

МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| УГНС | | 09.00.00 | Информатика и вычислительная техника | | |
| Направление подготовки | | 09.03.01 | Информатика и вычислительная техника | | |
| Направленность (профиль) | |  | Системы автоматизированного проектирования | | |
| Форма обучения | |  | очная | | |
|  | |  |  | | |
| Факультет | |  | Информационных технологий и управления | | |
| Кафедра | |  | Систем автоматизированного проектирования и управления | | |
| Учебная дисциплина | |  | Программирование | | |
| Курс | Ⅱ | | | Группа | 413 |

Отчёт по контрольной работе № 3

Вариант № 7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Исполнитель: |  |  |  |  |
| обучающийся группы 413 |  |  |  | Занин Егор Валерьевич |
|  |  | (дата, подпись) |  |  |
|  |  |  |  |  |
| Проверил(и): |  |  |  | Корниенко Иван Григорьевич |
|  |  | (дата, подпись) |  | Федин Алексей Константинович |
|  |  |  |  |  |

Содержание

[1. Постановка задачи 3](#_Toc98373326)

[2. Исходные данные 3](#_Toc98373327)

[3. Особые ситуации 3](#_Toc98373328)

[4. Форматы представления данных 4](#_Toc98373330)

[5. Структура программы 5](#_Toc98373331)

[6. Блок-схемы алгоритмов программы 6](#_Toc98373332)

[7. Описание хода выполнения контрольной работы 6](#_Toc98373333)

[8. Результаты работы программы 7](#_Toc98373334)

[9. Исходный текст программы 10](#_Toc98373335)

1. 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.

Построить график функции и вывести таблицы значений функции: Трактриса (1.1).

|  |  |
| --- | --- |
|  | (1.1) |

1. Исходные данные

В качестве исходных данных, пользователь задает правую и левую границу, шаг, коэффициенты.

1. Особые ситуации
2. При невозможности построить график функции в заданном интервале пользователю выдается предупреждение об этом с предложением сменить границы построения.
3. Если график функции из-за коэффициентов вырождается в точку или не может быть построен пользователь видит предупреждение.
4. Форматы представления данных

Программа использует следующие переменные:

Таблица 1 – Переменные, используемый в программе

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| label | label | Текст |
| textbox | textbox | Текстовое поле для ввода |
| button | button | Кнопка |

1. Структура программы

Основные функции программы:

Таблица 2 – Функции, составляющие программу

|  |  |
| --- | --- |
| Имя | Описание |
| InitializeComponent | Загрузка формы |
| Button\_Click | Событие кнопки |
| toolStripButton\_Click | Событие обработки нажатия инструмента панели инструментов |
| Form\_Closing | Обработка действий при закрытии формы |

1. Блок-схемы алгоритмов программы

Для удобства написания блок-схемы, было принято ввести обозначения:

a — коэффициент;

left — левая граница;

right — Правая граница;

step — шаг.

На рисунке 1 представлена блок-схема алгоритма решения задачи.



Рисунок 1 – Блок-схема решения задачи

1. Описание хода выполнения контрольной работы

При выполнении данной контрольной работой, мною было изучено: работа с WinForms.

1. Результаты работы программы

Результат тестирования всех возможностей программы приведен на рисунке 2.

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 2, лист 1 – Экранная копия результата работы разработанной программы

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 2, лист 2 – Экранная копия результата работы разработанной программы

Изображение выглядит как текст, диаграмма, График, линия

Автоматически созданное описание

Рисунок 2, лист 3 – Экранная копия результата работы разработанной программы

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 2, лист 4 – Экранная копия результата работы разработанной программы

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 2, лист 5 – Экранная копия результата работы разработанной программы

1. Исходный текст программы

Код с файла «Program.cs»:

using System;

using System.Windows.Forms;

namespace WinFormsKR3

{

internal static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new FunctionSolution());

}

}

}

Код с файла «Tractrix.cs»:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace TractrixNamespace

{

public class TractrixClass

{

public static double EquationTractrix(double a ,double y)

{

double rootNumber = Math.Sqrt((Math.Pow(a, 2) - Math.Pow(y, 2)));

double x = (a \* Math.Log(((a + rootNumber) / y), Math.E)) - rootNumber;

return Math.Round(x, 3);

}

}

}

Код с файла «FunctionSolutionForm.cs»:

using System;

using System.Windows.Forms;

using System.IO;

using TractrixNamespace;

namespace WinFormsKR3

{

public partial class FunctionSolution : Form

{

static double leftBorder, rightBorder, step, a, x, y;

//Построение пользовательского интерфейса

public FunctionSolution()

{

InitializeComponent();

LoadDefaultParametrs();

showStart\_ToolStripMenuItem.Checked = Properties.Settings.Default.checkBoxUser;

if (showStart\_ToolStripMenuItem.Checked == true)

{

new GreetUser().Show();

}

}

//Обработка события при закрытии формы

private void FunctionSolution\_FormClosing(object sender, FormClosingEventArgs e)

{

Properties.Settings.Default.checkBoxUser = showStart\_ToolStripMenuItem.Checked;

Properties.Settings.Default.Save();

}

private void information\_ToolStripMenuItem\_Click(object sender, EventArgs e)

{

new GreetUser().Show();

}

private void showStart\_ToolStripMenuItem\_Click(object sender, EventArgs e)

{

showStart\_ToolStripMenuItem.Checked = !showStart\_ToolStripMenuItem.Checked;

}

private void saveChart\_ToolStripMenuItem\_Click(object sender, EventArgs e)

{

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

return;

string filename = saveFileDialog.FileName;

using (StreamWriter writer = new StreamWriter(filename, false))

{

for (int i = 0; i < funcValue\_Table.RowCount; i++)

{

writer.Write(funcValue\_Table.Rows[i].Cells[0].Value);

writer.Write(" ");

writer.WriteLine(funcValue\_Table.Rows[i].Cells[1].Value);

}

}

chartTraktris\_Chart.SaveImage($"{filename}.png", System.Drawing.Imaging.ImageFormat.Png);

}

private void saveInitialData\_ToolStripMenuItem\_Click(object sender, EventArgs e)

{

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

return;

string filename = saveFileDialog.FileName;

using (StreamWriter writer = new StreamWriter(filename, false))

{

writer.WriteLine(a);

writer.WriteLine(step);

writer.WriteLine(leftBorder);

writer.WriteLine(rightBorder);

}

}

private void initialData\_Button\_Click(object sender, EventArgs e)

{

LoadDefaultParametrs();

}

private void clearChart\_Button\_Click(object sender, EventArgs e)

{

this.chartTraktris\_Chart.Series[0].Points.Clear();

}

private void downloadFromFile\_ToolStripMenuItem\_Click(object sender, EventArgs e)

{

string line;

if (openFileDialog.ShowDialog() == DialogResult.Cancel)

return;

string filename = openFileDialog.FileName;

using (StreamReader reader = new StreamReader(filename))

{

try

{

line = reader.ReadLine();

valueA\_TextBox.Text = line;

line = reader.ReadLine();

step\_TextBox.Text = line;

line = reader.ReadLine();

leftBorder\_TextBox.Text = line;

line = reader.ReadLine();

rightBorder\_TextBox.Text = line;

CheckInitialData();

}

catch

{

MessageBox.Show(this, "Произошла ошибка при попытке считывания данных из файла." +

"Отредактируйте данные и попробуйте снова", "Ошибка!",

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

private void buildChart\_Button\_Click(object sender, EventArgs e)

{

try

{

if (CheckInitialData())

{

funcValue\_Table.Rows.Clear();

this.chartTraktris\_Chart.Series[0].Points.Clear();

BuildChart();

}

else

{

funcValue\_Table.Rows.Clear();

this.chartTraktris\_Chart.Series[0].Points.Clear();

}

}

catch

{

MessageBox.Show(this, "Невозможно построить график", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void BuildChart()

{

double count = 0;

for (y = leftBorder; y <= rightBorder; y += step)

{

y = Math.Round(y, 3);

x = TractrixClass.EquationTractrix(a, y);

x = Math.Round(x, 3);

this.chartTraktris\_Chart.Series[0].Points.AddXY(-x, y);

funcValue\_Table.Rows.Add(y, -x);

count = y;

}

if ((count+step) == a)

{

this.chartTraktris\_Chart.Series[0].Points.AddXY(0, a);

funcValue\_Table.Rows.Add(a, 0);

}

else

{

this.chartTraktris\_Chart.Series[0].Points.AddXY(0, double.NaN);

}

for (y = count; y >= leftBorder; y -= step)

{

y = Math.Round(y, 3);

x = TractrixClass.EquationTractrix(a, y);

x = Math.Round(x, 3);

this.chartTraktris\_Chart.Series[0].Points.AddXY(x, y);

funcValue\_Table.Rows.Add(y, x);

}

}

private bool CheckInitialData()

{

try

{

leftBorder = Convert.ToDouble(leftBorder\_TextBox.Text);

rightBorder = Convert.ToDouble(rightBorder\_TextBox.Text);

step = Convert.ToDouble(step\_TextBox.Text);

a = Convert.ToDouble(valueA\_TextBox.Text);

leftBorder = Math.Round(leftBorder, 3);

rightBorder = Math.Round(rightBorder, 3);

step = Math.Round(step, 3);

a = Math.Round(a, 3);

leftBorder\_TextBox.Text = leftBorder.ToString();

rightBorder\_TextBox.Text = rightBorder.ToString();

step\_TextBox.Text = step.ToString();

valueA\_TextBox.Text = a.ToString();

if ((leftBorder <= 0) || (rightBorder < leftBorder))

{

MessageBox.Show(this, "Невозможно построить график в заданном интревале.\n" +

"Измените границы диапазона", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

return false;

}

y = leftBorder;

if ((leftBorder == rightBorder) || (y + step > rightBorder))

{

MessageBox.Show(this, "График выражется в точку.\n" +

"Измените границы диапазона", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

return false;

}

if ((a <= rightBorder))

{

MessageBox.Show(this, "Невозможно построить график с текущим коэффициентом.\n" +

"Измените коэффициент 'a' ", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

return false;

}

if ((step <= 0))

{

MessageBox.Show(this, "Невозможно построить график с заданным шагом.\n" +

"Измените шаг", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

return false;

}

}

catch

{

MessageBox.Show(this, "Невозможно построить график.\n" +

"Проверьте корректность введнных данных", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

return false;

}

return true;

}

private void LoadDefaultParametrs()

{

a = 1;

step = 0.1;

leftBorder = 0.1;

rightBorder = 0.9;

leftBorder\_TextBox.Text = leftBorder.ToString();

rightBorder\_TextBox.Text = rightBorder.ToString();

step\_TextBox.Text = step.ToString();

valueA\_TextBox.Text = a.ToString();

if (CheckInitialData())

{

funcValue\_Table.Rows.Clear();

this.chartTraktris\_Chart.Series[0].Points.Clear();

BuildChart();

}

}

}

}

Код с файла «GreetUserForm.cs»:

using System;

using System.Windows.Forms;

namespace WinFormsKR3

{

public partial class GreetUserForm : Form

{

public GreetUserForm()

{

InitializeComponent();

}

private void close\_Button\_Click(object sender, EventArgs e)

{

Close();

}

}

}