Aula 09 - Recursividade

TECNOLOGIA em **Gestão de Dados**

Programação I

Prof. Dr. Wener Sampaio

ReUni DIGITAL · EAD Centro de Educação Aberta e a Distância · UFPI Universidade Federal do Piauí · MEC

Recursividade

- Funções:
  - Uma função pode fazer chamadas a outras funções.
  - A recursividade permite que uma função chame a si mesma.
  - Muitas estruturas de dados e funções matemáticas são definidas de forma recursiva.
  - Recursividade é matematicamente mais elegante e clara.
  - Alto consumo de memória.

Recursividade

- Exemplo 1: Soma dos N termos positivos:
  - S(5) = 5+4+3+2+1 = 5+S(4) = ... = 15
  - Versão iterativa:

```
def S(n):
    soma=0
    for v in range(1,n+1):
        soma+=v
    return soma

n=int(input('Informe um valor positivo: '))
if n>0: print(S(n))
```

  - Versão recursiva:

```
def S(n):
    if n!=1: return n+S(n-1)
    else: return n

n=int(input('Informe um valor positivo: '))
if n>0: print(S(n))
```

Recursividade

- Exemplo 2: Fatorial
  - n ϵ naturais
  - fat(6)=6*5*4*3*2*1=720

$$\text{fat(n)}=\begin{cases} 1, & se\ n = 0\ ou\ 1 \\ n * fat(n-1), & caso\ contrário \end{cases}$$

Solução trivial

Solução recursiva

- Versão recursiva

```
1 def fat(n):
2     if not(n== 0 or n==1):
3         return n*fat(n-1)
4     else:
5         return 1
```

- Versão iterativa:

```
1 def fatI(n):
2     prod=1
3     i=1
4     while i<=n:
5         prod*=i
6         i+=1
7     return prod
```

Recursividade

- Exemplo 3: Fibonacci
  - n ∈ naturais
  - fib(6)={1,1,2,3,5,**8**}

$$\text{fib(n)}=\begin{cases} 1, & se\ n < 2 \\ fib(n-1)+fib(n-2), & caso\ contrário \end{cases}$$

- Versão recursiva

```
def fib(n):
    if n>2: return fib(n-1)+fib(n-2)
    else: return 1

n=int(input('Informe um valor positivo: '))
if n>0: print(fib(n))
```

- Versão iterativa:

```
def fib(n):
    cont,post,ant=1,1,0
    while(cont<n):
        aux=post
        post+=ant
        ant=aux
        cont+=1
    return post

n=int(input('Informe um valor positivo: '))
if n>0: print(fib(n))
```


wener.sampaio@ufpi.edu.br

Recursividade

- Exemplo 3: Fibonacci
  - Versão recursiva
    - Cada nova chamada da função cria mais duas novas variáveis
    - Se n=30, são necessárias 536.6870.917 variáveis
    - Alto consumo de memória.
  - Versão iterativa
    - Para qualquer n, sempre serão 4
    - Baixo consumo de memória

Chamada da função	Quantidade de variáveis n armanezadas na memória
1	1
2	2
3	4
4	8
5	16
6	32
7	64
8	128
9	256
10	512
11	1024
12	2048
13	4096
14	8192
15	16384
16	32768
17	65536
18	131072
19	262144
20	524288
21	1048576
22	2097152
23	4194304
24	8388608
25	16777216
26	33554432
27	67108864
28	134217728
29	268435456
30	536870912

Recursividade

- Exemplo 4: Soma A+B:
  - soma(4,3) = 1+soma(4,2) = 1+1+ soma(4,1) = ... = 7
  - Versão Recursiva:

```
1 def soma(a,b):
2     if abs(a) >= abs(b):
3         if b>0 :return 1+soma(a,b-1)
4         elif b==0: return a
5         else: return -1+soma(a,b+1)
6     else:
7         return soma(b,a)
```

Exercícios

- Use recursividade nas soluções a seguir:

1. Calcule o MDC de dois números inteiros x e y usando a definição recursiva:
   - MDC(x, y) = MDC(x − y, y), se x > y
   - MDC(x,y) = MDC(y,x)
   - MDC(x,x) = x
2. Transforme um inteiro decimal em binário.
3. Multiplicação de dois inteiros. Considere positivos e negativos.
4. Potenciação de dois inteiros. Considere positivos e negativos

Desafios

- Use recursividade nas soluções a seguir:

1. Calcule a soma de todos os valores de uma lista de reais
2. Dado uma lista de inteiros e o seu número de elementos, inverta a posição dos seus elementos
3. Determine quantas vezes um dígito K ocorre em um número natural N. Por exemplo, o dígito 2 ocorre 3 vezes em 762021192.