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 | I | | | Группа | 403 |

Отчёт по лабораторной работе № 4

Вариант № 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Исполнитель: |  |  |  |  |
| обучающийся группы 403 |  |  |  | Старков Силантий Денисович |
|  |  | (дата, подпись) |  |  |
|  |  |  |  |  |
| Проверили: |  |  |  | Корниенко Иван Григорьевич |
|  |  | (дата, подпись) |  | Федин Алексей Константинович |

Оглавление

[1. Постановка задачи 3](#_Toc99139425)

[2. Исходный данные 3](#_Toc99139426)

[3. Особые ситуации 3](#_Toc99139427)

[4. Математические методы и алгоритмы решения задач 3](#_Toc99139428)

[5. Форматы представления данных 4](#_Toc99139429)

[6. Структура программы 4](#_Toc99139430)

[7. Блок-схемы алгоритмов программы 6](#_Toc99139431)

[8. Описание хода выполнения лабораторной работы 6](#_Toc99139432)

[9. Результаты работы программы 7](#_Toc99139433)

[10. Исходный текст программы 8](#_Toc99139434)

## Постановка задачи

Необходимо написать приложение для хранения списка должников с использованием технологии WPF используя паттерн MVVM. Программа должна позволять добавлять новые сущности с использованием интерфейса и редактировать существующие. Сущности, добавленные в программу, должны сохраняться между запусками приложения. Для хранения данных необходимо спользовать СУБД SqLite. Необходимо предусмотреть возможность сохранения списка существующих сущностей в файл.

## Исходный данные

В качестве исходных данных программа использует информацию, уже хранящуюся в базе данных, и вводимые пользователем данные, которые присваиваются параметрам новой или уже существующей сущности.

## Особые ситуации

Необходимо рассмотреть следующие особые ситуации.

* Если пользователь ввёл значение, не соответствующее операбельному типу, программа осуществляет автоматическую валидацию и удаляет это значение.
* Если пользователь не установил фотографию профиля должника, фотография установится автоматически.
* Если пользователь ввел некорректное значение, будет показано сообщение об ощибке.

## Математические методы и алгоритмы решения задач

Согласно постановке задачи программа должна предоставлять возможность работы с набором сущностей-должников.

## Форматы представления данных

Программа использует следующие переменные:

Таблица 1 – Переменные, используемые в программе

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Описание |
| \_container | IContainer | Стандартный контейнер для разделения пользовательского интерфейса и бизнес-логики приложения |
| \_grafServ | IGraphics | Ссылка на объект, вычисляющий значения точек на графика |
| excel\_app | Application | Экземпляр приложения Excel для экспорта данных |
| IDictionary<double, double> | Points | Словарь для составления графика синусоиды |
| xMin,xMax,yMin,yMax | int | Пределы для построения графика |
| saveFileDialog | SaveFileDialog | Путь до файла сохранения |
| openFileDialog | OpenFileDialog | Путь до файла с исходными данными |
| showGreeting | bool | Переменная, по которой определяется, показывать приветствие для запуска или нет |
| step | double | Переменная, хранящая значение удаления оп горизонтальной оси точек графика |

Для нахождения значений точек графика используется класс Sinusoid, в котором определены соответствующие методы.

## Структура программы

Программа разделена на следующие модули:

1. Program: начальный модуль с функцией Main. В нем происходит запуск главного окна приложения.
2. MainForm: в нем находятся функции, инициализирующие компоненты формы и осуществляющие связь компонентов формы с бизнес-логикой проиложения.
3. IGraphics: интерфейс для переопределения методов определения значений точек.
4. Sinusoid: модуль, реализующий методы нахождения точек гарфика.
5. SinusoidTests: модуль, хранящий тесты основных расчётных функций.
6. MessageForm: модуль, инициализирующий компоненты окна с дополнительной информацией.
7. App.config: модуль, необходимый для сохранения пользовательских настроек.

Программа разделена на несколько функций:

Таблица 3 – Функции, составляющие программу

|  |  |
| --- | --- |
| Имя | Описание |
| CalculatePoints | Функция для нахождения точек графика функции |
| RegisterTypes | Функция для построения контейнера |
| exportToExelToolStripMenuItem\_Click | Функция для экспорта в Excel |
| openFileToolStripMenuItem\_Click | Функция для экспорта входящих данных |
| ShowGreeting | Функция, показывающая приветствие при запуске программы |
| SavelData | Функция, сохраняющая исходные данные |
| SavelResults | Функция, сохраняющая дерево в файл |
| saveInnerDataToolStripMenuItem\_Click | Функция для сохранения исходных данных |
| saveResultsToolStripMenuItem\_Click | Функция для сохранения результатов программы |
| Validation | Функция для валидации ввода с клавиатуры |
| Start | Функция, запускающая основные вычисления |

## Блок-схемы алгоритмов программы

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 – Блок схема алгоритма вычисления точек графика

## Описание хода выполнения лабораторной работы

1. В ходе лабораторной работы было создано решение (Solution) в интегрированной среде разработки Microsoft Visual Studio 2019. В нём был создан проект.
2. После написания приложения, было выяснено, что при определенных параметрах построения график выходит за пределы видимой области. Проблема была решена автоматическим изменением горизонтальной оси графика.
3. После написания приложения, было выяснено, что функциональных возможностей MessageBox для вывода сообщений недостаточно. Проблема была решена созданием аналога MessageBox с расширенными функциональными возможностями.

## Результаты работы программы

В результате работы программы, на экран выводится график функции и таблица значений точек.

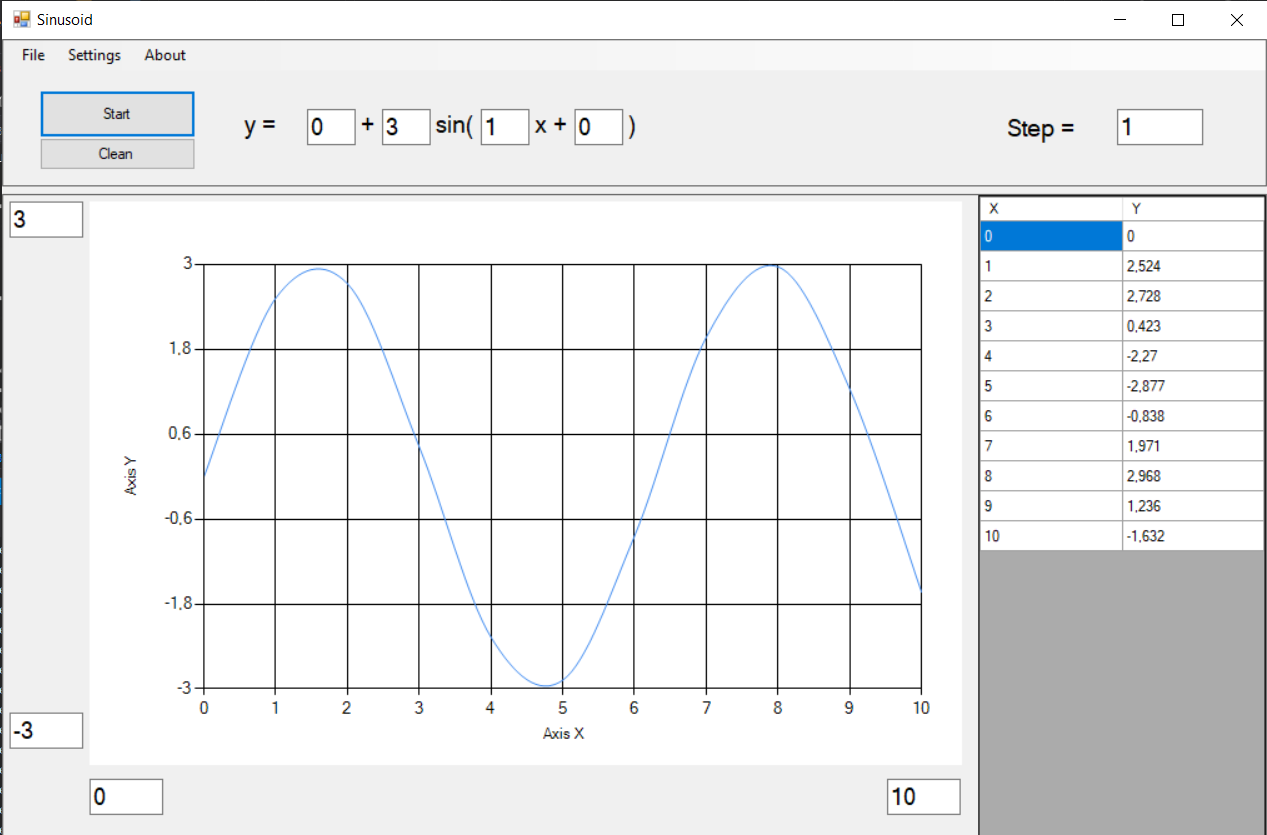


Рисунок 3 – Результат работы программы

## Исходный текст программы

[Исходный текст прогарммы --]

//Program.cs

using Autofac;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

using Lab3.Servises;

namespace Lab3

{

static class Program

{

private static IContainer \_container;

[STAThread]

static void Main()

{

RegisterTypes();

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(\_container.Resolve<MainForm>());

}

private static void RegisterTypes()

{

var builder = new ContainerBuilder();

builder.RegisterType<Sinusoid>().As<IGraphics>();

builder.RegisterType<MainForm>();

\_container = builder.Build();

}

}

}

//MessegeForm.sc

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 Lab3

{

public partial class MessageForm : Form

{

public MessageForm(int left, int top, string message)

{

InitializeComponent();

this.StartPosition = FormStartPosition.Manual;

this.textMessage.Text = message;

Point magin = textMessage.Location;

this.Size = new Size(magin.X \* 2 + textMessage.Width, magin.Y \* 3 + textMessage.Height);

this.MaximumSize = this.Size;

this.MinimumSize = this.Size;

this.FormBorderStyle = FormBorderStyle.FixedSingle;

this.Location = new Point(left - Width / 2, top - Height / 2);

}

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

{

}

}

}

//MainForm.sc

using Lab3.Servises;

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;

using System.Configuration;

using System.Windows.Forms.DataVisualization.Charting;

namespace Lab3 {

public partial class MainForm : Form {

private readonly IGraphics \_grafServ;

bool onValidation = true;

public MainForm(IGraphics grafServ) {

InitializeComponent();

\_grafServ = grafServ;

ChangeChartArea();

chart.ChartAreas[0].AxisY.Title = "Axis Y";

chart.ChartAreas[0].AxisX.Title = "Axis X";

chart.Series[0].Points.Add();

textBoxA.TextChanged += new EventHandler(MoveFormula);

textBoxB.TextChanged += new EventHandler(MoveFormula);

textBoxC.TextChanged += new EventHandler(MoveFormula);

textBoxD.TextChanged += new EventHandler(MoveFormula);

textBoxStep.TextChanged += new EventHandler(MoveFormula);

textBoxXMin.TextChanged += new EventHandler((s, e) => Validation(s, false));

textBoxXMax.TextChanged += new EventHandler((s, e) => Validation(s, false));

textBoxYMin.TextChanged += new EventHandler((s, e) => Validation(s, false));

textBoxYMax.TextChanged += new EventHandler((s, e) => Validation(s, false));

MoveFormula(textBoxA, default);

}

#region Greeting

private void ShowGreeting(object sender, EventArgs e) {

if (bool.Parse(ConfigurationManager.AppSettings["showGreeting"])) {

ShowGreetingForm();

showGreetingsToolStripMenuItem.Checked = true;

}

else {

showGreetingsToolStripMenuItem.Checked = false;

}

}

private void ShowGreetingForm() {

new Lab3.MessageForm(

Left + Width / 2,

Top + Height / 2,

"Студент СПБГТИ(ТУ) \n" +

"Старков Силантий Денисович \n" +

"403 группа\nЛабораторная работа №1\n" +

"Вариант №1\nПостроение простого бинарного дерева").ShowDialog();

}

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

ShowGreetingForm();

}

private void showGreetingsToolStripMenuItem\_Click(object sender, EventArgs e) {

var config = ConfigurationManager.OpenExeConfiguration(ConfigurationUserLevel.None);

bool show = !bool.Parse(ConfigurationManager.AppSettings["showGreeting"]);

config.AppSettings.Settings["showGreeting"].Value = (show).ToString();

config.Save();

showGreetingsToolStripMenuItem.Checked = show;

ConfigurationManager.RefreshSection("appSettings");

}

#endregion

#region File

private void saveInnerDataToolStripMenuItem\_Click(object sender, EventArgs e) {

SaveFileDialog saveFileDialog = new SaveFileDialog {

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

};

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

return;

string filename = saveFileDialog.FileName;

StringBuilder data = new StringBuilder();

data.Append("A = " + textBoxA.Text + "\n");

data.Append("B = " + textBoxB.Text + "\n");

data.Append("C = " + textBoxC.Text + "\n");

data.Append("D = " + textBoxD.Text);

System.IO.File.WriteAllText(filename, data.ToString());

}

private void saveResultsToolStripMenuItem\_Click(object sender, EventArgs e) {

SaveFileDialog saveFileDialog = new SaveFileDialog {

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

};

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

return;

string filename = saveFileDialog.FileName;

StringBuilder data = new StringBuilder();

foreach (System.Windows.Forms.DataVisualization.Charting.DataPoint item in chart.Series[0].Points) {

data.Append("X = " + item.XValue + " Y = " + item.YValues[0] + "\n");

}

System.IO.File.WriteAllText(filename, data.ToString());

}

private void openFileToolStripMenuItem\_Click(object sender, EventArgs e) {

OpenFileDialog openFileDialog = new OpenFileDialog {

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

};

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

return;

string filename = openFileDialog.FileName;

string[] data = System.IO.File.ReadAllText(filename).Split('\n');

try {

textBoxA.Text = data[0].Substring(data[0].IndexOf("= ") + "= ".Length);

textBoxB.Text = data[1].Substring(data[1].IndexOf("= ") + "= ".Length);

textBoxC.Text = data[2].Substring(data[2].IndexOf("= ") + "= ".Length);

textBoxD.Text = data[3].Substring(data[3].IndexOf("= ") + "= ".Length);

startBtn\_Click(sender, e);

}

catch (Exception) {

new Lab3.MessageForm(

Left + Width / 2,

Top + Height / 2,

"Ошибка:\nНеверный формат данных.").ShowDialog();

}

}

private void exportToExelToolStripMenuItem\_Click(object sender, EventArgs e) {

Microsoft.Office.Interop.Excel.Application excel\_app =

new Microsoft.Office.Interop.Excel.Application();

Visible = true;

Microsoft.Office.Interop.Excel.Workbook workbook =

excel\_app.Workbooks.Add(Type.Missing);

Microsoft.Office.Interop.Excel.Worksheet sheet = (Microsoft.Office.Interop.Excel.Worksheet)workbook.Sheets[1];

object[,] values = new object[gridView.Rows.Count + 1, 2];

values[0, 0] = "x";

values[0, 1] = "y";

for (int i = 0; i < gridView.Rows.Count; i++) {

values[i + 1, 0] = gridView.Rows[i].Cells[0].Value;

values[i + 1, 1] = gridView.Rows[i].Cells[1].Value;

}

Microsoft.Office.Interop.Excel.Range value\_range = sheet.get\_Range("A1", "B" + (gridView.Rows.Count + 1).ToString());

value\_range.Value2 = values;

Microsoft.Office.Interop.Excel.Range colA = (Microsoft.Office.Interop.Excel.Range)sheet.Columns[1, Type.Missing];

colA.ColumnWidth = 12;

Microsoft.Office.Interop.Excel.Shape chart\_shape = sheet.Shapes.AddChart(

Microsoft.Office.Interop.Excel.XlChartType.xlLine, 200, 5, 300, 200);

Microsoft.Office.Interop.Excel.Chart chart = chart\_shape.Chart;

Microsoft.Office.Interop.Excel.Range chart\_range = sheet.get\_Range("B2", "B" + (gridView.Rows.Count + 1).ToString());

chart.SetSourceData(chart\_range, Microsoft.Office.Interop.Excel.XlRowCol.xlColumns);

Microsoft.Office.Interop.Excel.Range axis\_range = sheet.get\_Range("A2", "A" + (gridView.Rows.Count + 1).ToString());

Microsoft.Office.Interop.Excel.Series series = (Microsoft.Office.Interop.Excel.Series)chart.SeriesCollection(1);

series.XValues = axis\_range;

/\*string filename = Application.StartupPath + "\\Chart.xlsx";

System.IO.File.Delete(filename);\*/

SaveFileDialog saveFileDialog = new SaveFileDialog {

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

};

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

return;

string filename = saveFileDialog.FileName;

workbook.SaveAs(filename, Type.Missing, Type.Missing,

Type.Missing, Type.Missing, Type.Missing,

Microsoft.Office.Interop.Excel.XlSaveAsAccessMode.xlExclusive, Type.Missing,

Type.Missing, Type.Missing, Type.Missing,

Type.Missing);

workbook.Close(true, Type.Missing, Type.Missing);

excel\_app.Quit();

}

#endregion

#region UI

private void MoveFormula(object sender, EventArgs e) {

Validation(sender, true);

textBoxA.Size = TextRenderer.MeasureText(textBoxA.Text + ' ', textBoxA.Font);

textBoxB.Size = TextRenderer.MeasureText(textBoxB.Text + ' ', textBoxB.Font);

textBoxC.Size = TextRenderer.MeasureText(textBoxC.Text + ' ', textBoxC.Font);

textBoxD.Size = TextRenderer.MeasureText(textBoxD.Text + ' ', textBoxD.Font);

labelPlus.Location = new Point(textBoxA.Location.X + textBoxA.Size.Width, labelPlus.Location.Y);

textBoxB.Location = new Point(labelPlus.Location.X + labelPlus.Size.Width, textBoxB.Location.Y);

labelSin.Location = new Point(textBoxB.Location.X + textBoxB.Size.Width, labelSin.Location.Y);

textBoxC.Location = new Point(labelSin.Location.X + labelSin.Size.Width, textBoxC.Location.Y);

labelXPlus.Location = new Point(textBoxC.Location.X + textBoxC.Size.Width, labelXPlus.Location.Y);

textBoxD.Location = new Point(labelXPlus.Location.X + labelXPlus.Size.Width, textBoxD.Location.Y);

labelClose.Location = new Point(textBoxD.Location.X + textBoxD.Size.Width, labelClose.Location.Y);

}

private void MoveGrafic(object sender, EventArgs e) {

TextBox tb = (TextBox)sender;

tb.Size = TextRenderer.MeasureText(tb.Text + ' ', tb.Font);

chart.Location = new Point(tb.Location.X + tb.Width, 0);

chart.Width = splitContainerMain.Width - chart.Location.X - gridView.Width;

}

/\* private void chart\_SizeChanged(object sender, EventArgs e)

{

chart.Location = new Point(Math.Max(textBoxYMax.Location.X + textBoxYMax.Width, textBoxYMin.Location.X + textBoxYMin.Width), 0);

chart.Width = gridView.Location.X- chart.Location.X;

chart.Height = textBoxXMax.Location.Y- chart.Location.Y;

}\*/

private void ChangeChartArea() {

if (onValidation) {

if (int.Parse(textBoxXMin.Text) < int.Parse(textBoxXMax.Text)) {

chart.ChartAreas.ElementAt(0).AxisX.Minimum = int.Parse(textBoxXMin.Text);

chart.ChartAreas.ElementAt(0).AxisX.Maximum = int.Parse(textBoxXMax.Text);

}

else {

new Lab3.MessageForm(

Left + Width / 2,

Top + Height / 2,

"Ошибка:\nМаксимальное значение Х меньше минимального!").ShowDialog();

}

if (int.Parse(textBoxYMin.Text) < int.Parse(textBoxYMax.Text)) {

chart.ChartAreas.ElementAt(0).AxisY.Minimum = int.Parse(textBoxYMin.Text);

chart.ChartAreas.ElementAt(0).AxisY.Maximum = int.Parse(textBoxYMax.Text);

}

else {

new Lab3.MessageForm(

Left + Width / 2,

Top + Height / 2,

"Ошибка:\nМаксимальное значение Y меньше минимального!").ShowDialog();

}

}

}

private void cleanBtn\_Click(object sender, EventArgs e) {

textBoxA.Text = "0";

textBoxB.Text = "0";

textBoxC.Text = "0";

textBoxD.Text = "0";

textBoxXMin.Text = "0";

textBoxXMax.Text = "10";

textBoxYMin.Text = "0";

textBoxYMax.Text = "10";

chart.Series[0].Points.Clear();

chart.Series[0].Points.Add();

ChangeChartArea();

gridView.Rows.Clear();

}

private void Validation(object sender, bool doDouble) {

TextBox tb = (TextBox)sender;

string sequenceTextNewValue = tb.Text;

if (sequenceTextNewValue.Length > 0) {

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

if (sequenceTextNewValue.Length > 0 && (sequenceTextNewValue.ElementAt(i) < '0' || sequenceTextNewValue.ElementAt(i) > '9') && sequenceTextNewValue.ElementAt(i) != '-' && (doDouble ? sequenceTextNewValue.ElementAt(i) != ',' : true)) {

sequenceTextNewValue = sequenceTextNewValue.Remove(i, 1);

i--;

}

}

if (tb.Name == textBoxStep.Name) {

for (int a = 0; a < sequenceTextNewValue.Length; a++) {

if (sequenceTextNewValue.ElementAt(a) == '-') {

sequenceTextNewValue = sequenceTextNewValue.Remove(a, 1);

a--;

}

}

}

if (doDouble) {

while (sequenceTextNewValue.IndexOf(',') != sequenceTextNewValue.LastIndexOf(',')) {

sequenceTextNewValue = sequenceTextNewValue.Remove(sequenceTextNewValue.LastIndexOf(','), 1);

}

while (sequenceTextNewValue.IndexOf("-0") != -1)

sequenceTextNewValue = sequenceTextNewValue.Replace("-0", "0");

while (sequenceTextNewValue.Length > 1 && sequenceTextNewValue.ElementAt(0) == '0' && sequenceTextNewValue.ElementAt(1) != ',') {

sequenceTextNewValue = sequenceTextNewValue.Remove(0, 1);

}

if (sequenceTextNewValue.LastIndexOf("-") > 0)

sequenceTextNewValue = sequenceTextNewValue.Remove(sequenceTextNewValue.LastIndexOf("-"), 1);

tb.Text = sequenceTextNewValue;

}

else {

while (sequenceTextNewValue.IndexOf("-0") != -1)

sequenceTextNewValue = sequenceTextNewValue.Replace("-0", "0");

while (sequenceTextNewValue.Length > 1 && sequenceTextNewValue.ElementAt(0) == '0') {

sequenceTextNewValue = sequenceTextNewValue.Remove(0, 1);

}

if (sequenceTextNewValue.LastIndexOf("-") > 0)

sequenceTextNewValue = sequenceTextNewValue.Remove(sequenceTextNewValue.LastIndexOf("-"), 1);

tb.Text = sequenceTextNewValue;

if ((sequenceTextNewValue.Length > 1 && sequenceTextNewValue.Contains("-") ||

sequenceTextNewValue.Length > 0 && !sequenceTextNewValue.Contains("-")))

ChangeChartArea();

}

return;

}

}

#endregion

private void startBtn\_Click(object sender, EventArgs e) {

onValidation = false;

double.TryParse(textBoxA.Text, out double a);

double.TryParse(textBoxB.Text, out double b);

double.TryParse(textBoxC.Text, out double c);

double.TryParse(textBoxD.Text, out double d);

double.TryParse(textBoxStep.Text, out double step);

int.TryParse(textBoxYMax.Text, out int yMax);

int.TryParse(textBoxYMin.Text, out int yMin);

int.TryParse(textBoxXMax.Text, out int xMax);

int.TryParse(textBoxXMin.Text, out int xMin);

var Points = \_grafServ.CalculatePoints(a, b, c, d, step, yMax, yMin, xMax, xMin);

chart.Series[0].Points.Clear();

while (gridView.Rows.Count != 0) {

gridView.Rows.RemoveAt(gridView.Rows.Count - 1);

}

chart.ChartAreas[0].AxisY.MajorTickMark.Interval = 0;

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

chart.Series[0].Points.AddXY(Points.Keys.ElementAt(i), Points.Values.ElementAt(i));

gridView.Rows.Add(Points.Keys.ElementAt(i), Points.Values.ElementAt(i));

}

textBoxYMax.Text = ((int)(chart.Series[0].Points.FindMaxByValue().YValues[0] + 1)).ToString();

textBoxYMin.Text = ((int)(chart.Series[0].Points.FindMinByValue().YValues[0] - 1)).ToString();

onValidation = true;

ChangeChartArea();

}

private void textBoxXMax\_TextChanged(object sender, EventArgs e) {

}

}

}

//App.config

<?xml version="1.0" encoding="utf-8"?>

<configuration>

<startup>

<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.6.1"/>

</startup>

<runtime>

<assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">

<dependentAssembly>

<assemblyIdentity name="System.Memory" publicKeyToken="cc7b13ffcd2ddd51" culture="neutral"/>

<bindingRedirect oldVersion="0.0.0.0-4.0.1.1" newVersion="4.0.1.1"/>

</dependentAssembly>

</assemblyBinding>

</runtime>

<appSettings>

<add key="showGreeting" value="true"/>

</appSettings>

</configuration>

//SinusoidTests.sc

using Microsoft.VisualStudio.TestTools.UnitTesting;

using Lab3.Servises;

using System;

using System.Collections.Generic;

using System.Text;

using System.Collections;

namespace Lab3.Servises.Tests {

[TestClass()]

public class SinusoidTests {

private static Sinusoid sinusoid = new Sinusoid();

[TestMethod()]

public void CalculatePointsTest1() {

IDictionary<double, double> Points = sinusoid.CalculatePoints(0, 0, 1, 0, 1, 10, 0, 10, 0);

List<double> p = new List<double>(Points.Values);

List<double> q = new List<double>() { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };

if (p.Count != q.Count)

Assert.Fail();

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

if (p[i] != q[i])

Assert.Fail();

}

}

[TestMethod()]

public void CalculatePointsTest2() {

IDictionary<double, double> Points = sinusoid.CalculatePoints(0, 1, 1, 0, 1, 10, 0, 10, 0);

List<double> p = new List<double>(Points.Values);

List<double> q = new List<double>() { 0, 0.841, 0.909, 0.141, -0.757, -0.959, -0.279, 0.657, 0.989, 0.412, -0.544 };

if (p.Count != q.Count)

Assert.Fail();

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

if (p[i] != q[i])

Assert.Fail();

}

}

[TestMethod()]

public void CalculatePointsTest3() {

IDictionary<double, double> Points = sinusoid.CalculatePoints(22, 1, 1, 0, 1, 10, 0, 10, 0);

List<double> p = new List<double>(Points.Values);

List<double> q = new List<double>() { 22, 22.841, 22.909, 22.141, 21.243, 21.041, 21.721, 22.657, 22.989, 22.412, 21.456 };

if (p.Count != q.Count)

Assert.Fail();

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

if (p[i] != q[i])

Assert.Fail();

}

}

[TestMethod()]

public void CalculatePointsTest4() {

IDictionary<double, double> Points = sinusoid.CalculatePoints(0, 1, 1, -100, 1, 10, 0, 10, 0);

List<double> p = new List<double>(Points.Values);

List<double> q = new List<double>() { 0.506, 0.999, 0.573, -0.38, -0.984, -0.683, 0.245, 0.948, 0.779, -0.106, -0.894 };

if (p.Count != q.Count)

Assert.Fail();

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

if (p[i] != q[i])

Assert.Fail();

}

}

}

}

//IGraphics.sc

using System;

using System.Collections.Generic;

using System.Text;

namespace Lab3.Servises

{

public interface IGraphics

{

IDictionary<double, double> CalculatePoints(

double a,

double b,

double c,

double d,

double step,

int yMax,

int yMin,

int xMax,

int xMin

);

}

}

//Sinusoid.sc

using System;

using System.Collections.Generic;

namespace Lab3.Servises {

public class Sinusoid : IGraphics {

public IDictionary<double, double> CalculatePoints(

double a,

double b,

double c,

double d,

double step,

int yMax,

int yMin,

int xMax,

int xMin

) {

IDictionary<double, double> Points = new Dictionary<double, double>();

for (double i = xMin; i <= xMax; i += step) {

double y =Math.Round( a + b \* Math.Sin(c \* i + d),3);

Points.Add(i, y);

}

return Points;

}

}

}

[-- Конец программы]