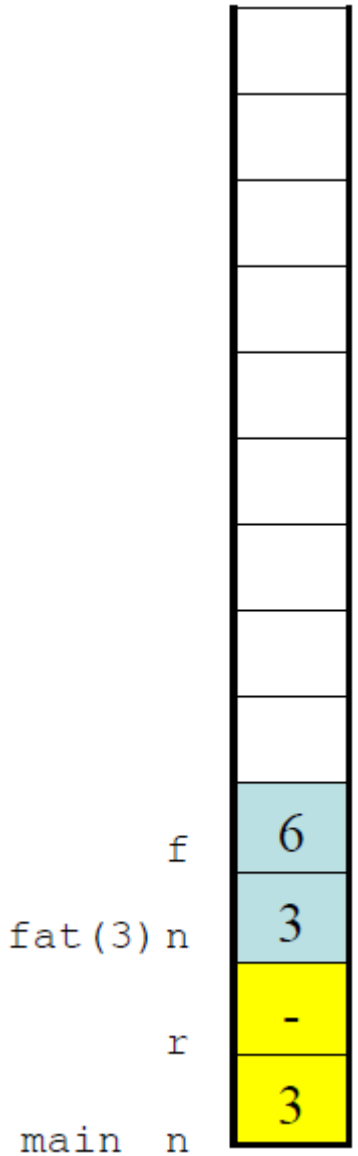


Funções Recursivas

```
#include <stdio.h>

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

int main(void)
{
  int n = 3, r;
  r = fat(n);
  printf("Fatorial de %d = %d \n", n, r);
  return 0;
}
```

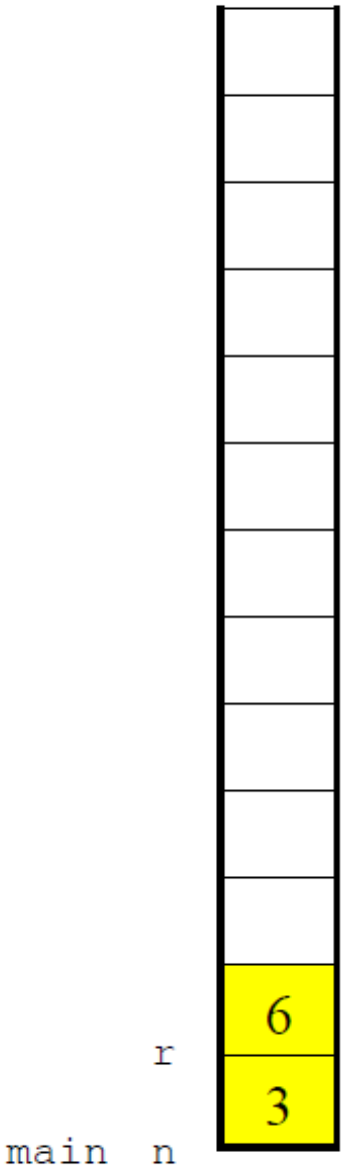


Funções Recursivas

```
#include <stdio.h>

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

int main(void)
{
  int n = 3, r;
  r = fat(n);
  printf("Fatorial de %d = %d \n", n, r);
  return 0;
}
```



Funções Recursivas

- **Exercício:** forneça a definição recursiva para a operação de potenciação:

$$x^n = \begin{cases} 1, se\ n = 0 \\ x \times x^{(n-1)}, se\ n > 0 \end{cases}$$

Caso Base

Passo Recursivo

```
int pot(int x, int n)
{
  if (n == 0)
    return 1;
  else
    return x * pot(x, n-1);
}
```

Manipulação de Cadeias de Caracteres

- É possível utilizar **funções recursivas** para manipular cadeias de caracteres;

- Baseiam-se em uma definição recursiva de cadeias de caracteres:
  - Uma cadeia de caracteres é:
    - a cadeia de caracteres vazia; ou
    - um caractere seguido de uma cadeia de caracteres.

Manipulação de Cadeias de Caracteres

- Implementação recursiva da função “imprime”:

```
void imprime_rec(char* s)
{
  if (s[0] != '\0')
  {
    printf("%c", s[0]);
    imprime_rec(&s[1]);
  }
}
```

```
int main (void)
{
  char cidade[] = "Rio de Janeiro";
  imprime_rec(&cidade[0]);
  return 0;
}
```

Manipulação de Cadeias de Caracteres

- Implementação recursiva da função “imprime invertido”?:

```
void imprime_inv(char* s)
{
  if (s[0] != '\0')
  {
    imprime_inv(&s[1]);
    printf("%c", s[0]);
  }
}
```

```
int main (void)
{
  char cidade[] = "Rio de Janeiro";
  imprime_inv(&cidade[0]);
  return 0;
}
```

Manipulação de Cadeias de Caracteres

- Implementação recursiva da função “comprimento”?:

versão iterativa

```
int comprimento(char s[])
{
   int i, n = 0;
   for (i=0; s[i] != '\0'; i++)
   {
      n++;
   }
   return n;
}
```

versão recursiva

```
int comprimento_rec(char* s)
{
  if (s[0] == '\0')
    return 0;
  else
    return 1 + comprimento_rec(&s[1]);
}
```

Leitura Complementar

- Waldemar Celes, Renato Cerqueira, José Lucas Rangel, **Introdução a Estruturas de Dados**, Editora Campus (2004).

- **Capítulo 4 – Funções**