Изображение выглядит как цепь

Автоматически созданное описание

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) | |  | Автоматизированные системы обработки информации и управления | | |
| Форма обучения | |  | очная | | |
|  | |  |  | | |
| Факультет | |  | Информационных технологий и управления | | |
| Кафедра | |  | Систем автоматизированного проектирования и управления | | |
| Учебная дисциплина | |  | Разработка программных систем | | |
| Курс | II | | | Группа | 403 |

Отчёт по лабораторной работе № 3

Вариант № 9

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Исполнитель: |  |  |  |  |
| обучающийся группы 403 |  |  |  | Шишко Даниил Юрьевич |
|  |  | (дата, подпись) |  |  |
|  |  |  |  |  |
| Проверили: |  |  |  | Корниенко Иван Григорьевич |
|  |  | (дата, подпись) |  | Федин Алексей Константинович |

**Оглавление**

[Постановка задачи 3](#_Toc96790610)

[Исходные данные 3](#_Toc96790611)

[Особые ситуации 3](#_Toc96790612)

[Математические методы и алгоритмы решения задач 3](#_Toc96790613)

[Форматы представления данных 3](#_Toc96790614)

[Структура программы 5](#_Toc96790615)

[Блок-схемы алгоритмов программы 6](#_Toc96790616)

[Описание хода выполнения работы 6](#_Toc96790617)

[Результат работы программы 7](#_Toc96790618)

[Исходный текст программы 8](#_Toc96790619)

[Документирование и комментирование исходного текста 18](#_Toc96790620)

# 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.

# Исходные данные

В качестве исходных данных программа использует, вводимое пользователем константу ‘a’, левую границу графика и правую границу графика, также шаг, с периодичностью которого будет меняться ‘x’.

# Особые ситуации

Необходимо рассмотреть следующие особые ситуации.

* Не введены один или несколько изначальных данных.
* График вырождается в точку.
* Левая граница больше правой, или правая меньше левой.
* Значение правой границы при отрицательной константе больше константы ‘a’.
* Значение левой границы при положительной константе меньше ‘a’.

# Математические методы и алгоритмы решения задач

Находим y при известных x, a.

# Форматы представления данных

Программа использует следующие переменные

Таблица 1 – Переменные, используемые в программе

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Описание** |
| welcome | TextBox | Описание создателя |
| information | TextBox | Описание цели работы |
| dontShowAgreement | TextBox | Описание согласия на непоявление |
| close | Button | Закрыть форму |
| file | StreamWriter | Взаимодействие с файлом |
| Таблица 1 – Переменные, используемые в программе | | |
| multiply | int | Приближение изображения |
| error | string | Ошибка ввода |
| errorOfConst | string | Ошибка в константе |
| borderError | string | Ошибка в барьере |
| \_concetrarionService | IChirngauzSquareInfrastructure | Сервис для управления нахождениям точек функции |
| constA | string | Константа «а» |
| leftBorder | string | Правая граница |
| rightBorder | string | Левая граница |
| step | string | Шаг «х» |
| graphics | Graphics | Класс для рисования |
| points | List<ChirgauzSquareModel> | Значения функции в точках |
| somePointsUp | PointF[] | Значения функции при положительных y |
| somePointsDown | PointF[] | Значения функции при отрицательных y |
| centerX | int | Центр по «х» |
| centerY | int | Центр по «у» |
| iterator | int | Счётчик |
| pen | Pen | Ручка для рисования |
| drawFont | Font | Шрифт для рисования |
| drawBrush | SolidBrush | Кисть для рисования |
| yesOrNo | bool | Число из файла про соглашение |
| table | Table | Форма для таблицы |
| saveFileDialog | SaveFileDialog | Сохранение файла |
| filePath | string | Путь к файлу |
| openFileDialog | OpenFileDialog | Чтение файла |
| fileStream | Stream | Поток файла |
| package | ExcelPackage | Взаимодействие с Excel |
| sheet | ExcelWorksheet | Доступ к ячейкам таблицы |
| greeting | Greeting | Форма приветствия |
| tableWithData | PictureBox | Таблица значений функции |
| counter | int | Счётчик номеров строк |
| back | Buttom | Закрытие формы |
| save | Buttom | Сохранение таблицы |
| x | double | Икс функции |
| y | double | Игрик функции |
| numerator | double | Числитель |
| denominator | double | Знаменатель |

Таблица 3 – Пользовательские типы

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| ChirgauzSquareModel | Значение функции в точке |

В файле каждое сообщения хранятся в одной строке.

# Структура программы

Программа разделена на следующие модули:

1. Greeting.cs – приветствие;
2. MainForm.cs – главная форма;
3. Table.cs – таблица значений функции в точках;
4. IChirngauzSquareInfrastructure.cs – интерфейс для создания точек функции;
5. ChirgauzSquareModel.cs – точки функции;
6. ChirngauzSquareServices.cs – реализация расчёта значений функции.

Таблица 4 – Функции, составляющие программу

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| GreetingInner | Создание элементов на форме приветствия |
| Close\_Click | Закрытие формы приветствия |
| CheckData | Проверка правильности введенных данных |
| Calculate\_Click | Расчёт точек функции |
| GetCalculations | Возвращает значения точек функции |
| GetAxes | Рисует координатные оси |
| PictureBox1\_MouseWheel | Увеличение/уменьшение графика |
| CreateFormGreeting | Создание формы приветствия |
| Table\_Click | Создание формы с таблицей точек функции |
| SaveInitialData\_Click | Сохранение изначальных данных |
| GetInitial\_Click | Чтение изначальных данных из файла |
| aboutToolStripMenuItem\_Click | Открытие формы приветствия |
| SaveExcel\_Click | Сохранение данных в таблицу Excel |
| WriteToExcelTable | Создание таблица Excel |
| CreateEntity | Создание элементов на таблице значений функции |
| LoadDataInTAble | Прорисовка изначальных значений в таблицу |
| TableWithData\_MouseWheel | Прокрутка значений колёсиком |
| PaintData | Прорисовка не изначальных значений таблицы |
| Back\_Click | Закрыть форму |
| Save\_Click | Сохранить значения функции |
| GetPoints | Посчитать значения функции в точках |
| GetY | Посчитать игрик функции |

# Блок-схемы алгоритмов программы



Рисунок 1 – Блок-схема алгоритма решения

# Описание хода выполнения работы

1. В ходе работы было создано решение (Solution) в интегрированной среде разработки Microsoft Visual Studio 2015. В нём был создан проект.
2. Создано оформление главной формы, реализовано масштабирование графика.
3. Создана форма приветствия.
4. Создана форма таблицы значений функции, реализована прокрутка значений функции.
5. Исправлены возникающие ошибки при вводе некорректных данных.

# Результат работы программы

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3 – Результат работы программы

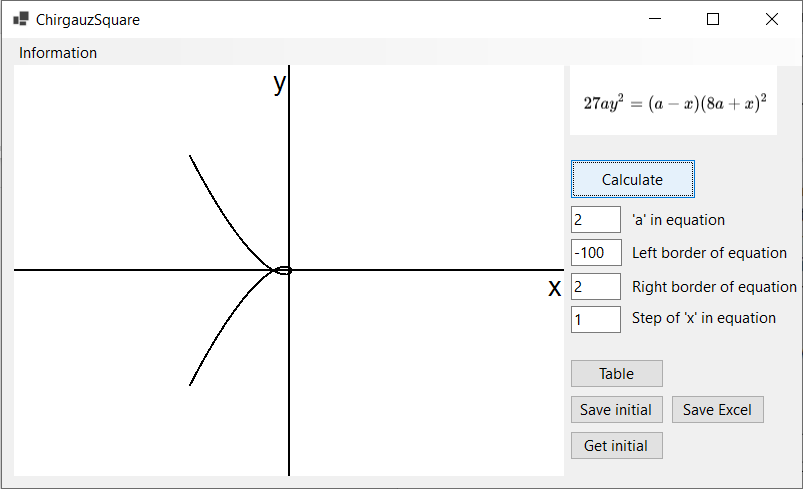


Рисунок 6 – Результат работы программы

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 7 – Результат работы программы

# Исходный текст программы

[ Начало программы ---]

[Greeting.cs]

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Text;

using System.Windows.Forms;

namespace FunctionGraph

{

public partial class Greeting : Form

{

public Greeting()

{

InitializeComponent();

GreetingInner();

}

CheckBox dontShowAgain;

public void GreetingInner()

{

this.Size = new Size(500, 350);

this.MinimumSize = this.Size;

this.MaximumSize = this.Size;

this.TopMost = true;

TextBox welcome = new TextBox();

welcome.ReadOnly = true;

welcome.Text = "Created by: Shishko Daniil Yrevich";

welcome.Size = new Size(this.Width / 2, 20);

welcome.Location = new Point(this.Width / 10, 5);

welcome.BorderStyle = BorderStyle.None;

this.Controls.Add(welcome);

TextBox information = new TextBox();

information.ReadOnly = true;

information.Text = "Option: 9" + Environment.NewLine + "Aim: Create Win Forms”+

“application to " +

"plotting a function graph and outputting a table of function values. User sets "

+

"rigth and left borders, step, coefficients (if presents). If unable to " +

"plot a finction graph in seted gap user get warning " +

"about this to change borders. If function graph in cause of" +

"coefficients becomes a dot or can't be ploted user also sees warning.";

information.Multiline = true;

information.TextAlign = HorizontalAlignment.Left;

information.Size = new Size(this.Width / 2 + 150, this.Height / 2);

information.Location = new Point(welcome.Location.X - 5, welcome.Height + 10);

information.BorderStyle = BorderStyle.None;

this.Controls.Add(information);

dontShowAgain = new CheckBox();

dontShowAgain.Enabled = true;

dontShowAgain.Size = new Size(20, welcome.Height);

dontShowAgain.Location = new Point(welcome.Location.X , information.Height + 20 +

welcome.Height);

dontShowAgain.Appearance = Appearance.Normal;

this.Controls.Add(dontShowAgain);

TextBox dontShowAgreement = new TextBox();

dontShowAgreement.ReadOnly = true;

dontShowAgreement.TextAlign = HorizontalAlignment.Left;

dontShowAgreement.Text = "Check if you don't want see this again";

dontShowAgreement.Size = new Size(welcome.Width + 20, 0);

dontShowAgreement.Location = new Point(welcome.Location.X + 30,

dontShowAgain.Location.Y + 3);

dontShowAgreement.BorderStyle = BorderStyle.None;

this.Controls.Add(dontShowAgreement);

Button close = new Button();

close.Text = "OK";

close.Location = new Point(dontShowAgain.Location.X + dontShowAgreement.Size.Width

+ dontShowAgain.Size.Width + 20, dontShowAgain.Location.Y);

close.Size = new Size(40, 30);

this.Controls.Add(close);

close.Click += Close\_Click;

}

public void Close\_Click(object sender, EventArgs e)

{

if (dontShowAgain.Checked)

{

StreamWriter file = new StreamWriter("Agreement.txt", false);

file.WriteLine(1);

file.Close();

}

if (!dontShowAgain.Checked)

{

StreamWriter file = new StreamWriter("Agreement.txt", false);

file.WriteLine(0);

file.Close();

}

this.Close();

}

}

}

[Greeting.cs]

[MainForm.cs]

using System;

using System.Drawing;

using System.Windows.Forms;

using Services.Infrustucture.Parametr;

using System.Windows.Input;

using System.IO;

using Services.ChirngauzSquare.Point;

using System.Collections.Generic;

using OfficeOpenXml;

namespace FunctionGraph

{

public partial class MainForm : Form

{

static int multiply = 1;

string error = "Wrong data";

string errorOfConst = "Graph is dot";

string borderError = "Wrong border edges";

private readonly IChirngauzSquareInfrastructure \_concentrationService;

public MainForm(IChirngauzSquareInfrastructure concentrationService)

{

\_concentrationService = concentrationService ?? throw new

ArgumentNullException(nameof(concentrationService));

InitializeComponent();

MouseWheel += new MouseEventHandler(PictureBox1\_MouseWheel);

CreateFormGreeting();

}

public void CheckData()

{

WrongData.Clear();

string constA = "", leftBorder = "", rightBorder = "", step = "";

if (ConstA.Text == "" || !double.TryParse(ConstA.Text, out double uselessResult))

{

WrongData.SetError(ConstA, error);

}

if (ConstA.Text != "" && double.TryParse(ConstA.Text, out uselessResult))

{

if (double.Parse(ConstA.Text) == 0)

{

WrongData.SetError(ConstA, errorOfConst);

}

else

{

constA = ConstA.Text;

}

}

if (LeftBorder.Text == "" || !double.TryParse(LeftBorder.Text, out uselessResult))

WrongData.SetError(LeftBorder, error);

else

{

leftBorder = LeftBorder.Text;

}

if (RightBorder.Text == "" || !double.TryParse(RightBorder.Text, out uselessResult))

WrongData.SetError(RightBorder, error);

else

{

rightBorder = RightBorder.Text;

}

if (Step.Text == "" || !double.TryParse(Step.Text, out uselessResult))

WrongData.SetError(Step, error);

else

{

step = Step.Text;

}

if (ConstA.Text != "" && ConstA.Text != "0" && WrongData.GetError(LeftBorder) == ""

&& WrongData.GetError(RightBorder) == "")

{

if (double.Parse(constA) > 0)

if (double.Parse(rightBorder) > double.Parse(constA))

WrongData.SetError(RightBorder, error);

if (double.Parse(constA) < 0)

if (double.Parse(leftBorder) < double.Parse(constA))

WrongData.SetError(LeftBorder, error);

}

if (RightBorder.Text != "" && double.TryParse(RightBorder.Text, out uselessResult))

{

if (LeftBorder.Text != "" && double.TryParse(LeftBorder.Text, out uselessResult))

{

if (double.Parse(LeftBorder.Text) > double.Parse(RightBorder.Text))

{

WrongData.SetError(LeftBorder, borderError);

WrongData.SetError(RightBorder, borderError);

}

}

}

}

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

{

CheckData();

Graphics graphics = pictureBox1.CreateGraphics();

graphics.Clear(Color.White);

GetAxes();

if (WrongData.GetError(LeftBorder) == "" && WrongData.GetError(RightBorder) == ""

&& WrongData.GetError(ConstA) == "" && WrongData.GetError(Step) == "")

{

var points = GetCalculations();

PointF[] somePointsUp = new PointF[points.Count / 2];

PointF[] somePointsDown = new PointF[points.Count / 2];

int centerX = pictureBox1.Width / 2;

int centerY = pictureBox1.Height / 2;

int iterator = 0;

for (int i = 0; i < points.Count; i += 2)

{

somePointsUp[iterator] = new PointF((float)(points[i].x \* multiply + centerX),

((float)points[i].y \* multiply + centerY));

somePointsDown[iterator] = new PointF((float)(points[i + 1].x \* multiply +

centerX), (float)(points[i + 1].y \* multiply + centerY));

iterator++;

}

Pen pen = new Pen(Color.Black, 2);

graphics.DrawCurve(pen, somePointsUp);

graphics.DrawCurve(pen, somePointsDown);

}

}

public List<ChirgauzSquareModel> GetCalculations()

{

CheckData();

if (WrongData.GetError(LeftBorder) == "" && WrongData.GetError(RightBorder) == ""

&& WrongData.GetError(ConstA) == "" && WrongData.GetError(Step) == "")

return \_concentrationService.GetPoints(double.Parse(ConstA.Text),

double.Parse(LeftBorder.Text), double.Parse(RightBorder.Text),

double.Parse(Step.Text));

return null;

}

private void GetAxes()

{

Graphics graphics = pictureBox1.CreateGraphics();

Pen pen = new Pen(Color.Black, 2);

graphics.DrawLine(pen, 0, pictureBox1.Height / 2, pictureBox1.Width,

pictureBox1.Height / 2);

graphics.DrawLine(pen, pictureBox1.Width / 2, 0, pictureBox1.Width / 2,

pictureBox1.Height);

Font drawFont = new Font("Arial", 16);

SolidBrush drawBrush = new SolidBrush(Color.Black);

Brush brush = Brushes.Black;

graphics.DrawString("x", drawFont, drawBrush, pictureBox1.Width - (20),

pictureBox1.Height / 2 + (0));

graphics.DrawString("y", drawFont, drawBrush, pictureBox1.Width / 2 - 40 / 2, 0);

}

private void PictureBox1\_MouseWheel(object sender, MouseEventArgs e)

{

if(e.Delta > 0 && multiply < 30)

{

multiply++;

Calculate\_Click(sender, e);

}

if(e.Delta < 0 && multiply > 1)

{

multiply--;

Calculate\_Click(sender, e);

}

}

private static void CreateFormGreeting()

{

StreamReader file = new StreamReader("Agreement.txt");

int Agreement = 0;

if (file != null)

{

string yesOrNo = file.ReadLine();

Agreement = int.Parse(yesOrNo);

}

file.Close();

if (Agreement != 1)

{

Greeting greeting = new Greeting();

greeting.Show();

}

}

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

{

var result = GetCalculations();

if (result != null) {

Table table = new Table(result);

table.Show();

}

}

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

{

CheckData();

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.InitialDirectory = "c:\\";

saveFileDialog.Filter = "txt files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*";

if (WrongData.GetError(LeftBorder) == "" && WrongData.GetError(RightBorder) == ""

&& WrongData.GetError(ConstA) == "" && WrongData.GetError(Step) == "")

{

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

{

var filePath = saveFileDialog.FileName;

StreamWriter file = new StreamWriter(filePath, false);

if(int.Parse(ConstA.Text) == 0)

{

WrongData.SetError(ConstA, errorOfConst);

}

else

{

file.WriteLine(double.Parse(ConstA.Text));

file.WriteLine(double.Parse(LeftBorder.Text));

file.WriteLine(double.Parse(RightBorder.Text));

file.WriteLine(double.Parse(Step.Text));

}

file.Close();

}

}

}

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

{

using (OpenFileDialog openFileDialog = new OpenFileDialog())

{

openFileDialog.InitialDirectory = "d:\\4 семестр\\РПС\\FunctionGraph";

openFileDialog.Filter = "txt files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*";

openFileDialog.FilterIndex = 2;

openFileDialog.RestoreDirectory = true;

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

{

var filePath = openFileDialog.FileName;

var fileStream = openFileDialog.OpenFile();

using (StreamReader reader = new StreamReader(fileStream))

{

ConstA.Text = reader.ReadLine();

LeftBorder.Text = reader.ReadLine();

RightBorder.Text = reader.ReadLine();

Step.Text = reader.ReadLine();

}

}

}

}

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

{

CreateFormGreeting();

}

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

{

WriteToExcelTable();

}

private void WriteToExcelTable()

{

CheckData();

if (WrongData.GetError(LeftBorder) == "" && WrongData.GetError(RightBorder) == ""

&& WrongData.GetError(ConstA) == "" && WrongData.GetError(Step) == "")

{

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.InitialDirectory = "d:\\4 семестр\\РПС\\FunctionGraph";

saveFileDialog.Filter = "Excel files (\*.xlsx)|\*.xlsx|All files (\*.xlsx)|\*.xlsx";

saveFileDialog.FilterIndex = 2;

saveFileDialog.RestoreDirectory = true;

var filePath = "";

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

{

filePath = saveFileDialog.FileName;

}

ExcelPackage.LicenseContext = LicenseContext.NonCommercial;

var package = new ExcelPackage();

var sheet = package.Workbook.Worksheets.Add("GraphPoint");

sheet.Cells[1,1].Value = "a - "; sheet.Cells[1, 2].Value =

double.Parse(ConstA.Text);

sheet.Cells[2, 1].Value = "Left border - "; sheet.Cells[2, 2].Value =

double.Parse(LeftBorder.Text);

sheet.Column(1).Width = 14;

sheet.Column(2).Width = 14;

sheet.Cells[3, 1].Value = "Right border - "; sheet.Cells[3, 2].Value =

double.Parse(RightBorder.Text);

sheet.Cells[4, 1].Value = "Step - "; sheet.Cells[4, 2].Value =

double.Parse(Step.Text);

sheet.Cells[5, 1, 5, 2].LoadFromArrays(new object[][] { new[] { "X", "Y" } });

sheet.Cells[5, 1].Style.HorizontalAlignment =

OfficeOpenXml.Style.ExcelHorizontalAlignment.Center;

sheet.Cells[5, 2].Style.HorizontalAlignment =

OfficeOpenXml.Style.ExcelHorizontalAlignment.Center;

var points = GetCalculations();

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

{

sheet.Cells[i + 6, 1].Value = points[i].x;

sheet.Cells[i + 6, 2].Value = points[i].y;

}

File.WriteAllBytes(filePath, package.GetAsByteArray());

}

}

}

}

[MainForm.cs]

[Table.cs]

using Services.ChirngauzSquare.Point;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Text;

using System.Windows.Forms;

namespace FunctionGraph

{

public partial class Table : Form

{

List<ChirgauzSquareModel> points;

PictureBox tableWithData;

int counter;

Pen pen = new Pen(Color.Black, 2);

SolidBrush drawBrush = new SolidBrush(Color.Black);

Font drawFont = new Font("Arial", 16);

public Table(List<ChirgauzSquareModel> outPoints)

{

InitializeComponent();

points = outPoints;

CreateEntity();

}

private void CreateEntity()

{

this.Size = new Size(500, 300);

this.MinimumSize = this.Size;

this.MaximumSize = this.Size;

this.TopMost = true;

tableWithData = new PictureBox();

tableWithData.Location = new Point(0, 0);

tableWithData.Size = new Size(this.Width, this.Height \* 2 / 3);

tableWithData.BackColor = Color.White;

this.Controls.Add(tableWithData);

Paint += new PaintEventHandler(LoadDataInTAble);

MouseWheel += new MouseEventHandler(TableWithData\_MouseWheel);

Button back = new Button();

back.Size = new Size(80, 30);

back.Location = new Point(40, tableWithData.Height + 10);

back.Text = "Return";

this.Controls.Add(back);

back.Click += Back\_Click;

Button save = new Button();

save.Size = new Size(back.Size.Width, back.Size.Height);

save.Location = new Point((this.Width / 2) + back.Size.Width + 20,

back.Location.Y);

save.Text = "Save";

this.Controls.Add(save);

save.Click += Save\_Click;

counter = 4;

}

private void LoadDataInTAble(object sender, PaintEventArgs e)

{

Graphics graphics = tableWithData.CreateGraphics();

graphics.Clear(Color.White);

graphics.DrawLine(pen, 120, 0, 120, tableWithData.Height);

graphics.DrawLine(pen, (tableWithData.Width / 2) - 10, 0, (tableWithData.Width / 2)

- 10, tableWithData.Height);

for (int i = 0; i < 5 && counter < points.Count; i++)

{

graphics.DrawLine(pen, 0, (tableWithData.Height / 5) \* i, tableWithData.Width,

(tableWithData.Height / 5) \* i);

graphics.DrawString((i + 1).ToString(), drawFont, drawBrush, tableWithData.Height

/ 20, (tableWithData.Height / 20) + (i \* 40));

graphics.DrawString(points[i].x.ToString(), drawFont, drawBrush,

(tableWithData.Width / 2) - (tableWithData.Width / 4),

(tableWithData.Height / 20) + (i \* 40));

graphics.DrawString(points[i].y.ToString(), drawFont, drawBrush,

(tableWithData.Width) - (tableWithData.Width / 2),

(tableWithData.Height / 20) + (i \* 40));

}

}

private void TableWithData\_MouseWheel(object sender, MouseEventArgs e)

{

if (e.Delta > 0 && counter - 6 >= 0)

{

counter--;

PaintData(counter - 5);

}

if (e.Delta < 0 && counter + 1 < points.Count)

{

counter++;

PaintData(counter - 5);

}

}

private void PaintData(int innerCounter)

{

Graphics graphics = tableWithData.CreateGraphics();

graphics.Clear(Color.White);

graphics.DrawLine(pen, 120, 0, 120, tableWithData.Height);

graphics.DrawLine(pen, (tableWithData.Width / 2) - 10, 0, (tableWithData.Width / 2)

- 10, tableWithData.Height);

for (int i = 0; i < 5; i++)

{

graphics.DrawLine(pen, 0, (tableWithData.Height / 5) \* i, tableWithData.Width,

(tableWithData.Height / 5) \* i);

graphics.DrawString((innerCounter + 1).ToString(), drawFont, drawBrush,

tableWithData.Height / 20, (tableWithData.Height / 20) + (i \* 40));

graphics.DrawString(points[innerCounter].x.ToString(), drawFont, drawBrush,

(tableWithData.Width / 2) - (tableWithData.Width / 4),

(tableWithData.Height / 20) + (i \* 40));

graphics.DrawString(points[innerCounter].y.ToString(), drawFont, drawBrush,

(tableWithData.Width) - (tableWithData.Width / 2),

(tableWithData.Height / 20) + (i \* 40));

innerCounter++;

}

}

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

{

this.Close();

}

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

{

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.InitialDirectory = "c:\\";

saveFileDialog.Filter = "txt files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*";

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

{

var filePath = saveFileDialog.FileName;

StreamWriter file = new StreamWriter(filePath, false);

foreach (var point in points)

{

file.WriteLine(point.x.ToString() + " " + point.y.ToString());

}

file.Close();

}

}

}

}

[Table.cs]

[IChirngauzSquareInfrastructure.cs]

using System;

using System.Collections.Generic;

using System.Text;

using Services.ChirngauzSquare.Point;

namespace Services.Infrustucture.Parametr

{

public interface IChirngauzSquareInfrastructure

{

public List<ChirgauzSquareModel> GetPoints(double a, double leftBorder, double rigthBorder, double step);

}

}

[IChirngauzSquareInfrastructure.cs]

[ChirgauzSquareModel.cs]

using System;

using System.Collections.Generic;

using System.Text;

namespace Services.ChirngauzSquare.Point

{

public class ChirgauzSquareModel

{

public double x { get; set; }

public double y { get; set; }

public ChirgauzSquareModel( double innerX, double innerY)

{

x = innerX;

y = innerY;

}

}

}

[ChirgauzSquareModel.cs]

[ChirngauzSquareServices.cs]

using System;

using System.Collections.Generic;

using System.Text;

using Services.Infrustucture.Parametr;

using Services.ChirngauzSquare.Point;

namespace Services.ChirngauzSquare.Services

{

public class ChirngauzSquareServices : IChirngauzSquareInfrastructure

{

public List<ChirgauzSquareModel> GetPoints(double a, double leftBorder, double

rigthBorder, double step)

{

List<ChirgauzSquareModel> points = new List<ChirgauzSquareModel>();

for(double x = leftBorder; x <= rigthBorder; x += step)

{

double y = GetY(x, a);

points.Add(new ChirgauzSquareModel((double)x, y));

points.Add(new ChirgauzSquareModel((double)x, -y));

}

return points;

}

private double GetY(double x, double a)

{

if (a > 0)

if(x > a)

throw new DivideByZeroException();

if(a < 0)

if (x < a)

throw new DivideByZeroException();

if (x + 0.1 > a && x - 0.1 < a)

return 0;

double y, numerator, denominator;

denominator = 27 \* a;

double leftTemp = a - x;

double rightTemp = (8 \* a) + x;

numerator = leftTemp \* Math.Pow(rightTemp, 2);

y = Math.Sqrt(numerator / denominator);

return y;

}

}

}

[ChirngauzSquareServices.cs]

[ --- Конец программы]

# Документирование и комментирование исходного текста