PUC-Minas - Ciência da Computação
AED1 – Estudo Dirigido 06

Tema: Introdução à programação II
Atividade: Funções e procedimentos recursivos em Java

01.) Editar e salvar um esboço de programa em Java:

```
/**
  * Exemplo0101
  *
  * @author
  * @version 01
  */

// ---------------------------------------------  dependencias

import IO.*;

// ---------------------------------------------  definicao da classe principal

public class Exemplo0101
{
// ---------------------------------------------  definicao de metodo auxiliar

   public static void metodo01 ( int x )
   {
     // testar se valor positivo
       if ( x > 0 )
       {
         // mostrar valor
           IO.println ( “Valor lido = ” + x );
         // tentar fazer de novo com valor menor
           metodo01 ( x-1 );
       } // fim se
   } // fim metodo01( )

// ---------------------------------------------  definicao do metodo principal

   /**
    * main() – metodo principal
    */
   public static void main ( String [ ] args )
   {
     // identificar
       IO.println ( "EXEMPLO0101 - Programa em Java" );
       IO.println ( "Autor: _____" );
     // executar o metodo auxiliar
       metodo01 ( 5 );
     // encerrar
       IO.pause   ( “Apertar ENTER para terminar.” );
   } // fim main( )
} // fim class Exemplo0101
```

02.) Compilar o programa novamente.
Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
Se não houver erros, seguir para o próximo passo.
Em caso de dúvidas, consultar a apostila, recorrer aos monitores ou apresentá-las ao professor.

03.) Executar o programa.
Observar as saídas.
Registrar os dados e os resultados.
Em caso de erro (ou dúvida), usar comentários para registrar a ocorrência e,
posteriormente, tentar resolvê-lo (ou esclarecer a dúvida).

04.) Copiar a versão atual do programa para outra nova – Exemplo0102.java.

05.) Editar mudanças no nome do programa e versão.
Acrescentar um método recursivo para mostrar um valor inteiro positivo.
Na parte principal, editar a chamada do método para o novo.
Prever novos testes.

```
public static void metodo02 ( int x )
{
  // testar se valor positivo
    if ( x > 0 )
    {
      // tentar fazer de novo com valor menor
        metodo02 ( x-1 );
      // mostrar valor ( acao pendente )
        IO.println ( "Valor = " + x );
    } // fim se
} // fim metodo02( )
```

OBS.:
A exibição do primeiro valor não ocorrerá enquanto
o parâmetro (x) não chegar a zero, e não for iniciar o processo de retorno.
Os valores pendentes ainda serão conhecidos.

06.) Compilar o programa novamente.
Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
Se não houver erros, seguir para o próximo passo.

07.) Executar o programa.
Observar as saídas.
Registrar os dados e os resultados.

08.) Copiar a versão atual do programa para outra nova – Exemplo0103.java.

09.) Editar mudanças no nome do programa e versão.
Acrescentar um método recursivo para mostrar um valor inteiro positivo em outra ordem.
Na parte principal, editar a chamada do método para o novo.
Prever novos testes.

```
// ----------------------------------------------  definicao de metodo auxiliar

   public static void metodo03 ( int x )
   {
     // testar se contador valido
        if ( x > 1 )
        {
          // mostrar valor relativo
             IO.print ( " " + x );
          // tentar fazer de novo com valor menor
             metodo03 ( x-1 );      // motor da recursividade
        }
        else
        {
          // mostrar ultimo
             IO.println ( " " + x );  // base da recursividade
        } // fim se
   } // fim metodo03( )
```

OBS.:
Diferente do anterior, a exibição do primeiro valor ocorrerá antes de avançar para o próximo valor (motor).
Observar também que o último valor será tratado de forma particular (base).

10.) Compilar o programa novamente.
Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
Se não houver erros, seguir para o próximo passo.

11.) Executar o programa.
Observar as saídas.
Registrar os dados e os resultados.

12.) Copiar a versão atual do programa para outra nova – Exemplo0104.java.

13.) Editar mudanças no nome do programa e versão.
Acrescentar outro método para mostrar valores da sequência: 1 2 4 6 8
em ordem decrescente.
Na parte principal, editar a chamada do método para o novo.
Prever novos testes.

```
// ----------------------------------------------  definicao de metodo auxiliar

   public static void metodo04 ( int x )
   {
     // testar se contador valido
       if ( x > 1 )
       {
         // mostrar valor relativo
            IO.print ( " " + 2*(x-1) );
         // tentar fazer de novo com valor menor
            metodo04 ( x-1 );
       }
       else
       {
         // mostrar ultimo
            IO.println ( " 1" );
       } // fim se
   } // fim metodo04( )
```

OBS.:
Observar que o último valor será tratado de forma particular.

14.) Compilar o programa novamente.
Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
Se não houver erros, seguir para o próximo passo.

15.) Executar o programa.
Observar as saídas.
Registrar os dados e os resultados.

16.) Copiar a versão atual do programa para outra nova – Exemplo0105.java.

17.) Editar mudanças no nome do programa e versão.
Acrescentar outro método para mostrar valores de parcelas do somatório:
1 + 2/3 + 4/5 + 6/7 + 8/9 ...
Na parte principal, editar a chamada do método para o novo.
Prever novos testes.

```
// ----------------------------------------------  definicao de metodo auxiliar

   public static void metodo05 ( int x )
   {
     // testar se contador valido
        if ( x > 1 )
        {
          // tentar fazer de novo com valor menor
             metodo05 ( x-1 );
          // mostrar valor relativo
             IO.print ( " + " + (2*(x-1)) + "/" + (2*x-1) );
        }
        else
        {
          // mostrar ultimo valor (primeiro na sequencia)
             IO.print ( " 1" );
        } // fim se
   } // fim metodo05( )
```

OBS.:
Observar que o primeiro na sequência será tratado de forma particular.

18.) Compilar o programa novamente.
Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
Se não houver erros, seguir para o próximo passo.

19.) Executar o programa.
Observar as saídas.
Registrar os dados e os resultados.

20.) Copiar a versão atual do programa para outra nova – Exemplo0106.java.

21.) Editar mudanças no nome do programa e versão.
Acrescentar uma função para calcular o somatório: 1 + 2 + 3 + 4 + 5.
Na parte principal, incluir uma chamada para essa função.
Prever novos testes.

```
// ------------------------------------------------  definicao de metodo auxiliar

   public static int funcao01 ( int x )
   {
     // definir dado
        int resposta = 0;
     // testar se contador valido
        if ( x > 1 )
        {
          // tentar fazer de novo com valor menor
             resposta = x + funcao01 ( x-1 );
          // mostrar valor relativo
             IO.print ( " + " + x );
        }
        else
        {
          // mostrar ultimo
             IO.print ( " 1" );
          // ultima resposta
             resposta = 1;
        } // fim se
     // retornar resposta
        return ( resposta );
   } // fim funcao01( )

   /**
     * main() – metodo principal
     */
   public static void main ( String [ ] args )
   {
     // identificar
        IO.println ( "EXEMPLO0106 - Programa em Java" );
        IO.println ( "Autor: _____" );
        IO.println ( );
     // executar o metodo auxiliar
        IO.println ( " = " + funcao01 ( 5 ) );
     // encerrar
        IO.pause   ( "Apertar ENTER para terminar." );
   } // fim main( )
```

22.) Compilar o programa novamente.
Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
Se não houver erros, seguir para o próximo passo.

23.) Executar o programa.
Observar as saídas.
Registrar os dados e os resultados.

24.) Copiar a versão atual do programa para outra nova – Exemplo0107.java.

25.) Editar mudanças no nome do programa e versão.
Acrescentar uma função para calcular o somatório: 1 + 2/3 + 4/5 + 6/7 + 8/9 ...
Na parte principal, incluir uma chamada para essa função.
Prever novos testes.

```
// ------------------------------------------------  definicao de metodo auxiliar

   public static double funcao02 ( int x )
   {
     // definir dado
        double resposta = 0.0;
     // testar se contador valido
        if ( x > 1 )
        {
          // calcular termo e tentar fazer de novo com valor menor
             resposta = (2.0*(x-1) / (2.0*x-1)) + funcao02 ( x-1 );
          // mostrar valor relativo
             IO.print ( " + " + 2*(x-1) + "/" + (2*x-1) );
        }
        else
        {
          // ultima resposta
             resposta = 1.0;
          // mostrar ultimo
             IO.print ( " 1" );
        } // fim se
     // retornar resposta
        return ( resposta );
   } // fim funcao02( )

   /**
     * main() – metodo principal
     */
   public static void main ( String [ ] args )
   {
     // identificar
        IO.println ( "EXEMPLO0107 - Programa em Java" );
        IO.println ( "Autor: _____" );
        IO.println ( );
     // executar o metodo auxiliar
        IO.println ( " = " + funcao02 ( 5 ) );
     // encerrar
        IO.pause   ( "Apertar ENTER para terminar." );
   } // fim main( )
```

26.) Compilar o programa novamente.
Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
Se não houver erros, seguir para o próximo passo.

27.) Executar o programa.
Observar as saídas.
Registrar os dados e os resultados.

28.) Copiar a versão atual do programa para outra nova – Exemplo0108.java.

29.) Editar mudanças no nome do programa e versão.
Acrescentar uma função para calcular a quantidade de dígitos de um valor inteiro.
Na parte principal, incluir chamadas para essa função.
Prever novos testes.

```
// ------------------------------------------------  definicao de metodo auxiliar

   public static int funcao03 ( int x )
   {
     // definir dado
        int resposta = 1;                              // base
     // testar se contador valido
        if ( x >= 10 )
        {
          // tentar fazer de novo com valor menor
             resposta = 1 + funcao03 ( x/10 );         // motor 1
        }
        else
        {
            if ( x < 0 )
            {
             // fazer de novo com valor absoluto
                resposta = funcao03 ( -x );            // motor 2
            } // fim se
        } // fim se
     // retornar resposta
        return ( resposta );
   } // fim funcao03( )

   /**
     * main() – metodo principal
     */
   public static void main ( String [ ] args )
   {
     // identificar
        IO.println ( "EXEMPLO0108 - Programa em Java" );
        IO.println ( "Autor: _____" );
        IO.println ( );
     // executar o metodo auxiliar
        IO.println ( " Digitos = " + funcao03 ( 123 ) );
        IO.println ( " Digitos = " + funcao03 (     0 ) );
        IO.println ( " Digitos = " + funcao03 (  -12 ) );
     // encerrar
        IO.pause   ( "Apertar ENTER para terminar." );
   } // fim main( )
```

30.) Compilar o programa novamente.
Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
Se não houver erros, seguir para o próximo passo.

31.) Executar o programa. Observar as saídas. Registrar os dados e os resultados.

32.) Copiar a versão atual do programa para outra nova – Exemplo0109.java.

33.) Editar mudanças no nome do programa e versão.
Acrescentar uma função para calcular termo da série de Fibonacci.
Na parte principal, incluir chamadas para essa função.
Prever novos testes.

```
// ------------------------------------------------  definicao de metodo auxiliar

   public static int funcao04 ( int x )
   {
     // definir dado
        int resposta = 0;
     // testar se contador valido
        if ( x == 1 || x == 2 )
        {
          // primeiros dois valores iguais a 1
             resposta = 1;            // bases
        }
        else
        {
            if ( x > 1 )
            {
              // fazer de novo com valor absoluto
                 resposta = funcao04 ( x-1 ) + funcao04 ( x-2 );
            } // fim se
        } // fim se
     // retornar resposta
        return ( resposta );
   } // fim funcao04( )

   /**
     * main() – metodo principal
     */
   public static void main ( String [ ] args )
   {
     // identificar
        IO.println ( "EXEMPLO0109 - Programa em Java" );
        IO.println ( "Autor: _____" );
        IO.println ( );
     // executar o metodo auxiliar
        IO.println ( " Fibonacci = " + funcao04 ( 1 ) );
        IO.println ( " Fibonacci = " + funcao04 ( 2 ) );
        IO.println ( " Fibonacci = " + funcao04 ( 3 ) );
        IO.println ( " Fibonacci = " + funcao04 ( 4 ) );
        IO.println ( " Fibonacci = " + funcao04 ( 5 ) );
     // encerrar
        IO.pause   ( "Apertar ENTER para terminar." );
   } // fim main( )
```

34.) Compilar o programa novamente. Se houver erros, resolvê-los; senão seguir para o próximo passo.

35.) Executar o programa. Observar as saídas. Registrar os dados e os resultados.

36.) Copiar a versão atual do programa para outra nova – Exemplo0110.java.

37.) Editar mudanças no nome do programa e versão.
Acrescentar uma função para contar letras minúsculas em uma cadeia de caracteres.
Na parte principal, incluir chamadas para essa função.
Prever novos testes.

```
// ------------------------------------------------  definicao de metodo auxiliar

  public static int funcao05 ( String cadeia, int x )
  {
    // definir dado
      int resposta = 0;
    // testar se contador valido
      if ( 0 <= x && x < cadeia.length( ) )
      {
       // testar se letra minuscula
          if ( cadeia.charAt (x) >= 'a' &&
               cadeia.charAt (x) <= 'z' )
          {
            // fazer de novo com valor absoluto
              resposta = 1;
          } // fim se
          resposta = resposta + funcao05 ( cadeia, x+1 );
      } // fim se
    // retornar resposta
      return ( resposta );
  } // fim funcao05( )

  /**
    * main() – metodo principal
    */
  public static void main ( String [ ] args )
  {
    // identificar
      IO.println ( "EXEMPLO0110 - Programa em Java" );
      IO.println ( "Autor: _____" );
      IO.println ( );
    // executar o metodo auxiliar
       IO.println ( "Minusculas (\"abc\",0) = "  + funcao05 ( "abc"  , 0 ) );
       IO.println ( "Minusculas (\"aBc\",0) = "  + funcao05 ( "aBc" , 0 ) );
       IO.println ( "Minusculas (\"AbC\",0) = " + funcao05 ( "AbC", 0 ) );
    // encerrar
      IO.pause   ( "Apertar ENTER para terminar." );
  } // fim main( )
```

Exercícios:
DICAS GERAIS: Consultar o Anexo Java 02 na apostila para outros exemplos.
Prever, realizar e registrar todos os testes efetuados.

01.) Fazer um programa (Exemplo0111) com método recursivo
para ler um valor inteiro do teclado e
mostrar essa quantidade em valores ímpares em ordem crescente começando em 3.

02.) Fazer um programa (Exemplo0112) com método recursivo
para ler um valor inteiro do teclado e
mostrar essa quantidade em múltiplos de 3 em ordem decrescente terminando em 3.

03.) Fazer um programa (Exemplo0113) com método recursivo
para ler um valor inteiro do teclado e
mostrar essa quantidade em valores da sequência: 1 3 6 9 12 15 ...

04.) Fazer um programa (Exemplo0114) com método recursivo
para ler um valor inteiro do teclado e
mostrar essa quantidade em valores decrescentes da sequência: ... 1/27 1/9 1/3 1.

05.) Fazer um programa (Exemplo0115) com método recursivo
para ler uma cadeia de caracteres e
para mostrar cada símbolo separadamente, um por linha.

06.) Fazer um programa (Exemplo0116) com função recursiva
para calcular a soma dos primeiros valores ímpares positivos começando em 3.
Testar essa função para quantidades diferentes.

07.) Fazer um programa (Exemplo0117) com função recursiva
para calcular a soma dos inversos dos primeiros valores ímpares positivos começando em 3.
Testar essa função para quantidades diferentes.

08.) Fazer um programa (Exemplo0118) com função recursiva
para calcular certo termo par da série de Fibonacci começando em 1.
Testar essa função para quantidades diferentes.

09.) Fazer um programa (Exemplo0119) com função recursiva
para calcular a quantidade de maiúsculas em uma cadeia de caracteres.
Testar essa função para quantidades diferentes.

10.) Fazer um programa (Exemplo0120) com função recursiva
para contar os dígitos em uma cadeia de caracteres.
Testar essa função para quantidades diferentes.

Tarefas extras

E1.) Fazer um programa com função recursiva
para calcular o valor da função definida abaixo,
lidos os valores de (x) e (n) do teclado:

$$f (x, n) = 1 + x^1 + x^2 + x^3 + x^4 + ... + x^n$$

E2.) Fazer um programa com função recursiva
para calcular o valor indicado abaixo,
lido o número de termos (n) do teclado:

$$e = 1 + 1/1 + 1/2 + 1/3 + 1/4 + ... + 1/(n-1)$$