Verificarea şi Testarea Sistemelor de Calcul

Temă de casă

November 5, 2018

Titlu: *Ecuaţia de gradul II*

Profesor: Ş.l. Dr. Ing. Nicolae Enescu

Student: Voiculescu Ioan-Valentin

Facultate: Automatică, Calculatoare şi Electronică

Anul: IV

Specializarea: Calculatoare Română

Grupa: CR 4.H1 A

Contents

1 Codul Sursă[1]

1.1 Implementarea

1.1.1 Program.cs

```
1   using   System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Lab1
8  {
9      class Program
10     {
11         static void Main(string[] args)
12         {
13             try
14             {
15                 InputProcessing inputProcessing = new
                       InputProcessing(new Input(), new
                       Output());
16                 OutputProcessing outputProcessing =
                       new OutputProcessing(new Output());
17                 QuadraticEquation quadraticEquation =
                       new QuadraticEquation(
                       inputProcessing.GetData());
18                 outputProcessing.PutData(
                       quadraticEquation.
                       SolveWithRealSolutions());
19             }
20             catch (Exception e)
21             {
22                 Console.WriteLine(e.Message);
23             }
24         }
25     }
26 }
```

1.1.2 QuadraticEquation.cs

```
1   using   System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Lab1
8  {
9      public class QuadraticEquation
10     {
11         //ax^2 + bx + c = 0
12         public Coefficients Coefficients { private set
               ; get; }
13         public double Delta
14         {
15             get
16             {
17                 return (double)(Math.Pow(Coefficients.
                       B, 2) - 4 * Coefficients.A *
                       Coefficients.C);
18             }
19         }
20         public QuadraticEquation (Coefficients
               coefficients)
21         {
22             if(coefficients != null)
23             {
24                 Coefficients = coefficients;
25             }
26             else
27             {
28                 throw new Exception("'coefficients'
                       cannot be null");
29             }
30         }
31         public Solution SolveWithRealSolutions()
32         {
33             Solution solution = null;
34             if (Delta.Equals(0))
35             {
36                 solution = new Solution(
37                     (double)(-Coefficients.B) / (
                           double)(2 * Coefficients.A),
38                     (double)(-Coefficients.B) / (
```

```
                        double)(2 * Coefficients.A)
39                      );
40              }
41              if(Delta > 0)
42              {
43                  solution = new Solution(
44                      (double)(-Coefficients.B + Math.
                            Sqrt(Delta)) / (double)(2 *
                            Coefficients.A),
45                      (double)(-Coefficients.B - Math.
                            Sqrt(Delta)) / (double)(2 *
                            Coefficients.A)
46                      );
47              }
48              if (Delta < 0)
49              {
50                  throw new Exception("the equation do
                        not have a real solution");
51              }
52              return solution;
53          }
54      }
55  }
```

1.1.3 Coefficients.cs

```
1   using   System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Lab1
8  {
9      public class Coefficients
10     {
11         public int A { private set; get; }
12         public int B { private set; get; }
13         public int C { private set; get; }
14         public Coefficients(int coefficientA, int
               coefficientB, int coefficientC)
15         {
16             if (coefficientA == 0)
17             {
18                 throw new Exception("The 'a'
                       coefficient cannot equal 0");
19             }
20             A = coefficientA;
21             B = coefficientB;
22             C = coefficientC;
23         }
24     }
25 }
```

1.1.4 Solution.cs

```
  using   System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab1
{
    public class Solution
    {
        public double Root1 { private set; get; }
        public double Root2 { private set; get; }
        public Solution(double root1, double root2)
        {
            Root1 = root1;
            Root2 = root2;
        }
    }
}
```

1.1.5 DecimalRound.cs

```
1  using System;
2
3  namespace Lab1
4  {
5      public static class Decimals
6      {
7          public static string[] Round(Solution solution
               , int numberDecimals)
8          {
9              string[] roots;
10             roots = new string[2]
11             {
12                 (Math.Truncate(Math.Pow(10,
                       numberDecimals) * solution.Root1) /
                        (double) Math.Pow(10,
                       numberDecimals)).ToString(),
13                 (Math.Truncate(Math.Pow(10,
                       numberDecimals) * solution.Root2) /
                        (double) Math.Pow(10,
                       numberDecimals)).ToString(),
14             };
15             return roots;
16         }
17     }
18 }
```

1.1.6 Constants.cs

```
  using   System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab1
{
    public static class Constants
    {
        public const int CoefficientAMinimumValue =
            -30;
        public const int CoefficientAMaximumValue =
            70;

        public const int CoefficientBMinimumValue =
            -50;
        public const int CoefficientBMaximumValue =
            10;

        public const int CoefficientCMinimumValue = 0;
        public const int CoefficientCMaximumValue =
            200;
    }
}
```

1.1.7 InputProcessing.cs

```
1   using   System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Linq.Expressions;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace Lab1
9  {
10     public class InputProcessing
11     {
12         private Input _input;
13         private Output _output;
14         public InputProcessing(Input input, Output
               output)
15         {
16             _input = input;
17             _output = output;
18         }
19         public Coefficients GetData()
20         {
21             Coefficients coefficients = new
                   Coefficients(
22                 SetCoefficient('a', Constants.
                       CoefficientAMinimumValue, Constants
                       .CoefficientAMaximumValue),
23                 SetCoefficient('b', Constants.
                       CoefficientBMinimumValue, Constants
                       .CoefficientBMaximumValue),
24                 SetCoefficient('c', Constants.
                       CoefficientCMinimumValue, Constants
                       .CoefficientCMaximumValue)
25                 );
26             return coefficients;
27         }
28         private int SetCoefficient(char c, int
               minimumValue, int maximumValue)
29         {
30             int coefficient;
31             if (_output != null)
32             {
33                 string line = c + "=";
34                 _output.WriteData(new string[1] {line
                       });
```

```
35              }
36              try
37              {
38                  coefficient = Int32.Parse(_input.
                        ReadDataByLine());
39              }
40              catch (FormatException)
41              {
42                  throw new Exception(c+" is not integer
                        ");
43              }
44              if(!(coefficient>=minimumValue &&
                    coefficient<=maximumValue))
45              {
46                  throw new Exception(c + " out of range
                        ");
47              }
48              return coefficient;
49          }
50      }
51  }
```

1.1.8 OutputProcessing.cs

```
  using   System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab1
{
    public class OutputProcessing
    {
        private Output _output;
        public OutputProcessing(Output output)
        {
            _output = output;
        }
        public void PutData(Solution solution)
        {
            string line1, line2;
            string[] newSolution = Decimals.Round(
               solution, 2);
            line1 = "x1=" + newSolution[0];
            line2 = "x2=" + newSolution[1];
            _output.WriteData(new string[2] {line1,
               line2});
        }
    }
}
```

1.1.9 Input.cs

```
using System;

namespace Lab1
{
    public class Input
    {
        public virtual string ReadDataByLine()
        {
            return Console.ReadLine();
        }
    }
}
```

1.1.10 Output.cs

```
using System;

namespace Lab1
{
    public class Output
    {
        public virtual void WriteData(string[] lines)
        {
            foreach (var line in lines)
            {
                Console.WriteLine(line);
            }
        }
    }
}
```

1.2 Testarea

1.2.1 Tests.cs

```
1   using System;
2  using NUnit.Framework;
3  using Lab1;
4
5  namespace Lab1Tests
6  {
7      [TestFixture]
8      public class Tests
9      {
10         private string[] testDates;
11         private InputProcessing inputProcessing;
12         private QuadraticEquation quadraticEquation;
13         private string[] testSolution;
14
15         [Test]
16         public void Test1()
17         {
18             testDates = new string[3]
19             {
20                 "5", "-2", "10"
21             };
22             inputProcessing = new InputProcessing(new
                   InputTest(testDates), null);
23             quadraticEquation = new QuadraticEquation(
                   inputProcessing.GetData());
24             Assert.That(() => quadraticEquation.
                   SolveWithRealSolutions(), Throws.
                   Exception.TypeOf<Exception>().With.
                   Message.EqualTo
25                  ("the equation do not have a real
                       solution")
26             );
27         }
28
29         [Test]
30         public void Test2()
31         {
32             testDates = new string[3]
33             {
34                 "-30", "-50", "0"
35             };
36             inputProcessing = new InputProcessing(new
```

```
            InputTest(testDates), null);
37          quadraticEquation = new QuadraticEquation(
                inputProcessing.GetData());
38          testSolution = new string[2]
39          {
40              "-1.66", "0"
41          };
42          Assert.AreEqual(Decimals.Round(
                quadraticEquation.
                SolveWithRealSolutions(),2),
                testSolution);
43      }
44
45      [Test]
46      public void Test3()
47      {
48          testDates = new string[2]
49          {
50              "-30", "11"
51          };
52          inputProcessing = new InputProcessing(new
                InputTest(testDates), null);
53          Assert.That(() => quadraticEquation = new
                QuadraticEquation(inputProcessing.
                GetData()), Throws.Exception.TypeOf<
                Exception>().With.Message.EqualTo
                 ("b out of range")
54          );
55      }
56
57      [Test]
58      public void Test4()
59      {
60          testDates = new string[3]
61          {
62              "20", "0", "-1"
63          };
64          inputProcessing = new InputProcessing(new
                InputTest(testDates), null);
65          Assert.That(() => quadraticEquation = new
                QuadraticEquation(inputProcessing.
                GetData()), Throws.Exception.TypeOf<
                Exception>().With.Message.EqualTo
                 ("c out of range")
66          );
67      }
68
69  }
```

```
70
71          [Test]
72          public void Test5()
73          {
74              testDates = new string[1]
75              {
76                  "-1.00"
77              };
78              inputProcessing = new InputProcessing(new
                    InputTest(testDates), null);
79              Assert.That(() => quadraticEquation = new
                    QuadraticEquation(inputProcessing.
                    GetData()), Throws.Exception.TypeOf<
                    Exception>().With.Message.EqualTo
80                   ("a is not integer")
81              );
82          }
83
84          [Test]
85          public void Test6()
86          {
87              testDates = new string[2]
88              {
89                  "20", "-11.65"
90              };
91              inputProcessing = new InputProcessing(new
                    InputTest(testDates), null);
92              Assert.That(() => quadraticEquation = new
                    QuadraticEquation(inputProcessing.
                    GetData()), Throws.Exception.TypeOf<
                    Exception>().With.Message.EqualTo
93                   ("b is not integer")
94              );
95          }
96
97          [Test]
98          public void Test7()
99          {
100             testDates = new string[3]
101             {
102                 "14", "6", "gogu"
103             };
104             inputProcessing = new InputProcessing(new
                    InputTest(testDates), null);
105             Assert.That(() => quadraticEquation = new
                    QuadraticEquation(inputProcessing.
```

```
            GetData()), Throws.Exception.TypeOf<
            Exception>().With.Message.EqualTo
106          ("c is not integer")
107      );
108  }
109
110  [Test]
111  public void Test8()
112  {
113      testDates = new string[1]
114      {
115          "Y"
116      };
117      inputProcessing = new InputProcessing(new
            InputTest(testDates), null);
118      Assert.That(() => quadraticEquation = new
            QuadraticEquation(inputProcessing.
            GetData()), Throws.Exception.TypeOf<
            Exception>().With.Message.EqualTo
119          ("a is not integer")
120      );
121  }
122
123  [Test]
124  public void Test9()
125  {
126      testDates = new string[2]
127      {
128          "22", "453g"
129      };
130      inputProcessing = new InputProcessing(new
            InputTest(testDates), null);
131      Assert.That(() => quadraticEquation = new
            QuadraticEquation(inputProcessing.
            GetData()), Throws.Exception.TypeOf<
            Exception>().With.Message.EqualTo
132          ("b is not integer")
133      );
134  }
135
136  [Test]
137  public void Test10()
138  {
139      testDates = new string[3]
140      {
141          "3", "0", "31s"
```

```
142             };
143             inputProcessing = new InputProcessing(new
                    InputTest(testDates), null);
144             Assert.That(() => quadraticEquation = new
                    QuadraticEquation(inputProcessing.
                    GetData()), Throws.Exception.TypeOf<
                    Exception>().With.Message.EqualTo
145                     ("c is not integer")
146             );
147         }
148
149         [Test]
150         public void Test11()
151         {
152             testDates = new string[3]
153             {
154                 "0", "3", "4"
155             };
156             inputProcessing = new InputProcessing(new
                    InputTest(testDates), null);
157             Assert.That(() => quadraticEquation = new
                    QuadraticEquation(inputProcessing.
                    GetData()), Throws.Exception.TypeOf<
                    Exception>().With.Message.EqualTo
158                     ("The 'a' coefficient cannot equal 0")
159             );
160         }
161
162         [Test]
163         public void Test12()
164         {
165             testDates = new string[3]
166             {
167                 "1", "0", "10"
168             };
169             inputProcessing = new InputProcessing(new
                    InputTest(testDates), null);
170             quadraticEquation = new QuadraticEquation(
                    inputProcessing.GetData());
171             Assert.That(() => quadraticEquation.
                    SolveWithRealSolutions(), Throws.
                    Exception.TypeOf<Exception>().With.
                    Message.EqualTo
172                     ("the equation do not have a real
                            solution")
173             );
```

```
174         }
175
176         [Test]
177         public void Test13()
178         {
179             testDates = new string[3]
180             {
181                 "-1", "6", "18"
182             };
183             inputProcessing = new InputProcessing(new
                    InputTest(testDates), null);
184             quadraticEquation = new QuadraticEquation(
                    inputProcessing.GetData());
185             testSolution = new string[2]
186             {
187                 "-2.19", "8.19"
188             };
189             Assert.AreEqual(Decimals.Round(
                    quadraticEquation.
                    SolveWithRealSolutions(),2),
                    testSolution);
190         }
191
192         [Test]
193         public void Test14()
194         {
195             testDates = new string[3]
196             {
197                 "11", "-27", "0"
198             };
199             inputProcessing = new InputProcessing(new
                    InputTest(testDates), null);
200             quadraticEquation = new QuadraticEquation(
                    inputProcessing.GetData());
201             testSolution = new string[2]
202             {
203                 "2.45", "0"
204             };
205             Assert.AreEqual(Decimals.Round(
                    quadraticEquation.
                    SolveWithRealSolutions(),2),
                    testSolution);
206         }
207
208         [Test]
209         public void Test15()
```

```
210         {
211             testDates = new string[3]
212             {
213                 "-12", "10", "0"
214             };
215             inputProcessing = new InputProcessing(new
                    InputTest(testDates), null);
216             quadraticEquation = new QuadraticEquation(
                    inputProcessing.GetData());
217             testSolution = new string[2]
218             {
219                 "0", "0.83"
220             };
221             Assert.AreEqual(Decimals.Round(
                    quadraticEquation.
                    SolveWithRealSolutions(),2),
                    testSolution);
222         }
223
224         [Test]
225         public void Test16()
226         {
227             testDates = new string[3]
228             {
229                 "2", "-14", "24"
230             };
231             inputProcessing = new InputProcessing(new
                    InputTest(testDates), null);
232             quadraticEquation = new QuadraticEquation(
                    inputProcessing.GetData());
233             testSolution = new string[2]
234             {
235                 "4", "3"
236             };
237             Assert.AreEqual(Decimals.Round(
                    quadraticEquation.
                    SolveWithRealSolutions(),2),
                    testSolution);
238         }
239
240         [Test]
241         public void Test17()
242         {
243             testDates = new string[1]
244             {
245                 ""
```

```
246             };
247             inputProcessing = new InputProcessing(new
                    InputTest(testDates), null);
248             Assert.That(() => quadraticEquation = new
                    QuadraticEquation(inputProcessing.
                    GetData()), Throws.Exception.TypeOf<
                    Exception>().With.Message.EqualTo
249                     ("a is not integer")
250             );
251         }
252 
253         [Test]
254         public void Test18()
255         {
256             testDates = new string[3]
257             {
258                 "1", "4", "4"
259             };
260             inputProcessing = new InputProcessing(new
                    InputTest(testDates), null);
261             quadraticEquation = new QuadraticEquation(
                    inputProcessing.GetData());
262             testSolution = new string[2]
263             {
264                 "-2", "-2"
265             };
266             Assert.AreEqual(Decimals.Round(
                    quadraticEquation.
                    SolveWithRealSolutions(),2),
                    testSolution);
267         }
268 
269         [Test]
270         public void Test19()
271         {
272             testDates = new string[3]
273             {
274                 "70", "-50", "0"
275             };
276             inputProcessing = new InputProcessing(new
                    InputTest(testDates), null);
277             quadraticEquation = new QuadraticEquation(
                    inputProcessing.GetData());
278             testSolution = new string[2]
279             {
280                 "0.71", "0"
```

```
281                 };
282                 Assert.AreEqual(Decimals.Round(
                        quadraticEquation.
                        SolveWithRealSolutions(),2),
                        testSolution);
283             }
284
285             [Test]
286             public void Test20()
287             {
288                 testDates = new string[3]
289                 {
290                     "8", "-30", "7"
291                 };
292                 inputProcessing = new InputProcessing(new
                        InputTest(testDates), null);
293                 quadraticEquation = new QuadraticEquation(
                        inputProcessing.GetData());
294                 testSolution = new string[2]
295                 {
296                     "3.5", "0.25"
297                 };
298                 Assert.AreEqual(Decimals.Round(
                        quadraticEquation.
                        SolveWithRealSolutions(),2),
                        testSolution);
299             }
300         }
301     }
```

1.2.2 InputTest.cs

```
using System;
using Lab1;

namespace Lab1Tests
{
    public class InputTest : Input
    {
        private string[] _inputDates;
        private int _index;

        public InputTest(string[] inputDates)
        {
            _inputDates = inputDates;
            _index = 0;
        }
        public override string ReadDataByLine()
        {
            return _inputDates[_index++];
        }
    }
}
```

2 Tabelul de test

Id	Val 'a'	Val 'b'	Val 'c'	Rezultatul asteptat	Rezultatul obtinut	Valid
1	5	-2	10	the equation do not have a real solution	the equation do not have a real solution	V
2	-30	-50	0	x1=-1.66 x2=0	x1=-1.66 x2=0	V
3	-30	11		b out of range	b out of range	V
4	20	0	-1	c out of range	c out of range	V
5	-1.00			a is not integer	a is not integer	V
6	20	-11.65		b is not integer	b is not integer	V
7	14	6	gogu	c is not integer	c is not integer	V
8	Y			a is not integer	a is not integer	V
9	22	453g		b is not integer	b is not integer	V
10	3	0	31s	c is not integer	c is not integer	V
11	0	3	4	The 'a' coefficient cannot equal 0	The 'a' coefficient cannot equal 0	V
12	1	0	10	the equation do not have a real solution	the equation do not have a real solution	V
13	-1	6	18	x1=-2.19 x2=8.19	x1=-2.19 x2=8.19	V
14	11	-27	0	x1=2.45 x2=0	x1=2.45 x2=0	V
15	-12	10	0	x1=0 x2=0.83	x1=0 x2=0.83	V
16	2	-14	24	x1=4 x2=3	x1=4 x2=3	V
17				a is not integer	a is not integer	V
18	1	4	4	x1=-2 x2=-2	x1=-2 x2=-2	V
19	70	-50	0	x1=0.71 x2=0	x1=0.71 x2=0	V
20	8	-30	7	x1=3.5 x2=0.25	x1=3.5 x2=0.25	V

3 Fiabilitatea

$$F = \frac{K}{n} \tag{1}$$

K-numarul de executii corecte
n-numarul total de executii

$F = \frac{20}{20} = 1$
=> Fiabilitate este de 100 %

4 Complexitatea Halstead

$$C_H = n_1 * \log_2 N_1 + n_2 * \log_2 N_2 \tag{2}$$

n_1-numarul de operatori distincti din program
n_2-numarul de operanzi distincti din program
N_1-numarul total de operatori distincti din program
N_2-numarul total de operanzi distincti din program

Lungimea programului:

$$N = N_1 + N_2 \tag{3}$$

Vocabularul programului:

$$n = n_1 + n_2 \tag{4}$$

Volumul programului:

$$V = N * \log_2 n \tag{5}$$

Dificultatea:

$$D = \frac{n_1}{2} * \frac{N_2}{n_2} \tag{6}$$

Efortul:

$$E = D * V \tag{7}$$

Timpul:

$$T = \frac{E}{18} sec \tag{8}$$

Nivelul:

$$L = \frac{1}{D} \tag{9}$$

4.1 QuadraticEquation.cs

Id	Operator	Aparitie
1	using	5
2	System	1
3	System.Collections.Generic	1
4	System.Linq	1
5	System.Text	1
6	System.Threading.Tasks	1
7	;	15
8	namespace	1
9	public	5
10	class	1
11	{}	12
12	private	1
13	set	1
14	get	2
15	double	10
16	return	2
17	()	32
18	Math.Pow()	1
19	,	3
20	-	6
21	*	6
22	if()	4
23	!=	1
24	null	2
25	=	4
26	else	1
27	throw	2
28	new	4
29	Exception()	2
30	Equals()	1
31	/	4
32	>	1
33	+	1
34	Math.Sqrt()	2
35	<	1
36	.	21

Id	Operand	Aparitie
1	Lab1	1
2	QuadraticEquation	1
3	"ax2 + bx + c = 0"	1
4	Coefficients[class]	2
5	Coefficients[property]	2
6	Coefficients.A	5
7	Coefficients.B	5
8	Coefficients.C	1
9	2	5
10	4	1
11	QuadraticEquation()	1
12	coefficients	3
13	"'coefficients' cannot be null"	1
14	Solution	4
15	SolveWithRealSolutions()	1
16	solution	4
17	Delta	6
18	0	3
19	"the equation do not have a real solution"	1

$n_1 = 36$
$n_2 = 19$
$N_1 = 159$
$N_2 = 48$

$C_H = 36 * \log_2 159 + 19 * \log_2 48 = 36 * 7.31 + 19 * 5.58 = 263.16 + 106.02 = 369.18$

$N = 159 + 48 = 207$

$n = 36 + 19 = 55$

$V = 207 * \log_2 55 = 207 * 5.78 = 1196.46$

$D = \frac{36}{2} * \frac{48}{19} = 18 * 2.52 = 45.36$

$E = 45.36 * 1196.46 = 54271.4256$

$T = \frac{54271.4256}{18} = 3015.0792sec$

$L = \frac{1}{45.36} = 0.022045$

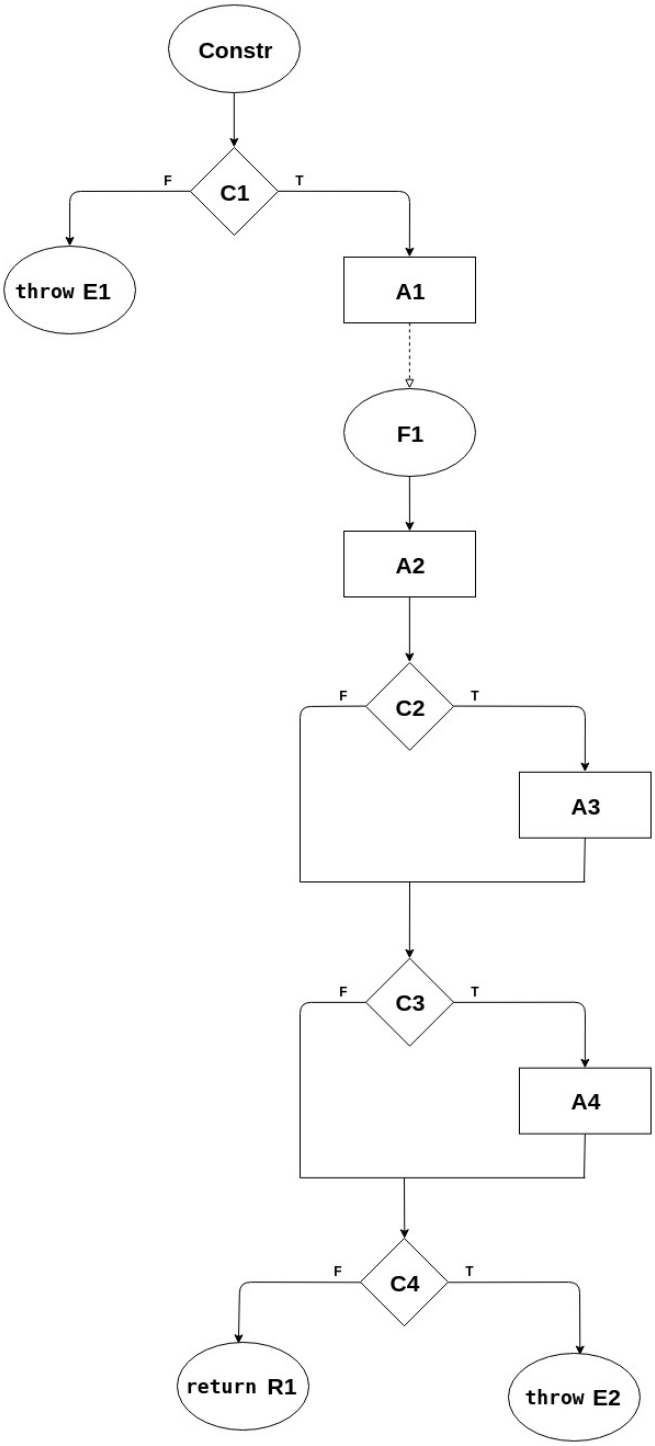
5 Complexitatea McCabe

$$C_{MC} = n_a - n_n + 2 \tag{10}$$

n_a-numarul de arce
n_n-numarul de noduri

5.1 QuadraticEquation.cs

Constr - QuadraticEquation (Coefficients coefficients)
$\mathbf{C}_1 - coefficients! = null$
$\mathbf{C}_2 - Delta.Equals(0)$
$\mathbf{C}_3 - Delta > 0$
$\mathbf{C}_4 - Delta < 0$
$\mathbf{A}_1 - Coefficients = coefficients;$
$\mathbf{A}_2 - Solutionsolution = null;$
$\mathbf{A}_3 - solution = newSolution((double)(-Coefficients.B)/(double)(2 * Coefficients.A), (double)(-Coefficients.B)/(double)(2 * Coefficients.A));$
$\mathbf{A}_4 - solution = newSolution((double)(-Coefficients.B + Math.Sqrt(Delta))/(double)(2 * Coefficients.A), (double)(-Coefficients.B - Math.Sqrt(Delta))/(double)(2 * Coefficients.A));$
$\mathbf{E}_1 - newException("'coefficients' cannot be null");$
$\mathbf{E}_2 - newException("the equation do not have a real solution");$
$\mathbf{F}_1 - SolveWithRealSolutions()$
$\mathbf{R}_1 - solution;$


Constr
C1
F
T
throw E1
A1
F1
A2
C2
F
T
A3
C3
F
T
A4
C4
F
T
return R1
throw E2

$n_a = 14$
$n_n = 13$

$C_{MC} = 14 - 13 + 2 = 3$

References

[1] https://github.com/vioan12/
Verificarea-si-Testarea-Sistemelor-de-Calcul