**Разработка и анализ требований к ПО**

**«Newton Method для поиска корня»**

Выполнил: Байгазиев Санжар

Бишкек 2018

Содержание

[Глава 1: Описание проблемы 3](#_Toc525912377)

[Глава 2: Описание Newton Method 3](#_Toc525912378)

[Глава 3: Спецификация требований к программному обеспечению 4](#_Toc525912379)

[3.1. Наименование программной разработки 4](#_Toc525912380)

[3.2. Описание бизнес-процесса и анализ расхождения 4](#_Toc525912381)

[3.3. Бизнес-цель разработки программного обеспечения 4](#_Toc525912382)

[3.4. Пользовательская история разработки программного обеспечения 4](#_Toc525912383)

[3.5. Функциональные требования к разработке программного обеспечения 5](#_Toc525912384)

[3.6. Нефункциональные требования на разработку программного обеспечения 6](#_Toc525912385)

[3.7. Ограничения на разработку программного обеспечения 6](#_Toc525912386)

[Глава 4: Стадии проектирования системы для поиска корня нелинейного уравнения f(x)=0, реализующей Newton Method: 8](#_Toc525912387)

[Глава 5: Дизайн интерфейсной формы системы, реализующей Newton method: 11](#_Toc525912388)

[Глава 6: Стадии конструирования системы для поиска корня нелинейного уравнения f(x)=0, реализующей Newton Method: 12](#_Toc525912389)

[Stage №.1: Код программы на C#, ассоцированный с интерфейсной формой “-Form1.cs” 12](#_Toc525912390)

[Stage No.2: Код программы на C#, ассоцированный с классом “NewtonMethod.cs” , который реализует логику Newton method по поиску корня нелинейного уравнения и составляет отдельный модуль в виде Public Class “ NewtonMethod ”: 16](#_Toc525912391)

[Stage No.3: Код программы на C#, ассоцированный с ***программным модулем*** “Newton Method.cs” , который реализует логику Newton method по поиску корня нелинейного уравнения и показывает код по управлению Progress Bar: 18](#_Toc525912392)

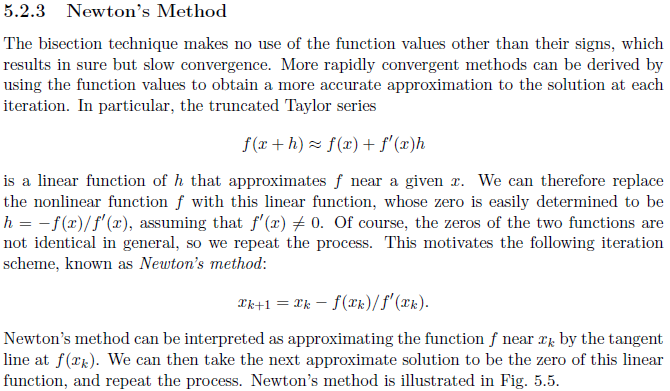
[Stage No.4: Подключение парсера 19](#_Toc525912393)

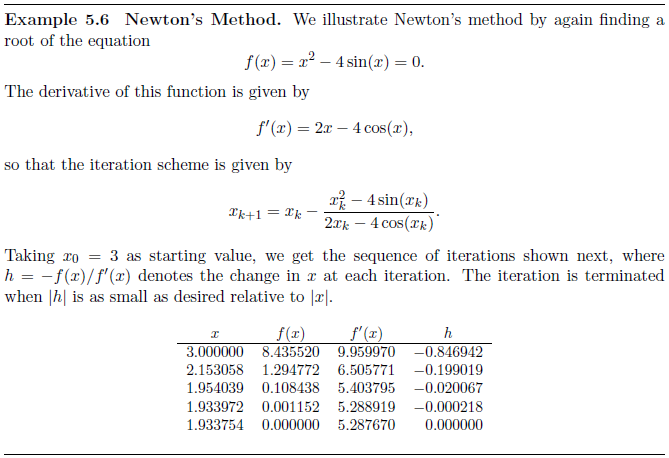
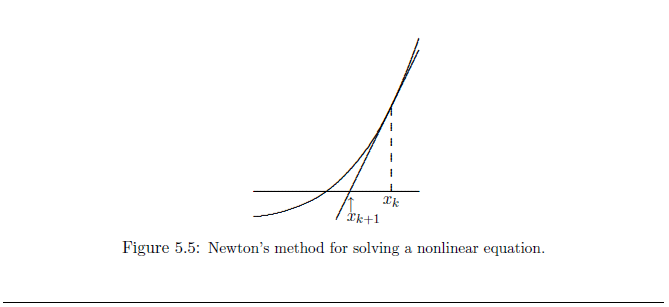
[Результат работы программы, реализующей Bisection method: 21](#_Toc525912394)

# 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 Ньютона (Newton 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.

# Глава 2: Описание Newton Method

****

****

[GoTo Contents](file:///D:\lessons\4-курс\РАТПО\Лабы\Bisection%20method_ver2\Отчет.docx#Contents)

# Глава 3: Спецификация требований к программному обеспечению

## 3.1. Наименование программной разработки

Система поиска корней нелинейного уравнения методом Ньютона

## 3.2. Описание бизнес-процесса и анализ расхождения

При решении различных бизнес-задач отделу исследований (users, clients, customers) 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 три 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.

Таким образом в отделе для достижения цели проводится следующие действия

1. формализуется задачи
2. определяется аналитический вид нелинейной функции
3. находят корня нелинейной функции

Для решения таких задач в отделе заняты три специалиста высокой квалификации и не всегда обеспечивается приемлемая точность решения задачи. После автоматизации, деятельность отдела существенно изменится, так как нахождением корня не линейного уравнения займется программа. Программа обеспечит любую допустимую погрешность до 1e-28 для решения задачи при любых видах нелинейной функции за короткое время.

## 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-28 (т.е., 0.000000000000000000000000001)***, 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)***;
* начальную точку **x**;
* значение допустимой погрешности ***Tolerance*** решения задачи;
* допустимое максимальное количество ***k\_max*** итераций.
* допустимое максимальное количество ***max\_time*** времени

***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 найти аналитическое выражение производной от f(x)
* должна провести parsing – синтаксический анализ выражения производной функции и преобразовать аналитическую запись нелинейной функции в виде символов в вычисляемую процедуру в виде подпрограммы-функции;
* должна найти решение задачи – корень нелинейного уравнения ***f(x)=0*** или ноль нелинейной функции ***f(x)*** – по алгоритму Newton Method;
* должна выводить аналитическое выражение производной функции
* должна выводить найденное значение ***X\**** корня заданного нелинейного уравнения;
* должна выводить значение заданной нелинейной функции ***f(x\*)*** в этой точке;
* должна выводить значение производной от 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 ***Newton Method для поиска корней*** нелинейного уравнения;
  + Система должна искать решение задачи по следующему алгоритму (Newton algorithm):

Do{

Xn-1 = Xn

Xn+1 = Xn – (f(Xn)/f’{Xn)

K=k+1}

While(***Abs(Xn – Xn-1)***) Tol;

* + Система должна обеспечивать ввод любых начальных числовых значений начальной точки ***[X]***;
  + Система должна обеспечить возможность продолжения поиска, если заданное время поиска ***max\_time*** истрачено и не достигнуто решение с требуемой погрешностью ***Tolerance***;
  + Система должна обеспечить возможность продолжения поиска решения задачи с последнем достигнутом на шаге ***k\_max*** значении точки ***[Xk(k\_max)]***, приняв ее за начальное значение начальной точки {т.е., 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 обеспечивать 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 ***x*** этой функции;
  + Система должна проводить ***parsing***\*\* – синтаксический анализ аналитического выражения производной функции f(x) – и автоматически конструировать подпрограмму для вычисления значения этой функции для любого значения аргумента ***x*** этой функции;
  + Система должна обеспечить ввод любого целого числового значения в качестве максимального допустимого количества итераций ***k\_max***;
  + Система должна обеспечивать возможность ввода любых числовых значений допустимой погрешности ***Tolerance*** как в формате с фиксированной точностью “0.00000000000000000000000001”, так и в экспоненциальном формате “1e-28”;
  + Система должна быть способна проверять валидность введенного аналитического выражения в качестве функции f(x) и сообщать пользователю об ошибке в аналитическом выражении функции;
  + Система должна быть способна проверять валидность формата введенных исходных числовых данных и сообщать пользователю об ошибке в формате исходных данных;
* 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 значения производной 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 вывод достигнутого значения погрешности решения задачи в качестве оценки точности этого метода (Newton Method);
  + Система должна обеспечить невозможность какого-либо ***исправления вручную*** выходных данных системы;
* Система должна обеспечить ***оценку производительности программы*** при решении задачи заданным методом (***Newton Method***) и реализации программы на языке ***C# в среде Visual Studio 2017*** путем измерения и вывода времени, затрачиваемой на поиск решения с заданной погрешностью;

## 3.6. Нефункциональные требования на разработку программного обеспечения

* Программа в процессе поиска решения должна отображать индикатор состояния процесса выполнения операции ***Progress 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, которая будет облегчать поиск источника 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;
* Сбой системы не должен происходить чаще чем один раз на тысячу запусков программы;
* Система должна обеспечить оценку качества решения задачи заданным методом (***Newton 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 в точке x0;
* Разработать ПО на основе IDE (integrated development environment – интегрированная среда разработки) – Visual Studio 2017;
* Разработать ПО на языке C#;
* Система должна решать задачи с допустимой погрешностью до 1e-28;
* Система должна решать задачи за время не более одной минуты;
* 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 10 сентября 2018 года, и система должна быть передана заказчику не позднее 1 октября 2018 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 любые аналитические выражения для нелинейной функции, которые включают следующие математические символы:

Cстандартных функций

"sqrt", "sin", "cos", "tan",

"atan", "acos", "asin", "acotan",

"exp", "ln", "log",

"sinh", "cosh", "tanh", "abs",

"ceil", "floor", "fac", "sfac", "round", "fpart"

* + 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.

# 

# Глава 4: Стадии проектирования системы для поиска корня нелинейного уравнения f(x)=0, реализующей Newton Method:

Stage No.1: Блок-схема Newton method

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



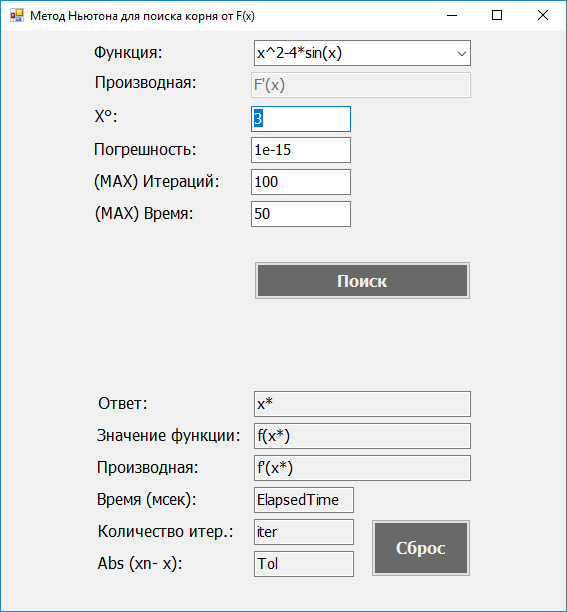
[GoTo Contents](file:///D:\lessons\4-курс\РАТПО\Лабы\Bisection%20method_ver2\Отчет.docx#Contents)

Таблица 1: Настройки для свойств элементов управления программной системой, реализующей Newton Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Number of control** | **Control** | **Property** | **Setting** |
| 1 | Label1 | Appearance (Text) | Функция |
| Label1 | Design (Name) | Function\_Label |
| 2 | Combobox1 | Appearance (items) | x^2-4\*sin(x)  x^2-exp(x)  (x-2)^2-ln(x)  (x-2)^2-log(x)  1600\*(1-exp(-x/5))-160\*x  x\*exp(-x)  4\*x^3-2\*x-6  1500\*x^3-0,001\*exp(2\*x)-0,01\*exp(z)^2 |
| Combobox1 | Design (Name) | Textbox1 |
| 3 | Label2 | Appearance (Text) | Производная |
| Label2 | Design (Name) | Derivative\_label |
| 4 | Textbox2 | Appearance (Text) | F’(x) |
| Textbox2 | Behavior (enabled) | false |
| Textbox2 | Design (Name) | Derivative\_text |
| 5 | Label3 | Appearance (Text) | X0: |
| Label3 | Design (Name) | A\_label |
| 6 | Textbox3 | Appearance (Text) | 3 |
| Textbox3 | Design (Name) | A\_textbox |
| 7 | Label4 | Appearance (Text) | Погрешность: |
| Label4 | Design (Name) | Tolerance\_label |
| 8 | Textbox4 | Appearance (Text) | 1e-15 |
| Textbox4 | Design (Name) | Tolerance\_textBox |
| 9 | Label5 | Appearance (Text) | (MAX) Итераций: |
| Label5 | Design (Name) | MAXIteration\_label |
| 10 | Textbox5 | Appearance (Text) | 100 |
| Textbox5 | Design (Name) | MAXIteration\_TextBox |
| 11 | Label11 | Design (Name) | MAXTime\_label |
| Label11 | Appearance (Text) | (MAX) Время: |
| 12 | Textbox6 | Appearance (Text) | 50 |
| Textbox6 | Design (Name) | MAXTime\_TextBox |
| 13 | Button1 | Appearance (Text) | Поиск |
| Button1 | Design (Name) | Solve\_button1 |
| 14 | ProgressBar1 | Behavior (Visible) | False |
| ProgressBar1 | Design (Name) | PB\_ProgressBar |
| 15 | Label13 | Appearance (Text) | Результат: |
| Label13 | Design (Name) | Res\_label |
| 16 | Label14 | Appearance (Text) |  |
| Label14 | Design (Name) | Message\_label |
| 17 | Label10 | Appearance (Text) | Ответ: |
| Label10 | Design (Name) | ResultX\_label |
| 18 | Textbox10 | Design (Name) | ResultX\_textBox |
| Textbox10 | Behavior (enabled) | false |
| Textbox10 | Appearance (Text) | X\* |
| 19 | Label9 | Appearance (Text) | Значение функции: |
| Label9 | Design (Name) | ResultFunction\_label |
| 20 | Textbox9 | Behavior (enabled) | false |
| Textbox9 | Design (Name) | ResultFunction\_TextBox |
| Textbox9 | Appearance (Text) | F(x\*) |
| 21 | Label11 | Appearance (Text) | Производная: |
| Label11 | Design (Name) | DerivativeFunction\_label |
| 22 | TextBox11 | Behavior (enabled) | false |
| TextBox11 | Design (Name) | DerivativeFunction \_TextBox |
| TextBox11 | Appearance (Text) | F’(x\*) |
| 23 | Label8 | Appearance (Text) | Время (мсек): |
| Label8 | Design (Name) | ResultTime\_label |
| 24 | Textbox8 | Behavior (enabled) | false |
| Textbox8 | Design (Name) | ResultTime\_TextBox |
| Textbox8 | Appearance (Text) | ElapsedTime |
| 25 | Label7 | Appearance (Text) | Количество итер.: |
| Label7 | Design (Name) | ResultIteration\_label |
| 26 | Textbox7 | Behavior (enabled) | false |
| Textbox7 | Design (Name) | ResultIteration \_TextBox |
| Textbox7 | Appearance (Text) | iter |
| 27 | Button2 | Appearance (Text) | Сброс: |
| Button2 | Design (Name) | button2 |
| 28 | Label6 | Appearance (Text) | Abs(xn-x) |
| Label6 | Design (Name) | ABS\_label |
| 29 | Textbox6 | Behavior (enabled) | false |
| Textbox6 | Design (Name) | ABS\_TextBox |
| Textbox6 | Appearance (Text) | Tol |

[GoTo Contents](file:///D:\lessons\4-курс\РАТПО\Лабы\Bisection%20method_ver2\Отчет.docx#Contents)

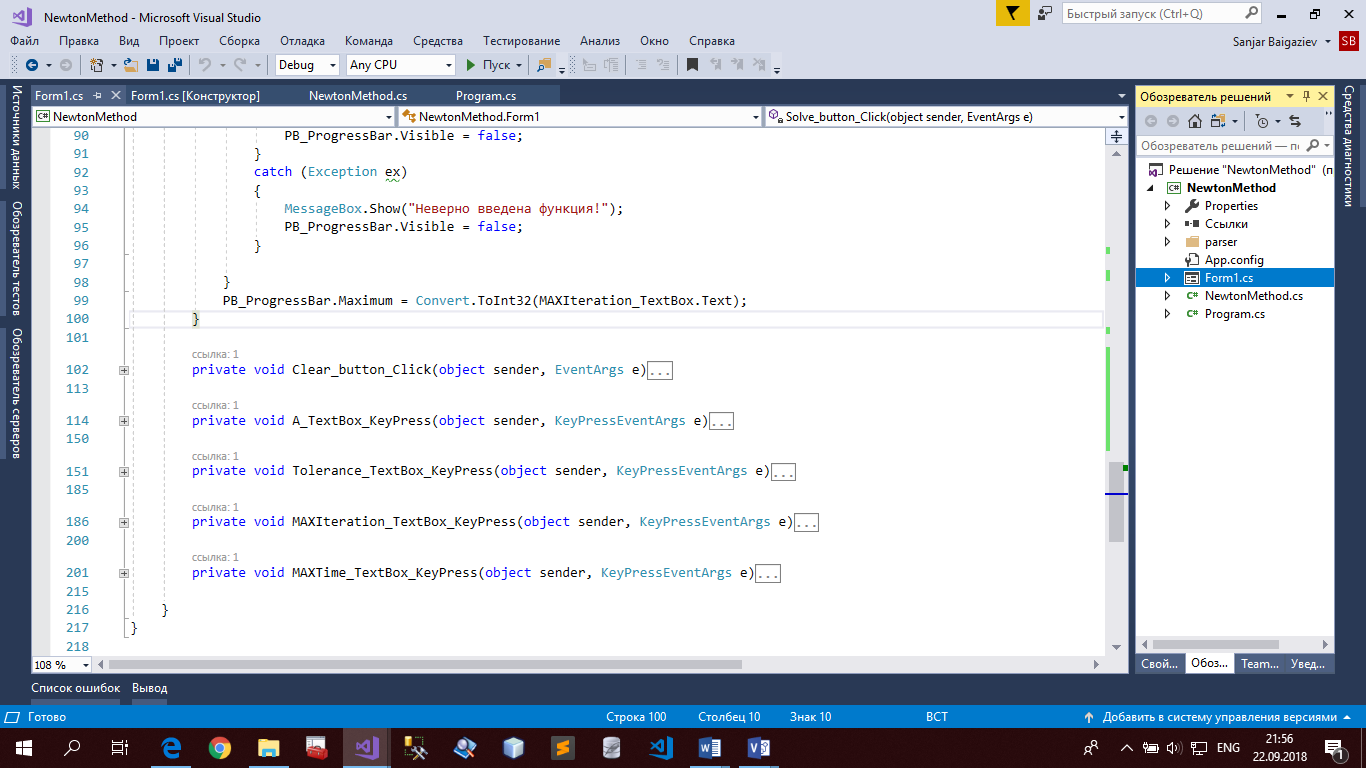
# Глава 5: Дизайн интерфейсной формы системы, реализующей Newton method:



[GoTo Contents](file:///D:\lessons\4-курс\РАТПО\Лабы\Bisection%20method_ver2\Отчет.docx#Contents)

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

## Stage №.1: Код программы на C#, ассоцированный с интерфейсной формой “-Form1.cs”



using System;

using System.Drawing;

using System.Globalization;

using System.Windows.Forms;

namespace NewtonMethod

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void Solve\_button\_Click(object sender, EventArgs e)

{

Message\_label.Text = "";

if (Function\_ComboBox.Text == "" || A\_TextBox.Text == "" || Tolerance\_TextBox.Text == "" || MAXIteration\_TextBox.Text == "" || MAXTime\_TextBox.Text == "")

{

MessageBox.Show("Не все данные введены!");

}

else

{

try

{

decimal tol1;

decimal x\_result;

decimal f\_result;

decimal iter\_result;

decimal time\_result;

decimal tol\_result;

decimal der\_result;

int error;

PB\_ProgressBar.Visible = true;

PB\_ProgressBar.Maximum = Convert.ToInt32(MAXIteration\_TextBox.Text) + 1;

NewtonMethod method = new NewtonMethod();

x\_result = method.Calculate(Function\_ComboBox, A\_TextBox, Tolerance\_TextBox, MAXIteration\_TextBox, MAXTime\_TextBox, PB\_ProgressBar, Derivative\_text);

tol1 = Decimal.Parse(Tolerance\_TextBox.Text, NumberStyles.AllowExponent | NumberStyles.AllowDecimalPoint);

f\_result = method.f\_result;

iter\_result = method.iter\_result;

time\_result = method.time\_result;

tol\_result = method.tol\_result;

der\_result = method.df\_result;

error = method.cond\_result;

ResultX\_TextBox.Text = x\_result.ToString();

ResultFunction\_TextBox.Text = f\_result.ToString();

DerivativeFunction\_TextBox.Text = der\_result.ToString();

ResultIteration\_TextBox.Text = iter\_result.ToString();

ABS\_TextBox.Text = tol\_result.ToString("0e0");

ResultTime\_TextBox.Text = time\_result.ToString();

if (iter\_result == 0)

{

Message\_label.Visible = true;

Message\_label.ForeColor = Color.Red;

Message\_label.Text = "Ошибка";

}

else

{

if (error == 4)

{

Message\_label.Visible = true;

Message\_label.ForeColor = Color.Red;

Message\_label.Text = "Ошибка";

}

else if (error == 1)

{

Message\_label.Visible = true;

Message\_label.ForeColor = Color.Red;

Message\_label.Text = "Результат с заданной точностью не достигнут \n из за лимита времени!";

}

else if (error == 2)

{

Message\_label.Visible = true;

Message\_label.ForeColor = Color.Red;

Message\_label.Text = "Результат с заданной точностью не достигнут \n из за лимита итерации!";

}

else

{

Message\_label.Visible = true;

Message\_label.ForeColor = Color.Green;

Message\_label.Text = "Результат с заданной точностью достигнут !";

}

}

}

catch (FormatException ex)

{

MessageBox.Show(ex.Message + "Неверно введены начальные значения!");

PB\_ProgressBar.Visible = false;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message + " Неверно введена функция!");

PB\_ProgressBar.Visible = false;

}

}

PB\_ProgressBar.Maximum = Convert.ToInt32(MAXIteration\_TextBox.Text);

}

private void Clear\_button\_Click(object sender, EventArgs e)

{

Message\_label.Text = "";

ResultX\_TextBox.Text = "";

ResultFunction\_TextBox.Text = "";

DerivativeFunction\_TextBox.Text = "";

ResultTime\_TextBox.Text = "";

ResultIteration\_TextBox.Text = "";

ABS\_TextBox.Text = "";

Derivative\_text.Text = "";

}

private void A\_TextBox\_KeyPress(object sender, KeyPressEventArgs e)

{

if ((e.KeyChar >= '0') && (e.KeyChar <= '9'))

{

return;

}

if (e.KeyChar == '.')

{

e.KeyChar = ',';

}

if (e.KeyChar == ',')

{

if (A\_TextBox.Text.IndexOf(',') != -1)

{

e.Handled = true;

}

return;

}

if (e.KeyChar == '-')

{

if (A\_TextBox.Text.IndexOf('-') != -1)

{

e.Handled = true;

}

return;

}

if (Char.IsControl(e.KeyChar))

{

return;

}

e.Handled = true;

}

private void Tolerance\_TextBox\_KeyPress(object sender, KeyPressEventArgs e)

{

if ((e.KeyChar >= '0') && (e.KeyChar <= '9'))

{

return;

}

if (e.KeyChar == 'e')

{

return;

}

if (e.KeyChar == '-')

{

return;

}

if (e.KeyChar == '.')

{

e.KeyChar = ',';

}

if (e.KeyChar == ',')

{

if (Tolerance\_TextBox.Text.IndexOf(',') != -1)

{

e.Handled = true;

}

return;

}

if (Char.IsControl(e.KeyChar))

{

return;

}

e.Handled = true;

}

private void MAXIteration\_TextBox\_KeyPress(object sender, KeyPressEventArgs e)

{

if ((e.KeyChar >= '0') && (e.KeyChar <= '9'))

{

return;

}

if (Char.IsControl(e.KeyChar))

{

return;

}

e.Handled = true;

}

private void MAXTime\_TextBox\_KeyPress(object sender, KeyPressEventArgs e)

{

if ((e.KeyChar >= '0') && (e.KeyChar <= '9'))

{

return;

}

if (Char.IsControl(e.KeyChar))

{

return;

}

e.Handled = true;

}

private void Form1\_Load(object sender, EventArgs e)

{

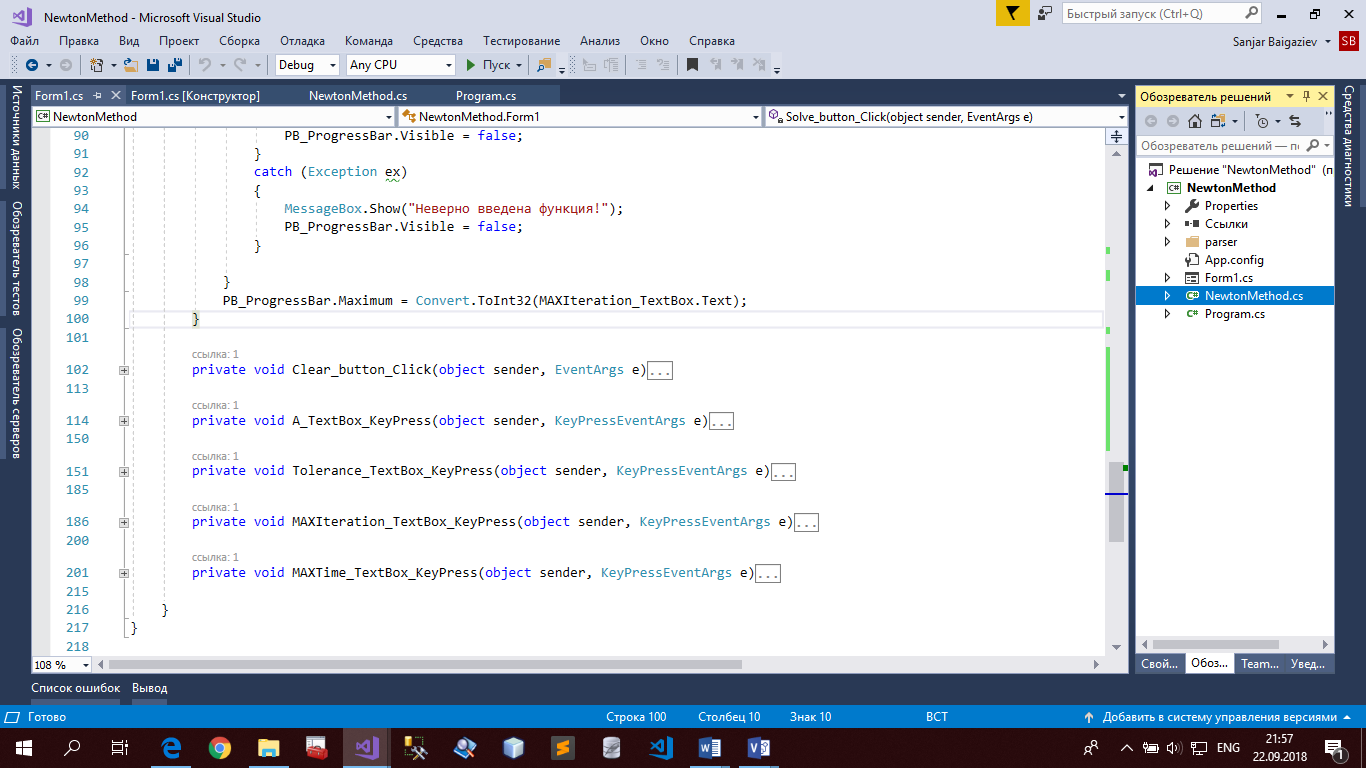
}

}

}

[GoTo Contents](file:///D:\\lessons\\4-курс\\РАТПО\\Лабы\\Bisection%20method_ver2\\Отчет.docx" \l "Contents)

## Stage No.2: Код программы на C#, ассоцированный с классом “NewtonMethod.cs” , который реализует логику Newton method по поиску корня нелинейного уравнения и составляет отдельный модуль в виде Public Class “ NewtonMethod ”:



using parserDecimal.Parser;

using System;

using System.Diagnostics;

using System.Globalization;

using System.Windows.Forms;

namespace NewtonMethod

{

class NewtonMethod

{

public decimal x\_result { get; set; }

public decimal f\_result { get; set; }

public decimal df\_result { get; set; }

public decimal iter\_result { get; set; }

public int time\_result { get; set; }

public decimal tol\_result { get; set; }

public int cond\_result { get; set; }

Computer f\_parser;

Derivative df\_parser;

public decimal Fx(string F, decimal v)

{

f\_parser = new Computer();

return f\_parser.Compute(F, v);

}

public string Fd(string F)

{

df\_parser = new Derivative();

return df\_parser.ReturnDerivative(F);

}

public decimal Calculate(ComboBox func\_text, TextBox x\_text, TextBox tol\_text, TextBox maxiter\_text, TextBox maxtime\_text, ProgressBar pb,TextBox Der)

{

Stopwatch stopwatch = new Stopwatch();

int iter;

string FD\_text;

decimal x, xn;

decimal f, fd;

int cond=3; //cond=1: Остановка по времени, cond=2; Остановка по итерации, cond=3: Решения найдена с заданной точностью, cond=4: деление на ноль

x = Convert.ToDecimal(x\_text.Text);

string F\_text = func\_text.Text;

decimal Tol = Decimal.Parse(tol\_text.Text, NumberStyles.AllowExponent | NumberStyles.AllowDecimalPoint); ;

int max\_iter = Convert.ToInt16(maxiter\_text.Text);

int max\_iter2 = max\_iter;

int max\_time = Convert.ToInt16(maxtime\_text.Text);

int max\_time2 = max\_time;

pb.Value = 0;

FD\_text = Fd(F\_text);

decimal h;

xn = x;

f = Fx(F\_text, xn);

fd = Fx(FD\_text, xn);

h = (f / fd);

x = xn - h;

iter = 0;

stopwatch.Start();

while (Math.Abs(xn - x) >= Tol)

{

if (max\_time <= Convert.ToInt16(stopwatch.ElapsedMilliseconds)) //Проверка на время

{

stopwatch.Stop();

pb.Value = pb.Maximum;

DialogResult dr = MessageBox.Show("Время вышло. Продолжить вычисление? Будет добавлено " + max\_time2 + " миллисекунд", "Продолжить вычисления?", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (dr == DialogResult.No)

{

cond = 1;

break;

}

else if (dr == DialogResult.Yes)

{

max\_time += max\_time2;

maxtime\_text.Text = max\_time.ToString();

stopwatch.Start();

}

}

if (iter >= max\_iter) //проверка на итерацию

{

stopwatch.Stop();

DialogResult dr = MessageBox.Show("Указанная точность за " + iter + " итераций не достигнута.Продолжить вычисление ? Будет добавлено " + max\_iter + " итераций.", "Продолжить вычисления ? ", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (dr == DialogResult.No)

{

cond = 2;

break;

}

else if (dr == DialogResult.Yes)

{

max\_iter = max\_iter + max\_iter2;

pb.Value = pb.Maximum;

maxiter\_text.Text = (max\_iter).ToString();

stopwatch.Start();

}

}

xn = x;

f = Fx(F\_text, xn);

fd = Fx(FD\_text, xn);

if (fd == 0)

{

stopwatch.Stop();

cond = 4;

break;

}

else

{

h = (f / fd);

x = xn - h;

iter++;

pb.Maximum = iter + 1;

pb.Value = iter;

}

}

stopwatch.Stop();

cond\_result = cond;

Der.Text = FD\_text;

f\_result = Fx(F\_text, x);

df\_result = Fx(FD\_text, x);

tol\_result = Math.Abs(xn - x);

iter\_result = iter;

time\_result = Convert.ToInt16(stopwatch.ElapsedMilliseconds);

pb.Value = pb.Maximum;

pb.Visible = false;

return x;

}

}

}

[GoTo Contents](file:///D:\lessons\4-курс\РАТПО\Лабы\Bisection%20method_ver2\Отчет.docx#Contents)

## Stage No.3: Код программы на C#, ассоцированный с ***программным модулем*** “Newton Method.cs” , который реализует логику Newton method по поиску корня нелинейного уравнения и показывает код по управлению Progress Bar:

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Здесь в коде реализована логика Newton Method and Progress Bar**

while (cond==0)

{

xn = Convert.ToDecimal(x);

f = Convert.ToDecimal(Fx(F\_text, xn));

fd = Convert.ToDecimal(Fx(FD\_text, xn));

if (Math.Abs(fd) <= Tol)

{

stopwatch.Stop();

cond = 4;

//break;

}

else

{

h = Convert.ToDecimal((f / fd));

x = Convert.ToDecimal(xn - h);

iter++;

pb.Maximum = iter + 1;

pb.Value = iter;

if (max\_time <= Convert.ToInt16(stopwatch.ElapsedMilliseconds)) //Проверка на время

{

stopwatch.Stop();

pb.Value = pb.Maximum;

DialogResult dr = MessageBox.Show("Время вышло. Продолжить вычисление? Будет добавлено " + max\_time2 + " миллисекунд", "Продолжить вычисления?", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (dr == DialogResult.No)

{

//tol2 = 0.0000000000000000000000000001m;

tol2 = Math.Abs(xn - x);

cond = 1;

}

else if (dr == DialogResult.Yes)

{

max\_time += max\_time2;

maxtime\_text.Text = max\_time.ToString();

stopwatch.Start();

}

}

if (iter >= max\_iter) //проверка на итерацию

{

stopwatch.Stop();

DialogResult dr = MessageBox.Show("Указанная точность за " + iter + " итераций не достигнута.Продолжить вычисление ? Будет добавлено " + max\_iter + " итераций.", "Продолжить вычисления ? ", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (dr == DialogResult.No)

{

//tol2 = 0.0000000000000000000000000001m;

tol2 = Math.Abs(xn - x);

cond = 2;

}

else if (dr == DialogResult.Yes)

{

max\_iter = max\_iter + max\_iter2;

pb.Value = pb.Maximum;

maxiter\_text.Text = (max\_iter).ToString();

stopwatch.Start();

}

}

}

if (Convert.ToDecimal(Math.Abs(xn - x)) <= Tol)

{

cond = 3;

tol2 = Math.Abs(xn - x);

}

}

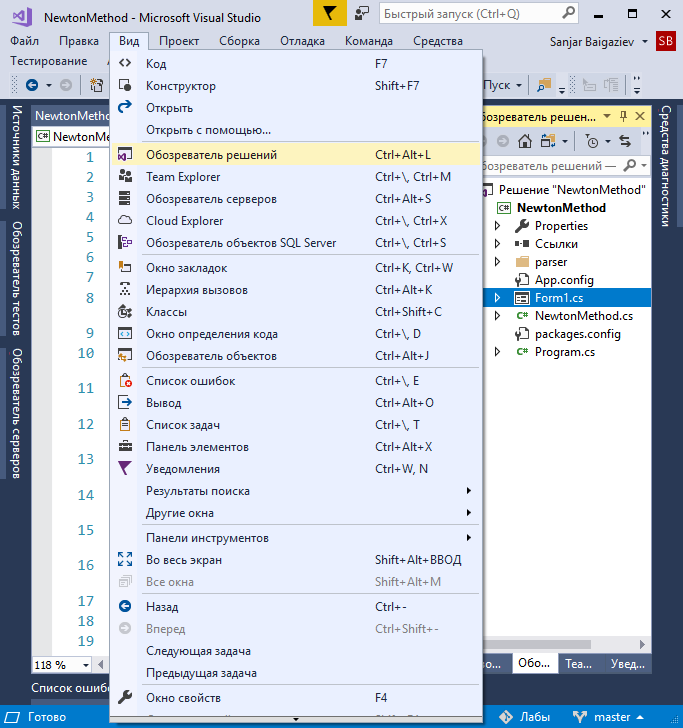
## Stage No.4: Подключение парсера

***Примечание***: *Если вставить приведенные в этом отчете коды программы, то Visual Studio выделит строки листинга программы, в которых обявлены обьекты следующих классов*

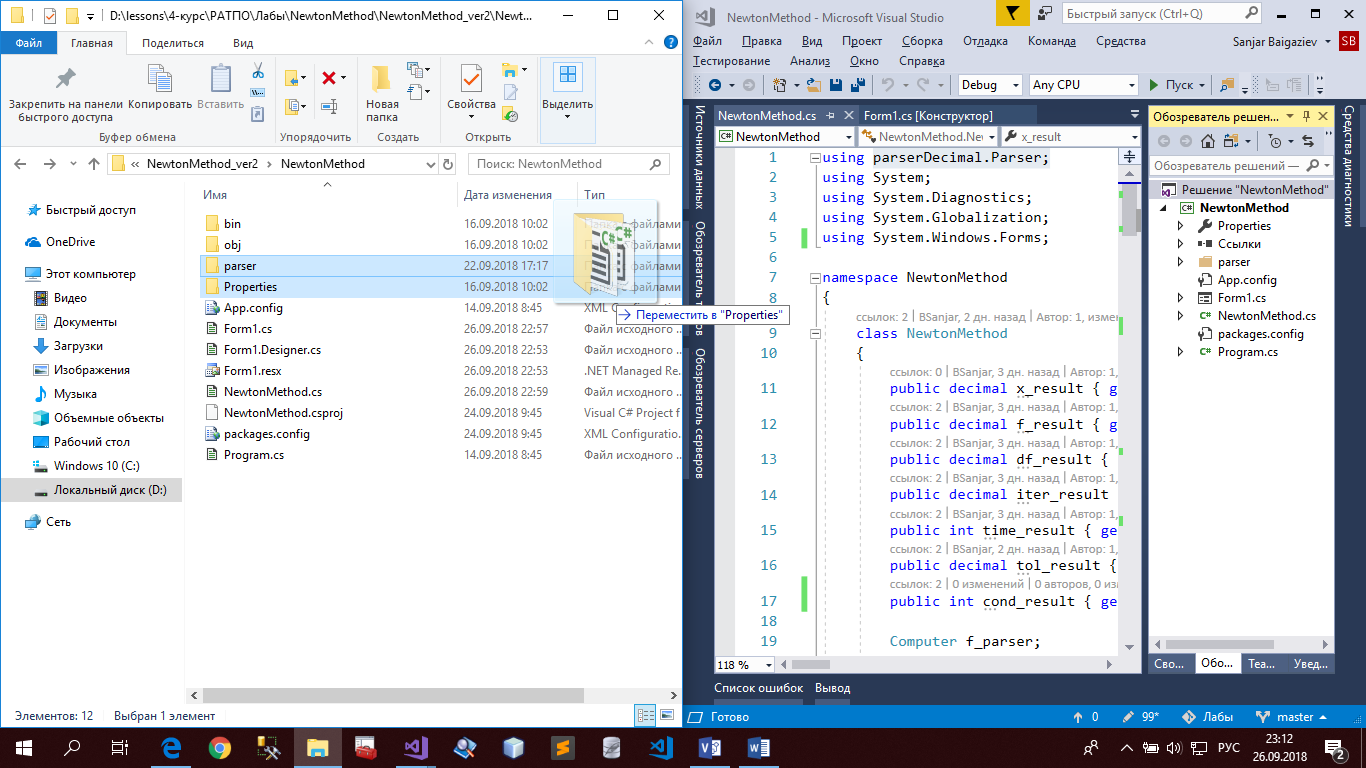
1. Computer f\_parser;
2. Derivative df\_parser;

Чтобы заработала программа, надо Добавить папку parser с классами который содержит AziretParser - parserDecimal.Parser;

Шаг №4.1: Откроем «обозреватель решений»



Шаг №4.2: Перетаскиваем папку с классами в наш проект «NewtonMethod»



Шаг №4.3: В листинге программы исчезнут все пометки об ошибках в коде программы, связанных с тем, что оператор «using parserDecimal.Parser» 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 “parserDecimal.Parser”

Computer f\_parser;

Derivative df\_parser;

public decimal Fx(string F, decimal v)

{

f\_parser = new Computer();

return f\_parser.Compute(F, v);

}

public string Fd(string F)

{

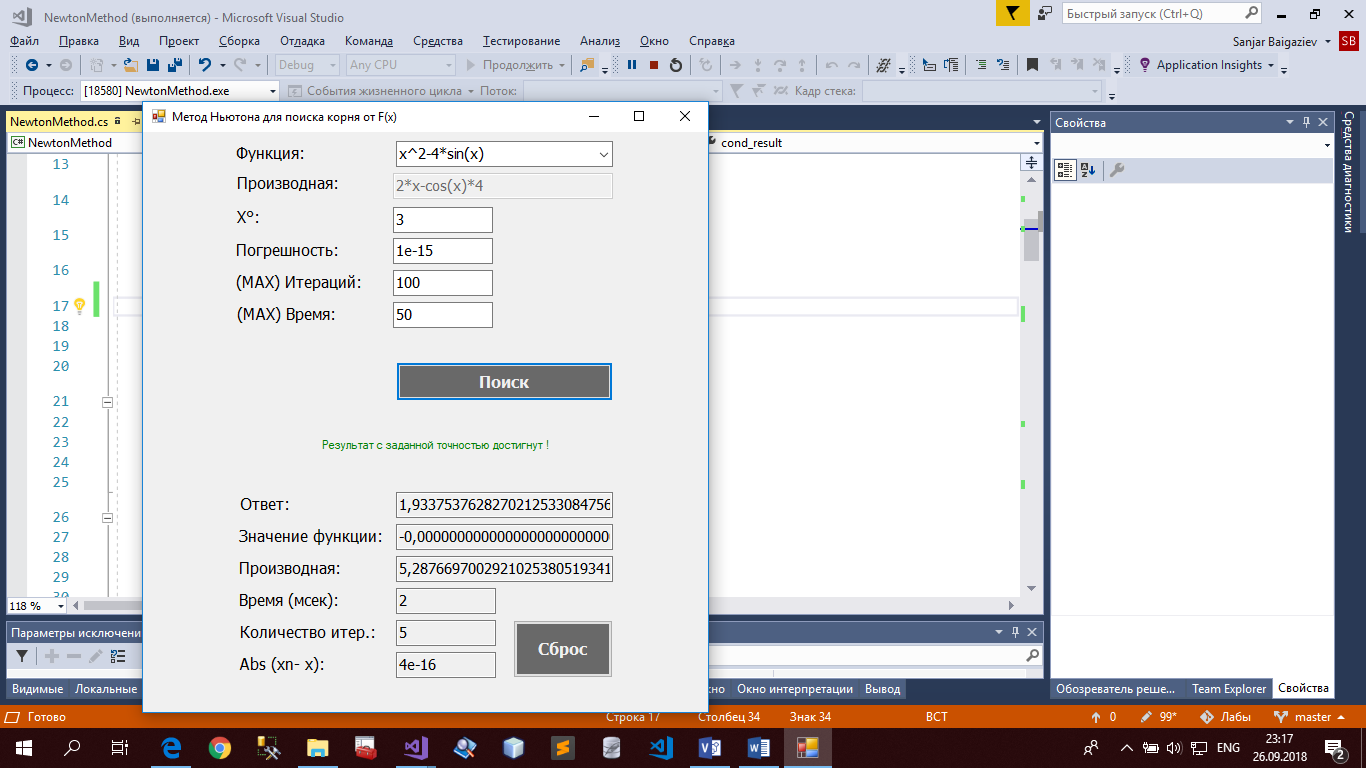
df\_parser = new Derivative();

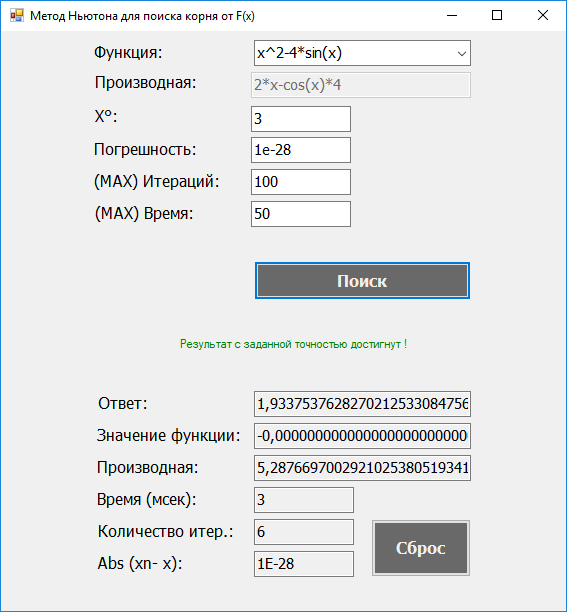
return df\_parser.ReturnDerivative(F);

} \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*это примеры использования парсинга “parserDecimal.Parser”

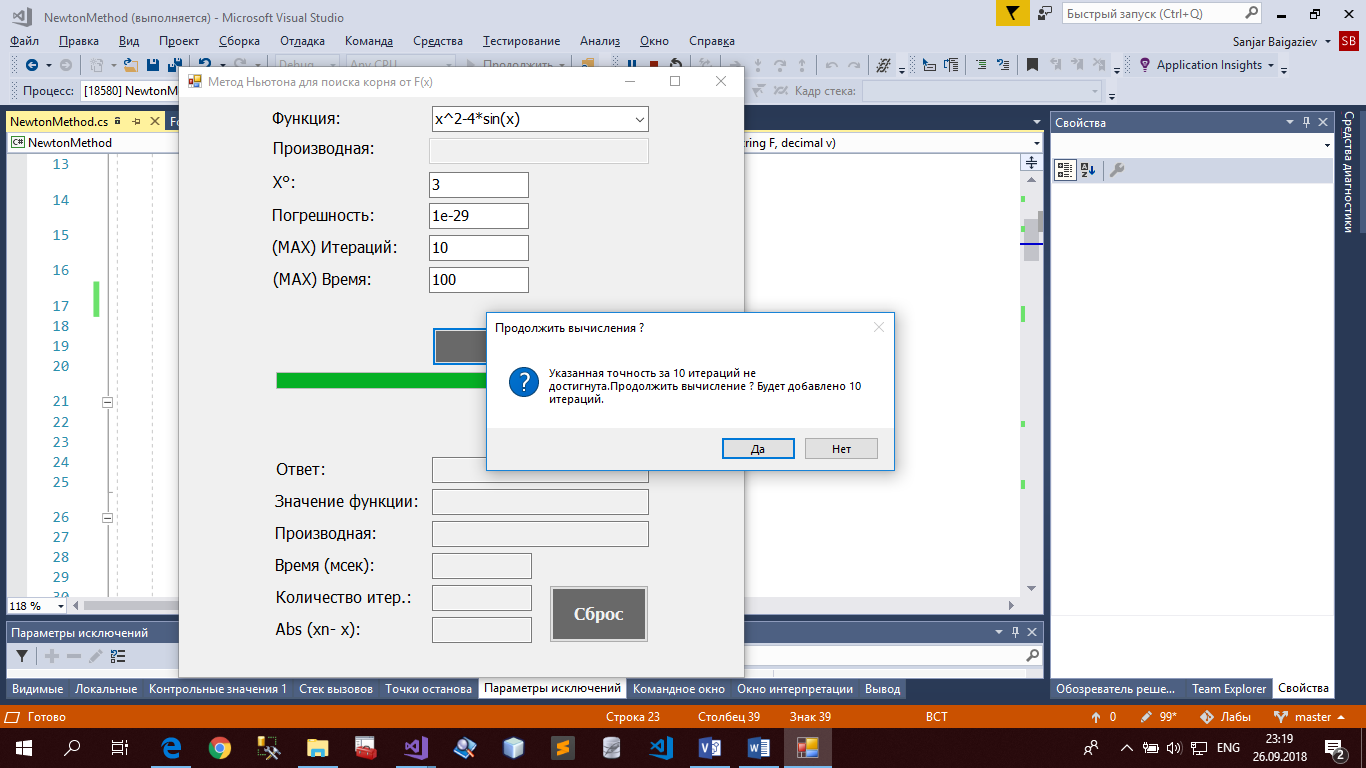
# Результат работы программы, реализующей Newton method:

1. Проверка программы на разные точности

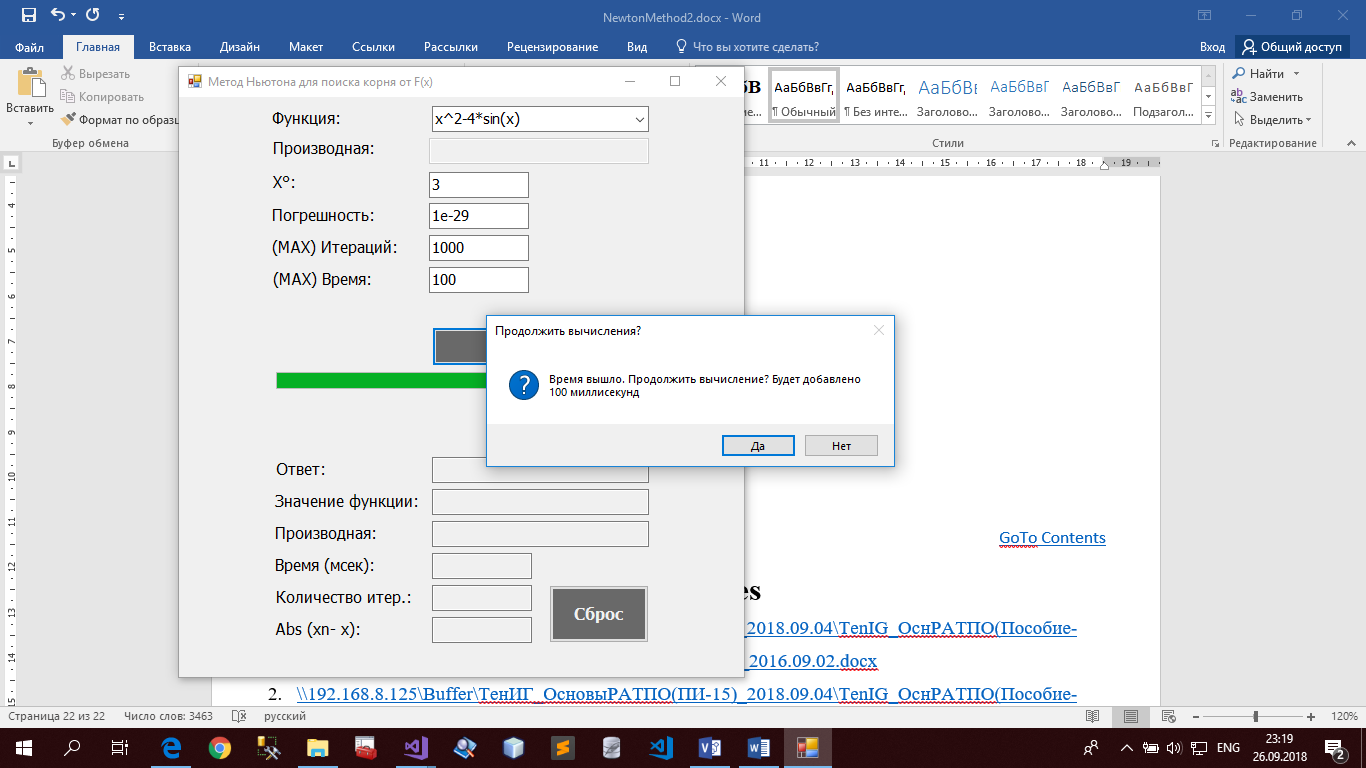




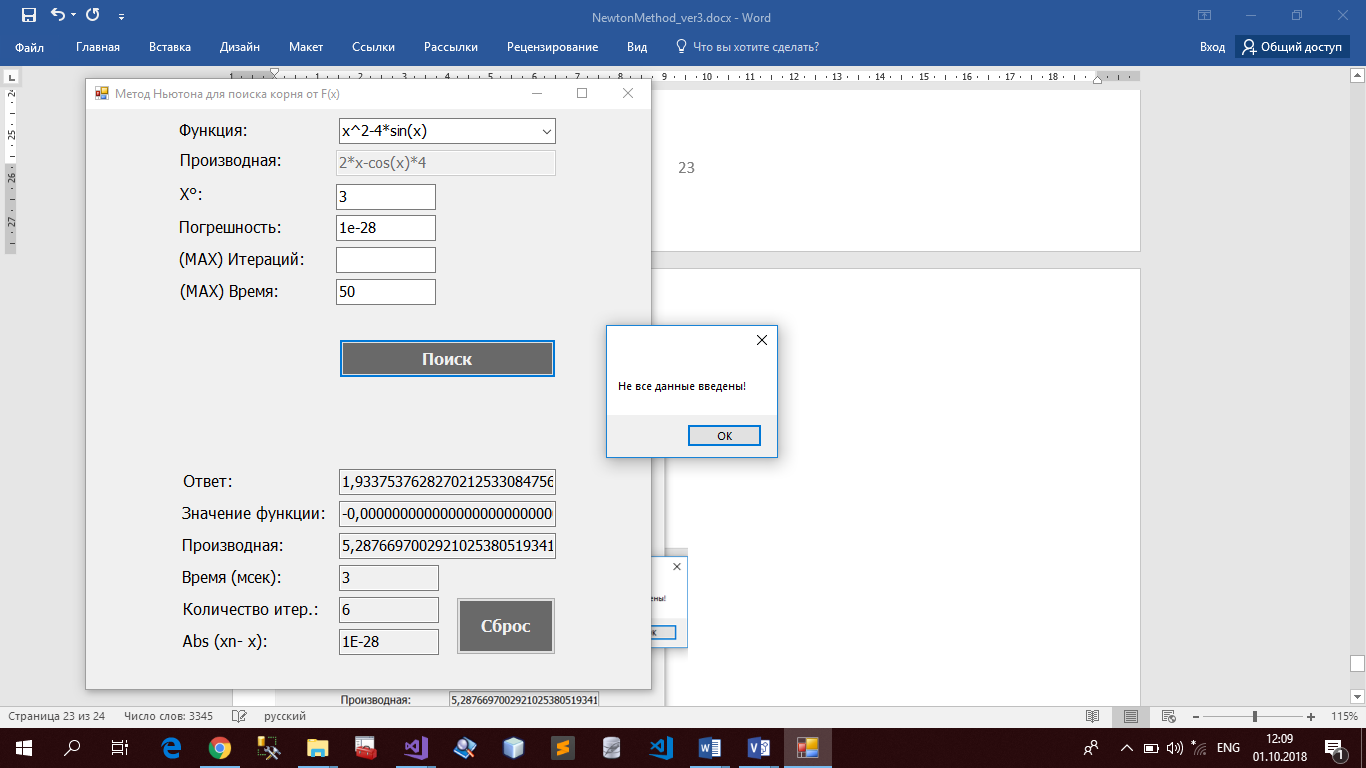
1. Остановка программы по лимиту итераций



1. Остановка программы по лимиту времену



1. Некоторые поля не заполнены



[GoTo Contents](file:///D:\lessons\4-курс\РАТПО\Лабы\Bisection%20method_ver2\Отчет.docx#Contents)