

ALGORITMOS AVANÇADOS

RECURSÃO

Prof. Michael da Costa Móra

Recursão

- Algoritmo recursivo é aquele que contém, em sua descrição, uma ou mais chamadas a si mesmo
- Em termos de programação
 - Uma rotina é recursiva quando ela chama a si mesma de forma direta ou indireta
 - Indireta é quando uma rotina X contém uma chamada a outra rotina Y, que por sua vez contém uma chamada a X

Recursão

- CUIDADO !!!
 - Devemos garantir que uma chamada recursiva não criará um *loop* (ou laço infinito)
 - Devemos sempre construir o algoritmo de modo que um caso base permita que a recursão termine
 - Deve haver uma expressão lógica que, em algum instante, tornar-se-á falsa e permitirá que a recursão termine

Recursão

- Vamos analisar o exemplo do fatorial
 - Cálculo recursivo do fatorial de um número

$$\left\{ \begin{array}{l} \textit{fatorial}(0) = 1 \\ \textit{fatorial}(1) = 1 \\ \textit{fatorial}(2) = 2 \\ \textit{fatorial}(3) = 6 \\ \textit{fatorial}(n) = n * \textit{fatorial}(n-1), n \geq 2 \end{array} \right.$$

- Caso base: se o número é 0 ou 1, o resultado é 1
- Passo da recursão: se o número é maior ou igual a 2, o resultado é o número multiplicado pelo fatorial do número decrescido de uma unidade

Recursão

- Vamos analisar o exemplo do fatorial
 - A entrada é o valor 3
 - O método chama fatorial() com o valor 2
 - A entrada é o valor 2
 - O método chama fatorial() com o valor 1
 - A entrada é o valor 1
 - O método retorna o resultado = 1
 - O método retorna o resultado = 2×1
 - O método retorna o resultado = 3×2

Recursão

- Vamos analisar o exemplo do fatorial: Java

```
public class MyMath
{
    public static int fatorial (int n)
    {
        int fat = -1; // -1 indica erro
        if ( n == 0 || n == 1 )
            fat = 1;
        else if (n>=2)
            fat = n * fatorial(n-1);
        return fat;
    }
}
```


Recursão

- Vamos analisar o exemplo do fatorial: Java

```
import java.util.*;
public class Teste {
    public static void main(String[] args)
    {
        Scanner entrada = new Scanner(System.in);
        System.out.println("Digite um número inteiro
                           não negativo:");
        int valor = entrada.nextInt();
        int resultado = MyMath.fatorial(valor);
        System.out.println(valor+"! = "+resultado);
    }
}
```

Recursão

```
int resultado = MyMath.fatorial(3);
```



```
public static int fatorial(int n) {  
    int fat = -1;  
    if (n==0 || n==1) fat = 1;  
    else if(n>=2) fat=n*fatorial(n-1);  
    return fat;  
}
```


Recursão

```
int resultado = MyMath.fatorial(3);
```

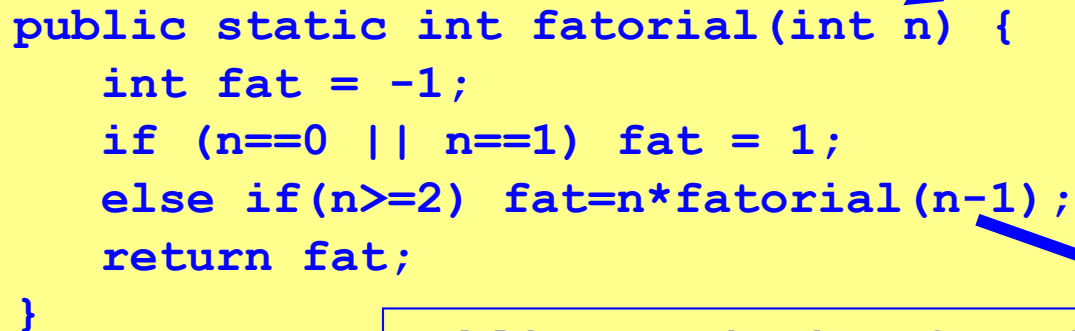
```
public static int fatorial(int n) {  
    int fat = -1;  
    if (n==0 || n==1) fat = 1;  
    else if(n>=2) fat=n*fatorial(n-1);  
    return fat;  
}
```

```
public static int fatorial(int n) {  
    int fat = -1;  
    if (n==0 || n==1) fat = 1;  
    else if(n>=2) fat=n*fatorial(n-1);  
    return fat;  
}
```

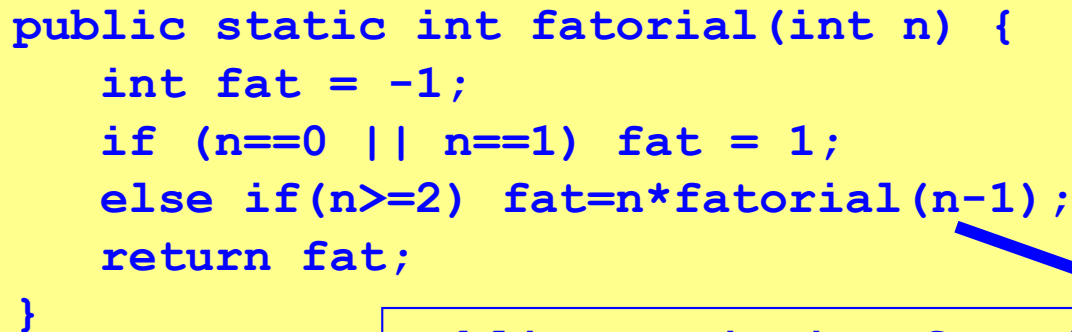
Recursão

```
int resultado = MyMath.fatorial(3);
```

```
public static int fatorial(int n) {  
    int fat = -1;  
    if (n==0 || n==1) fat = 1;  
    else if(n>=2) fat=n*fatorial(n-1);  
    return fat;  
}
```



```
public static int fatorial(int n) {  
    int fat = -1;  
    if (n==0 || n==1) fat = 1;  
    else if(n>=2) fat=n*fatorial(n-1);  
    return fat;  
}
```



```
public static int fatorial(int n) {  
    int fat = -1;  
    if (n==0 || n==1) fat = 1;  
    else if(n>=2) fat=n*fatorial(n-1);  
    return fat;  
}
```

Recursão

```
int resultado = MyMath.fatorial(3);
```

```
public static int fatorial(int n) {  
    int fat = -1;  
    if (n==0 || n==1) fat = 1;  
    else if(n>=2) fat=n*fatorial(n-1);  
    return fat;  
}
```

```
public static int fatorial(int n) {  
    int fat = -1;  
    if (n==0 || n==1) fat = 1;  
    else if(n>=2) fat=n*fatorial(n-1);  
    return fat;  
}
```

```
public static int fatorial(int n) {  
    int fat = -1;  
    if (n==0 || n==1) fat = 1;  
    else if(n>=2) fat=n*fatorial(n-1);  
    return fat;  
}
```

2

1

Recursão

```
int resultado = MyMath.fatorial(3);
```

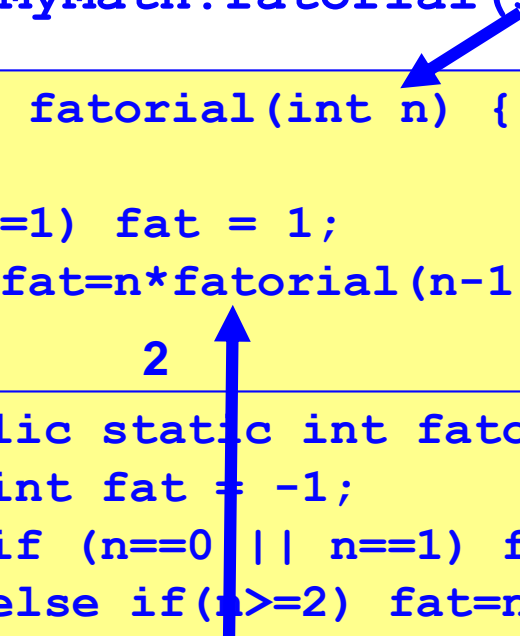
```
public static int fatorial(int n) {  
    int fat = -1;  
    if (n==0 || n==1) fat = 1;  
    else if(n>=2) fat=n*fatorial(n-1);  
    return fat;  
}
```

```
public static int fatorial(int n) {  
    int fat = -1;  
    if (n==0 || n==1) fat = 1;  
    else if(n>=2) fat=n*fatorial(n-1);  
    return fat;  
}
```

Recursão

```
int resultado = MyMath.fatorial(3);
```


```
public static int fatorial(int n) {  
    int fat = -1;  
    if (n==0 || n==1) fat = 1;  
    else if(n>=2) fat=n*fatorial(n-1);  
    return fat;  
}
```



```
public static int fatorial(int n) {  
    int fat = -1;  
    if (n==0 || n==1) fat = 1;  
    else if(n>=2) fat=n*fatorial(n-1);  
    return fat;  
}
```

Recursão

```
int resultado = MyMath.fatorial(3);
```




```
public static int fatorial(int n) {  
    int fat = -1;  
    if (n==0 || n==1) fat = 1;  
    else if(n>=2) fat=n*fatorial(n-1);  
    return fat;          3  2  
}
```

Recursão

```
int resultado = MyMath.fatorial(3);
```

6



```
public static int fatorial(int n) {  
    int fat = -1;  
    if (n==0 || n==1) fat = 1;  
    else if(n>=2) fat=n*fatorial(n-1);  
    return fat;  
}
```

Recursão

fatorial(1)
fatorial(2)
fatorial(3)
main

fatorial(1)
fatorial(2)
fatorial(3)
main

$n = 1; \text{fat} = 1$

$n = 2; \text{fat} = 2 * \text{fatorial}(1)$

$n = 3; \text{fat} = 3 * \text{fatorial}(2)$

return 1

return fat = 2 * 1

return fat = 3 * 2

Recursão

- Vamos analisar o exemplo do fatorial
 - O cálculo do fatorial possui um exemplo não-recursivo bastante trivial utilizando um laço de repetição
 - A essa solução chamamos de solução iterativa!

```
int f = 1;  
for (int i = n; i > 1; i--)  
    f = f * i;
```

Recursão

- Recursividade
 - Poderosa ferramenta de programação
 - Apesar de bastante empregada, nem sempre ela pode ser aplicada
 - É preciso analisar o problema e ver se necessita de uma solução recursiva.
- Quando bem empregada, a recursão pode tornar uma solução clara, simples e concisa

Recursão

- Vantagens

- Rotinas mais concisas
- Relação direta com uma prova por indução matemática (facilita verificar a correção)

- Desvantagem

- Custo adicional de execução
 - Cada chamada recursiva implica em um custo (tempo e espaço)
 - Alocação de memória
 - Salvamento do estado
 - A cada chamada variáveis locais da nova chamada são recriadas e informações da chamada corrente são armazenadas na pilha