INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATAR
CAMPUS FLORIANÓPOLIS

PROGRAMAÇÃO DE COMPUTADORES I

Décima Segunda Parte

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA
CAMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA

PROGRAMAÇÃO DE COMPUTADORES I

Décima Segunda Parte

Capítulo V

Funções Recursivas

Marco Villaça
Fernando Pacheco

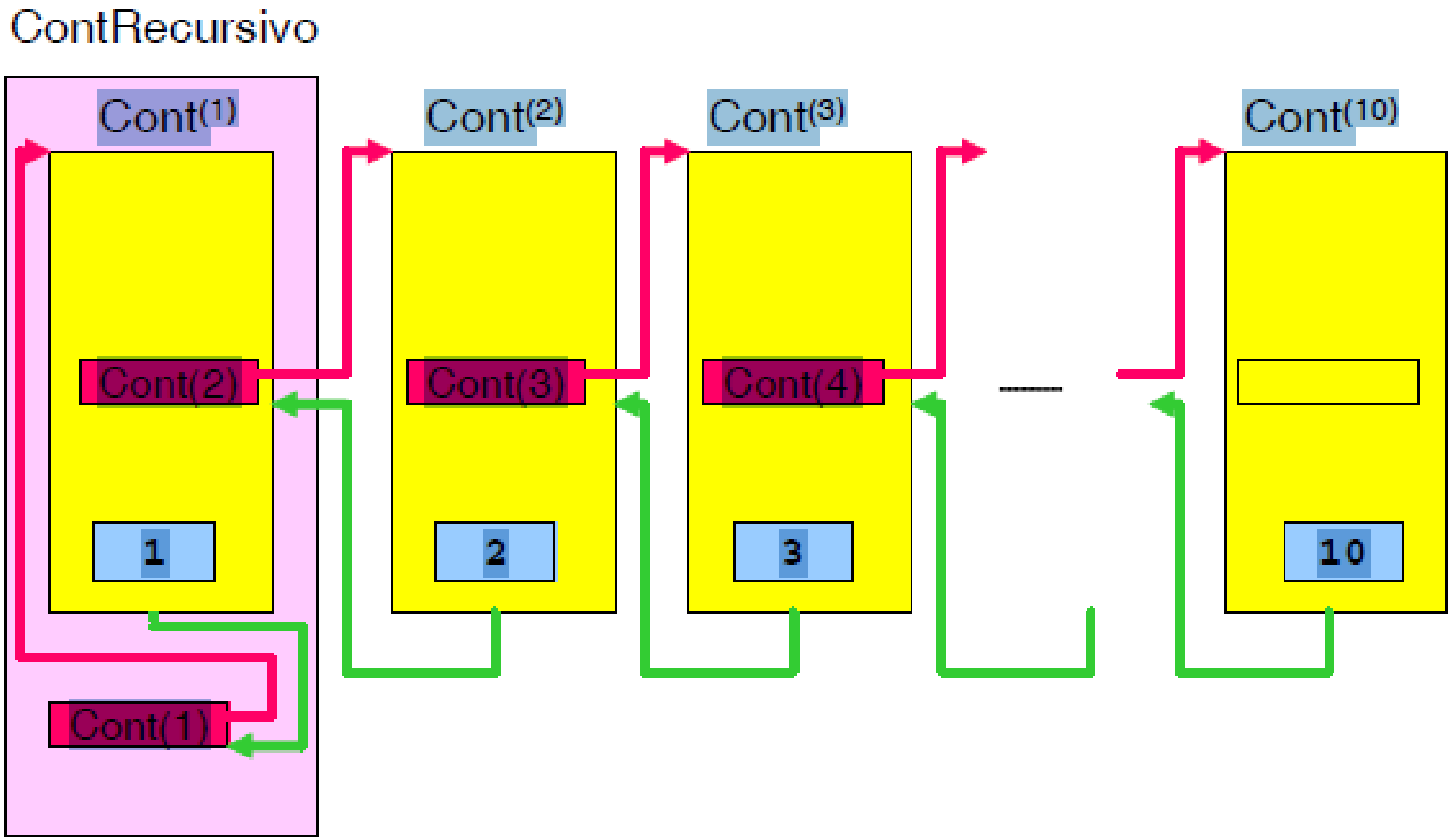
Recursão em funções

- Função chama a si mesma de forma direta ou indireta
- Duas regras básicas
  - Deve existir um ponto de parada
  - Deve simplificar o problema
- Exemplo: fatorial de um número
  - 0! = 1
  - n! = n·(n-1)!

Contador Recursivo
Entendo a recursão

```
#include <stdio.h>
#include <stdlib.h>
void cont (int);
int main()
{
   cont(1);
   return 0;
}
void cont(int n)
{
   if (n < 10)
      cont(n+1);
   printf("%d\n", n);
}
```

Contador Recursivo
Entendendo a recursão



Exemplo

Programa que utiliza uma função para calcular o fatorial de um número sem recursão, lembrando que:

- n! = n*(n-1)!
- 1! = 1
- 0! = 1

Fatorial

```
#include <stdio.h>
#include <stdlib.h>
double fatorial(int);
int main()
{   double fat;
    int num;
    printf("Entre com o numero que se deseja calcular o fatorial\n");
    scanf("%d",&num);
    fat=fatorial(num);
    printf("O fatorial de %d eh %.0f",num,fat);
    return 0;
}
double fatorial(int n)
{   double fat = 1;
    while (n>0)
    {
        fat=fat*n;
        n--;
    }
    return fat;
}
```

Recursão em funções

- Função recursiva para cálculo do fatorial

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

Para n = 3
*fat= 3 * fatorial(2)*
*fat =3 * 2 * fatorial(1)*
*fat =3 * 2 * 1*

Recursão em funções
Exercícios

1. Elabore uma função recursiva para calcular a soma dos n primeiros números inteiros.

2. Na série de Fibonacci, cada termo resulta da adição dos dois que o antecedem [F(n) = F(n-1) + F(n-2)] originando assim os números 0; 1; 1; 2;3; 5; 8; 13; 21 e, assim por diante. Formalmente,

$$F(n) = \begin{cases} 0 & \text{para } n = 1 \\ 1 & \text{para } n = 2 \text{ ou } n = 3 \\ F(n-1) + F(n-2) & \text{para } n > 3 \end{cases}$$

Escreva uma função recursiva de Fibonacci

Recursão em funções
Exercícios

3. Analisando o código, responda sem compilar, qual o valor de X(4)?

```
int X(int n)
{
    if (n == 1 || n == 2)
return n;
    else
return X(n-1) + n * X(n-2);
}
```

Recursão em funções
Exercícios

4. Qual é o valor de f(1,10)? Escreva uma função equivalente que seja mais simples.

```
double f( double x, double y)
{
    if (x >= y)
return (x + y)/2;
    else
return f( f(x+2, y-1), f(x+1, y-2));
}
```

Recursão em funções
Exercícios

5. A função abaixo calcula o maior divisor comum dos inteiros positivos m e n. Escreva uma função recursiva equivalente.

```
int Euclides(int m, int n)
{
    int r;
    do {
        r = m % n;
        m = n;
        n = r;
    } while (r != 0);
    return m;
}
```

Recursão em funções
Exercícios

6. Exponenciação. Escreva uma função recursiva eficiente que receba inteiros positivos k e n e calcule k^n. (Suponha que k^n cabe em um `int`.)

Referências

- OUALLINE, S. *Practical C Programming*. 3. ed. O'Reilly, 1997.