Departamento de Sistemas de Computação
Universidade de São Paulo
SSC 501 – Introdução a Ciência de Computação I

Funções III
Recursão

Seiji Isotani, Rafaela V. Rocha

sisotani@icmc.usp.br

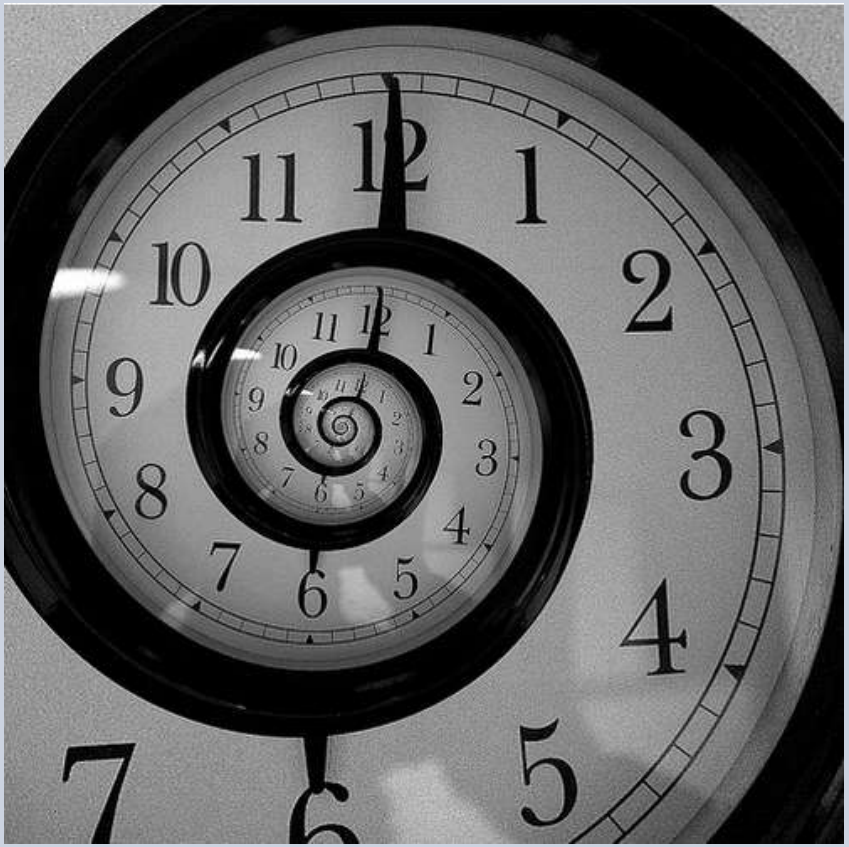
rafaela.vilela@gmail.com

PAE: Armando M. Toda, Geiser Chalco

armando.toda@gmail.com

geiser.gcc@gmail.com

Aplicação Visual



Definição

- O conceito de recursão é fundamental em computação.
- Um objeto é dito ser recursivo se ele é definido parcialmente em termos de si próprio.
- Recursão é uma técnica poderosa em definições matemáticas. O poder da recursão está na possibilidade de se definir elementos com base em versões mais simples deles mesmos.

Definição

- A recursão permite definir um problema em termos de uma ou mais versões menores deste mesmo problema.
- Em computação temos recursividade quando um dos passos de um algoritmo é repetir o mesmo algoritmo para uma parte menor do problema.
- Exemplos:
  - Soma dos N primeiros números naturais
  - Calculo de potência
  - Calculo de Fatorial

Definição

Ex: definir os números naturais

- Definindo os números naturais:
  - Seja 0 um número natural. Para todo número natural N existe (N +1) que é seu sucessor.

Soma dos números naturais.

Supondo N = 5

- S(1) = 1 = 1
  - **S(1) = 1** ---------->solução trivial

- S(2) = 1+2 = 3
  - **S(2)** = **S(1) + 2** -> 1 + 2 = 3

- S(3) = 1+2+3 = 6
  - **S(3)** = **S(2)** + 3 -> 3 + 3 = 6

- S(4) = 1+2+3+4 = 10
  - **S(4)** = **S(3) + 4** -> 6 + 4 = 10

- S(5) = 1+2+3+4+5 = 15
  - **S(5)** = **S(4) + 5** -> 10 + 5 = 15

Como definir e resolver um problema recursivo?

Recursividade

- Método da divisão e conquista.
  - Dividir o problema em subproblemas do mesmo tipo, ou seja, problemas menores. Resolver os problemas menores e ter a solução do problema fina.
- Se o problema é pequeno
  - Resolva-o diretamente.
- Senão
  - Reduza a um problema menor do mesmo problema.
  - Resolve o problema menor.
  - Volte para o problema original.

Função Recursiva

"É uma função que chama ela mesma."

```
SOMA(N)
    SE N = 1
        RETORNA 1
    SENÃO
        RETORNA N + SOMA(N-1)
    FIM SE
FIM SOMA
```

Função Recursiva

3 passos para definir a função recursiva:

- Verificar qual a condição de parada (caso base) onde o algoritmo termina.
- Pensar em como dividir o problema em problemas menores do mesmo tipo.
- Encontrar uma maneira de chamar recursivamente a função passando para ela um parâmetro do problema menor.

O que faz esta função?

```
int recursive(int x, int n)
{
    if(n == 1) //condição de parada
    {
        return x;          }
    else
    {
            return x * recursive(x, n-1);
    }
}
```

Chama recursivamente a própria função passando para ela um parâmetro do problema menor

Exercício

- Faça uma função recursiva para calcular o fatorial de um número.

- int fatorial(int n)

Fatorial

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

```
int fatorial(int n) {
    int r;
    if(n == 1 || n== 0) {
        r = 1;
        printf("Fatorial de %d= %d; \n", n, r);
        return r;
    }
    else  {
        r = n*fatorial(n-1);
        printf("Fatorial de %d= %d; \n", n, r);
        return  r;
    }
}

int main(){
        int x = 4;
        int f;
        f = fatorial(x);
        printf("Resultado: Fatorial de %d= %d  \n", x, f);
}
```

Execução do Fatorial

```
1  int fatorial(int n)
2  {
3     if(n == 1)
4     {
5        return 1;
6     }
7     else
8     {
9        return n*fatorial(n-1);
10    }
11 }
12 main (){
13    int n=4;
14    n= fatorial(n);
15 }
```

```
12  main
13  n=4
14  fatorial(4)
 1  fatorial(4)
…
 9      4*fatorial(3)
…
 9      4*[3*fatorial(2)]
…
 9              4*[3*[2*fatorial(1)]]
…
 5                      4*[3*[2*1]]
 9              4*[3*[2]]
 9      4*[6]
 9   24
14   n=24
```

Sequência de Fibonacci

- **Sequência de Fibonacci**), é uma sequência de números inteiros, começando normalmente por 0 e 1, na qual, cada termo subsequente corresponde a **soma dos dois anteriores**
- Ex: crescimento de uma população de coelhos

$$f(n) = \begin{cases} 0, & n = 0 \\ 1, & n = 1 \\ f(n-1) + f(n-2), & n > 1 \end{cases}$$

http://pt.wikipedia.org/wiki/Número_de_Fibonacci

0,1, **1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, ...**

Sequência de Fibonacci

- Versão Iterativa

```
int fibonacci(int n)
 a = 0;
  b = 1;

 for(i=0;i<n;i++)
    {
       aux = b
       b = a+b
       a = aux
    }
    return b;
}
```

Exercício

- Faça a versão Recursiva da sequencia de Fibonacci.

int fibonacci(int n);

Sequência de Fibonacci

- Versão Recursiva

```
int fibonacciR(int n)
{
   if(n < 2)
   {
      return n;
   }
   else
   {
      return fibonacciR(n-1) + fibonacciR(n-2);
   }
}
```