



Entendendo a Recursão II

A


recursão é bastante poderosa, pois permite que uma função ou um procedimento possa chamar a execução dentro mesmo. Uma vez que a função ou o procedimento forem definidos, é possível executar a mesma função ou procedimento várias vezes.

ALGORITMOS DE RECURSÃO SEM CAUDA

A recursão sem cauda acontece quando, durante a execução da recursão, ela não deixa um rastro na memória, isto é, não tem uma cauda.

A recursão sem cauda, normalmente, acontece quando a sua recursão é um procedimento e não retorna um valor.

Para exemplificar a aplicação da recursão sem cauda, vamos desenvolver um exercício simples. Um deles é o desenvolvimento de um algoritmo em pseudocódigo, que calcula o fatorial de um número natural. Neste caso, vamos utilizar modularização e recursão sem cauda.

Vamos revisar o entendimento do fatorial de um número natural, lembrando que o fatorial é o produto do número com os seus antecessores até o número 1.  fatorial de zero é 1.

FATORIAL

se $n = 0$, então, o $\text{fat}(n)$ é 1

caso contrário, $\text{fat}(n) = n * \text{fat}(n-1)$, para $n > 0$

se $n = 0$ então, o valor de $n!$ é 1

caso contrário, $n! = 1 * 2 * 3 * \dots * (n-2) * (n-1) = (n-1)! * n$

O módulo 'procedimento' para o cálculo do fatorial segue com algoritmo. Observe que os comandos do então do se, referem-se à regra 1 da recursão e os comandos do 'senão' do 'se' referem-se às regras 2 e 3 da recursão.

```
fatP (n inteiro, x inteiro, f inteiro)
  início
    se (x = 0 ou x = 1) então
      escrever("O fatorial de " + n + " é " + f); // regra 1
    senão
      // chamada da função recursiva
      fatP(n, x-1, f*x); // regra 2 e 3
    fimse;
  fim_módulo;
```

Segue o algoritmo que realiza a chamada do módulo procedimento recursivo do cálculo do fatorial.

Algoritmo FatorialP

```
início
    nro inteiro;

    escrever ("Digite um valor que você deseja saber o fatorial");
    ler(nro);
    se (nro < 0)
        então
            escrever ("Valor inválido para cálculo de fatorial, o
            valor precisa ser maior ou igual a zero");
            //sair do programa
        senão
            fatP(nro, nro, 1); // chamada da função recursiva
    fimse;
fim_módulo;
```



Ao realizar o teste de mesa, para o fatorial do número 5 e executar a chamada do método fat(5, 1), temos o que se segue:



fat (5, 1)

fat (4, 5)

fat (3, 20)

fat (2, 60)

fat (1, 120)

O fatorial é 120

RECURSÃO SEM CAUDA EM JAVA

A linguagem de programação que estamos utilizando é a linguagem Java.

Lembre-se de que você deve deixar seu programa bem alinhado e indentado para que, posteriormente, consiga entender e dar suporte para o programa.



A documentação é uma parte importante quando se desenvolve um programa. Faça sempre comentários no seu código.

Para exemplificar a aplicação da recursão sem cauda com a Linguagem de Programação Java, vamos desenvolver um exercício simples.

Uma delas é o desenvolvimento de um algoritmo, em Java, usando a IDE NetBeans, que calcula o número fatorial de um número natural. Neste caso, vamos utilizar o método procedimento e recursão sem cauda.

O método para o cálculo do fatorial segue com algoritmo. Observe que os comandos do então do if se refere à regra 1 da recursão e os comandos do else do if se referem às regras 2 e 3 da recursão.

```
import javax.swing.;
```

```
class FatorialP {
```

```
static void fatP (int n, int x, int f)
```

```
{
```

```
if (x == 0 || x == 1) {
```



```
    System.out.println("O fatorial de " + n + " é " + f); // regra 1
```

```
}
```

```
else
```

```
{
```

```
    // chamada da função recursiva
```

```
    fatP(n, x-1, fx); // regra 2 e 3
```

```
}
```

```
}
```


Segue o programa em Java ao usar a IDE NetBeans:

```
1  import javax.swing.*;
2
3  class FatorialP {
4      static void fatP (int n, int x, int f)
5          {...10 lines }
15
16     public static void main (String arg [ ])
17         {...15 lines }
32
33 }
```

```

class FatorialP {
    static void fatP (int n, int x, int f)
    {
        if (x == 0 || x == 1) {
            System.out.println("O fatorial de " + n + " é " + f); // regra 1
        }
        else
        {
            // chamada da função recursiva
            fatP(n, x-1, f*x); // regra 2 e 3
        }
    }
}

```



Segue o método main que realiza a chamada do método procedimento recursivo do cálculo do fatorial.

```

public static void main (String arg [])

```

```

{

```

```

    int nro;

```

```

        nro = Integer.parseInt(JOptionPane.showInputDialog("Digite um valor que
você deseja saber o fatorial"));

```

```

        if (nro < 0)

```

```

        {

```

```

            System.out.println ("Valor inválido para cálculo de fatorial, o valor precisa
ser maior ou igual a zero");

```

```

            System.exit(0);

```

```

        }

```

else



{

fatP(nro, nro, 1); // chamada da função recursiva

}

System.exit(0);

}

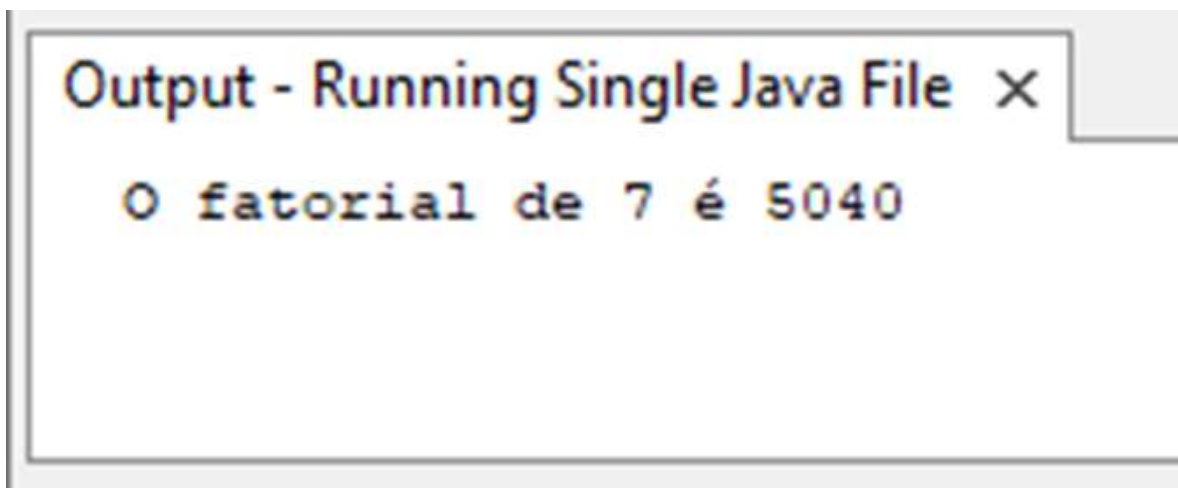
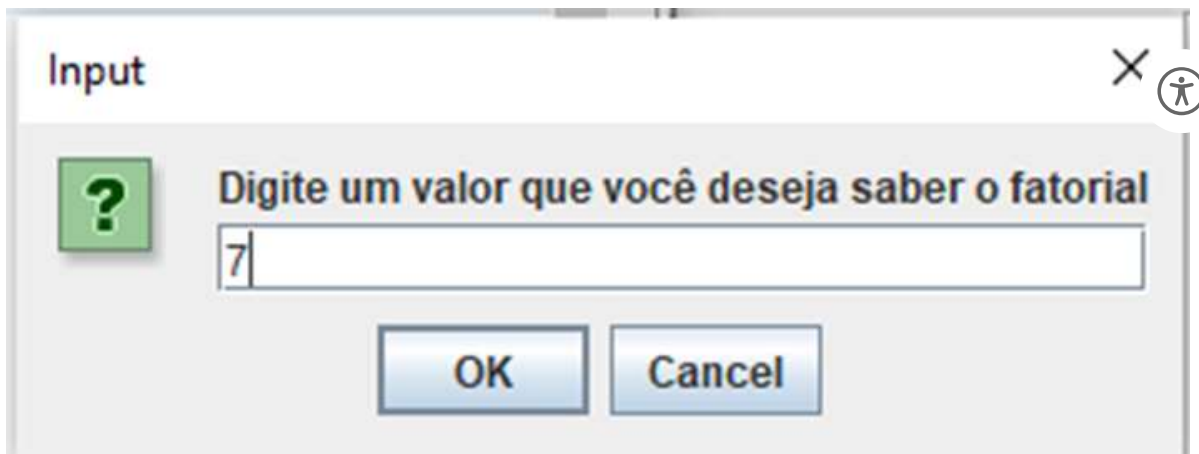
}

Segue o método main em Java escrito no NetBeans.

```
public static void main (String arg [ ])
{
    int nro;

    nro = Integer.parseInt(JOptionPane.showInputDialog("Digite um
valor que você deseja saber o fatorial"));
    if (nro < 0)
    {
        System.out.println ("Valor inválido para cálculo de
fatorial, o valor precisa ser maior ou igual a zero");
        System.exit(0);
    }
    else
    {
        fatP(nro, nro, 1); // chamada da função recursiva
    }
    System.exit(0);
}
```

Após a compilação e execução do programa Java no NetBeans, seguem a entrada de dados para o cálculo do fatorial de 7 e a saída de resultados com a mensagem “O fatorial de 7 é 5040”.



ALGORITMOS DE RECURSÃO COM CAUDA

A recursão com cauda acontece quando, durante a execução da recursão, ele deixa um rastro na memória, o que chamamos de cauda.

A recursão com cauda, normalmente, acontece quando a sua recursão é uma função e retorna um valor.

Para exemplificar a aplicação da recursão com cauda, vamos desenvolver um exercício simples. Uma delas é o desenvolvimento de um algoritmo, em pseudocódigo, que calcula o número fatorial de um número natural. Neste caso, vamos utilizar modularização e a recursão com cauda.



FATORIAL

se $n = 0$, então, o $\text{fat}(n)$ é 1

caso contrário, $\text{fat}(n) = n * \text{fat}(n-1)$, para $n > 0$

se $n = 0$ então, o valor de $n!$ é 1

caso contrário, $n! = 1 * 2 * 3 * \dots * (n-2) * (n-1) = (n-1)! * n$

O módulo 'função' para o cálculo do fatorial segue com algoritmo. Observe que os comandos do 'então' do 'se' está relacionado à regra 1 da recursão, e os comandos do 'senão' do 'se' às regras 2 e 3 da recursão.

```
numérico_inteiro fat (numérico inteiro n)
início
    Declarar
        f numérico_inteiro;

    se (n = 0)
        então
            retornar 1; // regra 1
        senão
            f <- n * fat(n-1); // regra 2 e 3
            retornar f;
    fimse;
fim_módulo;
```

Segue o algoritmo que realiza a chamada do módulo função recursivo do cálculo do fatorial.



```
Algoritmo Fatorial
início
  Declarar
    f, nro numérico_inteiro;

  escrever("Digite um valor que você deseja saber o fatorial");
  ler(nro);
  se (nro < 0)
    então
      escrever ("Valor inválido para cálculo de fatorial, o valor
        precisa ser maior ou igual a zero");
      <sair do programa>;
    senão
      f <- fat(nro);
      escrever("O fatorial de " , nro , " é " , f);
  fimse;
fim_algoritmo.
```

Ao realizar o teste de mesa, para o fatorial do número 5 e executar a chamada do método fat(5), temos o que se segue:

```
fat (5)
5 * fat(4)
5 * 4 * fat(3)
5 * 4 * 3 * fat(2)
5 * 4 * 3 * 2 * fat(1)
5 * 4 * 3 * 2 * 1 * fat(0)
5 * 4 * 3 * 2 * 1 * 1
5 * 4 * 3 * 2 * 1
5 * 4 * 3 * 2
5 * 4 * 6
5 * 24
120 = resultado do fatorial de 5.
```



A linguagem de programação que estamos utilizando é a linguagem Java.

Lembre-se de que você deve deixar seu programa bem alinhado e indentado para que, posteriormente, consiga entender e dar suporte para o programa.

A documentação é uma parte importante quando se desenvolve um programa. Faça sempre comentários no seu código.

Para exemplificar a aplicação da recursão com cauda com a Linguagem de Programação Java, vamos desenvolver um exercício simples.

Uma delas é o desenvolvimento de um algoritmo, em Java, usando a IDE NetBeans, que calcula o número fatorial de um número natural. Neste caso, vamos utilizar o método função e a recursão com cauda.

O método para o cálculo do fatorial segue com algoritmo. Observe que os comandos do então do if referem-se à regra 1 da recursão, e os comandos do else do if às regras 2 e 3 da recursão.

```
import javax.swing.*;
```



```
class Fatorial {  
  
    static int fat (int n) {  
  
        int f;  
  
        if (n == 0) {  
  
            return 1; // regra 1  
  
        }  
  
        else  
  
        {  
  
            // chamada da função recursiva  
  
            f = n * fat(n-1); // regra 2 e 3  
  
            return f;  
  
        }  
  
    }  
  
}
```

Usando a IDE NetBeans, segue o programa Java:

```

1  [-] import javax.swing.*;
2
3  class Fatorial {
4  [+   static int fat (int n) {...13 lines }
17
18       public static void main (String arg [ ])
19  [+       {...16 lines }
35
36     }
37

```

```

static int fat (int n) {
    int f;

    if (n == 0) {
        return 1; // regra 1
    }
    else
    {
        // chamada da função recursiva
        f = n * fat(n-1); // regra 2 e 3
        return f;
    }
}

```

Segue o método main que realiza a chamada do método função recursivo do cálculo do fatorial.

```
public static void main (String arg [ ])
```

```
{
```

int f, nro;



```
nro = Integer.parseInt(JOptionPane.showInputDialog("Digite um valor que  
você deseja saber o fatorial"));
```

```
if (nro < 0)
```

```
{
```

```
    System.out.println ("Valor inválido para cálculo de fatorial, o valor precisa  
ser maior ou igual a zero");
```

```
    System.exit(0);
```

```
}
```

```
else
```

```
{
```

```
    f = fat(nro); // chamada da função recursiva
```

```
    System.out.println("O fatorial de " + nro + " é " + f);
```

```
}
```

```
System.exit(0);
```

```
} // fim do void main
```

```
} // fim da classe
```

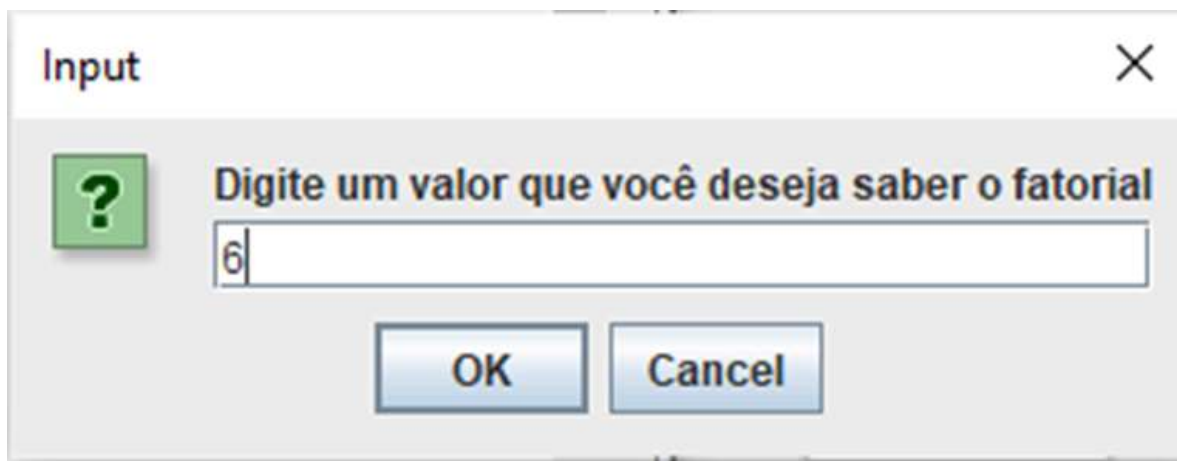
Segue em NetBeans.



```
public static void main (String arg [ ])
{
    int f, nro;

    nro = Integer.parseInt(JOptionPane.showInputDialog("Digite um
valor que você deseja saber o fatorial"));
    if (nro < 0)
    {
        System.out.println ("Valor inválido para cálculo de
fatorial, o valor precisa ser maior ou igual a zero");
        System.exit(0);
    }
    else
    {
        f = fat(nro); // chamada da função recursiva
        System.out.println("O fatorial de " + nro + " é " + f);
    }
    System.exit(0);
}
```

Após a compilação e execução do programa Java no NetBeans, seguem a entrada de dados para o cálculo do fatorial de 6 e a saída de resultados com a mensagem “O fatorial de 6 é 720”.



Output - Running Single Java File X



```
O fatorial de 6 é 720
```

Atividade extra

Indicação de leitura: **Estrutura de Dados: algoritmos, análise da complexidade e implementações em Java e C/C++**, da Ana Fernanda Gomes Ascencio e Graziela Santos de Araújo, capítulo 2.

Referência Bibliográfica

FORBELLONE, A.L.V.; EBERSPACHER, H.F. **Lógica de Programação: a construção de algoritmos e estruturas de dados**. 3ª ed. São Paulo: Prentice Hall, 2005.

PUGA, S.; RISSETTI, G. **Lógica de Programação e Estruturas de Dados, com aplicações em Java**. 3ª ed. São Paulo: Editora Pearson, 2016.



Atividade Prática 05 – Entendendo recursão II

Título da Prática: Recursão de potência de dois números inteiros

Objetivos: Entender como utilizar o netbeans para desenvolver programas em Java para o cálculo de potência de dois números com recursão

Materiais, Métodos e Ferramentas: Computador, netbeans, Java.

Atividade Prática

Quando a gente está desenvolvendo um programa ou algoritmo, muitas vezes, você precisa repetir um pedaço do código algumas vezes. E, na maioria das vezes, utilizamos as estruturas de repetição. Porém, em alguns casos, os códigos podem ficar muito complexos utilizando as estruturas de repetição.

Neste caso, temos a possibilidade de substituir as estruturas de repetição pela estrutura recursiva ou a Recursão.

A recursão é bastante poderosa de forma que consegue permitir que uma função ou um procedimento possa ser definido dentro dele mesmo. Uma vez que a função ou o procedimento foram definidos, você pode executar a mesma função ou procedimento várias vezes. Isso acontece, pois, a chamada do módulo ou da função acontecem dentro deles mesmos.

O Algoritmo de recursão para calcular a potência de dois números inteiros pode ser escrito como segue.

Desenvolva o programa em Java deste algoritmo no NetBeans.

Este é a função recursiva chamada pot que recebe dois números inteiros e retorna um número inteiro.

inteiro pot(inteiro base, inteiro exp)

início

se (base = 0) // parada da recursão, caso 1 da recursão

então

retornar 0;

senão

se (exp = 0) // parada da recursão, caso 1 da recursão

então



retornar 1;

senão

retornar pot(base , exp-1) * base; // chamada recursiva, casos 2 e 3 da
recursão

fimse;

fimse;

fim_módulo;

Agora, vamos desenvolver o módulo que faz a chamada da função recursiva.

Algoritmo Potencia

início

Declarar

b, e, p **inteiro;**

escrever(“Digite a base e o expoente para cálculo da potência”);

ler (b,e);

se (b < 0 **ou** e < 0) //garantir que os números digitados são maiores ou iguais a
zero

então

escrever("valor digitado incorretamente, valores devem ser maiores ou iguais a zero");



senão

$p \leftarrow \text{pot}(b,e)$; // chamada da função recursiva

escrever(b , " elevado a " , e , "é igual a " , p); // saída de dados

fimse;

fim_algoritmo.

Gabarito Atividade Prática

```
import javax.swing.*;
```

```
class Potencia
```

```
{
```

```
// Este é o método da função recursiva chamada pot
```

```
// que recebe dois números inteiros e retorna um número inteiro
```

```
static int pot(int base , int exp)
```



```
{
```

```
    if (base == 0) // parada recursiva, caso 1 da recursão
```

```
    {
```

```
        return 0;
```

```
    }
```

```
else
```

```
{
```

```
    if (exp == 0) // parada recursiva, caso 1 da recursão
```

```
    {
```

```
        return 1;
```

```
    }
```

```
else
```

```
{
```

```
    return pot(base , exp-1) * base; // chamada recursiva, casos 2 e 3 da
```

```
recursão
```

```
}
```

```
}
```

```
} // fim da função pot
```



//escrevendo o método principal no Java

```
public static void main(String entrada [ ])
```

```
{
```

```
    int b, e, p;
```

```
        b = Integer.parseInt(JOptionPane.showInputDialog("Digite a base para  
cálculo da potência"));
```

```
        e = Integer.parseInt(JOptionPane.showInputDialog("Digite o expoente para  
cálculo da potência"));
```

```
        if ( b < 0 || e < 0)
```

```
        {
```

```
            System.out.println("valor digitado incorretamente, valores devem ser  
maiores ou iguais a zero");
```

```
        }
```

```
        else
```

```
        {
```

```
            p = pot(b,e);
```

```
            System.out.println(b + " elevado a " + e + "é igual a " + p);
```

```
        }
```

```
System.exit(0);
```



```
} // fim do void main
```

```
} // fim da classe
```

Ir para exercício