

Aula 18

MC 102 - Algoritmos e Programação de Computadores

Recursão

Recursividade

Um objeto é dito recursivo se pode ser definido em termos de si próprio.

Recursão é o processo de se usar um objeto recursivo

Uma **função** é dita **recursiva** se **invoca a si mesma**, direta ou indiretamente.

Exemplo:

- Os pais de uma pessoa são seus antepassados
- Os pais de qualquer antepassado são também antepassados da pessoa em consideração

Recursão

Toda recursão é composta por:

- **Caso base**

- Uma instância do problema que pode ser solucionada facilmente

-

- **Chamadas Recursivas**

- O objeto define-se em termos de si próprio, procurando convergir para o caso base. Por exemplo, a soma de uma lista de n elementos pode ser definida a partir da lista da soma de $n-1$ elementos.

Recursão

Uma maneira mais simples de entender recursão:

Nós já terminamos?

-Se sim, retorne os resultados.

-Se não, simplifique o problema; resolva o(s) problema(s) mais simples, e então encaixe os resultados na solução do problema original. Ao final, retorne a solução.

Sem uma condição de parada como esta, uma recursão iria se repetir eternamente.

Recursão

- Se você já sabe o que é a recursão, apenas se lembre da resposta.
- Caso contrário, encontre alguém que esteja mais próximo de um oráculo do que você e então pergunte a ele ou a ela o que é a recursão.

Recursão – Exemplo 1

Soma de uma lista de n elementos

Caso base: somar 1 elemento (neste caso $n=1$)

Chamadas Recursivas:

soma dos n elem = somar elemento n à soma dos n-1 elementos

soma dos n-1 elem = somar o elemento n-1 à soma dos n-2 elementos

soma dos n-2 elem = somar o elemento n-2 à soma dos n-3 elementos

....

soma dos 3 elem = somar o elemento 3 à soma dos 2 primeiros elementos

soma dos 2 elem = somar elemento 1 ao elemento 2

soma do 1 elem = somar 1 elemento (**Caso base**)

Recursão – Exemplo 1

Soma de uma lista de n elementos

Caso base: somar 1 elemento (neste caso n=1)

Chamadas Recursivas:

soma dos n elem =

soma (n + soma(n-1)) =

soma(n-1+ soma(n-2)) =

soma(n-1+ soma(n-2 + soma(n-3))) = ...

soma(n-1+ soma(n-2 + soma(n-3 + soma(n-4 + ...

...+ soma(elem 3 + soma(elem 2+ (**soma(1)**))))))))))))))

Recursão - Exemplo 1 - Código

Realizar a soma de n elementos.

```
int soma_n (n) {  
    if (n <= 1)  
        return n;  
    else  
        return (n+ soma_n (n-1));  
}
```

Recursão - Exemplo 1 - Código

Realizar a soma de n elementos.

```
int soma_n (n) {  
    if (n <= 1)  
        return n;  
    else  
        return (n+ soma_n (n-1) );  
}
```

Caso base, condição em que facilmente se resolve o problema.

Recursão - Exemplo 1 - Código

Realizar a soma de n elementos.

```
int soma_n (n) {  
    if (n <= 1)  
        return n;  
    else  
        return (n+ soma_n (n-1) );  
}
```

Chamadas recursivas,
procurando simplificar o
problema

Recursão - Exemplo 2

Realizar a soma de todos os inteiros no intervalo [m,n].

Definição Recursiva:

$$\sum_{k=m}^n k = \begin{cases} m & \text{se } n = m \\ \sum_{k=m}^{n-1} k + n & \text{se } n > m \end{cases}$$

Recursão - Exemplo 2 - Código

Realizar a soma de todos os inteiros no intervalo [m,n].

```
int soma_mn(int m, int n) {
    if (n == m)
        return m;
    else if (m < n)
        return soma_mn(m, n-1) + n;
    else /* m > n */
        return soma_mn(m, n+1) + n;
}
```

Recursão - Exemplo 2 - Código

Realizar a soma de todos os inteiros no intervalo [m,n].

```
int soma_mn(int m, int n) {  
    if (n == m)  
        return m;  
    else if (m < n)  
        return soma_mn(m, n-1) + n;  
    else /* m > n */  
        return soma_mn(m, n+1) + n;  
}
```

Caso base, condição em que facilmente se resolve o problema.

Recursão - Exemplo 2 - Código

Realizar a soma de todos os inteiros no intervalo [m,n].

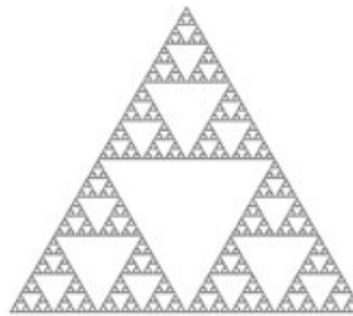
```
int soma_mn(int m, int n) {  
    if (n == m)  
        return m;  
    else if (m < n)  
        return soma_mn(m, n-1) + n;  
    else /* m > n */  
        return soma_mn(n, m);  
}
```

Chamadas recursivas,
procurando simplificar o
problema

Recursão – Exemplo 3

Triângulo de Sierpinski: dado um triângulo, faça:

Reduza-o pela metade de seus lados, faça três cópias e posicione-as uma sobre as outras duas, de forma que um encoste nos outros dois. Repita o passo até enquanto é possível dividir o triângulo



Recursão - Exemplo 4

Calcular o número fatorial para um inteiro n qualquer não negativo

Definição Recursiva:

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

Recursão - Exemplo 4 - Código

Calcular o número fatorial para um inteiro n qualquer não negativo

```
long fatorial(unsigned int n) {  
    if (n == 0)  
        return 1;  
    else /* n > 0 */  
        return n * fatorial(n-1);  
}
```

Recursão - Exemplo 4 - Código

Calcular o número fatorial para um inteiro n qualquer não negativo

```
long fatorial(unsigned int n) {  
    if (n == 0)  
        return 1;  
    else /* n > 0 */  
        return n * fatorial(n-1);  
}
```

Caso base, condição em que facilmente se resolve o problema.

Recursão - Exemplo 4 - Código

Calcular o número fatorial para um inteiro n qualquer não negativo

```
long fatorial(unsigned int n) {  
    if (n == 0)  
        return 1;  
    else /* n > 0 */  
        return n * fatorial(n-1);  
}
```

Chamadas recursivas,
procurando simplificar o
problema

Recursão – Exercício 1

Calcular a potência (expoente inteiro) de um número ponto flutuante qualquer

Definição Recursiva:

$$x^n = \begin{cases} \frac{1}{x^{(-n)}} & \text{se } n < 0 \\ 1 & \text{se } n = 0 \\ x \cdot x^{(n-1)} & \text{se } n > 0 \end{cases}$$

Recursão – Exercício 1 - Código

Calcular a potência (expoente inteiro) de um número ponto flutuante qualquer

```
double potencia(double x, int n) {  
    if (n == 0)  
        return 1;  
    else if (n > 0)  
        return x * potencia(x, n-1);  
    else /* n < 0 */  
        return 1 / potencia(x, -n);  
}
```