Министерство образования и науки Кыргызской Республики

**Кыргызский Государственный Технический Университет**

**им. И. Раззакова**

Кафедра программного обеспечения компьютерных систем

НАПРАВЛЕНИЕ МАГИСТЕРСКОЙ ПОДГОТОВКИ 710400

**ПРОГРАММНАЯ ИНЖЕНЕРИЯ**

**Основы разработки и анализа требований к ПО**

**Отчет**

**на программное обеспечение для**

**Bisection Method**

к курсу «Основы разработки и анализа требований к ПО»

**Выполнил: ст.гр. ПИ(б) 1-13 Темиров А.А.**

**Проверил: к.т.н проф. Тен И.Г.**

**Бишкек 2016г.**

# СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 2](#_Toc465241454)

[Глава 1: Описание проблемы 3](#_Toc465241455)

[Глава 2: Описание Bisection Method 3](#_Toc465241456)

[Глава 3: Спецификация требований к программному обеспечению 5](#_Toc465241457)

[3.1. Наименование программной разработки 5](#_Toc465241458)

[3.2. Описание бизнес-процесса и анализ расхождения 5](#_Toc465241459)

[3.3. Бизнес-цель разработки программного обеспечения 5](#_Toc465241460)

[3.4. Пользовательская история разработки программного обеспечения 5](#_Toc465241461)

[3.5. Функциональные требования к разработке программного обеспечения 6](#_Toc465241462)

[3.6. Нефункциональные требования на разработку программного обеспечения 7](#_Toc465241463)

[3.7. Ограничения на разработку программного обеспечения 7](#_Toc465241464)

[Глава 4: Интерфейс программы 9](#_Toc465241465)

[Глава 5: Стадии проектирования системы для поиска корня нелинейного уравнения f(x)=0, реализующей Bisection Method: 10](#_Toc465241466)

[Глава 6: Стадии конструирования системы для поиска корня нелинейного уравнения f(x)=0, реализующей Bisection Method: 14](#_Toc465241467)

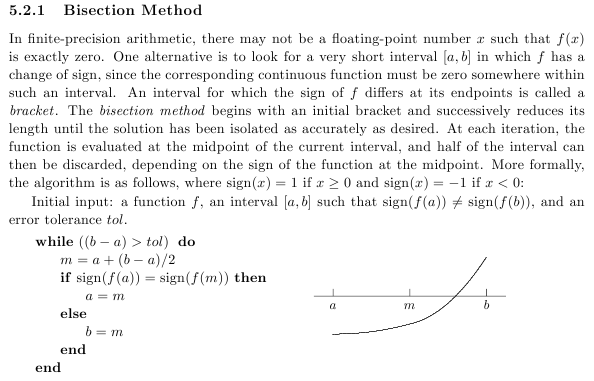
[Глава 7. Тестирование 20](#_Toc465241468)

[References 32](#_Toc465241469)

# Глава 1: Описание проблемы

Найти корень произвольного нелинейного уравнения – ноль нелинейной функции f(x) с заданной допустимой погрешностью Tolerance не более величины 1E–28 методом деления отрезка пополам (Bisection Method). Нелинейная функция f(x) имеет произвольный аналитический вид, составленный из математических функций (полиномов различных степеней, тригонометрических – sin(x), cos(x), exp(x), ln(x), log(x) и. т. д.), которая имеет математический смысл, и для которой существует хотя бы одно решение задачи. Полное описание метода деления отрезка пополам – Bisection Method – приведено по описанию метода из интернет-ресурса <https://ru.wikipedia.org/wiki/Метод_бисекции>.

# Глава 2: Описание Bisection Method



[GoTo Contents](#Contents)

# Глава 3: Спецификация требований к программному обеспечению

## 3.1. Наименование программной разработки

Система поиска корней нелинейного уравнения методом деления отрезка пополам.

## 3.2. Описание бизнес-процесса и анализ расхождения

При решении различных бизнес-задач отделу исследований (пользователям) требуется находить корень произвольного нелинейного уравнения с высокой точностью. Для решения таких задач в этом отделе имеется 5 ставок специалистов. Два из них занимаются формализацией задачи и определением/нахождением аналитического вида нелинейной функции, описывающей как можно точнее модель поведения реальной системы, а остальные занимаются решением задачи – поиском корня нелинейного уравнения для найденной нелинейной функции. При больших трудозатратах (заняты два специалиста высокой квалификации), не всегда обеспечивается приемлемая и единообразная/унифицированная точность решения задачи. Бизнесу желательно было бы гарантированно обеспечить любую допустимую погрешность решения задачи при любых видах нелинейной функции f(x) с минимальными трудозатратами.

## 3.3. Бизнес-цель разработки программного обеспечения

Минимизировать затраты времени на поиск корня нелинейного уравнения ***до одной минуты*** при максимально высокой точности решения – допустимая погрешность решения должна быть ***не менее 1e-15 (т.е., 0.000000000000001)***, что позволит сократить ***две ставки*** специалиста в отделе исследований, поскольку теперь с такой задачей может справиться уже один специалист с помощью разрабатываемой системы.

## 3.4. Пользовательская история разработки программного обеспечения

***Пользователь*** ***должен вводить:***

* аналитическое выражение для требуемой нелинейной функции ***f(x)***;
* значения концов интервала ***[a, b]***;
* значение допустимой погрешности ***Tolerance*** решения задачи;
* допустимое максимальное количество ***k\_max*** итераций.

***Программа:***

* должна провести parsing – синтаксический анализ выражения введенной нелинейной функции и преобразовать аналитическую запись нелинейной функции в виде символов в вычисляемую процедуру в виде подпрограммы-функции;
* должна ввести символьные значения концов интервала ***[a, b]*** и преобразовать их в числовые значения концов интервала;
* должна найти решение задачи – корень нелинейного уравнения ***f(x)=0*** или ноль нелинейной функции ***f(x)*** – по алгоритму Bisection Method;
* должна выводить найденное значение ***X\**** корня заданного нелинейного уравнения;
* должна выводить значение заданной нелинейной функции ***f(x\*)*** в этой точке;
* должна выводить общее затраченное время (в миллисекундах) на поиск решения ***Elapsed Time*** для оценки производительности программы;
* должна выводить затраченное на поиск решения с заданной погрешностью количество итераций ***Amount Of Iterations***;
* должна выводить достигнутое значение абсолютной ошибки решения ***Abs(b–a)***.

## 3.5. Функциональные требования к разработке программного обеспечения

* Система должна использовать Bisection Method для поиска корней нелинейного уравнения;
  + Система должна искать решение задачи по следующему алгоритму (Bisection algorithm):
  + Система должна обеспечивать ввод любых начальных числовых значений концов интервала ***[a; b]***;
  + Система должна обеспечивать проверку условия . Выполнение этого условия гарантирует корректность значений введенных начальных числовых значений концов интервала ***[a; b]***, которые необходимы для правильной работы Bisection Method;
  + Система должна распознавать ситуации , которые недопустимы для корректного применения метода – Bisection Method – поиска корней нелинейного уравнения, и сообщать пользователю о возникновении подобной ситуации;
  + Система должна обеспечить возможность продолжения поиска решения задачи с последних достигнутых на шаге ***k\_max*** концов интервала ***[a(k\_max), b(k\_max)]***, приняв их за начальные значения концов интервала {т.е., приняв a(0) =a(k\_max) и b(0)=b(k\_max) система должна заново выполнить не менее ***k\_max*** итераций}, если за данное количество итераций ***k\_max*** не достигнуто решение с требуемой погрешностью ***Tolerance***;
* Система должна обеспечить возможность ввода исходных данных, имеющих следующие особенности:
* Система должна обеспечивать возможность ввода аналитического выражения для любой нелинейной функции;
* Система должна проводить parsing\*\* синтаксический анализ аналитического выражения нелинейной функции f(x) и автоматически конструировать подпрограмму для вычисления значения этой функции для любого значения аргумента этой функции;
* Система должна обеспечить ввод любого целого числового значения для максимального допустимого количества итераций ***k\_max***;
* Система должна обеспечивать возможность ввода любых числовых значений допустимой погрешности ***Tolerance*** как в формате с фиксированной точностью “0.000000000000001”, так и в экспоненциальном формате “1e-15”;
* Система должна быть способна проверять валидность введенного аналитического выражения в качестве функции f(x) и сообщать пользователю об ошибке в аналитическом выражении;
* Система должна обеспечить вывод следующих результатов решения задачи:
* Система должна обеспечить вывод результата решения задачи в виде числа x\* - корня нелинейного уравнения или нуля нелинейной функции;
* Система должна обеспечить вывод значения функции в этой точке f(x\*);
* Система должна обеспечить вывод количества итераций, которые потребовались системе, чтобы найти решение задачи с требуемой погрешностью в качестве критерия производительности системы;
* Система должна обеспечить только просмотр выходных данных системы;
* Система должна обеспечить оценку производительности программы при решении задачи заданным методом (***Bisection Method***) и реализации программы на языке ***Visual Basic в среде Visual Studio 2013*** путем измерения и вывода времени, затрачиваемой на поиск решения с заданной погрешностью;

\*\* ***Parsing*** — синтаксический анализатор; анализатор; парсер часть компилятора, выполняющая чтение исходного текста программы, проверку её синтаксиса и создание промежуточного файла для блоков, выполняющих дальнейшие стадии компиляции. Различают нисходящий (top-down parser) и восходящий (bottom-up parser) синтаксические анализаторы. Первый раскрывает продукции грамматики, начиная со стартового символа, до получения требуемой последовательности лексем, т. е. превращает входной поток символов в поток лексем (см. token ), а второй восстанавливает продукции из правых частей, начиная с лексем и кончая стартовым символом.

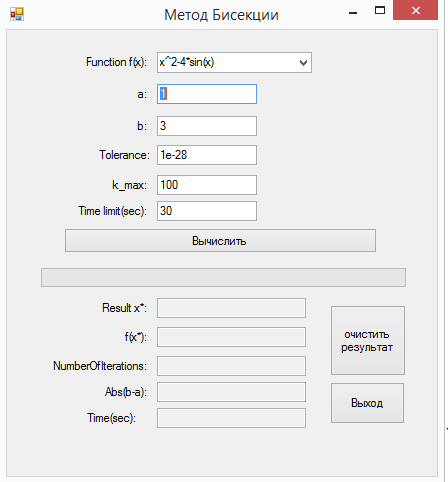
## 3.6. Нефункциональные требования на разработку программного обеспечения

* Программа в процессе поиска решения должна отображать индикатор состояния процесса выполнения операции ***Progressive Bar***, чтобы пользователь понимал, что система не зависла, а находится в рабочем состоянии;
* В случае ошибки во введенном аналитическом выражении нелинейной функции f(x) система должна предоставлять достаточно подробную информацию об ошибке, чтобы пользователь системы легко смог локализовать причину возникновения ошибки;
* Система должна обеспечить возможность очищения текстовых полей ***выходного*** интерфейса от результатов предыдущих вычислений при новом запуске вычислений. При этом данные в текстовых полях ***входного*** интерфейса должны сохраняться. Это позволит пользователю увидеть более наглядно, что получены новые результаты, соответствующие измененным входным данным;
* Система должна обеспечить оценку качества решения задачи заданным методом (***Bisection Method***) путем измерения и вывода количества итераций, потраченных на поиск решения с заданной погрешностью;

## 3.7. Ограничения на разработку программного обеспечения

* Нелинейная функция ***f(x)*** должна быть непрерывной на интервале ***[a; b]***;
* Разработать ПО на основе IDE (integrated development environment – интегрированная среда разработки) – Visual Studio 2013;
* Разработать ПО на языке Visual Basic;
* Система должна решать задачи за время не более 30 секунд;
* Временные затраты на разработку ПО не должны превышать одного месяца;
* Финансовые затраты на разработку ПО и соответствующую документацию (техническое задание на ПО, руководство пользователя и руководство программиста) не должны превышать $1000;
* Система должна быть разработана для эксплуатации на компьютерах типа пентиум с оперативной памятью не более 2 гигабайт;
* Разработка программы должна начаться не позже 16 сентября 2016 года и система должна быть передана заказчику не позднее 26 октября 2016 года;
* Система должна позволять пользователю решать не менее 100 задач поиска корня нелинейного уравнения за один час для любой нелинейной функции, описываемой в виде аналитического выражения, корректной с точки зрения математики;
* Система должна понимать и обрабатывать аналитические выражения для нелинейной функции, которые включают следующие математические символы:
  + Cтандартных функций – *sin, cos, tg, ctg, arcsin, arccos, arctg, arcctg, sh, ch, th, cth, exp, lg, ln, sqrt*;
  + Основных вычислительных операций: сложение (+), вычитание (-), умножение (\*), деление (/), остаток от деления (%) и возведение в степень (^);
  + круглых и квадратных скобок любой вложенности;
  + вектора переменных, на которые можно ссылаться, используя запись вида x[N] или x(N), где N - индекс переменной.
* Система не предназначена для ввода логических и алгоритмически заданных нелинейных функций, потому что понятие «корень нелинейного уравнения» относится к понятию числа, которое несовместимо с понятием логических значений «ложь-истина» вышеуказанных нелинейных функций.

# Глава 4: Интерфейс программы



[GoTo Contents](#Contents)

# Глава 5: Стадии проектирования системы для поиска корня нелинейного уравнения f(x)=0, реализующей Bisection Method:

Stage No.1: Блок-схема Bisection method

 Figure 1: Flowchart of the Bisection Method with a design of the input and output interfaces

[GoTo Contents](#Contents)

Stage No.2: Проектирование интерфейса системы, реализующей Bisection method и соответсвующих блок-схеме этого метода.

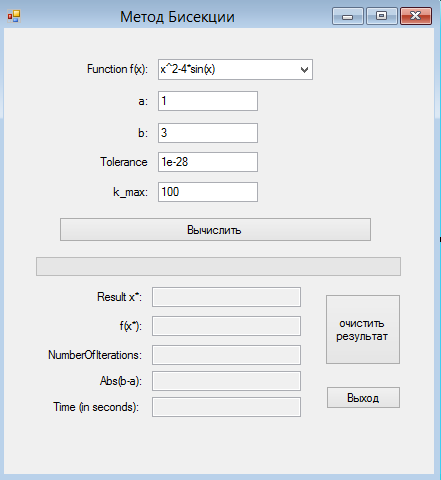
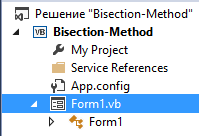


Table 2: Settings for control properties of the software system implementing Bisection method (the table is intended to document the activity of a student to settings control properties of the interface form – the table is ***mandatory***)

|  |  |  |  |
| --- | --- | --- | --- |
| **Number of control** | **Control** | **Property** | **Setting** |
| 1 | Label1 | Appearance (Text) | Function f(x) |
| Label1 | Design (Name) | Label1 |
| 2 | ComboBox1 | Appearance (Text) | x^2-4\*sin(x) |
| ComboBox1 | Design (Name) | function\_Box |
| 3 | Label9 | Appearance (Text) | a: |
| Label9 | Design (Name) | Label9 |
| 4 | Textbox2 | Appearance (Text) | 1 |
| Textbox2 | Design (Name) | a\_Box |
| 5 | Label8 | Appearance (Text) | b: |
| Label8 | Design (Name) | Label8 |
| 6 | Textbox3 | Appearance (Text) | 3 |
| Textbox3 | Design (Name) | b\_Box |
| 7 | Label2 | Appearance (Text) | Tolerance |
| Label2 | Design (Name) | Label2 |
| 8 | Textbox4 | Appearance (Text) | 1e-28 |
| Textbox4 | Design (Name) | Tol\_Box |
| 9 | Label7 | Appearance(Text) | k\_max |
| Label7 | Design(Name) | Label7 |
| 10 | TextBox5 | Appearance (Text) | 100 |
| TextBox5 | Design (Name) | k\_max\_Box |
| 11 | Label11 | Appearance(Text) | Time limit |
| Label11 | Design(Name) | Label11 |
| 12 | TextBox6 | Appearance (Text) | 30 |
| TextBox6 | Design (Name) | Time\_max\_Box |
| 13 | Button1 | Appearance (Text) | Вычислить |
| Button1 | Design (Name) | ButtonRun |
| 14 | ProgressBar1 | Behavior (Visible) | false |
| ProgressBar1 | Design (Name) | ProgressBar1 |
| 15 | Label6 | Appearance (Text) | Results x\*: |
| Label6 | Design (Name) | Label6 |
| 16 | Textbox7 | Design (Name) | resultX\_Box |
| Textbox7 | **Behavior (ReadOnly)** | True |
| 17 | Label5 | Appearance (Text) | f(x\*) |
| Label5 | Design (Name) | Label5 |
| 18 | Textbox8 | **Behavior (ReadOnly)** | True |
| Textbox8 | Design (Name) | resultF\_Box |
| 19 | Label4 | Appearance (Text) | NumberOfIterations: |
| Label4 | Design (Name) | Label9 |
| 20 | Textbox9 | **Behavior (ReadOnly)** | True |
| Textbox9 | Design (Name) | k\_Box |
| 21 | Label3 | Appearance (Text) | Abs(b-a): |
| Label3 | Design (Name) | Label3 |
| 22 | Textbox10 | **Behavior (ReadOnly)** | True |
| Textbox10 | Design (Name) | Abs\_Box |
| 23 | Label10 | Appearance(Text) | Time: |
| Label10 | Design (Name) | Label10 |
| 24 | Textbox11 | **Behavior (ReadOnly)** | true |
| Textbox11 | Design (Name) | Time\_Box |
| 25 | Button2 | Appearance (Text) | Очистить |
| Button2 | Design (Name) | ButtonClearOutPut |
| 26 | Button3 | Appearance (Text) | Выход |
| Button3 | Design (Name) | ButtonExit |

# Глава 6: Стадии конструирования системы для поиска корня нелинейного уравнения f(x)=0, реализующей Bisection Method:

Stage №.1: Код программы на Visual Basic, ассоцированный с интерфейсной формой “Form1.vb”



Imports info.lundin.Math

Imports System.Math

Imports System.Windows.Markup

Imports System.Threading

Public Class Form1

Dim SW As New Stopwatch

Dim f As String

Function Func(a As Decimal)

Dim parser As New ExpressionParser()

parser.Values.Add("x", a)

Return parser.Parse(f)

End Function

Private Sub calc()

SW.Reset()

SW.Start()

Dim a, b, m, tol1, f1, f2, f3, tol As Decimal

Dim k\_max, k\_max2, time\_max, time As Integer

Dim msg, title As String

Dim style As MsgBoxStyle

Dim response As MsgBoxResult

time = Double.Parse(Time\_max\_Box.Text)

f = function\_Box.Text 'это для проверки правильно ли введена функция'

' MsgBox("f = " & f)

a = Decimal.Parse(a\_Box.Text)

b = Decimal.Parse(b\_Box.Text)

tol1 = Decimal.Parse(Tol\_Box.Text, System.Globalization.NumberStyles.Float)

k\_max = Integer.Parse(k\_max\_Box.Text)

k\_max2 = k\_max

time\_max = time

ProgressBar1.Visible = True

tol = tol1

f1 = Func(a)

f2 = Func(b)

If (Math.Sign(f1) = Math.Sign(f2)) Then

MessageBox.Show("F(a) and F(b) must be opposite! Check [a,b]")

Else

Dim k As Integer

k = 0

While (k < k\_max And b - a > tol)

m = a + ((b - a) / 2)

f1 = Func(a)

f3 = Func(m)

If (Sign(f1) = Sign(f3)) Then

a += 1

a = m

Else

b += 1

b = m

End If

k = k + 1

If (k = k\_max And tol > 1.0E-29) Then

msg = "Не хватает итерации для точного решения, добавить итерацию?"

style = MsgBoxStyle.DefaultButton2 Or MsgBoxStyle.Critical Or MsgBoxStyle.YesNo

title = "MsgBox Demonstaration"

response = MsgBox(msg, style, title)

If response = MsgBoxResult.Yes Then

k\_max += k\_max2

End If

End If

If (ProgressBar1.Value <> 100) Then

ProgressBar1.Value += 1

End If

If (time\_max <= SW.ElapsedMilliseconds) Then

SW.Stop()

msg = "Не хватает времени, добавить время?"

style = MsgBoxStyle.DefaultButton2 Or MsgBoxStyle.Critical Or MsgBoxStyle.YesNo

title = "MsgBox Demonstaration"

response = MsgBox(msg, style, title)

If response = MsgBoxResult.Yes Then

SW.Start()

time\_max = time\_max + 10000

Else

Exit While

End If

End If

End While

If (ProgressBar1.Value <> 100) Then

ProgressBar1.Value = 100

SW.Stop()

End If

If (Abs(b - a) > tol) Then

If (k >= k\_max) Then

Label11.Text = ("Программа не может вычислить с погрешностью " & Tol\_Box.Text & vbCr & "при заданном k\_max")

Else

Label11.Text = ("Программа не может вычислить с погрешностью " & Tol\_Box.Text & vbCr & "при заданном Time limit")

End If

End If

SW.Stop()

ProgressBar1.Value = 100

resultX\_Box.Text = m

Abs\_Box.Text = Convert.ToDouble(Abs(b - a))

k\_Box.Text = k

resultF\_Box.Text = Func(m)

Time\_Box.Text = SW.ElapsedMilliseconds

End If

End Sub

Private Sub ButtonRun\_Click(sender As Object, e As EventArgs) Handles ButtonRun.Click

Label11.Text = ""

Try

If (function\_Box.Text = "" Or a\_Box.Text = "" Or b\_Box.Text = "" Or Tol\_Box.Text = "" Or k\_max\_Box.Text = "" Or Time\_max\_Box.Text = "") Then MsgBox("Заполните все поля!") : Exit Sub

calc()

ProgressBar1.Visible = False

Catch ex As Exception

MsgBox(ex.Message & " Проверьте правильность ввода Function f(x):")

End Try

End Sub

Private Sub ButtonClearOutPut\_Click(sender As Object, e As EventArgs) Handles ButtonClearOutPut.Click

resultX\_Box.Text = ""

resultF\_Box.Text = ""

k\_Box.Text = ""

Abs\_Box.Text = ""

Time\_Box.Text = ""

Label11.Text = ""

ProgressBar1.Value = 0

SW.Reset()

End Sub

Private Sub ButtonExit\_Click(sender As Object, e As EventArgs) Handles ButtonExit.Click

Me.Close()

End Sub

Private Sub a\_Box\_KeyPress(sender As Object, e As KeyPressEventArgs) Handles a\_Box.KeyPress

If a\_Box.SelectionStart = 0 Then

If Not Char.IsDigit(e.KeyChar) And Asc(e.KeyChar) <> 8 And e.KeyChar <> "-" And Asc(e.KeyChar) <> 44 And Asc(e.KeyChar) <> 46 Then

e.Handled = True

End If

Else

If Not Char.IsDigit(e.KeyChar) And Asc(e.KeyChar) <> 8 And Asc(e.KeyChar) <> 44 And Asc(e.KeyChar) <> 46 Then

e.Handled = True

End If

End If

If InStr(a\_Box.Text, ".") Then

If Not Char.IsDigit(e.KeyChar) And Asc(e.KeyChar) <> 8 And e.KeyChar <> "-" Then

e.Handled = True

End If

End If

End Sub

Private Sub b\_Box\_KeyPress(sender As Object, e As KeyPressEventArgs) Handles b\_Box.KeyPress

If b\_Box.SelectionStart = 0 Then

If Not Char.IsDigit(e.KeyChar) And Asc(e.KeyChar) <> 8 And e.KeyChar <> "-" And Asc(e.KeyChar) <> 44 And Asc(e.KeyChar) <> 46 Then

e.Handled = True

End If

Else

If Not Char.IsDigit(e.KeyChar) And Asc(e.KeyChar) <> 8 And Asc(e.KeyChar) <> 44 And Asc(e.KeyChar) <> 46 Then

e.Handled = True

End If

End If

If InStr(b\_Box.Text, ".") Then

If Not Char.IsDigit(e.KeyChar) And Asc(e.KeyChar) <> 8 And e.KeyChar <> "-" Then

e.Handled = True

End If

End If

End Sub

Private Sub k\_max\_Box\_KeyPress(sender As Object, e As KeyPressEventArgs) Handles k\_max\_Box.KeyPress

If Not Char.IsDigit(e.KeyChar) And Asc(e.KeyChar) <> 8 And Asc(e.KeyChar) <> 48 And Asc(e.KeyChar) <> 48 Then

e.Handled = True

End If

End Sub

End Class

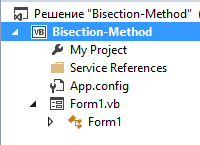
[GoTo Contents](#Contents)

Stage No.3: Подключение библиотечной программы “info.lundin.math.dll” к программному проекту для выполнения функции парсинга

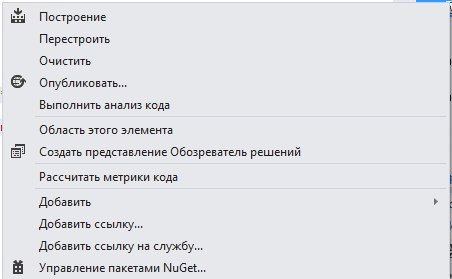
***Примечание***: *Если вставить приведенные в этом отчете коды программы, то Visual Studio 2013 выделит строки листинга программы, в которых есть ссылки на библиотечную функцию «info.lundin.math», как ошибки. Это связано с тем, что в проект не включена ссылка на эту функцию. Если у вас нету этой библиотеки вам необходимо его скачать с интернета. Ниже приведена инструкция по включению в проект парсера.*

Шаг №3.1: Правой кнопкой мыши открыть контекстное меню на выделенной темным цветом строке:

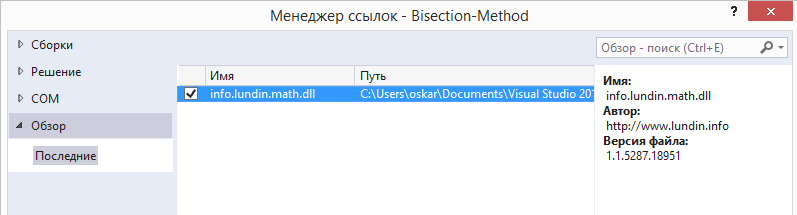
|  |  |
| --- | --- |
| VB | WindowsApplication10 |



Шаг №3.2: Щелкнуть мышкой на строчке «Добавить ссылку...»:



Шаг №3.3: В диалоговом окне менеджера ссылок ввести имя файла “info.lundin.math.dll”:



Шаг №3.4: В листинге программы исчезнут все пометки об ошибках в коде программы, связанных с тем, что оператор «Imports info.lundin.math» был неопределен, если нижеприведенные строки кода программы были уже введены до введения ссылки на эту библиотечную функцию. Если же эти строки кода не были до сих пор введены, то теперь можно ввести эти коды, в которых используется функция парсинга, как это показано ниже:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*это примеры использования парсинга “info.lundin.math.dll”

Function F(a As Double)

Dim Parser As New ExpressionParser()

Parser.Values.Add("x", a)

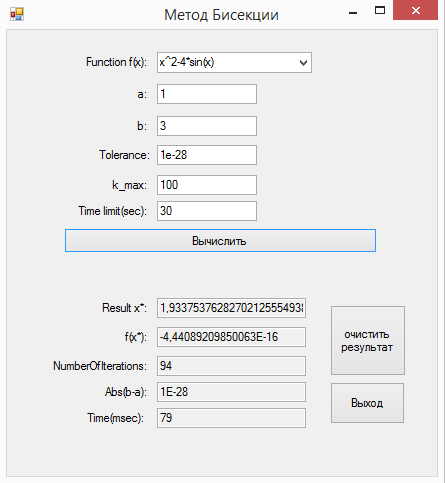
Return Parser.Parse(func)

End Function

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*это примеры использования парсинга “info.lundin.math.dll”

[GoTo Contents](#Contents)

Результат работы программы, реализующей Bisection method:



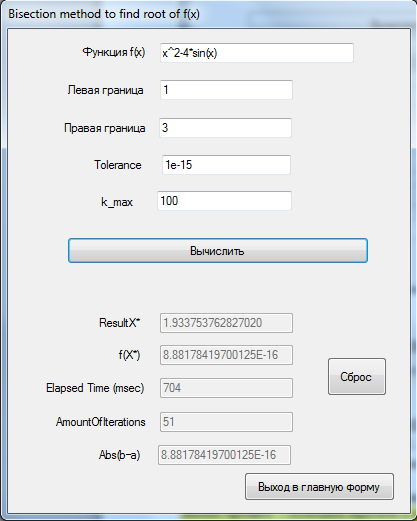
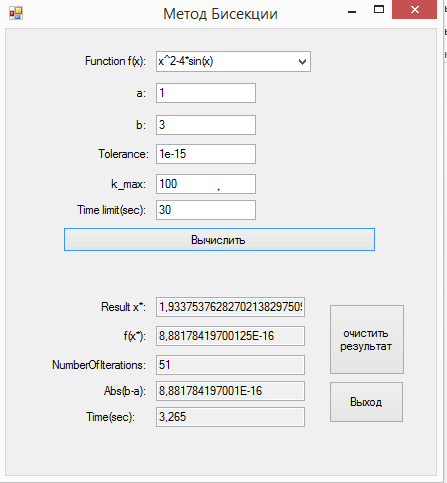
[GoTo Contents](#Contents)

# Глава 7. Тестирование

**Тест №1:**

Проверка функции f(x)=x^2-4\*sin(x) со значениями a=1, b=3, tolerance=1e-15 на валидность работы программы. На Рис 1.1 отображены результаты программы профессора Тена И.Г., а на Рис 1.2 отображены результаты моей программы и эти результаты равны.

Рис 1.1 Рис 1.2

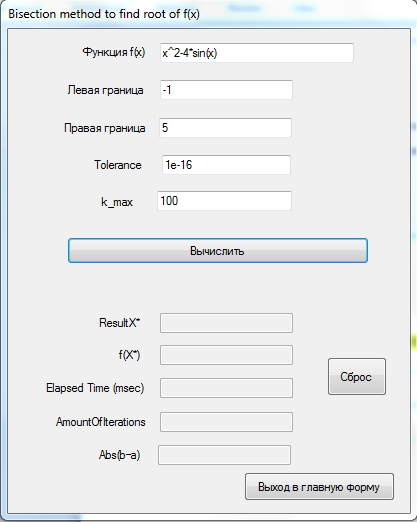
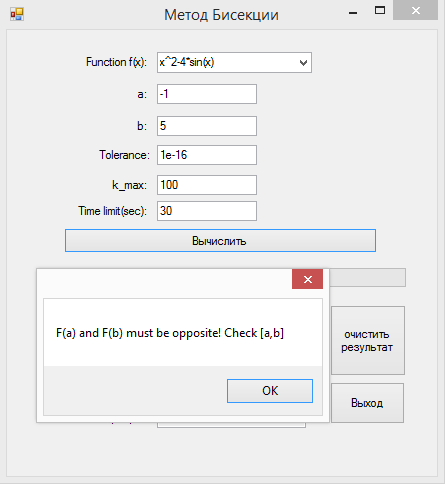
 

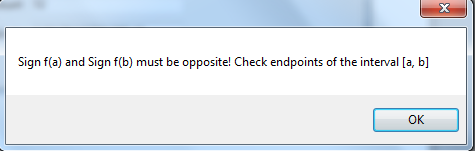
**Тест №2:**

Тест №1: Проверка функции f(x)=x^2-4\*sin(x) со значениями a=-1, b=5, tolerance=1e-16 на валидность работы программы. На Рис 2.1 отображен результат программы профессора Тена И.Г., а на Рис 2.2 отображен результат моей программы в виде ошибки “F(a) and F(b) must be opposite! Check [a, b]”.

Это ошибка сообщает что неправильно выбран интервал, выберите другой интервал.

Рис 2.1 Рис 2.2

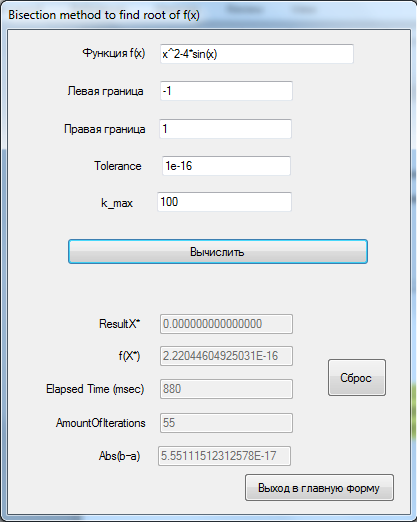
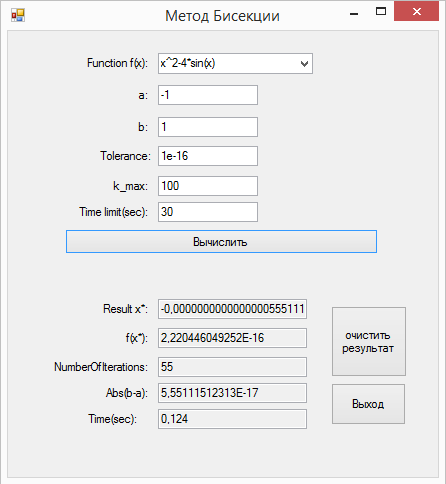
 



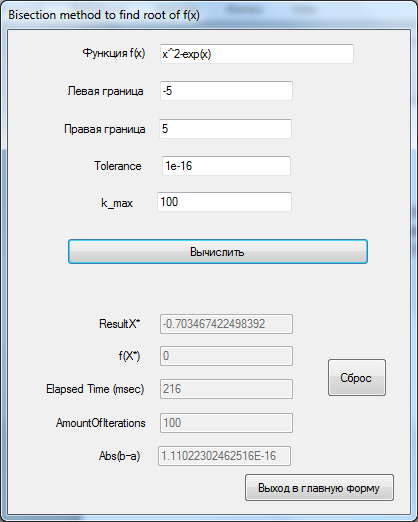
**Тест №3:**

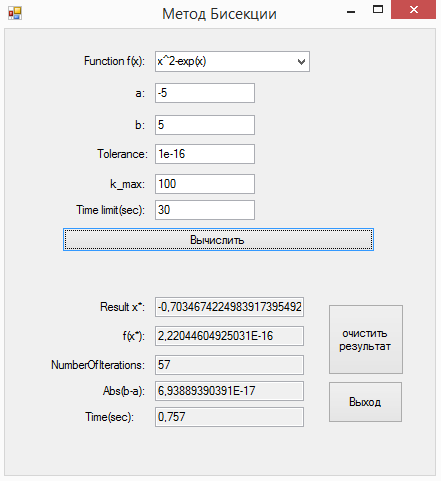
Проверка функции f(x)=x^2-4\*sin(x) со значениями a=-1, b=1, tolerance=1e-16 на валидность работы программы. На Рис 3.1 отображены результаты программы профессора Тена И.Г., а на Рис 3.2 отображены результаты моей программы и эти результаты равны.

Рис 3.1 Рис 3.2

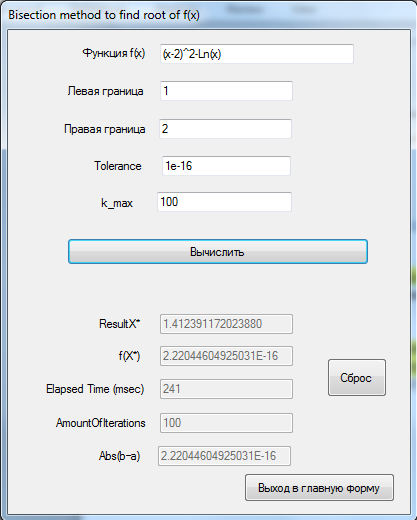
 

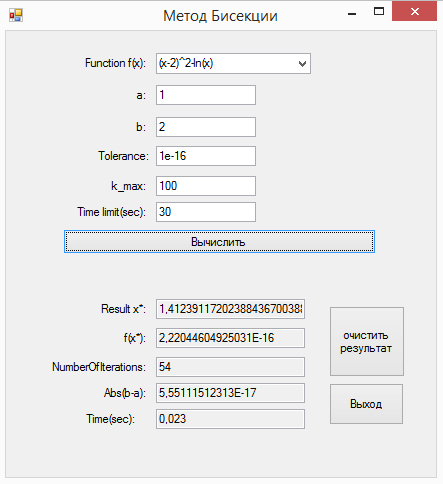
Тест №5:



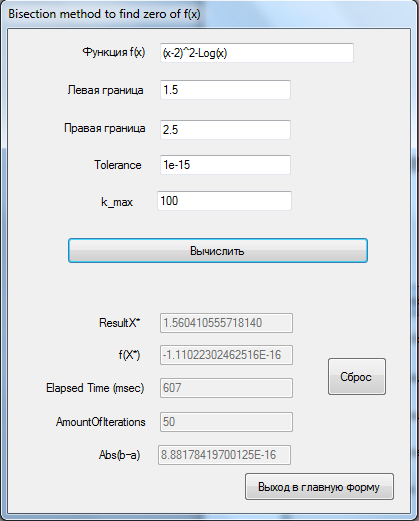


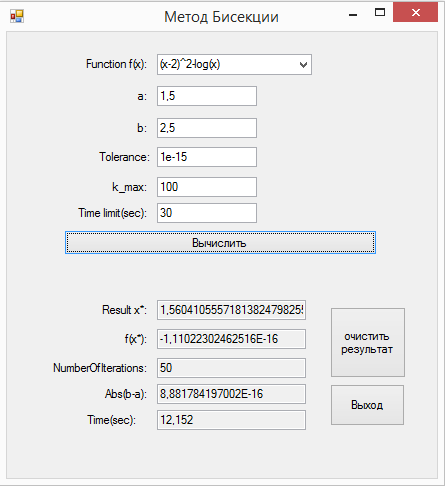
Тест №6:



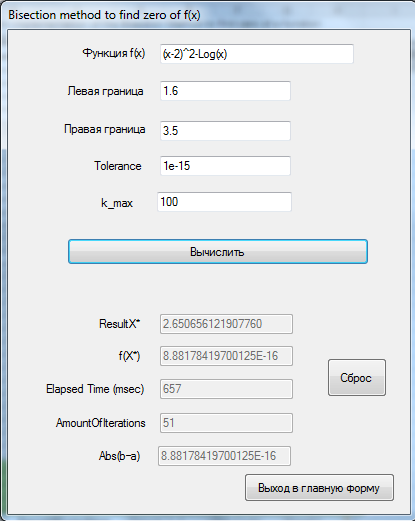


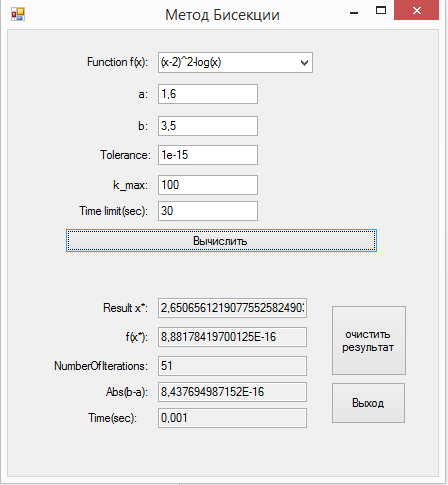
Тест №7:



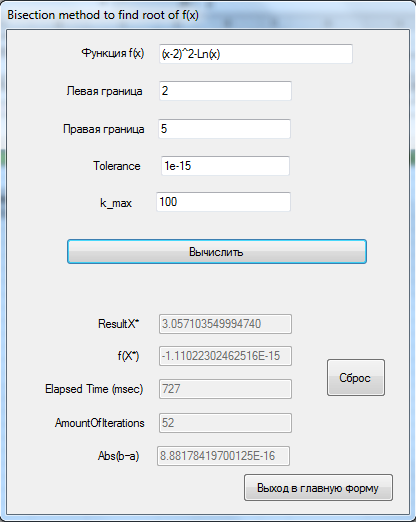


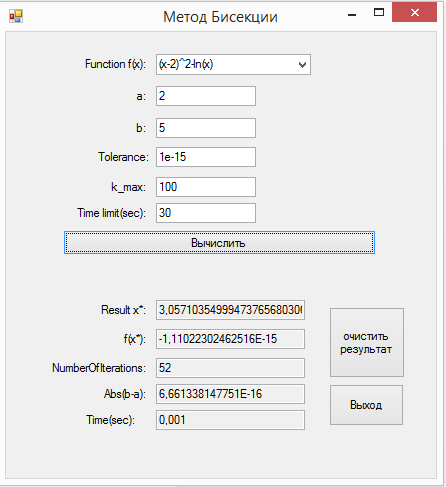
Тест №8:



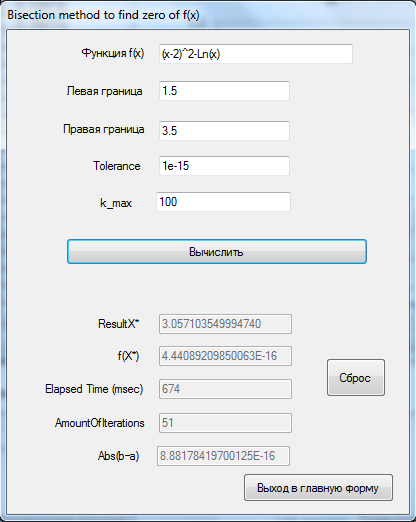


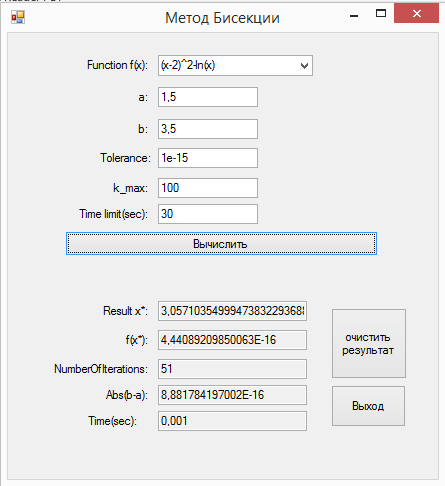
Тест №9:



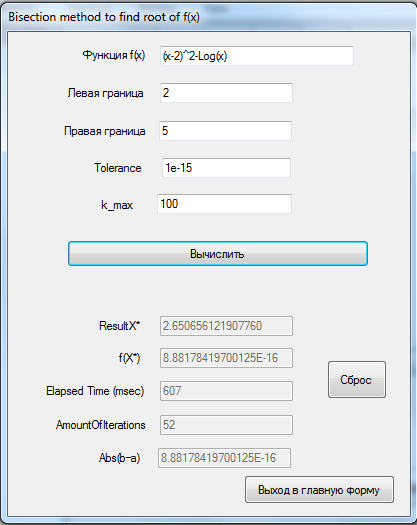


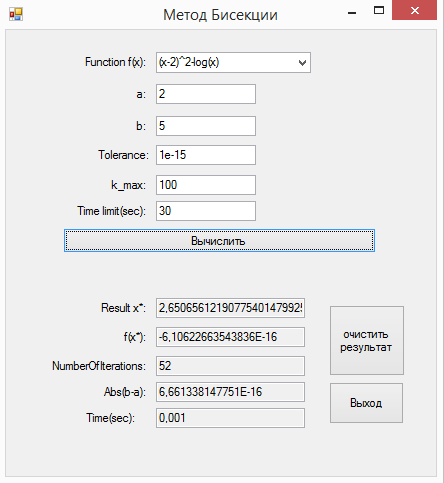
Тест №10:



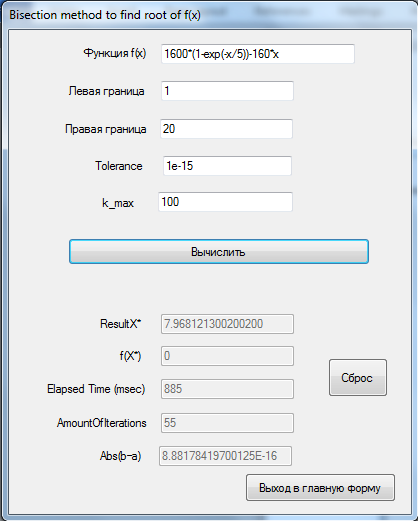


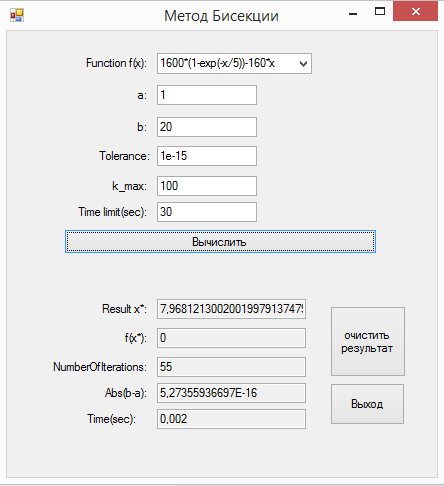
Тест №11:



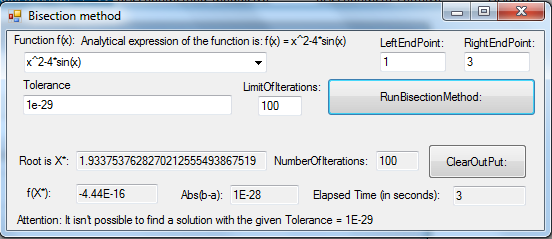


Тест №12:





Тест №13:



# References

1. Basic references
2. Wiegers K., Beatty J. Software Requirements. Third Edition. Redmond. Microsoft Press of Microsoft Corporation: Karl Wiegers and Seilevel, 2013. – 673 pages. ISBN: 978-0-7356-7966-5
3. Вигерс Карл, Битти Джой. Разработка требований к программному обеспечению. 3-е изд., дополненное /Пер. с англ. — М. : Издательство «Русская редакция» ; СПб. : БХВ-Петербург, 2014. — 736 стр. : ил. ISBN 978-5-7502-0433-5 («Русская редакция»); ISBN 978-5-9775-3348-5 («БХВ-Петербург»); УДК 004.738.5; ББК 32.973.202
4. Wiegers, Karl Eugene. Software requirements. Second edition. ISBN 0-7356-1879-8. Redmond, Washington, 2003.
5. Карл И. Вигерс. Разработка требований к программному обеспечению. — Русская редакция, 2004. — ISBN 5-7502-0240-2.
6. Кобёрн А. Современные методы описания функциональных требований к системам. — М.: Лори, 2002. — ISBN 0-201-70225-8, ISBN 5-85582-152-8.
7. Леффингуелл Д., Уидриг Д. Принципы работы с требованиями к программному обеспечению. — М.: Вильямс, 2002. — ISBN 5-8459-0275-4.
8. Alan Mark Davis. Just Enough Requirements Management: Where Software Development Meets Marketing. — Dorset House, 2005. — ISBN 978-0932633644.
9. Supplemental references
10. Douglas Bell. Software Engineering. A programming approach. 3rd Edition. Pearson Limited Edition, 2000. 470 pages. Chapter 3: Requirements engineering and analysis.
11. Тен И.Г. Методическое пособие к дипломному проектированию. Требования и состав ТЗ к дипломному проекту. Примеры реализации ТЗ. Файлы находятся по адресу: \\192.168.8.125\Buffer\ ПОКС\_ТЗдляДиплВыпПроекта\_2015
12. E-books of Software Engineering department:

**Address:** [\\192.168.8.125\books](file:///\\192.168.8.125\books)\RequirementsEngineering

[GoTo Contents](#Contents)