Методика выполнения контрольного задания № 1.3
«Построение таблицы функции с заданной точностью»

Постановка задачи. Построить алгоритм и отладить программу вычисления таблицы функции, заданной в виде аналитического представления

$$f_0(x) = \sum_{k=0}^{\infty} \frac{(-1)^k \cdot \left[\cos\left(\frac{b}{k+1}x\right) + \sin\left(\frac{b}{k+a}x\right)\right]}{4 \cdot (2k+a) \cdot (k+1)!} \left(\frac{11x}{7}\right)^{k+1}, \qquad (1)$$

на интервале $x \in [\, x_0, x_n \,]$ с заданным количеством строк n и абсолютной погрешностью ε, где a и b – некоторые заданные значения вещественных параметров данной функции.

На основе данных, имеющихся в таблице функции, установить наибольшее и наименьшее значения функции на всем заданном интервале $[\, x_0, x_n \,]$ изменения аргумента x.

Собственно создание таблицы функции $f = f(x)$, должно быть реализовано в теле конструктора нового класса – **`MyTableOfFunction`**, наследника от класса **`MyTable`**.

Указанный конструктор должен принимать конкретную функцию $f(x)$ в виде параметра.

Метод, реализующий вычисление значений функции $f(x)$ в соответствии с представлением (1), должен быть размещен в отдельном классе **`MyFunctions`** динамической библиотеки **`MAC_DLL`**:

```
7   namespace MAC_DLL
8   {
9   public class MyFunctions
10  {
11    public static double f0(double x, double a, double b, double eps)
12    {
13      if (x == 0) return 0.0;
14      double xz = 11.0 * x / 7.0, bx = b * x;
15      double Ao = (Math.Cos(bx) + Math.Sin(bx / a)) / a;
16      double pk = 1.0, Ak = 1.0, summa = 0.0;
17      for (int k = 1; Math.Abs(Ak) > eps; k++)
18      {
19        pk = -pk * xz / (k + 1.0);
20        Ak = (Math.Cos(bx / (k + 1.0)) + Math.Sin(bx / (k + a)))
21              * pk / (2.0 * k + a);
22        summa += Ak;
23      }
24      return 0.25 * xz * (Ao + summa);
25    }
26  }
27  }
```

В имеющемся рабочем пространстве **MAC_Solution** сгенерируем (добавим) новый проект **MAC_CheckTask_1_3** консольного приложения, который сделаем стартовым проектом.

Добавим в раздел **References** проекта **MAC_CheckTask_1_3** ссылку на проект **MAC_DLL**.

Можно приступать к разработке программной реализации класса **MyTableOfFunction**.

По условию задания конструктор **MyTableOfFunction()** должен принимать в качестве входного параметра метод с сигнатурой, соответствующей функции от одной действительной переменной (1), т.е. $f = f(x)$.

Это обязывает нас добавить в раздел делегатов библиотеки **MAC_DLL** определение нового делегата – **Function_of_x(double x)**.

Выполним это в уже имеющемся файле – **MAC_Common.cs** – в разделе «Делегаты»:

```
1   using ...
8
9   namespace MAC_DLL
10  {
11      # region     <--- Делегаты --->
12
13      public delegate double Member_of_Numeric_Series(int Index);
14      public delegate double Function_of_x(double x);
15
16      #endregion <--- Делегаты --->
17
18      <--- Классы --->
243
244 }
```

Алгоритм расчета таблицы функции, когда она задана аналитически, достаточно прост.

Непосредственно в теле конструктора класса **MyTableOfFunction** выполняется цикл по значениям аргумента от левой границы x_0 до правой границы x_n области определения $x \in [\, x_0, x_n \,]$ с постоянным шагом $h = (x_n - x_0) / n$.

В этом же цикле одновременно выполняется отбор узлов таблицы с наименьшим и наибольшим значениями функции.

Также следует переопределить виртуальный метод **ToPrint()**.

Класс `MyTableOfFunction`. Предлагается следующая реализация класса:

```
1   using System;
2   using System.Collections.Generic;
3   using CI = System.Globalization.CultureInfo;
4   using System.IO;
5   using System.Text;
6   using System.Threading.Tasks;
7   using System.Linq;
8
9   namespace MAC_DLL
10  {
11      <--- Делегаты --->
17
18      #region     <--- Классы --->
19
20      public class Point_xf...
39
40      public abstract class MyTable...
140
141     public class MyTableOfFunction : MyTable
142     {
143         readonly Function_of_x Fx;  // Указатель на функцию f(x)
144
145         public MyTableOfFunction(double xo, double xn, int n,
146                                  Function_of_x f_x, string title)
147         {
148             Title = title; Fx = f_x;
149             Points = new Point_xf[n + 1];
150
151             Minimum = new Point_xf(double.NaN, double.MaxValue);
152             Maximum = new Point_xf(double.NaN, double.MinValue);
153
154             double hx = (xn - xo) / n, xi;
155             for (int i = 0; i <= n; i++)
156             {
157                 xi = xo + i * hx; Points[i] = new Point_xf(xi, Fx(xi));
158                 if (Minimum.f > Points[i].f) { Minimum = Points[i]; }
159                 if (Maximum.f < Points[i].f) { Maximum = Points[i]; }
160             }
161         }
162
163         #region     <--- Переопределение методов класса MyTable --->
164         public override string ToPrint(string Comment)
165         {
166             return Comment + "\r\n" + Table_of_Function();
167         }
168         #endregion <--- Переопределение методов класса MyTable --->
169     }
170
171     public class MyTableOfData ...
265
266     #endregion <--- Классы --->
267 }
```

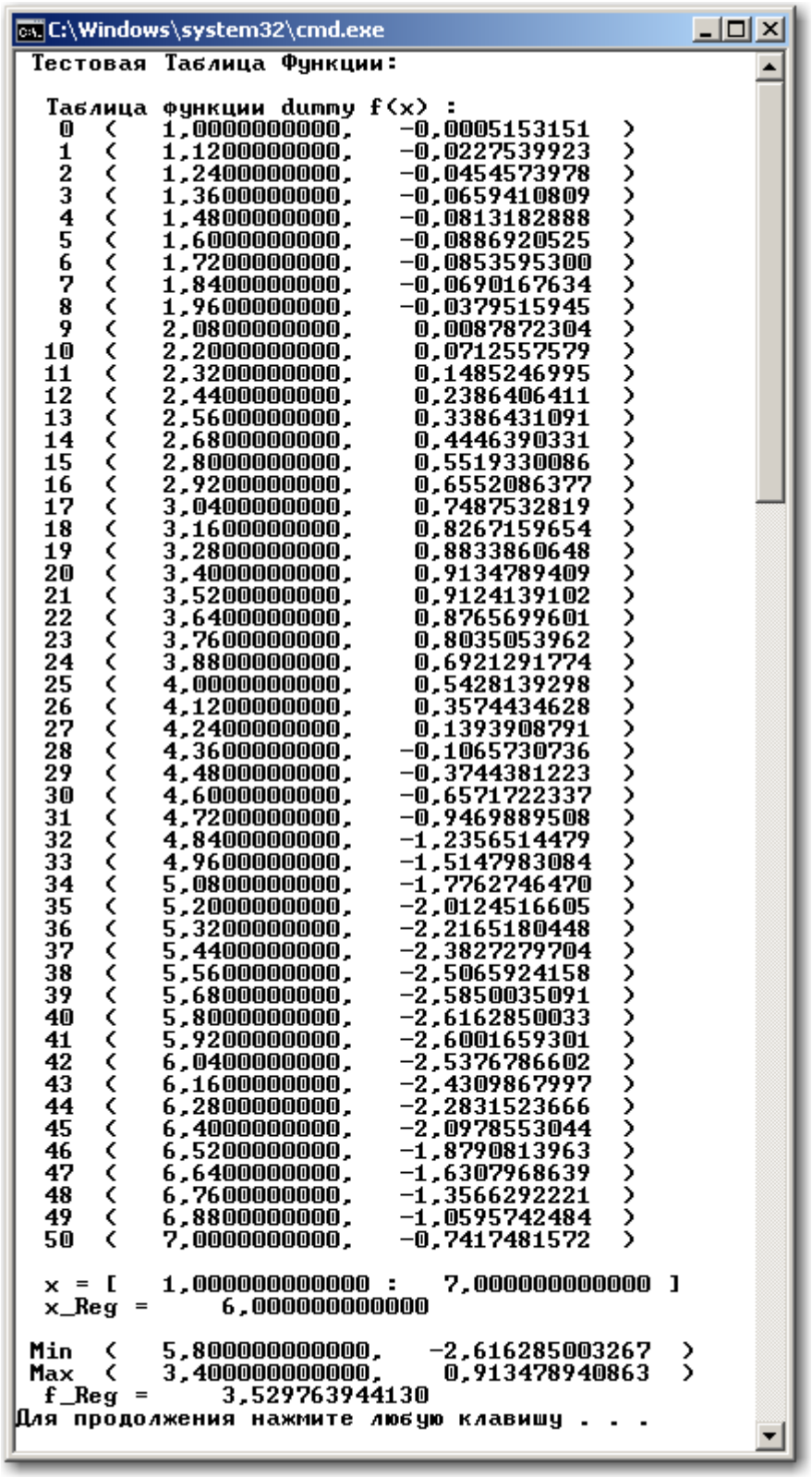
По условиям задания, параметры конструктора **`MyTableOfFunction()`** имеют следующий смысл:

- **`xo`** – левая граница интервала, на котором рассчитывается таблица функции;
- **`xn`** – правая граница этого интервала;
- **`n`** – количество равных интервалов, на которое требуется разбить отрезок [x_0, x_n];
- **`f_x`** – делегат, определяющий сигнатуру метода, непосредственно вычисляющего значение заданной функции;
- **`title`** – строковая переменная, содержащая «идентификатор» таблицы функции.

Теперь можно выполнить первую проверку и вычислить таблицу функции (1) в основной программе **`MAC_CheckTask_1_3`**:

```
1   using System;
2   using MyF = MAC_DLL.MyFunctions;
3   using MyTF = MAC_DLL.MyTableOfFunction;
4   // using System.Collections.Generic; ...
8
9   namespace MAC_Check_Task_1_3
10  {
11    class Main_CT_1_3
12    {
13      public static double par_a, par_b, par_e;
14
15      static void Main(string[] args)
16      {
17        par_a = 3.0; par_b = 1.8; par_e = 1.0E-9;
18
19        MyTF TF = new MyTF(1.0, 7.0, 50, dummy_fx, "dummy f(x)");
20        Console.Write(TF.ToPrint(" Тестовая Таблица Функции:"));
21      }
22      public static double dummy_fx(double x)
23      {
24        return MyF.f0(x, par_a, par_b, par_e);
25      }
26    }
27  }
```

Имеем следующий результат:



Визуальный анализ таблицы подтверждает правильность программного определения ее узлов с наибольшим и наименьшим значениями функции, а также длин области определения и области значений для данной функции.

Подведем предварительные итоги проделанной работы.

В нашем распоряжении имеется абстрактный класс **MyTable**, описывающий обобщенную совокупность числовых данных типа «Таблица» и алгоритмы (методы и свойства) их обработки вне зависимости от способа формирования самой таблицы.

На основе этого абстрактного класса мы создали два класса-наследника **MyTableOfData** и **MyTableOfFunction**, которые формируют таблицу двумя принципиально различными способами. При этом классы-наследники пользуются общими методами и свойствами класса **MyTable**, и при необходимости переопределяют их.

Мы и в дальнейшем будем придерживаться такой стратегии в формировании новых свойств и методов для указанных классов.

Общие для двух типов таблиц алгоритмы мы будем размещать в родительском классе **MyTable**. А все специфические алгоритмы – как собственные члены в классах-наследниках.

Продемонстрируем сказанное следующим образом.

Результаты выполнения двух последних заданий **MAC_LabWork_1_3** и **MAC_CheckTask_1_3** содержали достаточно большие объемы числовой информации, которая выводилась непосредственно в окно консольного приложения.

Просмотр и анализ результатов в такой форме нельзя признать удобным, в особенности, когда ее надо сохранить с целью сопоставления с подобными результатами.

Очевидным является способ сохранения результатов в виде текстового именованного файла, который, кроме всего прочего, можно включить в соответствующий проект рабочего пространства в качестве его обновляемого компонента.

Поскольку сохранение результатов обработки таблицы не зависит от способа генерации ее числовых данных, метод, который будет нам обеспечивать вывод результата в файл, мы разместим в абстрактном классе и сделаем виртуальным, чтобы при необходимости его можно было переопределить.

Основным аргументом этого метода является полный путь к файлу – параметр **path**, в который планируется сохранять результаты вычислений.

Если путь к файлу содержит только его имя, то файл размещается непосредственно в рабочей директории исполняемого файла данного проекта.

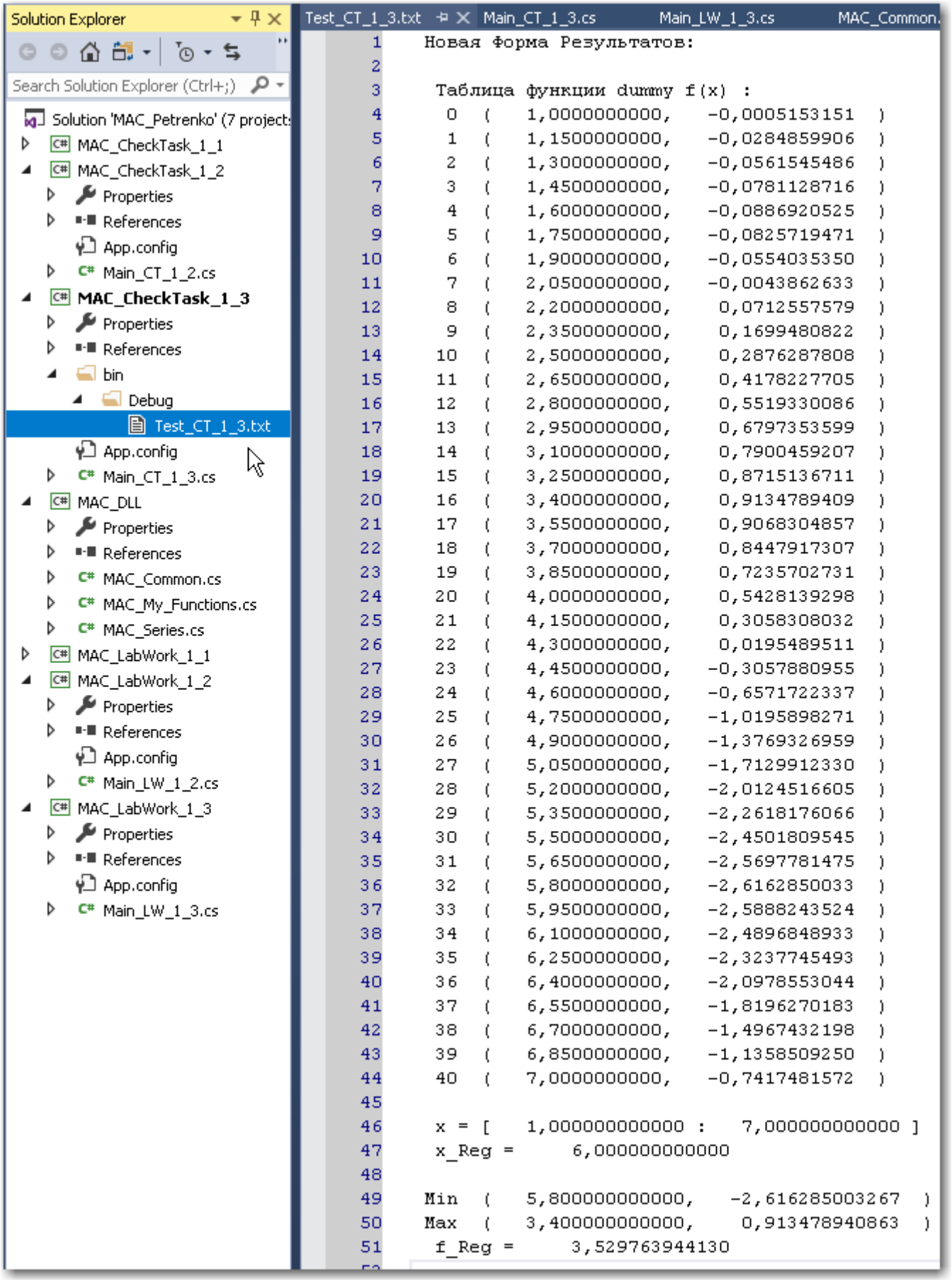
Если имя не указано – выбирается имя «по умолчанию» – **My_Table.txt**:

```
40  public abstract class MyTable
41  {
42      <--- Основные Свойства MyTable --->
72
73      #region     <--- Основные Методы MyTable --->
74
75      // Метод - возвращает значение x для строки таблицы с номером index
76      public double X(int index)...
81
82      // Метод - возвращает значение f для строки таблицы с номером index
83      public double F(int index)...
88
89      // Метод, формирующий таблицу функции в текстовой форме
90      public virtual string Table_of_Function()...
121
122     // Метод - возвращает два массива со значениями аргумента x и функции f
123     public void ToArrays(out double[] x, out double[] f)...
129
130     // Дополнительный переопределяемый метод, предназначенный для операций
131     //  вывода в текстовой форме различной информации по данной таблице
132
133     public virtual string ToPrint(string comment)...
137
138     public virtual void To_txt_File(string path, string comment)
139     {
140         if (path == "") path = "My_Table.txt";
141         FileInfo file = new FileInfo(path);
142         if (file.Exists) file.Delete();
143         StreamWriter SW = new StreamWriter(file.OpenWrite());
144         SW.Write(comment + "\r\n" + Table_of_Function());
145         SW.Close();
146     }
147
148     #endregion <--- Основные Методы MyTable --->
149 }
```

Добавим вызов нового метода в приложение **`MAC_CheckTask_1_3`**:

```
12  class Main_CT_1_3
13  {
14      public static double par_a, par_b, par_e;
15
16      static void Main(string[] args)
17      {
18          par_a = 3.0; par_b = 1.8; par_e = 1.0E-9;
19
20          MyToF TF = new MyToF(1.0, 7.0, 40, dummy_fx, "dummy f(x)");
21          //Console.Write(TF.ToPrint(" Тестовая Таблица Функции:"));
22          TF.To_txt_File("Test_CT_1_3.txt", " Новая Форма Результатов:");
23      }
24      public static double dummy_fx(double x)
25      {
26          return MyF.f0(x, par_a, par_b, par_e);
27      }
28  }
```

Файл **`Test_CT_1_3.txt`** присоединим к проекту и откроем для просмотра:



Переходим к первой задаче математической обработки данных в таблице функции.

Из курса математического анализа известно, что если функция $f(x)$ строго монотонна на интервале $[x_{i-1}, x_i]$ и выполняется неравенство $f(x_{i-1}) \cdot f(x_i) < 0$, то на этом интервале имеется точка с координатой x_*, в которой функция обращается в нуль, т.е. $f(x_*) = 0$.

Поставим перед собой задачу программно установить для уже созданной таблицы все интервалы $[x_{i-1}, x_i]$ изменения аргумента, на концах которых функция $f(x)$ имеет разные знаки, т.е. выполняется неравенство $f(x_{i-1}) \cdot f(x_i) < 0$.

Очевидно, что мы не можем наперед угадать количество таких интервалов.

Нам придется последовательно «обрабатывать» узлы таблицы и определять из них те, для которых выполняются указанные выше условия.

Как только такой интервал $[x_{i-1}, x_i]$ будет обнаружен – мы должны сохранить информацию о нем в какой-то переменной и перейти к следующим узлам таблицы.

Для подобного «сбора» информации подходит тип данных – список.

Но нам еще предстоит обобщить понятие «интервал, содержащий нуль функции» в виде некоторого пользовательского типа данных (класса) и определить количество и качество составляющих его членов.

Будем исходить из того, что интервал, содержащий нуль функции, описывается тремя действительными числами: это координаты концов интервала $[x_{i-1}, x_i]$ и собственно координата x_*, в которой функция обращается в нуль, т.е. $f(x_*) = 0$.

В численных методах координата x_* вычисляется приближенно (с использованием определенного алгоритма) с некоторой погрешностью, которая характеризует «качество» найденного значения x_*. Эту погрешность можно отождествить со значением $\varepsilon = \left|f(x_*)\right|$.

Чем меньше эта величина, тем «качественнее» нами вычислено значение x_*.

Мы включим эту величину в качестве члена в создаваемый класс.

Кроме того, алгоритмы уточнения нулей функции, как правило, выполняются итерационным путем. Нас будет интересовать количество итераций, затраченных тем или иным численным методом на вычисление нуля функции с заданной погрешностью.

Чем меньше таких итераций – тем лучше численный метод.

Количество итераций – целое число – также будет членом нового класса.

Класс `Root`.

Ниже предлагается код нового класса **`Root`**, который разместим в разделе общих классов файла **`MAC_Common.cs`**:

```
 9   namespace MAC_DLL
10   {
11     <--- Делегаты --->
17
18     #region     <--- Классы --->
19
20     public class Point_xf...
39
40     public class Root
41     {
42       public double xL, xR, x, err; public int iters;
43       public Root(double x_left, double x_right)
44       {
45         xL = x_left; xR = x_right; x = double.NaN; err = double.NaN; iters = 0;
46       }
47       public string ToPrint()
48       {
49         return "   [" + string.Format("{0:F5},", xL).PadLeft(10)
50                       + string.Format("{0:F5}", xR).PadLeft(10)
51                       + "   ]   root ="
52                       + string.Format("{0:F12}", x).PadLeft(16)
53                       + "   err ="
54                       + string.Format("{0:E1}", err).PadLeft(10)
55                       + "   iters = "
56                       + string.Format("{0}", iters);
57       }
58     }
59
60     public abstract class MyTable...
170
171    public class MyTableOfFunction ...
200
201    public class MyTableOfData ...
295
296    #endregion <--- Классы --->
297  }
```

Добавим в совокупность членов абстрактного класса **`MyTable`** новое свойство – **`List<Root> Roots`** – список объектов типа **`Root`**.

Алгоритм инициализации этого списка и заполнения его соответствующими экземплярами класса **`Root`** реализуем в виде закрытого метода **`Roots_Location()`** абстрактного класса **`MyTable`**. Одновременно создадим вспомогательный метод **`Table_of_Roots()`**, «формирующий» таблицу искомых интервалов. Оба этих метода разместив в отдельном регионе «Дополнительные Методы **`MyTable`**»:

```
60  public abstract class MyTable
61  {
62    <--- Основные Свойства MyTable --->
92
93    #region    <--- Дополнительные Свойства MyTable --->
94
95    protected List<Root> Roots { get; set; }
96
97    #endregion <--- Дополнительные Свойства MyTable --->
98
99    <--- Основные Методы MyTable --->
175
176   #region    <--- Дополнительные Методы MyTable --->
177
178   protected void Roots_Location()
179   {
180     int counter = 0;
181     for (int i = 1; i < Length; i++)
182     {
183       if (Points[i - 1].f * Points[i].f < 0)
184       {
185         counter++;
186         if (counter == 1) { Roots = new List<Root>(); }
187         Roots.Add(new Root(Points[i - 1].x, Points[i].x));
188       }
189     }
190   }
191   public string Table_of_Roots(string comment)
192   {
193     string table = comment + "\r\n";
194     if (Roots != null)
195     {
196       table += " Таблица нулей " + Title + " функции: \r\n";
197       for (int j = 0; j < Roots.Count; j++)
198       {
199         table += string.Format("{0}", j).PadLeft(3)
200                  + Roots[j].ToPrint() + "\r\n";
201       }
202     }
203     else { table += " Таблица нулей " + Title + " пустая!\r\n"; }
204     return table;
205   }
206
207   #endregion <--- Дополнительные Методы MyTable --->
208 }
```

Расширим функционал метода **`Table_of_Function()`**:

```
99      #region    <--- Основные Методы MyTable --->
100
101     // Метод - возвращает значение x для строки таблицы с номером index
102     public double X(int index)...
107
108     // Метод - возвращает значение f для строки таблицы с номером index
109     public double F(int index)...
114
115     // Метод, формирующий таблицу функции в текстовой форме
116     public virtual string Table_of_Function()
117     {
118       string txt = "\r\n  Таблица функции " + Title + " : \r\n";
119
120       for (int i = 0; i < Length; i++)
121       {
122         txt += string.Format("{0}", i).PadLeft(4)
123              + Points[i].ToPrint() + "\r\n";
124       }
125
126       txt += "\r\n  x = ["
127              + string.Format("{0:F12}", Points[0].x).PadLeft(17) + " :"
128              + string.Format("{0:F12}", Points[Length - 1].x).PadLeft(17)
129              + " ] \r\n";
130
131       txt += "  x_Reg = "
132              + string.Format("{0:F12}", Region_x).PadLeft(18) + "\r\n";
133
134       txt += "\r\n Min  ("
135              + string.Format("{0:F12},", Minimum.x).PadLeft(18)
136              + string.Format("{0:F12}", Minimum.f).PadLeft(18) + "  )";
137
138       txt += "\r\n Max  ("
139              + string.Format("{0:F12},", Maximum.x).PadLeft(18)
140              + string.Format("{0:F12}", Maximum.f).PadLeft(18) + "  )";
141
142       txt += "\r\n  f_Reg = "
143              + string.Format("{0:F12}", Region_f).PadLeft(18) + "\r\n";
144
145       return txt + Table_of_Roots("");
146     }
147
148     // Метод - возвращает два массива со значениями аргумента x и функции f
149     public void ToArrays(out double[] x, out double[] f)...
155
156     // Дополнительный переопределяемый метод, предназначенный для операций ...
158
159     public virtual string ToPrint(string comment)...
163
164     public virtual void To_txt_File(string path, string comment)...
173
174     #endregion <--- Основные Методы MyTable --->
```

Добавим теперь вызов метода **Roots_Location()** в коды конструкторов классов-наследников **MyTableOfData** и **MyTableOfFunction**:

```
241  public class MyTableOfData : MyTable
242  {
243    // Свойство - Таблица данных непосредственно в файле
244    public string Table_in_File { get; }
245
246    // <--- Конструктор экземпляра --->
247    public MyTableOfData(string path, string title)
248    {
249      List<Point_xf> Temp = new List<Point_xf>();
250      FileInfo file = new FileInfo(path);
251      Table_in_File = "\r\n Таблица " + title +
252                      " в файле " + file.Name + " :\r\n";
253
254      <--- Чтение бинарного файла в список данных --->
275
276      <--- Чтение форматного файла в список данных --->
306
307      <--- Формирование Таблицы Данных --->
326
327      Roots_Location();
328    }
329    <--- Переопределение методов класса MyTable --->
335  }
```

```
210  public class MyTableOfFunction : MyTable
211  {
212    readonly Function_of_x Fx;  // Указатель на функцию f(x)
213
214    public MyTableOfFunction(double xo, double xn, int n,
215                             Function_of_x f_x, string title)
216    {
217      Title = title; Fx = f_x;
218      Points = new Point_xf[n + 1];
219
220      Minimum = new Point_xf(double.NaN, double.MaxValue);
221      Maximum = new Point_xf(double.NaN, double.MinValue);
222
223      double hx = (xn - xo) / n, xi;
224      for (int i = 0; i <= n; i++)
225      {
226        xi = xo + i * hx; Points[i] = new Point_xf(xi, Fx(xi));
227        if (Minimum.f > Points[i].f) { Minimum = Points[i]; }
228        if (Maximum.f < Points[i].f) { Maximum = Points[i]; }
229      }
230      Roots_Location();
231    }
232
233    <--- Переопределение методов класса MyTable --->
239  }
```

Перекомпилируем проект **MAC_DLL** и убедимся в отсутствии ошибок.

Выполним заново приложение **MAC_CheckTask_1_3** (не внося в него никаких изменений) и получим новый результат – таблицу функции с выделенными интервалами, на концах которых функция меняет знак (нижняя часть таблицы):

```
 6      2  (   1,3000000000,   -0,0561545486  )
 7      3  (   1,4500000000,   -0,0781128716  )
 8      4  (   1,6000000000,   -0,0886920525  )
 9      5  (   1,7500000000,   -0,0825719471  )
10      6  (   1,9000000000,   -0,0554035350  )
11      7  (   2,0500000000,   -0,0043862633  )
12      8  (   2,2000000000,    0,0712557579  )
13      9  (   2,3500000000,    0,1699480822  )
14     10  (   2,5000000000,    0,2876287808  )
15     11  (   2,6500000000,    0,4178227705  )
16     12  (   2,8000000000,    0,5519330086  )
17     13  (   2,9500000000,    0,6797353599  )
18     14  (   3,1000000000,    0,7900459207  )
19     15  (   3,2500000000,    0,8715136711  )
20     16  (   3,4000000000,    0,9134789409  )
21     17  (   3,5500000000,    0,9068304857  )
22     18  (   3,7000000000,    0,8447917307  )
23     19  (   3,8500000000,    0,7235702731  )
24     20  (   4,0000000000,    0,5428139298  )
25     21  (   4,1500000000,    0,3058308032  )
26     22  (   4,3000000000,    0,0195489511  )
27     23  (   4,4500000000,   -0,3057880955  )
28     24  (   4,6000000000,   -0,6571722337  )
29     25  (   4,7500000000,   -1,0195898271  )
30     26  (   4,9000000000,   -1,3769326959  )
31     27  (   5,0500000000,   -1,7129912330  )
32     28  (   5,2000000000,   -2,0124516605  )
33     29  (   5,3500000000,   -2,2618176066  )
34     30  (   5,5000000000,   -2,4501809545  )
35     31  (   5,6500000000,   -2,5697781475  )
36     32  (   5,8000000000,   -2,6162850033  )
37     33  (   5,9500000000,   -2,5888243524  )
38     34  (   6,1000000000,   -2,4896848933  )
39     35  (   6,2500000000,   -2,3237745493  )
40     36  (   6,4000000000,   -2,0978553044  )
41     37  (   6,5500000000,   -1,8196270183  )
42     38  (   6,7000000000,   -1,4967432198  )
43     39  (   6,8500000000,   -1,1358509250  )
44     40  (   7,0000000000,   -0,7417481572  )
45
46     x = [   1,000000000000 :   7,000000000000 ]
47     x_Reg =     6,000000000000
48
49    Min  (   5,800000000000,   -2,616285003267  )
50    Max  (   3,400000000000,    0,913478940863  )
51     f_Reg =     3,529763944130
52
53  Таблица нулей dummy f(x) функции:
54    0  [  2,05000,   2,20000  ]  root =           NaN  err =       NaN  iters = 0
55    1  [  4,30000,   4,45000  ]  root =           NaN  err =       NaN  iters = 0
```

Теперь вернемся к проекту **MAC_LabWork_1_3** и внесем в код соответствующие изменения, после чего выполним проверку корректности работы новых методов:

```
using System;
using MyTD = MAC_DLL.MyTableOfData;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MAC_LabWork_1_3
{
  class Main_LW_1_3
  {
    static void Main(string[] args)
    {
      MyTD T1 = new MyTD("MAC_LW_1_3_v00.bin", "binary file");
      string txt = "\r\n";
      txt += string.Format(" Прочитано строк: {0}", T1.Length);
      txt += T1.ToPrint("   Обработка файла *.bin");
      Console.WriteLine(txt);

      MyTD T2 = new MyTD("MAC_LW_1_3_v00.txt", "text   file");
      txt = "\r\n";
      txt = string.Format("  Прочитано строк: {0}", T2.Length);
      txt += T2.ToPrint("   Обработка файла *.txt");
      Console.WriteLine(txt);
      T2.To_txt_File("Test_LW_1_3.txt", " Новая Форма Результатов:");
    }
  }
}
```

Не забудьте включить файл результатов **`Test_LW_1_3.txt`** в состав проекта **`MAC_LabWork_1_3`** и открыть его для просмотра в **`MS VS`**:

```
Новая Форма Результатов:

 Таблица функции text   file :
  0  (  -3,1000000000,   -1,5810000000  )
  1  (  -2,8791666667,    1,6974586950  )
  2  (  -2,6583333333,    4,2309959491  )
  3  (  -2,4375000000,    6,0842285156  )
  4  (  -2,2166666667,    7,3217731481  )
  5  (  -1,9958333333,    8,0082466001  )
  6  (  -1,7750000000,    8,2082656250  )
  7  (  -1,5541666667,    7,9864469763  )
  8  (  -1,3333333333,    7,4074074074  )
  9  (  -1,1125000000,    6,5357636719  )
 10  (  -0,8916666667,    5,4361325231  )
 11  (  -0,6708333333,    4,1731307147  )
 12  (  -0,4500000000,    2,8113750000  )
 13  (  -0,2291666667,    1,4154821325  )
 14  (  -0,0083333333,    0,0500688657  )
 15  (   0,2125000000,   -1,2202480469  )
 16  (   0,4333333333,   -2,3308518519  )
 17  (   0,6541666667,   -3,2171257957  )
 18  (   0,8750000000,   -3,8144531250  )
 19  (   1,0958333333,   -4,0582170862  )
 20  (   1,3166666667,   -3,8838009259  )
 21  (   1,5375000000,   -3,2265878906  )
 22  (   1,7583333333,   -2,0219612269  )
 23  (   1,9791666667,   -0,2053041811  )
 24  (   2,2000000000,    2,2880000000  )

 x = [  -3,100000000000 :   2,200000000000 ]
 x_Reg =     5,300000000000

Min  (   1,095833333333,   -4,058217086227  )
Max  (  -1,775000000000,    8,208265625000  )
 f_Reg =    12,266482711227

Таблица нулей text   file функции:
 0  [ -3,10000,   -2,87917  ]  root =          NaN  err =       NaN  iters = 0
 1  [ -0,00833,    0,21250  ]  root =          NaN  err =       NaN  iters = 0
 2  [  1,97917,    2,20000  ]  root =          NaN  err =       NaN  iters = 0
```