UNIVERSIDADE FEDERAL DO CEARÁ – UFC SOBRAL

TÉCNICAS DE PROGRAMAÇÃO – 2019.1 – PROF. WENDLEY

AULA PRÁTICA: 05
06/05/2019

Recursividade

www.ec.ufc.br/ wendley

UM POUCO SOBRE RECURSIVIDADE

Um objeto é dito recursivo se pode ser definido em termos de si próprio. Exemplo: "Para fazer um iogurte, você precisa de leite e de um pouco de iogurte". A recursão é uma forma interessante de resolver problemas, pois o divide em problemas menores de mesma natureza. A chamada recursiva executa enquanto a chamada original para o método ainda estiver ativa. O passo de recursão (chamada recursiva) pode resultar em muitas outras chamadas recursivas à medida que o método divide cada novo subproblema em duas partes conceituais.

Para que a recursão termine, toda vez que o método chamar a si próprio com uma versão mais simples do problema original, a sequência de problemas cada vez menores deve convergir em um caso básico. Nesse ponto, o método reconhece o caso básico e retorna um resultado à cópia anterior do método. Uma sequência de retornos segue até a chamada do método original retornar o resultado final para o "chamador".

Um processo recursivo consiste de duas partes:

- O caso trivial, cuja solução é conhecida
- Um método geral, que reduz o problema a um ou mais problemas menores de mesma natureza

Em decorrência disso, as funções recursivas precisam ter duas partes:

- Ponto de parada
- Regra geral

Exemplo de cálculo do fatorial que utiliza recursão:

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

Para realizar os exercícios práticos adiante, utilize o programa NetBeans com Java.

EXERCÍCIO PRÁTICO – 1 – Implemente em Java os métodos a seguir e use uma classe main para verificar as suas funcionalidades

Método 1 – Fatorial

```
public int fatorial(int f) {
      if (f == 0)
          return 1;
      else
          return (f * fatorial(f - 1));
}
```

Método 2 – Receber um número N e calcular a soma dos números pares de 0 até N

```
public int somaPares(int n) {
        int soma = 0;
        if (n == 0)
            return 0;
        else {
            if(n%2==0)
                soma = n;
            return (soma + somaPares(n - 1));
        }
    }
```

Método 3 – Escrever na tela os números de 1 a 50

```
public int mostrar(int m){
        if(m == 50)
            return 50;
        else
            System.out.println(m);
            return(mostrar(m+1));
    }
```

Método 5 – Somar um intervalo fornecido

```
public int soma_mn(int m, int n){
        if(n == m)
            return ________ ;
        else if(m < n)
            return ________ (________,________) + n;
        else
            return ________ (________,________) + ________;
    }
```

Método 6 – Mostrar série de Fibonacci

```
public int fib(int num){
        int aux;
        if(num < 2)
            return ________ ;
        else {
            aux = fib(num - 1) + fib(num - ________);
            return ________ ;
        }
    }
```