# DOCUMENTAÇÃO DE ***SOFTWARE***

## 1. Introdução

O que é documentação de ***software***?

Qualquer ***software***, se encarado como a implementação de uma solução de problema em computador, pode ser comparado a um mapa. Entretanto, se a esse mapa faltarem legendas, marcas e orientações, ele será de pouco valor. Desse mesmo modo, pode-se imaginar um programa sem documentação.

Para quê é necessária?

- Para decidir  a compra
- Para instalar o produto
- Para treinar  o usuário
- Para permitir o uso eficiente
- Para permitir a alteração
- Para informar a correção de erros

Como deve ser a documentação de ***software***?

- Completa e pertinente ao serviço a que se destina
- Com o nível adequado de detalhes
- Apropriada ao tipo de usuário e inteligível
- Testada como o ***software***
- Bem apresentada
- Fácil de ser trabalhada

## 2. Documentação de programa

### O que é documentação de programa?

Dentre os diversos documentos que acompanham um determinado ***software***, a documentação de programa é aquele mais próximo da descrição da solução implementada, comentando-a e fornecendo informações técnicas que possibilitem a sua melhor compreensão.

### Para que serve a documentação de programa?

- Para responder às consultas sobre a performance operacional do ***software***
- Para a identificação e correção de erros
- Para auxiliar na alteração do ***software***, seja através do desenvolvimento de seus recursos, seja pelo acréscimo de novas funções
- Para manter sempre atualizada a descrição

### Quem deve fazer documentação de programa?

Cabe principalmente ao profissional responsável pela implementação determinar a forma, o conteúdo e a adequação do documento.

### Quando deve ser feita a documentação de programa?

- - Desde a fase de planejamento
- - Durante a fase de implementação
- - Durante a fase de desenvolvimento
- - Após qualquer revisão

3. O que deve conter a documentação de programa?

0. Apresentação

- capa
    - identificação do sistema
    - identificação do programa
    - data da última atualização
    - responsável pelo produto
- índice geral
- índice de figuras, diagramas, fluxogramas
- índice de anexos
- índice de listagens
- referências

1. Identificação

- nome do programa
- identificação do sistema
- identificação do programa
- data de implementação
- data da última revisão
- responsável pelo produto
- localização do produto

2. Ambiente

- equipamento de desenvolvimento
- sistema operacional (versão)
- compilador / interpretador (versão)
- forma de compilação e edição (montagem)
- equipamento de uso
- requisitos de operação

3. Conteúdo

- sumário dos objetivos
- descrição de ambiente
- estrutura e relacionamento dos módulos
- descrição de cada módulo
- lógica de cada módulo
  - método / algoritmo
  - diagramas / fluxogramas
  - formulário técnico
  - limitações / extensões
- descrição de dados e abstrações
  - dicionário de dados
    - nome, tipo, faixa de valores, valores iniciais e tolerâncias
    - verificações e observações
  - fluxo de dados
  - importações / exportações
  - opções
- listagens
  - numeração de linhas e páginas
  - destaque das palavras-chaves, dos títulos, dos módulos
- tratamento de erros
  - situações de erros previstas
  - classificação e forma de tratamento dos erros
  - mensagens de erros
  - explicações
- operação
  - carga
  - fornecimento de dados
  - resultados esperados
  - exemplos
- testes
  - dados
  - resultados
  - condições avaliadas
  - limitações
  - extensões possíveis

# 4. Recomendações para a documentação de programas

## 4.1 O uso de comentários

Comentários devem ser empregados no desenvolvimento de programas com o objetivo de ilustrar e explicar detalhes de seu funcionamento.

Os programas são as traduções de algoritmos expressos através de linguagens de programação. Sua legibilidade deve ser buscada sempre, como indicador de qualidade, e o uso de comentários pode ser útil nesse sentido.

Após as etapas de análise do problema e confecção de sua solução algorítmica, pode passar-se à escolha da linguagem de programação mais apropriada à natureza do problema e às exigências da solução elaborada. A tradução do algoritmo para a linguagem deve ser uma atividade realizada de maneira sistemática, cuidadosa, buscando preservar todas as informações contidas no algoritmo, inclusive os comentários acrescidos durante o desenvolvimento.

Além disso, os comentários devem servir para esclarecer detalhes específicos da implementação. É sugerido incluir:

- uma breve descrição dos objetivos do programa ou da abstração de comandos que estiver sendo desenvolvida;
- o autor e datas de escrita e de última atualização; e neste caso, quais as modificações realizadas;
- uma breve descrição de como utilizá-lo; no caso de abstrações, incluir também a descrição e utilidade de cada parâmetro;
- uma descrição sucinta de tipos, variáveis e constantes empregadas;
- uma descrição sucinta de arquivos e dispositivos de entrada e/ou saída necessários;
- uma breve descrição ou referência para métodos ou procedimentos especiais empregados;
- avaliação de custo de processamento (tempo e memória requeridos).

Dentre outras recomendações para a melhoria de legibilidade pode-se citar:

- a utilização de espaços em branco:
  - entre definições;
  - entre operadores, quando conveniente;
  - antes e depois de comentários ou de seções de código
  - para separar trechos que atendam a objetivos distintos;
  - para destacar a parte principal de um programa;
- a escolha de identificadores representativos para tipos, constantes, variáveis, funções, procedimentos e arquivos;
- o uso de um comando por linha;
- a utilização de parênteses para explicitar prioridades;
- o emprego de endentação (deslocamento para a direita) e identificação de blocos associados às estruturas ou às abstrações;
- a busca da clareza na escolha de representações de dados e estruturas de controle;
- o detalhamento de testes e verificações de condições especiais para utilização;
- o cuidado no emprego e na explicação de particularidades de uma linguagem ou de característica de um compilador ou interpretador.

É pensamento freqüente a crença de que a qualidade de um programa é avaliada apenas pelo seu funcionamento correto. A correção de um programa é uma atividade criteriosa que deve levar em conta vários fatores, inclusive a própria documentação. Um bom programa pode não estar funcionando bem, mas ter uma boa documentação que sugira suas fraquezas, onde encontrá-las e, talvez, como superá-las; um programa que funcione, mas não possa ser alterado ou expandido com facilidade, por falta de informações, é de pouca utilidade.

5. Exemplo

PUCMinas – Curso de Ciência da Computação TP _____

ALGORITMOS E ESTRUTURAS DE DADOS I DATA __/__/__

NOME DO PROGRAMA : _____
PROGRAMADOR :
_____

1. Identificação

Nome do Programa : Equacao.java

Objetivo:

Este programa tem por objetivo o calculo das raízes reais de uma equação de segundo grau, dados os coeficientes.

Os dados serão fornecidos em um arquivo em disco de nome DADOS.TXT.

2. Ambiente

Equipamento

Desenvolvimento : PC compatível
Uso : PC compatível

Sistema Operacional

Versão : qualquer
Compatibilidade : qualquer

Linguagem

Compilador : Sun Java Compiler
Versão : v.1.8.0_241

Forma de Edição : jGrasp v.2.6.02_01
Forma de Compilação : do próprio
Forma de Execução : Run in console window

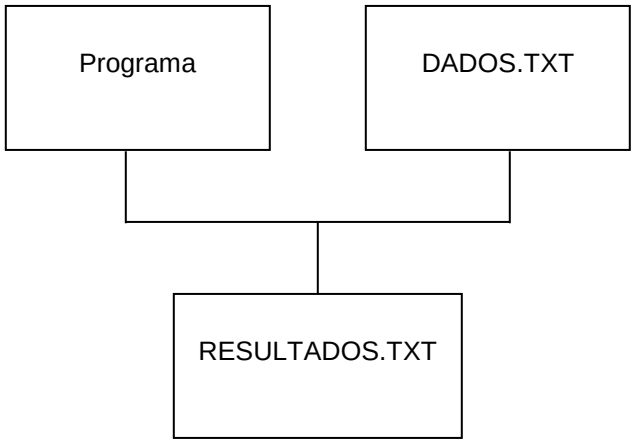
Exemplo alternativo:

C:\>javac Equação.java <ENTER>
C:\>java Equacao <ENTER>

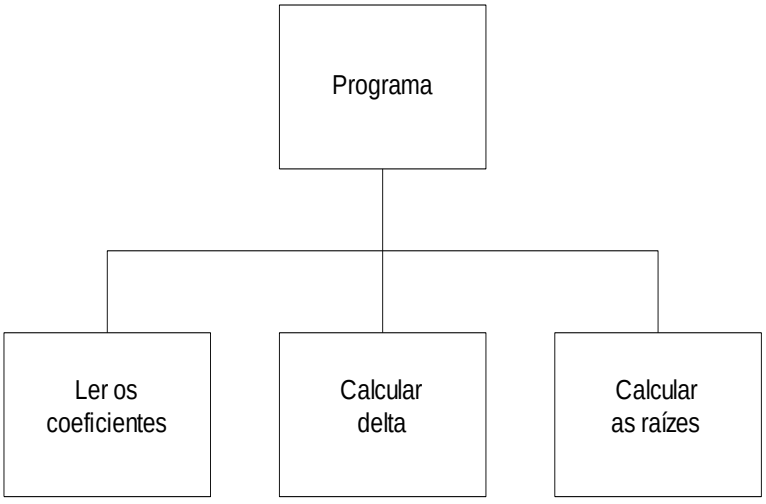
3. Conteúdo

3.1 Descrição sumária de funcionamento

3.1.1 Descrição de ambiente




3.1.2 Diagrama funcional



3.1.3 Descrição de funcionamento

Acionado o programa, ele procurará pelo arquivo DADOS.TXT, e lerá os coeficientes A, B, C. O valor de A será testado, para verificar se a equação é mesmo do segundo grau; se for, será calculado o delta. Se o valor de delta for positivo ou nulo, serão calculadas e exibidas as raízes. Os resultados também serão guardados em arquivo.

Exemplo:

C:\>Equacao
Calculo de raizes reais de equacao do 2º. grau :

Dados:

Equacao : 1 x ^ 2 + 4 x + 4 = 0

Resultados:

X1 = -2.000
X2 = -2.000

Apertar <ENTER> para continuar.
C:\>

3.2 Descrição de dados

| Nome | Tipo | Valor Inicial | Limites | Observações |
|---|---|---|---|---|
| dados | Coeficientes | -- | -- | dados do arquivo |
| a | inteiro | -- | -- | primeiro coeficiente |
| b | inteiro | -- | -- | segundo coeficiente |
| c | inteiro | -- | -- | terceiro coeficiente |
| delta | real | -- | -- | valor de delta |
| x | Raízes | -- | -- | resultados |

3.3 Descrição de métodos

3.3.1 Descrição de funções

<table>
<tr><td colspan="4">Função</td></tr>
<tr><td colspan="4">Nome: calcular_Raiz</td></tr>
<tr><td colspan="4">Objetivo:<br>Calcular uma raiz de equação de segundo grau,<br>dados os valores dos coeficientes (a), (b) e (c),<br>o sinal da raiz de delta, e o valor de delta</td></tr>
<tr><td>Retorno</td><td>Tipo</td><td colspan="2">Observações</td></tr>
<tr><td>raiz</td><td>real</td><td colspan="2">valor calculado de uma raiz da equação</td></tr>
<tr><td colspan="4">Parâmetros</td></tr>
<tr><td>Nome</td><td>Tipo</td><td>Passagem</td><td>Observações</td></tr>
<tr><td>a</td><td>inteiro</td><td>valor</td><td>primeiro coeficiente</td></tr>
<tr><td>b</td><td>inteiro</td><td>valor</td><td>segundo coeficiente</td></tr>
<tr><td>c</td><td>inteiro</td><td>valor</td><td>terceiro coeficiente</td></tr>
<tr><td>sinal</td><td>inteiro</td><td>valor</td><td>sinal da raiz de delta</td></tr>
<tr><td>delta</td><td>real</td><td>valor</td><td>valor calculado de delta</td></tr>
<tr><td colspan="4">Testes</td></tr>
<tr><td colspan="4">Para os valores:<br>a = 1, b = 4, c =4, sinal = +1, delta = 0.0<br>Resultado:<br>raiz = -2.00</td></tr>
<tr><td colspan="4">Limites / Extensões / Observações</td></tr>
<tr><td colspan="4">- Somente calcula uma raiz;<br>- Poderia calcular o sinal.</td></tr>
</table>

3.3.2 Descrição de procedimentos

<table>
<tr><td colspan="4">Procedimento</td></tr>
<tr><td colspan="4">Nome: Calcular_Delta</td></tr>
<tr><td colspan="4">Objetivo:<br>Calcular o valor de delta por meio da expressão:<br>$delta = b^2 - 4 \cdot a \cdot c$</td></tr>
<tr><td colspan="4">Parâmetros</td></tr>
<tr><td>Nome</td><td>Tipo</td><td>Passagem</td><td>Observações</td></tr>
<tr><td>a</td><td>inteiro</td><td>valor</td><td>primeiro coeficiente</td></tr>
<tr><td>b</td><td>inteiro</td><td>valor</td><td>segundo coeficiente</td></tr>
<tr><td>c</td><td>inteiro</td><td>valor</td><td>terceiro coeficiente</td></tr>
<tr><td>delta</td><td>real</td><td>referência</td><td>valor calculado de delta</td></tr>
<tr><td colspan="4">Testes</td></tr>
<tr><td colspan="4">Para os valores:<br>a = 1, b = 4, c = 4<br>Resultado:<br>Delta = 0.0</td></tr>
<tr><td colspan="4">Limites / Extensões / Observações</td></tr>
<tr><td colspan="4"></td></tr>
</table>

Procedimento

Nome: ler_Coeficientes

Objetivo:
Ler um arquivo (do tipo texto), contendo os três coeficientes (a), (b) e (c), um por linha

Parâmetros

| Nome | Tipo | Passagem | Observações |
|---|---|---|---|
| nome | cadeia | valor | nome do arquivo |
| dados | real(3) | referência | conjunto de dados |
| | | | |
| | | | |

Testes

Fornecido um arquivo com o conteúdo:
1
4
4
os valores foram convenientemente lidos.

Limites / Extensões / Observações

3.4 Descrição de arquivo

| Arquivo | |
|---|---|
| Nome interno: arquivo | |
| Nome externo:<br>DADOS.TXT | Tamanho: 08 bytes |
| Organização | Acesso |
| texto | seqüencial |
| Registro | |
| linhas de texto, com um dado em cada | |
| Objetivo:<br>Armazenar os coeficientes de uma equação do segundo grau. | |

| Arquivo | |
|---|---|
| Nome interno: Arquivo | |
| Nome externo:<br>RESULTADOS.TXT | Tamanho: 08 bytes |
| Organização | Acesso |
| texto | seqüencial |
| Registro | |
| linhas de texto, com dados e resultados, um em cada linha | |
| Objetivo:<br>Armazenar os coeficientes de uma equação do segundo grau,<br>e seus resultados.. | |

4. Previsão de testes

4.1 Execução normal:

| Dados: | Resultados esperados: |
|---|---|
| a = 1 | X1 = -2 |
| b = 4 | X2 = -2 |
| c = 4 | |

4.2 Teste do tipo de equação :

| Dados: | Resultados esperados: |
|---|---|
| a = 0 | A equacao nao e' do segundo grau. |
| b = 4 | |
| c = 4 | |

4.3 Teste do valor de delta:

| Dados: | Resultados esperados: |
|---|---|
| a = 1 | Delta menor que zero, raizes complexas. |
| b = 4 | |
| c = 5 | |

5. Execução de testes

5.1 Execução normal:

| Dados | Resultados esperados | Resultados obtidos |
|---|---|---|
| a = 1<br>b = 4<br>c = 4 | x1 = -2<br>x2 = -2 | x1 = - 2.0<br>x2 = - 2.0 |

5.2 Teste do tipo de equação:

| Dados | Resultados esperados | = | Resultados obtidos |
|---|---|---|---|
| a = 0<br>b = 4<br>c = 4 | A equacao nao e’ do segundo grau. | | |

5.3 Teste do valor de delta:

| Dados | Resultados esperados | = | Resultados obtidos |
|---|---|---|---|
| a = 1<br>b = 4<br>c = 5 | Delta menor que zero, raizes complexas. | | |

6. Sugestão de documentação de programa em Java:

```
/**
 * ---------------------------------------------- identificacao
 * Pontificia Universidade Catolica de Minas Gerais
 * Curso de Ciencia da Computacao
 * Algoritmos e Estruturas de Dados 1
 *
 * Trabalho pratico: 00 ( Exemplo de documentacao )
 *
 * Objetivo:
 * Modelo de programa para a disciplina AED I
 * Sugestao para montar um trabalho pratico
 *
 * Programa para resolver uma equcao do segundo grau:
 * ax ^ 2 + bx + c = 0,
 * lidos os valores reais de (a), (b) e (c)
 *
 * Source files: {@files}
 * @author   Matricula: 999999  Nome: xxx yyy zzz
 * @version  1.0                Data: 99/99/9999
 *
 * Dados:
 * - opcao de execucao escolhida pelo usuario
 *
 * Resultados:
 * - execucao de um teste de cada vez a escolha do usuario
 *
 * Condicoes:
 * - so' aceita as opcoes numericas oferecidas.
 *
 * Forma de compilacao:
 * - acionar o compilador no modo console:
 *
 *   <drive>:>javac   Equacao.java
 *   ou mediante uso de um lançador de programas (do sistema operacional)
 *   <drive>:>compile Equacao
 *
 *   OBS:
 *   A variavel CLASSPATH deve ter sido definida para os
 *   diretorios adequados (ex: C:\java\aed1); ou ter sua
 *   definicao equivalente no arquivo classpath.bat.
 *
 * Forma de uso:
 * - acionar o programa no modo console:
 *
 *   <drive>:>java Equacao
 *   ou mediante uso de um lançador de programas (do sistema operacional)
 *   <drive>:>run  Equacao
 *
 * - escolher uma das opcoes oferecidas.
 *
 * Referencias:
 * - Exemplos mostrados em sala de aula
 * - Apostila: Fundamentos de Programacao
 **/
```

```
// ---------------------------------------------- classes nativas

import IO.*;

// ---------------------------------------------- definicao de classe

/**
  * classe Equacao - para raizes de equacao do segundo grau.
  * <br>
  * @param mensagem  - identificacao do erro
  * <br>
  */

class Equacao
{
// ---------------------------------------------- conversao para String

   /**
     * toString - metodo converter dados para caracteres.
     * <br>
     */

   public String toString ( )
   {
     return ( "Equacao" );
   } // end Equacao.toString ( )

// ---------------------------------------------- tratamento de erros

   /**
     * tratar_Erro - metodo para tratar erro.
     * <br>
     * @param mensagem  - identificacao do erro
     * <br>
     */
   private static void tratar_Erro
                    ( String mensagem )
   {
     IO.println ( );
     IO.println ( mensagem );
     IO.println ( );
     IO.pause ( "Apertar <ENTER> para continuar." );
     IO.println ( );
   } // fim Equacao.tratar_Erro ( )
```

```
// ---------------------------------------------- definicoes globais

// ---------------------------------------------- definicao de constantes

// ---------------------------------------------- definicao de armazenadores

// ---------------------------------------------- definicao de construtor(es)

// ---------------------------------------------- definicao de metodos restritos

  /**
   * identificar - metodo para mostrar dados do programa e do autor.
   * <br><pre>
   * @param tp          - identificacao do programa
   * @param versao    - versao do programa
   * @param data        - data da liberacao
   * @param titulo      - titulo do programa
   * @param matricula - numero de matricula
   * @param nome      - nome do autor
   * <br></pre>
   */
  private static void identificar
                        ( String tp, String versao, String data,
                          String titulo,
                          String matricula, String nome )
  {
    IO.clrscr  ( );
    IO.println ( tp  + "\t" + versao + "\t" + data );
    IO.println ( );
    IO.println ( titulo );
    IO.println ( );
    IO.print   ( "Matricula:" + matricula + "\t" + "Nome:" + nome );
    IO.println ( );
  } // fim Equacao.identificar ( )
```

```
// ---------------------------------------------- definicao de metodos publicos

  /**
   * ler_Coeficientes - metodo para ler dados de arquivo.
   * <br>
   * @param nome  - nome do arquivo
   * @param dados - armazenador dos coeficientes
   * <br>
   * OBS.: Supor a existencia de tres valores em arquivo, um em cada linha.
   */
  public static void ler_Coeficientes
                          ( String nome, double [ ] dados )
  {
  // definicao de dados locais
     FILE arquivo = new FILE ( FILE.INPUT, nome );
     String linha;

  // ler dado
     linha = arquivo.readln ( );
     dados [ 0 ] = IO.getdouble ( linha );
     linha = arquivo.readln ( );
     dados [ 1 ] = IO.getdouble ( linha );
     linha = arquivo.readln ( );
     dados [ 2 ] = IO.getdouble ( linha );
  } // fim Equacao.ler_Coeficientes ( )

  /**
   * calcular_Delta - metodo para calcular o valor de delta,
   *                  conhecidos os coeficientes da equacao.
   * <br>
   * @return valor do delta da equacao
   * <br>
   * @param a - coeficiente do termo de grau 2
   * @param b - coeficiente do termo de grau 1
   * @param c - coeficiente do termo de grau 0
   * <br>
   * OBS.: Supor que esta ordem tambem esta' refletida no arquivo.
   */
  public static double calcular_Delta
                          ( double a, double b, double c )
  {
  // definicao de dado local
     double resposta = 0.0;
  // calcular a resposta
     resposta = b * b - 4.0 * a * c;
  // retornar resposta
     return ( resposta );
  } // fim Equacao.calcular_Delta ( )
```

```
  /**
   * calcular_Raiz - metodo para calcular uma raiz real
   *                 da equacao.
   * <br>
   * @return valor de uma raiz real da equacao
   * <br>
   * @param a      - coeficiente do termo de grau 2
   * @param b      - coeficiente do termo de grau 1
   * @param delta - valor calculado do delta
   * @param sinal - valor calculado do delta
   * <br>
   */
  public static double calcular_Raiz
                        ( double a, double b, double sinal, double delta  )
  {
  // definicao de dado local
     double resposta = 0.0;
  // calcular a resposta
     resposta = ( -b + sinal * Math.sqrt ( delta ) ) / ( 2.0 * a );
  // retornar resposta
     return ( resposta );
  } // fim Equacao.calcular_Delta ( )

// ---------------------------------------------- definicao de metodos

  /**
   * parte principal - calcular raizes reais de
   *                   uma equacao do segundo grau
   *                   com dados lidos de arquivo.
   * <br>
   * <pre>
   *------------------------------------------- Testes

     Versao  Teste
     0.0     01. ( OK  ) identificacao de programa
             02. ( OK  ) definicao de variaveis
             03. ( OK  ) teste do primeiro arquivo
                         equacao: 1 x^2 + 4 x + 4
                         previsto: ( x-2 ) ( x-2 )
             04. ( OK  ) teste do segundo  arquivo
                         equacao: 0 x^2 + 4 x + 4
                         previsto: "A equacao nao e' do segundo grau."
             05. ( OK  ) teste do terceiro arquivo
                         equacao: 1 x^2 + 4 x + 5
                         previsto: "Delta nulo: raizes complexas."
   * </pre>
   */
```

```
public static void main ( String [ ] args )
{
 //   identificar
     identificar ( "Equacao", "v.1.0", "99/99/9999",
                   "Exemplo de documentacao",
                   "999999", "xxx yyy zzz" );

 //   definir dados
     double [ ] dados              // em arquivo
     = new double [ 3 ];
     double a, b, c,               // coeficientes da equacao
            delta,                 // delta = b^2 - 4*a*c
            x1, x2;                // raizes reais da equacao
     FILE   arquivo                // para os resultados
     = new FILE ( FILE.OUTPUT, "RESULTADOS.TXT" );

 //   processar dados

     // ler dados
        ler_Coeficientes ( "DADOS.TXT", dados );
        a = dados [ 0 ];
        b = dados [ 1 ];
        c = dados [ 2 ];

     // mostrar dados
        IO.println ( );
        IO.println ( "Equacao: " + a + " x ^2 + " + b + " x + " + c + " = 0" );
     // gravar o erro em arquivo, tambem
        arquivo.println ( "Arquivo gerado pelo programa Equacao.java" );
        arquivo.println ( "Equacao: " + a + " x ^2 + " + b + " x + " + c + " = 0" );
        IO.println ( );

     // verificar o grau da equacao
        if ( a == 0.0 )
        {
         // erro: nao e' equacao do segundo grau
            tratar_Erro ( "A equacao nao e' do segundo grau." );
         // gravar o erro em arquivo, tambem
            arquivo.println ( "A equacao nao e' do segundo grau." );
        }

        {
         // calcular delta
            delta = calcular_Delta ( a, b, c );
         // testar delta
            if ( delta < 0 )
            {
             // nao ha' raizes reais
                tratar_Erro ( "Delta nulo: raizes complexas." );
             // gravar o erro em arquivo, tambem
                arquivo.println ( "Delta nulo: raizes complexas." );
            }
```

```
            else // ( ha' raizes reais )
            {
             // calcular raizes
                 x1 = calcular_Raiz ( a, b, +1, delta );
                 x2 = calcular_Raiz ( a, b, -1 , delta );
             // mostrar raizes
                IO.println ( "x1 = " + x1 );
                IO.println ( "x2 = " + x2 );
             // gravar as raizes em arquivo, tambem
                arquivo.println ( "x1 = " + x1 );
                arquivo.println ( "x2 = " + x2 );
            } // fim se ( delta < 0 )
          } // fim se (equacao do segundo grau)

     //   encerrar a execucao
         arquivo.close ( );      // fechar arquivo

         IO.pause ( "Apertar <ENTER> para terminar." );

     } // fim da parte principal - main ( )

} // fim da classe Equacao

// ------------------------------------------------ Documentacao complementar
/*
   ------------------------------------------------ Manual

   Formas de uso:
   a.) Pela console/terminal:
      - ir ate' a pasta que contem programa
      - executa-lo acionando-o pelo nome, sem parametros:

        java Equacao

      OBS:
        Os arquivos de dados devem estar presentes na pasta.

   Descricao de funcionamento:
   O programa buscara' os dados automaticamente no arquivo de dados.
   Os resultados serao apresentados em seguida.

   Observacao:
   Para mudar os dados, o nome do arquivo devera' ser editado,
   e o programa recompilado.
   Para evitar isso, basta copiar e renomear o arquivo de dados
   de forma conveniente.
```

```
  ------------------------------------------------ Historico

  Versao        Data    Modificacao
  0.0           01/02   esboco

  ------------------------------------------------ Testes

  Versao        Teste
  0.0            01. ( OK  ) identificacao de programa
                 02. ( OK  ) definicao de variaveis
                 03. ( OK  ) teste do primeiro arquivo
                             equacao: 1 x^2 + 4 x + 4
                             previsto: ( x-2 ) ( x-2 )
                 04. ( OK  ) teste do segundo  arquivo
                             equacao: 0 x^2 + 4 x + 4
                             previsto: "A equacao nao e' do segundo grau."
                 05. ( OK  ) teste do terceiro arquivo
                             equacao: 1 x^2 + 4 x + 5
                             previsto: "Delta nulo: raizes complexas."

 ------------------------------------------------ Desempenho

  Tipo da funcao de custo do numero de operacoes: linear

  ------------------------------------------------
*/
```