

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Санкт-Петербургский государственный технологический институт (технический университет)»

Дисциплина: «Разработка программных систем»

Отчёт по лабораторной работе № 3

**Задание:** Написать приложение с использованием технологии WinForms для построения графика функции и вывода таблицы значений функции.

**Выполнил студент группы №485:**

Зобнин Илья Михайлович

**Проверили:**

Иван Григорьевич Корниенко

Алексей Константинович Федин

Санкт-Петербург

2020

# Постановка задачи

Необходимо написать приложение с использованием технологии WinForms для построения графика функции и вывода таблицы значений функции. Пользователь задает правую и левую границу, шаг, коэффициенты (при их наличии). При невозможности построить график функции в заданном интервале пользователю выдается предупреждение об этом с предложением сменить границы построения. Если график функции из-за коэффициентов вырождается в точку или не может быть построен пользователь также видит предупреждение. Добавить в программу возможность экспорта исходных данных и результатов расчета в MS Excel.

# Особые ситуации

* Если при введенных границах и коэффициентах некоторые значения функции не будут отображаться на графике, то необходимо предупредить пользователя.
* Если пользователь попытается сохранить результат работы программы в уже существующий файл с данными, программа оповестит пользователя и предложит перезаписать файл либо же указать другой файл.

# Математические методы и алгоритмы решения задач

Согласно постановке задачи для вычисления массива значений функции и построения соответствующих таблицы и графика, будет использована блок-схема, представленная на Рисунке 1.

Формула, используемая в данном алгоритме:

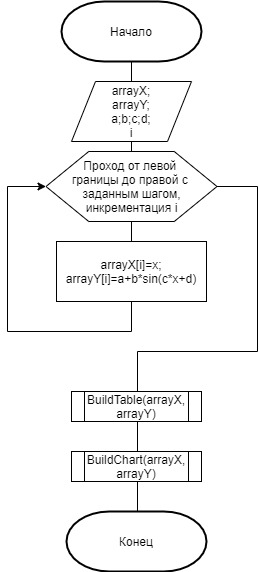


Рисунок 1 – Блок-схема алгоритма вычисления массива значений функции и построения соответствующих таблицы и графика

# Форматы представления данных

Программа использует следующие переменные:

Таблица 1 – Переменные, используемы в программе

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Описание** |
| a | double | Коэффициент a |
| b | double | Коэффициент b |
| c | double | Коэффициент c |
| d | double | Коэффициент d |
| leftLimitX | double | Левая граница x |
| rightLimitX | double | Правая граница x |
| stepX | double | Шаг x |
| arrayX | double | Массив значений x |
| arrayY | double | Массив значений функции y |

# Структура программы

Программа разделена на шесть классов, из которых один является основным и отвечает за запуск программ, пять оставшихся содержат в себе функции, необходимые для работы программы.

Класс Program.cs:

Таблица 2 – Функции, составляющие Класс Program.cs

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| Main | Запуск программы |

Класс ExcelExport.cs:

Таблица 3 – Функции, составляющие класс ExcelExport.cs

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| ExportToExcel | Функция для вывода данных программы в файл Excel |
| ExcelExportBtn\_Click | Обработка события нажатия на кнопку сохранения в Excel |

Класс FuncCalculation.cs:

Таблица 4 – Функции, составляющие класс FuncCalculation.cs

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| FuncCalc | Расчет значений функции |

Класс Interface.cs:

Таблица 5 – Функции, составляющие класс Interface.cs

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| ShowOrHideWarnings | Показ или скрытие предупреждений о неправильном вводе |
| RestoreTextBoxes | Заполнение текстовых полей старыми значениями при запуске программы |
| SaveTextBoxesAfterClose | Запоминание значений в текстовых полях |
| AboutProgram\_Click | Обработчик события при нажатии кнопки «О программе» |
| ExitProgram\_Click | Обработчик события при нажатии кнопки «Выйти» |

Класс RedrawFuncs.cs:

Таблица 6 – Функции, составляющие класс RedrawFuncs.cs

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| TableAndGraphRedraw | Вызов функций перерисовки таблицы значений и графика функции |
| ResultFuncLabelRedraw | Перерисовка конечного вида функции |
| TableOfFunctionValuesRedraw | Перерисовка таблицы значений функции |
| FunctionGraphRedraw | Перерисовка графика функции |

Класс TextBoxesChecks.cs:

Таблица 7 – Функции, составляющие класс TextBoxesChecks.cs

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| TextBoxFormatCheckAndRedraw | Проверка на правильность формата введённой строки и вызов функции перерисовки |
| ParamsAndXCheckAndRedraw | Вызывается при изменении любого текстового поля в форме, проверяет его на соответствие формату |
| BoardsAndStepXCheck | Проверка границ и шага X |

# Описание хода выполнения лабораторной работы

* При получении пользовательского ввода необходимо было добавить проверку, на его корректность.
* При выполнении лабораторной работы были созданы тесты для проверки правильности работы алгоритмов.

# Результат работы программы

В результате работы программа выводит построенный график функции.

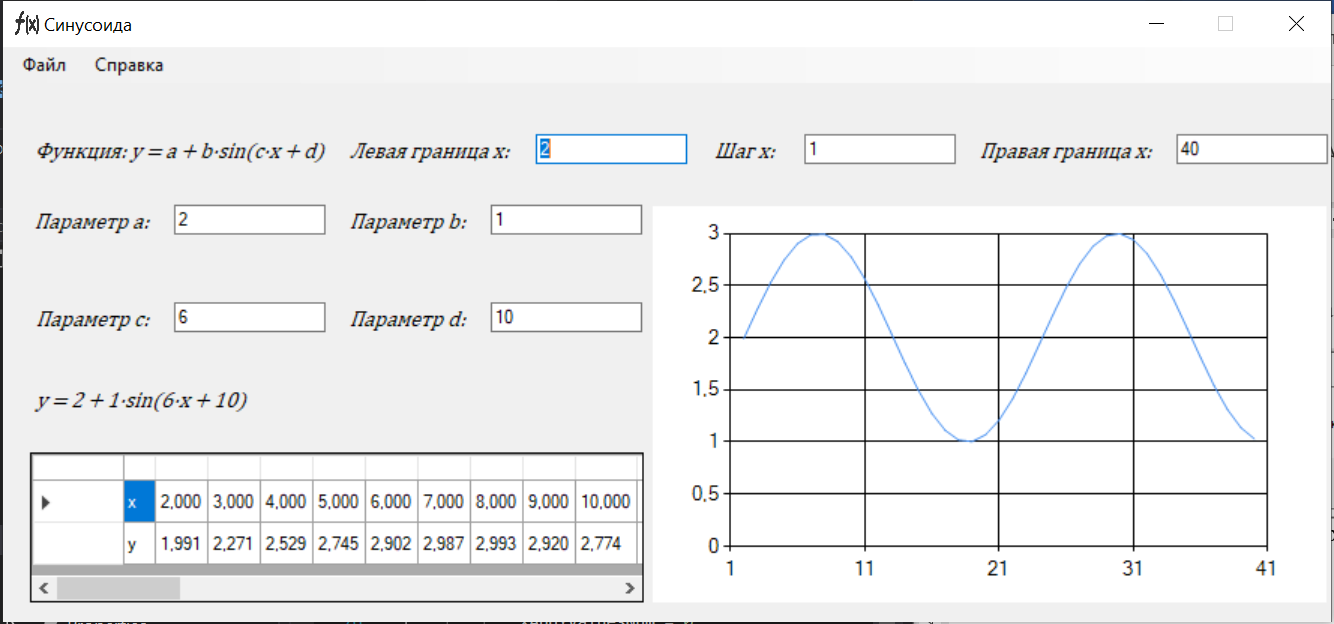


Рисунок 2 – Успешная работа программы.

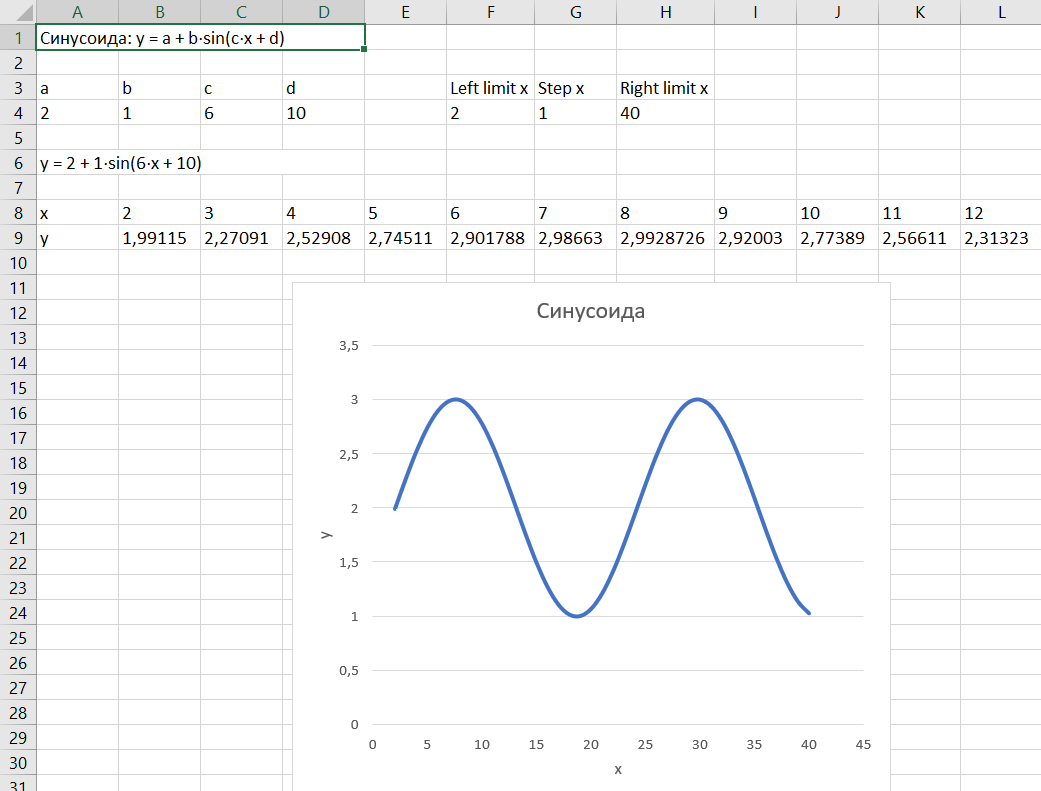


Рисунок 3 – Успешный вывод исходных данных и результата в Excel файл.

# Текст программы

**[--- Начало программы ---]**

**// Program.cs**

**// Лабораторная работа №3.**

**// Студент группы 485, Зобнин Илья Михайлович. 2020 год**

using System;

using System.Windows.Forms;

namespace Lab3

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Interface());

}

}

}

**// ExcelExport.cs**

**// Лабораторная работа №3.**

**// Студент группы 485, Зобнин Илья Михайлович. 2020 год**

using Microsoft.Office.Interop.Excel;

using System;

using System.Windows.Forms;

namespace Lab3

{

public partial class Interface : Form

{

private void ExportToExcel()

{

var excel = new Microsoft.Office.Interop.Excel.Application();

excel.DisplayAlerts = false;

var wb = excel.Workbooks.Add();

Worksheet ws = (Worksheet)excel.ActiveSheet;

ws.Cells[1, 1] = "Синусоида: y = a + b⋅sin(c⋅x + d)";

ws.Range["A1:D1"].Merge();

ws.Cells[3, 1] = "a";

ws.Cells[4, 1] = a;

ws.Cells[3, 2] = "b";

ws.Cells[4, 2] = b;

ws.Cells[3, 3] = "c";

ws.Cells[4, 3] = c;

ws.Cells[3, 4] = "d";

ws.Cells[4, 4] = d;

ws.Cells[3, 6] = "Left limit x";

ws.Cells[4, 6] = leftBoardX;

ws.Cells[3, 7] = "Step x";

ws.Cells[4, 7] = stepX;

ws.Cells[3, 8] = "Right limit x";

ws.Cells[4, 8] = rightBoardX;

ws.Columns["F"].AutoFit();

ws.Columns["H"].AutoFit();

ws.Range["A3:H3"].Style.HorizontalAlignment = XlHAlign.xlHAlignLeft;

ws.Cells[6, 1] = ResultFuncLabel.Text;

ws.Range["A6:D6"].Merge();

ws.Cells[8, 1] = "x";

ws.Cells[9, 1] = "y";

for (int i = 0; i < xAndYValuesNum; ++i)

{

ws.Cells[8, i + 2] = xValues[i];

ws.Cells[9, i + 2] = yValues[i];

}

Range xValuesRange = ws.Range[ws.Cells[8, 2], ws.Cells[8, xAndYValuesNum + 1]];

Range yValuesRange = ws.Range[ws.Cells[9, 2], ws.Cells[9, xAndYValuesNum + 1]];

Chart chart = ws.Shapes.AddChart2(Type.Missing, Type.Missing, 150, 150, 350, 300).Chart;

chart.ChartWizard(yValuesRange, XlChartType.xlXYScatterSmoothNoMarkers, Type.Missing, Type.Missing, Type.Missing, Type.Missing, Type.Missing,

"Синусоида", "x", "y");

Series series = chart.SeriesCollection(1);

series.XValues = xValuesRange;

ws.SaveAs(saveFileDialog.FileName);

wb.Close(true);

excel.Quit();

ws = null;

wb = null;

excel = null;

GC.Collect();

}

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

{

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

ExcelExportBtn.Enabled = false;

ExportToExcel();

MessageBox.Show("Успешно сохранено!", "Сохранение", MessageBoxButtons.OK, MessageBoxIcon.Information);

ExcelExportBtn.Enabled = true;

}

}

}

}

**// FuncCalculation.cs**

**// Лабораторная работа №3.**

**// Студент группы 485, Зобнин Илья Михайлович. 2020 год**

using System;

using System.Windows.Forms;

namespace Lab3

{

public partial class Interface : Form

{

int xAndYValuesNum = 0;

private void FuncCalc()

{

xValues = new double[xAndYValuesNum];

yValues = new double[xAndYValuesNum];

int i = 0;

for (double x = leftBoardX; x <= rightBoardX; x += stepX, ++i)

{

xValues[i] = x;

yValues[i] = a + b \* Math.Sin(c \* x + d);

}

}

}

}

**// Interface.cs**

**// Лабораторная работа №3.**

**// Студент группы 485, Зобнин Илья Михайлович. 2020 год**

using System;

using System.IO;

using System.Windows.Forms;

namespace Lab3

{

public partial class Interface : Form

{

readonly static string NL = Environment.NewLine;

readonly string AboutText = "Эта программа вычисляет значения функции синусоиды в точках x, " +

$"заданных левой и правой границами, а также шагом и строит соответствующий график.{NL}" +

"Программа студента 485 группы Зобнина Ильи.";

public Interface()

{

InitializeComponent();

// обработчики изменения текста

LeftBoardXTextBox.TextChanged += delegate (object sender, EventArgs e)

{

ParamsAndXCheckAndRedraw((TextBox)sender, out leftBoardXChecked, LeftBoardXWarningLabel,

LeftBoardXWarningPicture, out leftBoardX, "X");

};

StepXTextBox.TextChanged += delegate (object sender, EventArgs e)

{

ParamsAndXCheckAndRedraw((TextBox)sender, out stepXChecked, StepXWarningLabel,

StepXWarningPicture, out stepX, "X");

};

RightBoardXTextBox.TextChanged += delegate (object sender, EventArgs e)

{

ParamsAndXCheckAndRedraw((TextBox)sender, out rightBoardXChecked, RightBoardXWarningLabel,

RightBoardXWarningPicture, out rightBoardX, "X");

};

ParametrATextBox.TextChanged += delegate (object sender, EventArgs e)

{

ParamsAndXCheckAndRedraw((TextBox)sender, out parametrAChecked, ParametrAWarningLabel, ParametrAWarningPicture,

out a, "Param");

};

ParametrBTextBox.TextChanged += delegate (object sender, EventArgs e)

{

ParamsAndXCheckAndRedraw((TextBox)sender, out parametrBChecked, ParametrBWarningLabel, ParametrBWarningPicture,

out b, "Param");

};

ParametrCTextBox.TextChanged += delegate (object sender, EventArgs e)

{

ParamsAndXCheckAndRedraw((TextBox)sender, out parametrCChecked, ParametrCWarningLabel, ParametrCWarningPicture,

out c, "Param");

};

ParametrDTextBox.TextChanged += delegate (object sender, EventArgs e)

{

ParamsAndXCheckAndRedraw((TextBox)sender, out parametrDChecked, ParametrDWarningLabel, ParametrDWarningPicture,

out d, "Param");

};

RestoreTextBoxes();

}

private void ShowOrHideWarnings(Label label, PictureBox picture, bool showOrHide,

string labelMessage = "")

{

if (labelMessage != "")

label.Text = labelMessage;

label.Visible = showOrHide;

picture.Visible = showOrHide;

}

private void RestoreTextBoxes()

{

using (var streamReader = new StreamReader("TextBoxesSave.txt"))

{

var controls = Controls;

for (int i = 0; i < controls.Count; ++i)

if (controls[i].GetType().Name == "TextBox")

controls[i].Text = streamReader.ReadLine();

}

}

private void SaveTextBoxesAfterClose(object sender, FormClosingEventArgs e)

{

using (var streamWriter = new StreamWriter("TextBoxesSave.txt"))

{

var controls = Controls;

for (int i = 0; i < controls.Count; ++i)

if (controls[i].GetType().Name == "TextBox")

streamWriter.WriteLine(controls[i].Text);

}

}

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

{

MessageBox.Show(AboutText);

}

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

{

this.Close();

}

}

}

**// RedrawFuncs.cs**

**// Лабораторная работа №3.**

**// Студент группы 485, Зобнин Илья Михайлович. 2020 год**

using System.Windows.Forms;

namespace Lab3

{

public partial class Interface : Form

{

#region BoardsAndParamsChecksForRedraw

bool leftBoardXChecked = false;

bool stepXChecked = false;

bool rightBoardXChecked = false;

public bool parametrAChecked = false;

bool parametrBChecked = false;

bool parametrCChecked = false;

bool parametrDChecked = false;

#endregion

#region FuncParamsAndXYValues

double leftBoardX;

double stepX;

double rightBoardX;

double a;

double b;

double c;

double d;

double[] xValues;

double[] yValues;

#endregion

private void TableAndGraphRedraw()

{

if (leftBoardXChecked && stepXChecked && rightBoardXChecked && parametrAChecked && parametrBChecked &&

parametrCChecked && parametrDChecked)

{

FuncCalc();

TableOfFunctionValuesRedraw();

FunctionGraphRedraw();

ExcelExportBtn.Enabled = true;

}

else

ExcelExportBtn.Enabled = false;

}

private void ResultFuncLabelRedraw()

{

ResultFuncLabel.Text = "y = ";

if (parametrAChecked)

{

string toAdd = ParametrATextBox.Text;

if (ParametrATextBox.TextLength > 4)

toAdd = ParametrATextBox.Text.Substring(0, 4) + "...";

ResultFuncLabel.Text += toAdd;

}

else

ResultFuncLabel.Text += "a";

ResultFuncLabel.Text += " + ";

if (parametrBChecked)

{

string toAdd = ParametrBTextBox.Text + "⋅";

if (ParametrBTextBox.TextLength > 4)

toAdd = ParametrBTextBox.Text.Substring(0, 4) + "..." + "⋅";

ResultFuncLabel.Text += toAdd;

}

else

ResultFuncLabel.Text += "b⋅";

ResultFuncLabel.Text += "sin(";

if (parametrCChecked)

{

string toAdd = ParametrCTextBox.Text + "⋅";

if (ParametrCTextBox.TextLength > 4)

toAdd = ParametrCTextBox.Text.Substring(0, 4) + "..." + "⋅";

ResultFuncLabel.Text += toAdd;

}

else

ResultFuncLabel.Text += "c⋅";

ResultFuncLabel.Text += "x + ";

if (parametrDChecked)

{

string toAdd = ParametrDTextBox.Text;

if (ParametrDTextBox.TextLength > 4)

toAdd = ParametrDTextBox.Text.Substring(0, 4) + "...";

ResultFuncLabel.Text += toAdd;

}

else

ResultFuncLabel.Text += "d";

ResultFuncLabel.Text += ")";

}

private void TableOfFunctionValuesRedraw()

{

while (TableOfFunctionValues.RowCount != 0)

TableOfFunctionValues.Rows.RemoveAt(0);

while (TableOfFunctionValues.ColumnCount != 0)

TableOfFunctionValues.Columns.RemoveAt(0);

TableOfFunctionValues.Visible = true;

for (int i = 0; i < xValues.Length + 1; ++i)

TableOfFunctionValues.Columns.Add("", "");

string[] xStr = new string[xValues.Length + 1];

xStr[0] = "x";

string[] yStr = new string[xValues.Length + 1];

yStr[0] = "y";

for (int i = 0; i < xValues.Length; ++i)

{

xStr[i + 1] = xValues[i].ToString("F3");

yStr[i + 1] = yValues[i].ToString("F3");

}

TableOfFunctionValues.Rows.Add(xStr);

TableOfFunctionValues.Rows.Add(yStr);

}

private void FunctionGraphRedraw()

{

FunctionGraph.Series["FunctionGraph"].Points.Clear();

FunctionGraph.Visible = true;

for (int i = 0; i < xValues.Length; ++i)

FunctionGraph.Series["FunctionGraph"].Points.AddXY(xValues[i], yValues[i]);

}

}

}

**// TextBoxesChecks.cs**

**// Лабораторная работа №3.**

**// Студент группы 485, Зобнин Илья Михайлович. 2020 год**

using System.Windows.Forms;

namespace Lab3

{

public partial class Interface : Form

{

const double lowerLimit = -10000000000000000000.0;

const double upperLimit = 10000000000000000000.0;

const int xAndYValuesNumLimit = 200;

private void TextBoxFormatCheckAndRedraw(TextBox ParamTextBox, out bool paramCheck, Label ParamWarningLabel,

PictureBox ParamWarningPic, out double value)

{

paramCheck = double.TryParse(ParamTextBox.Text, out value);

if (!paramCheck)

{

if (ParamTextBox.TextLength != 0)

ShowOrHideWarnings(ParamWarningLabel, ParamWarningPic, true);

else

ShowOrHideWarnings(ParamWarningLabel, ParamWarningPic, false);

ExcelExportBtn.Enabled = false;

}

else

{

if (value <= lowerLimit || value >= upperLimit)

{

ShowOrHideWarnings(ParamWarningLabel, ParamWarningPic, true,

"Значение слишком большое или маленькое!");

paramCheck = false;

ExcelExportBtn.Enabled = false;

}

else

{

ShowOrHideWarnings(ParamWarningLabel, ParamWarningPic, false);

ExcelExportBtn.Enabled = false;

}

}

}

private void ParamsAndXCheckAndRedraw(TextBox ParamTextBox, out bool paramCheck,

Label ParamWarningLabel, PictureBox ParamWarningPic, out double value, string paramOrXCheck)

{

TextBoxFormatCheckAndRedraw(ParamTextBox, out paramCheck, ParamWarningLabel, ParamWarningPic, out value);

if (paramOrXCheck == "Param")

{

ResultFuncLabelRedraw();

TableAndGraphRedraw();

}

else

BoardsAndStepXCheck();

}

private void BoardsAndStepXCheck()

{

if (!leftBoardXChecked || !stepXChecked || !rightBoardXChecked)

return;

if (leftBoardX >= rightBoardX)

{

ShowOrHideWarnings(WrongBoardsOrStepWarningLabel, WrongBoardsOrStepWarningPicture, true,

"Левая граница не может быть больше или равна правой!");

return;

}

if (stepX <= 0)

{

ShowOrHideWarnings(WrongBoardsOrStepWarningLabel, WrongBoardsOrStepWarningPicture, true,

"Шаг не может быть меньше или равен нулю!");

return;

}

if ((leftBoardX + stepX) > rightBoardX)

{

ShowOrHideWarnings(WrongBoardsOrStepWarningLabel, WrongBoardsOrStepWarningPicture, true,

"График не может состоять из одной точки!");

return;

}

//подсчёт количества точек x

xAndYValuesNum = 0;

for (double x = leftBoardX; x <= rightBoardX; x += stepX, ++xAndYValuesNum) { }

if (xAndYValuesNum > xAndYValuesNumLimit)

{

ShowOrHideWarnings(WrongBoardsOrStepWarningLabel, WrongBoardsOrStepWarningPicture, true,

"Слишком много точек x! Лимит 200.");

return;

}

ShowOrHideWarnings(WrongBoardsOrStepWarningLabel, WrongBoardsOrStepWarningPicture, false);

TableAndGraphRedraw();

}

}

}

**[--- Конец программы ---]**