НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Дисциплина: «Программирование»

**Контрольное домашнее задание №1**

**за 2 модуль**

Выполнил: Ярных Роман,

студент группы 101ПИ.

Преподаватель: Подбельский Вадим Валериевич,

доктор технических наук,

профессор программной инженерии

факультета компьютерных наук

Москва 2014

**Введение**

**Цели и задачи контрольного домашнего задания:**

Разработать удобную программу для вычисления заданной функции и построения ее графика.

**Использованные инструменты:**

Среда .NET 4.5, C# 5.0, Visual Studio 2013, Windows 7 64-bit.

**Этапы выполнения контрольного домашнего задания:**

-анализ поставленной задачи;

-разработка алгоритма, отвечающего все заявленным требованиям;

-подготовка среды разработки;

-прототипирование программного проекта, его интерфейса и принципа работы;

-разработка приложения;

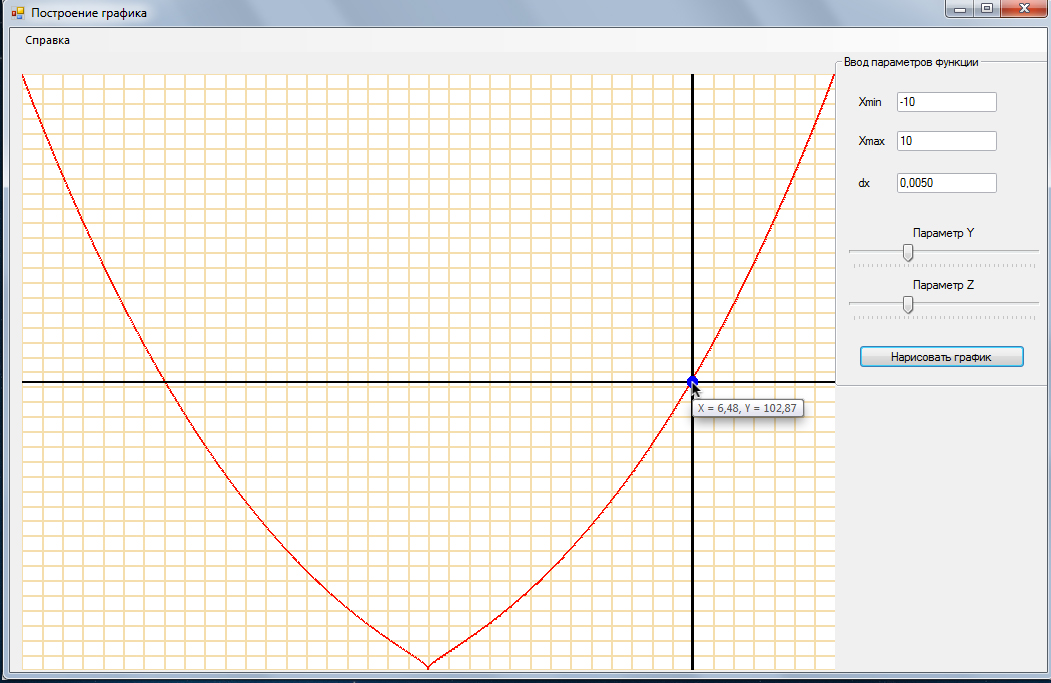
-тестирование;

-окончательная верификация и выпуск.

**Описание программы**

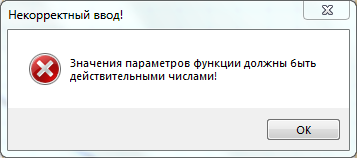
Проект КДЗ представляет собой графическое приложение, предназначенное для вычисления заданной функции и построения ее графика. Главное окно программы включает в себя кнопку для инициализации построения, поля ввода, трек бары для параметров.

Скриншот 1. Главное окно приложения

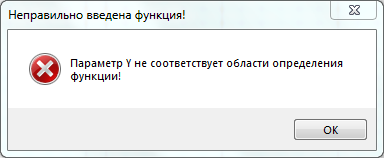


Масштабирование и прорисовка графика происходит автоматически в зависимости от числовых данных, введенных пользователем: минимальное и максимальное значение ***x***, шаг функции ***dx*** и параметры ***y***, ***z***. На этапе ввода данных приложение проверяет их соответствие области определения функции, а также формату вещественных чисел. Для отображения всех ошибок предусмотрены специальные окна сообщений двух типов: *warning message*(предупреждение пользователя о возможных проблемах с точностью вывода или скоростью выполнения программы) и *error message*(некорректность введенных данных, неправильные значения параметров функции). Примеры предупреждающих окон:

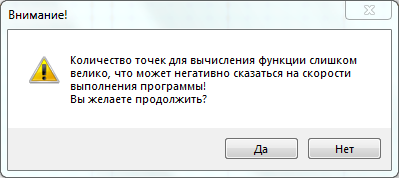
Скриншот 2. Окно предупреждения о введении нечисловых данных

****

Скриншот 3. Окно предупреждения о введении неправильного значения Y

****

Скриншот 4. Окно предупреждения о проблемах со скоростью выполнения программы при очень малом шаге функции

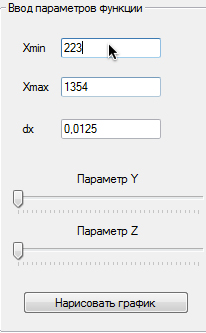
****

**Принцип работы с программой**

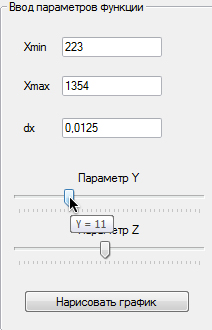
Интерфейс приложения изначально проектировался с учетом того, как должно приложение взаимодействовать с пользователем. Работа с программой состоит из нескольких шагов. Для наглядности приведу все эти шаги в виде скриншотов:

1. Ввести значения *xmin*, *xmax*, *dx.*
2. Изменить положения ползунков, задающих значения *Y*, *Z*.
3. Нажать на кнопку “Построить график”.
4. Ждать построения графика функции
5. По желанию, чтобы узнать значение в конкретной точки, можно двигать курсором, по координатам которого будет выводится *x*, *y* функции.

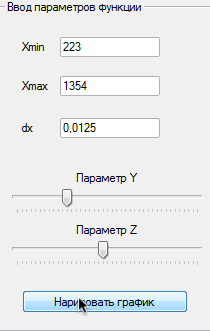
Шаг №1. Ввод значений ***Xmin***, ***Xmax*** и шага функции ***dx***



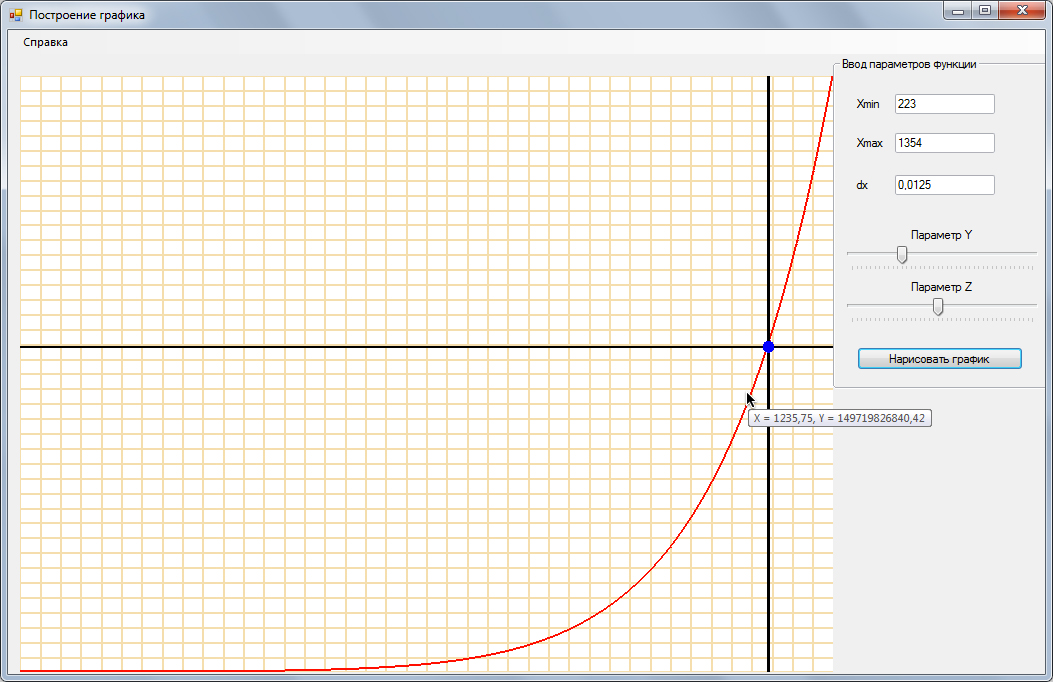
Шаг №2. Изменение положения ползунка для ввода значений параметров ***Y***, ***Z***



Шаг №3. Нажатие кнопки для рисования графика



Шаг №4. Курсор при наведении на график функции выводит ее значение в данной точки



**Структура классов приложения**

Весь проект состоит из трех классов:

-class Program;

-class GrapherWindow;

-class Function;

Главный функционал приложения содержится в классе окна GrapherWindow, который задает основное поведение программы(обработка событий на действия пользователей, вывод графики, верификация введенных данных и т.д.). Класс GrapherWindow включает в себя:

1. Методы

* private void PlotFunction(Function func, Graphics g)
* private void DrawAxes(Function func, double x, Graphics g)
* private void DrawGrid(Graphics g)
* private void InitGraphics()

1. Обработчики событий(методы)

* private void butDrawGraph\_Click(object sender, EventArgs e)
* private void graphicsBox\_SizeChanged(object sender, EventArgs e)
* private void graphicsBox\_MouseMove(object sender, MouseEventArgs e)
* private void graphicsBox\_Paint(object sender, PaintEventArgs e)

1. Контролы окна ***butDrawGraph, graphicsBox, tboxXmin, tboxXmax, tboxDx, trbarY, trBarZ***

**Исходный код программы**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace КДЗ\_\_\_Ярных\_Роман

{

public partial class GrapherWindow : Form

{

private float Xunit, Yunit;

private float XRight, YDown;

private float XMiddle, YMiddle;

private double XMin, XMax, Dx, YMin, YMax;

private float XBound, YBound;

private double XMouse;

private bool IsDrawing;

private Function myfunction;

public GrapherWindow()

{

IsDrawing = false;

InitializeComponent();

}

/// <summary>

/// Рисуем график функции

/// </summary>

/// <param name="func"></param>

private void PlotFunction(Function func, Graphics g)

{

double xstart = 0.0;

double ystart, yend;

ystart = 0.0;

yend = 0.0;

//Переводим декартовые координаты в координаты graphicsBox

Func<double, double, Point> decartes2fulldevice = (X, Y) =>

{

return new Point((int)((X - XMin) / (XMax - XMin) \* (XRight - XBound)), (int)((YDown - (Y - YMin) / (YMax - YMin) \* (YDown - YBound))));

};

Pen pen = new Pen(Color.Red, 2.0f);

for (double x = XMin; x <= XMax; x += Dx)

{

if (x == XMin)

{

xstart = x;

ystart = func[xstart];

}

else

{

if (!func.GapPoint) //Проверяем, не является ли данная точка точкой разрыва

{

yend = func[x];

g.DrawLine(pen, decartes2fulldevice(xstart, ystart), decartes2fulldevice(x, yend));

}

xstart = x;

ystart = yend;

}

}

}

/// <summary>

/// Рисуем линии абциссы и ординаты

/// </summary>

private void DrawAxes(Function func, double x, Graphics g)

{

Func<double, float> decartes2deviceX = X =>

{

return (float)((X - XMin) / (XMax - XMin) \* (XRight - XBound));

};

Func<double, float> decartes2deviceY = Y =>

{

return (float)((YDown - (Y - YMin) / (YMax - YMin) \* (YDown - YBound)));

};

double y = func[x];

Pen pen = new Pen(Color.Black, 2.5f);

g.DrawLine(pen, decartes2deviceX(x), 0.0f, decartes2deviceX(x), YDown);//Рисуем ось Y

g.DrawLine(pen, 0.0f, decartes2deviceY(y), XRight, decartes2deviceY(y));//Рисуем ось X

SolidBrush brush = new SolidBrush(Color.Blue);

g.FillEllipse(brush, new RectangleF(decartes2deviceX(x) - 6.0f, decartes2deviceY(y) - 6.0f, 12.0f, 12.0f));

funcvaluestip.SetToolTip(graphicsBox, String.Format("X = {0:F2}, Y = {1:F2}", x, y));

}

/// <summary>

/// Рисуем сетку координат

/// </summary>

private void DrawGrid(Graphics g)

{

Func<float, float> device2decartesX = (X) =>

{

return (float)(X \* (XMax - XMin) / XRight + XMin);

};

Func<float, float> device2decartesY = (Y) =>

{

return (float)((YDown - Y) \* (YMax - YMin) / YDown);

};

Pen pen = new Pen(Color.Wheat, 2.0f);

SolidBrush brush = new SolidBrush(Color.Black);

for (float Y = YBound; Y <= YDown - YBound; Y += Yunit)

{

for (float X = XBound; X <= XRight - XBound; X += Xunit)

{

g.DrawLine(pen, new PointF(X, YBound), new PointF(X, YDown - YBound));

//g.DrawString(String.Format("{0:F2}", device2decartesX(X)), new Font("Ariel", 7.0f, FontStyle.Bold), brush, new PointF(X + 2.0f, YDown + 5.0f));

}

g.DrawLine(pen, new PointF(XBound, Y), new PointF(XRight - XBound, Y));

//g.DrawString(String.Format("{0:F2}", device2decartesY(Y)), new Font("Ariel", 7.0f, FontStyle.Bold), brush, new PointF(XBound + 2.0f, Y + 2.0f));

}

}

/// <summary>

/// Обрабатываем событие перерисовки

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void graphicsBox\_Paint(object sender, PaintEventArgs e)

{

if (IsDrawing)

{

Graphics g = e.Graphics;

DrawGrid(g);

PlotFunction(myfunction, g);

DrawAxes(myfunction, XMouse, g);

}

}

/// <summary>

/// Инициализируем параметры для рисования графика

/// </summary>

private void InitGraphics()

{

XRight = graphicsBox.Size.Width;

YDown = graphicsBox.Size.Height;

Xunit = XRight / 40.0f;

Yunit = YDown / 40.0f;

XBound = 0.0f;

YBound = 0.0f;

}

private void butDrawGraph\_Click(object sender, EventArgs e)

{

bool tboxXMinIsNum, tboxXMaxIsNum, tboxDxIsNum;

tboxXMinIsNum = Double.TryParse(tboxXmin.Text, out XMin);

tboxXMaxIsNum = Double.TryParse(tboxXmax.Text, out XMax);

tboxDxIsNum = Double.TryParse(tboxDx.Text, out Dx);

if(tboxXMinIsNum && tboxXMaxIsNum && tboxDxIsNum){//Проверяем, являются ли значения полей ввода числами

errorProvider.Clear(); //Очищаем все прошлые ошибки

if (XMin >= XMax)

{

MessageBox.Show("XMin не может быть больше XMax!", "Некорректный ввод данных!", MessageBoxButtons.OK, MessageBoxIcon.Error);

IsDrawing = false;

}

else if (trBarY.Value <= Math.Abs(XMin) && XMin < 0)

{

MessageBox.Show("Параметр Y не соответствует области определения функции!", "Неправильно введена функция!", MessageBoxButtons.OK, MessageBoxIcon.Error);

IsDrawing = false;

}

else if (Dx == 0.0)

{

errorProvider.SetError(tboxDx, "Значение dx не может быть нулевым!");

MessageBox.Show("Шаг функции не может быть нулевым!", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

IsDrawing = false;

}

else

{

DialogResult dialog\_result = System.Windows.Forms.DialogResult.Yes;

if ((XMax - XMin) / Dx > 10e6)//Если число точек очень велико, то выводим предупреждающее сообщение

dialog\_result = MessageBox.Show(String.Format("{0}\n{1}", "Количество точек для вычисления функции слишком велико, что может негативно сказаться на скорости выполнения программы!", "Вы желаете продолжить?"), "Внимание!", MessageBoxButtons.YesNo, MessageBoxIcon.Warning, MessageBoxDefaultButton.Button2);

if (Dx >= 0.1)

MessageBox.Show("Шаг функции достаточно большой. Точность графика будет снижена!", "Внимание!", MessageBoxButtons.OK, MessageBoxIcon.Warning);

if (dialog\_result == System.Windows.Forms.DialogResult.Yes)

{

myfunction = new Function();

myfunction.Y = trBarY.Value;

myfunction.Z = trBarZ.Value;

List<double> func\_values = new List<double>();

for (double x = XMin; x <= XMax; x += Dx)

func\_values.Add(myfunction[x]);

YMax = func\_values.Max();//Найдем максимальное значение функции

YMin = func\_values.Min();//Найдем минимальное значение функции

XMiddle = (float)(Math.Abs(XMin) / (XMax - XMin) \* XRight);

IsDrawing = true;

graphicsBox.Refresh();

}

}

}

else

{

errorProvider.Clear();

if(!tboxXMinIsNum)

errorProvider.SetError(tboxXmin, "Значение Xmin не действительное число!");

if(!tboxXMaxIsNum)

errorProvider.SetError(tboxXmax, "Значение Xmax не действительное число!");

if(!tboxDxIsNum)

errorProvider.SetError(tboxDx, "Значение dx не действительное число!");

MessageBox.Show("Значения параметров функции должны быть действительными числами!", "Некорректный ввод!", MessageBoxButtons.OK, MessageBoxIcon.Error);

IsDrawing = false;

}

}

/// <summary>

/// Обработчик события загрузки окна

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

this.DoubleBuffered = true;

InitGraphics();

graphicsBox.Refresh();

}

/// <summary>

/// Обработчик события изменения размеров экрана

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void graphicsBox\_SizeChanged(object sender, EventArgs e)

{

InitGraphics();

graphicsBox.Refresh();

}

private void оПрограммеToolStripMenuItem\_Click(object sender, EventArgs e)

{

MessageBox.Show(String.Format("{0}\n{1}", "КДЗ по программированию выполнено студентом", "101ПИ ФКН НИУ ВШЭ Ярных Романом."));

}

/// <summary>

/// Обработчик события изменения параметров Y и Z

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void trBar\_Scroll(object sender, EventArgs e)

{

if((TrackBar)sender == trBarY)

hintProvider.SetToolTip(trBarY, String.Format("Y = {0}", trBarY.Value));

if((TrackBar)sender == trBarZ)

hintProvider.SetToolTip(trBarZ, String.Format("Z = {0}", trBarZ.Value));

}

private void graphicsBox\_MouseMove(object sender, MouseEventArgs e)

{

if (IsDrawing)

{

//Переводим координаты graphicsBox в декартовые координаты

Func<int, double> device2decartesX = (X) =>

{

return X \* (XMax - XMin) / XRight + XMin;

};

XMouse = device2decartesX(e.X);

graphicsBox.Refresh();

}

}

}

}