RECURSIVIDADE

ESTRUTURA DE DADOS

Profa. Ma. Andréa Zotovici

COMO CALCULAR O FATORIAL

4! = 4 x 3 x 2 x 1 = 24

A partir deste valor

Chegar até este

fat = fat * N;
N = N – 1;

N	fat
4	1
3	4
2	12
1	24

Elemento neutro da operação

PARTES DE UMA ESTRUTURA DE REPETIÇÃO

Toda estrutura de repetição possui quatro partes fundamentais:

1ª Inicialização de uma variável que controla a repetição

2ª Teste da variável para decidir se continua a repetição ou se a finaliza

3ª Alteração da variável que controla a repetição

4ª Instrução(ões) a repetir

PARTES DE UMA ESTRUTURA DE REPETIÇÃO PARA O CÁLCULO DO FATORIAL

Para o fatorial as três partes fundamentais são:

1ª Inicialização de uma variável que controla a repetição

N = valor fornecido por parâmetro

2ª Teste da variável para decidir se continua a repetição ou se a finaliza

N > 1

3ª Alteração da variável que controla a repetição

N - 1

4ª Instrução(ões) a repetir

fat = fat * N; → para funcionar corretamente fat deve ser inicializada com 1 antes da repetição

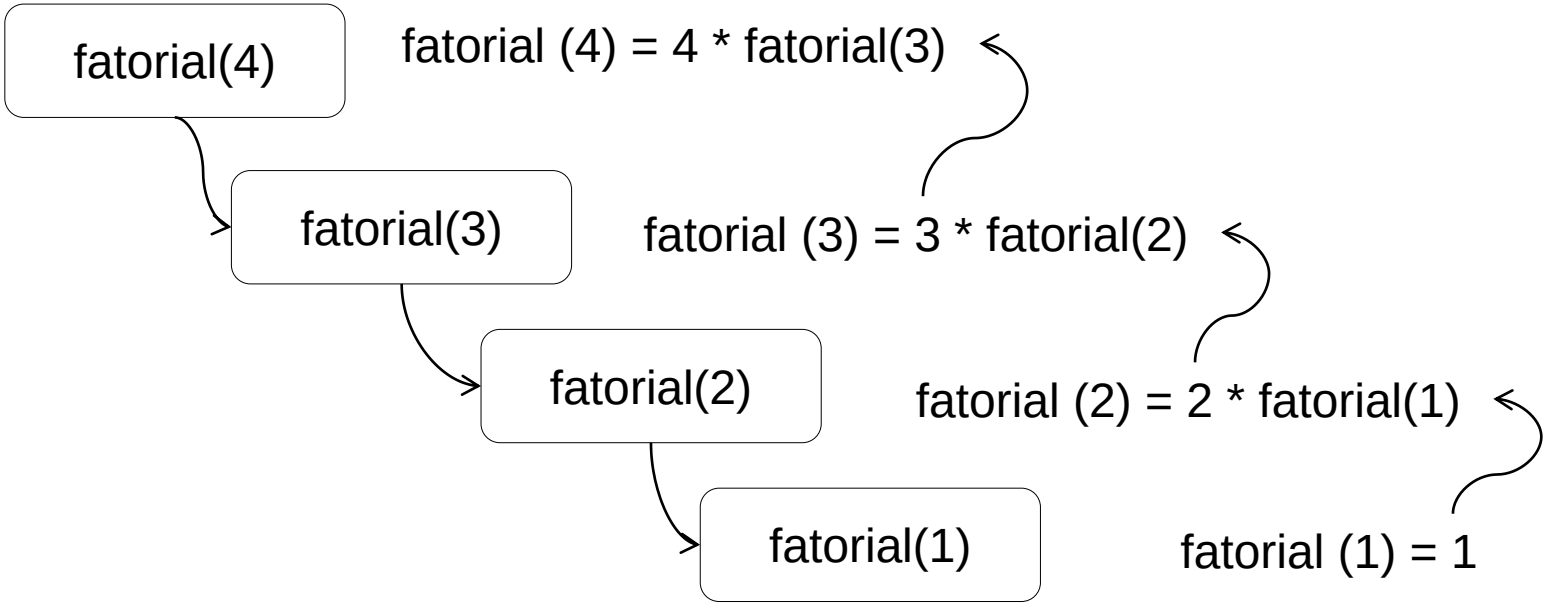
IMPLEMENTAÇÃO NÃO RECURSIVA PARA O CÁLCULO DO FATORIAL

```
public int fatorial (int n) {              //Inicializa a variável de controle

      int fat = 1;

      while (n > 1) {                      // Testa a variável de controle
            fat = fat * n;
            n = n – 1;                     // Atualiza a variável de controle
      }

      return fat ;

}
```

RECURSIVIDADE

Realizar a repetição de comandos sem utilizar laços como **for** e **while**. É utilizada a chamada de um método por si mesmo, denominada **recursão**.

Exemplo: desenvolver um método recursivo para o cálculo do fatorial de um número (N). Considere que se deseja calcular 4!:



RECURSIVIDADE

Considere que se deseja calcular 0!:

fatorial (0) = 1

LÓGICA PARA O CÁLCULO DO FATORIAL

Assim:

Critério de Parada

fatorial (N) =

1	se N == 0 OU N==1
N . fatorial (N-1)	se N > 1

LÓGICA PARA O CÁLCULO DO FATORIAL

Assim:

fatorial (N) =

1 se N == 0 OU N==1

N . fatorial (N-1) se N > 1

Critério de Continuidade

NÃO RECURSIVA X RECURSIVA

```
public int fatorial (int n) {

      int fat = 1;

      while (n > 1) {

            fat = fat * n;
            n = n – 1;

      }
      return fat ;
}
```

```
public int fatorialR (int n) {


      if  (n > 1) {

            return fatorialR (n-1) * n;

      }

      return 1 ;
}
```

IMPLEMENTAÇÃO RECURSIVA PARA O CÁLCULO DO FATORIAL

```
public int fatorialR (int n) {


      if  (n == 1 || n == 0) {

            return 1;

      }

      return fatorialR (n-1) * n ;
}
```

IMPLEMENTAÇÃO DE UMA CLASSE E MÉTODO PRINCIPAL PARA TESTAR O CÁLCULO DO FATORIAL

```
public class Inteiro {

        // adicione aqui o método fatorial

        public static void main (String args[]) {

                Inteiro objetoInt = new Inteiro ();

                System.out.println(objetoInt.fatorial(4));
        }
}
```

RECURSIVIDADE

Gerar o termo N da seqüência de Fibonacci recursivamente.

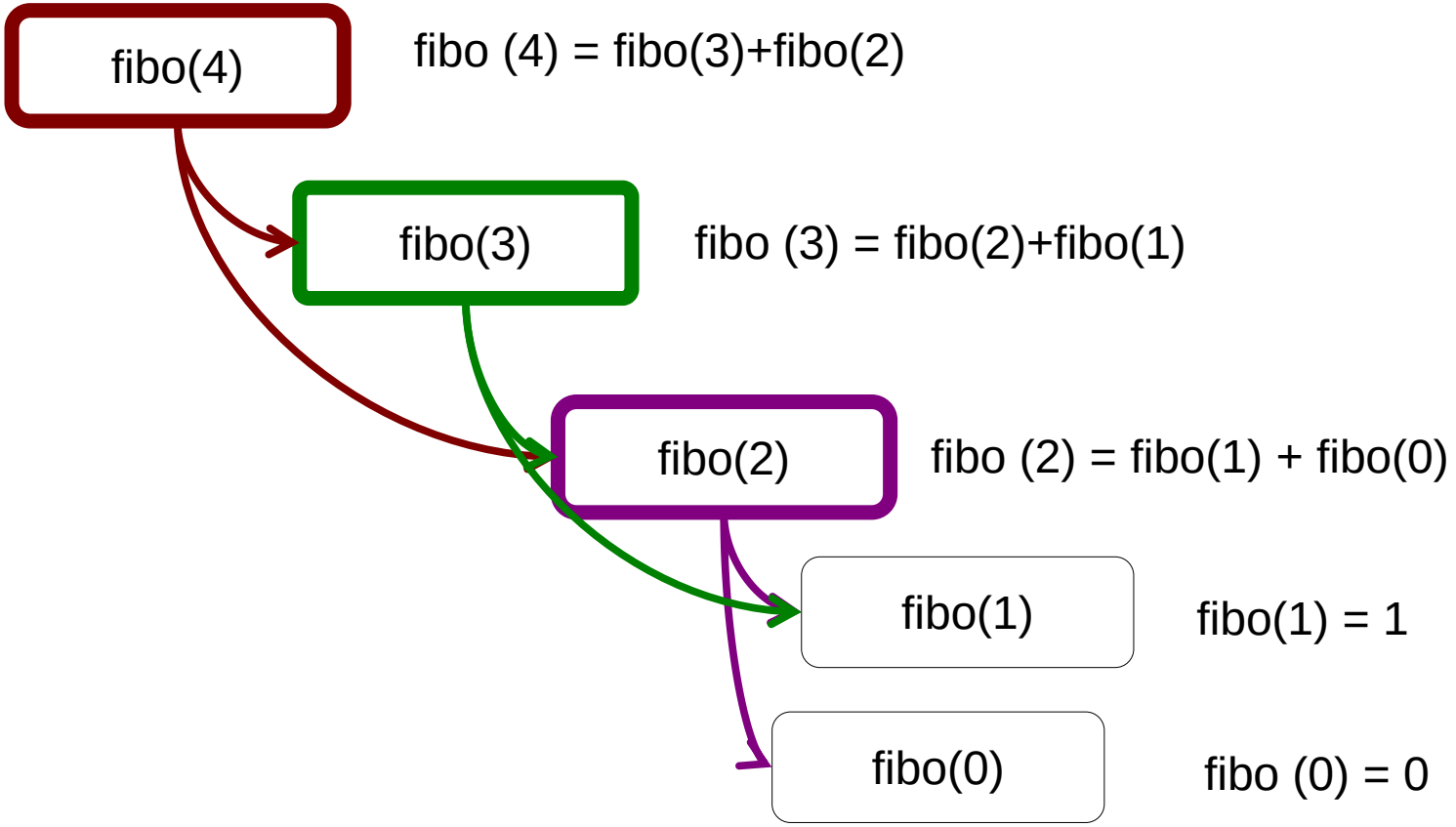
Considerando N=4

Termos gerados até o 4o. :

1 1 2 **3**

SEQUÊNCIA DE FIBONACCI

Gerar o termo N. Considerando N=4



RECURSIVIDADE

```
public int fibonacci (int n) {

        if ( n == 0 ) {
                return 0;
        }

        if ( n == 1 ) {
                return 1;

        }

        return fibonacci (n-1) + Fibonacci (n-2) ;
}
```

EXERCÍCIOS

Para resolver os exercícios a seguir, leia o capítulo 7 (páginas: 347 – 360) do livro de Koffman e Wolfgang (2008)

EXERCÍCIOS

1. Dado o método:

```
public static int metodoB(int n, int m){
        if (m>n){
                return 0;
        }
        if (m==n){
                return n;
        }
        return (metodoB(n-1,m+1)+(n+m));
}
```

a) Qual o valor de metodoB(5,1)?

b) Ilustre passo-a-passo cada chamada feita no item a

2. Dado o método:

```
public static int metodoA(int n){
        if (n == 1){
                return 1;
        }
        return metodoA(n-1)+n;
}
```

a) Qual o valor de metodoA(5)?

b) Ilustre passo-a-passo cada chamada feita no item a

EXERCÍCIOS

3. (ENADE-ADS-2014) Uma função é denominada recursiva quando ela é chamada novamente dentro de seu corpo. Implementações recursivas também tendem a ser menos eficientes, porém facilitam a codificação e seu entendimento.
CELES, W.; CERQUEIRA, R.; RANGEL, J.L. Introdução a estruturas de dados. Rio de Janeiro, 2004 (adaptado).

Considere a função recursiva f(), a qual foi escrita em linguagem C:

```
int f ( int v[], int n){
        if(n == 0)
                return 0;
        else {
                int s;
                s = f (v, n-1);
                if (v[n-1] > 0) s = s + v [n-1];
                        return s;
        }
}
```

Suponha que a função f() é acionada com os seguintes parâmetros de entrada:
f({2,-4, 7, 0, -1, 4}, 6);
Nesse caso, o valor de retorno da função f() será: a) 8 b) 10 c) 13 d) 15 e) 18

EXERCÍCIOS

Considere m = 250 e n = 150. Calcular o máximo divisor comum:

250, 150	**2**	250 % 150 = 100
125, 75	3	150 % 100 = 50
125, 25	**5**	100 % 50 = 0
25, 5	**5**	**50**
5, 1	5	
1, 1		

m.d.c.(250, 150) = 2 x 5 x 5 = 50

4. Escreva um método em Java para calcular o máximo divisor comum de dois números inteiros.

EXERCÍCIOS

5. Escreva um método em Java para calcular um termo da sequência de Fibonacci. Utilize um vetor para guardar cada termo anteriormente calculado e use-os no cálculo dos termos seguintes.

6. Escreva um método em Java para calcular o número de Euler:

$$e = \sum_{n=0}^{\infty} \frac{1}{n!}$$

7. Escreva um método em Java para calcular o π:

$$\pi = \sum_{n=0}^{\infty} (-1)^n \ \frac{4}{2n+1}$$

8. Escreva um método em Java para calcular:

$$\frac{1}{2} + \frac{1}{3} + \frac{1}{5} + \frac{1}{7} + \frac{1}{11} + \ldots + \frac{1}{N}$$

EXERCÍCIOS

9. Baseado no ENADE 2014 - Dado o método a seguir, informe o valor que ele devolve no final da sua execução considerando que o valor inicial de n (parâmetro) é 27. Mostre cada chamada feita ao método recursivo a seguir.

```
int recursao (int n){
       if (n > 10) {
              return recursao ( recursao (n/3) );
       } else {
              return n * 2;
       }
}
```

EXERCÍCIOS

10. A sequência de Fibonacci é uma sequência de números inteiros que começa em 1, a que se segue 1, e na qual cada elemento subsequente é a soma dos dois elementos anteriores. A função fib a seguir calcula o n-ésimo elemento da sequência de Fibonacci:

```
unsigned int fib (unsigned int n) {
       if (n < 2)
              return 1;
       return fib (n-2) + fib (n-1);
}
```

Considerando a implementação acima, avalie as afirmações a seguir.

I. A complexidade de tempo da função fib é exponencial no valor de n
II. A complexidade de espaço da função fib é exponencial no valor de n
III. É possível implementar uma versão iterativa da função fib com complexidade de tempo linear no valor de n e a complexidade de espaço constante.

É correto o que se afirma em:

a) I, apenas
b) II, apenas
c) I e III, apenas
d) II e III, apenas
e) I, II e III

BIBLIOGRAFIA

KOFFMAN, Elliot B.; WOLFGANG, Paul A. Objetos, abstração, estruturas de dados e projeto usando C++. Tradução Sueli Cunha: revisão técnica Orlando Bernardo Filho, João Araújo Ribeiro. Rio de Janeiro: LTC, 2008.