

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 | |  | Разработка программных систем | | |
| Курс | II | | | Группа | 423 |

Отчёт по контрольной работе №3

Вариант № 14

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Исполнитель: |  |  |  |  |
| обучающийся группы 423 |  |  |  | Орехов Даниил Сергеевич |
|  |  | (дата, подпись) |  |  |
|  |  |  |  |  |
| Проверили: |  |  |  | Корниенко Иван Григорьевич |
|  |  | (дата, подпись) |  | Федин Алексей Константинович |
|  |  |  |  |  |

Оглавление

[1 Постановка задачи 3](#_Toc161642198)

[2 Исходные данные 3](#_Toc161642199)

[3 Особые ситуации 3](#_Toc161642200)

[4 Математические методы и алгоритмы решения задач 3](#_Toc161642201)

[5 Блок-схема алгоритма решения задачи 4](#_Toc161642202)

[6 Форматы представления данных 5](#_Toc161642203)

[7 Структура программы 5](#_Toc161642204)

[8 Описание хода выполнения 5](#_Toc161642205)

[9 Результаты работы программы 6](#_Toc161642206)

[10 Выводы по заданию 6](#_Toc161642207)

[11 Исходный код полученного программного решения 6](#_Toc161642208)

## 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.

## 2 Исходные данные

Исходные данные состоят из коэффициента «R», величины шага, верхней и нижней границы и уравнения Квадратрисы.

## 3 Особые ситуации

Необходимо рассмотреть следующие особые ситуации:

- если пользователь ввёл некорректные данные, программа выделит соответствующую ошибку и не позволит пользователю продолжить,

- если при попытке считать данные из файла программа не сможет это сделать, она выдаст соответствующую ошибку и попросит повторить ввод,

- если график не может быть построен на отрезке или имеет на нём лишь одну точку, программа выдаст соответствующую ошибку и попросит повторить ввод.

## 4 Математические методы и алгоритмы решения задач

Согласно поставленной задаче, необходимо построить график и таблицу значений, исходя из уравнения Квадратрисы и исходных данных.

## 5 Блок-схема алгоритма решения задачи

На рисунке 1 представлена блок-схема алгоритма решения задачи.



Рисунок 1 – Блок-схема алгоритма решения задачи

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 6 Форматы представления данных Таблица 1 - Основные переменные, используемые в программе   |  |  |  | | --- | --- | --- | | Имя | Тип | Описание | | R | double | Коэффициент уравнения | | step | double | Шаг расчёта | | y1 | double | Нижняя граница графика | | y2 | double | Верхняя граница графика | | XList | List<double> | Массив координат х | | YList | List<double> | Массив координат у | | x | double | Координата х | | y | double | Координата у |  7 Структура программы Таблица 2 – Методы программы   |  |  |  | | --- | --- | --- | | Класс | Имя метода | Описание | | Form1 | CountDigitsAfterDecimal | Округление числа с плавающей запятой | | BuildGraph | Построение графика | | TextBoxGeneral\_KeyPress | Обработка ввода пользователя | | TextBoxPos\_KeyPress | Обработка ввода пользователя положительных чисел | |  | TextBox\_TextChanged | Проверка всех введенных полей | |  | BuildBtn\_Click | Кнопка построения графика | |  | LoadBtn\_Click | Кнопка загрузки данных | |  | SaveBtn\_Click | Кнопка сохранения данных | |  | InfoBtn\_Click | Кнопка вывода информации | | InfoForm | InfoForm | Информация о программе | |  |

## 8 Описание хода выполнения

Для выполнения задания необходимо было дополнительно изучить принципы работы с Windows Forms и особенности работы языка C# в них.

Возникли сложности при работе с Windows Forms, а также при работе с различными встроенными классами в С#. Проблемы были ликвидированы при помощи специализированной литературы и интернет-ресурсов.

## 9 Результаты работы программы

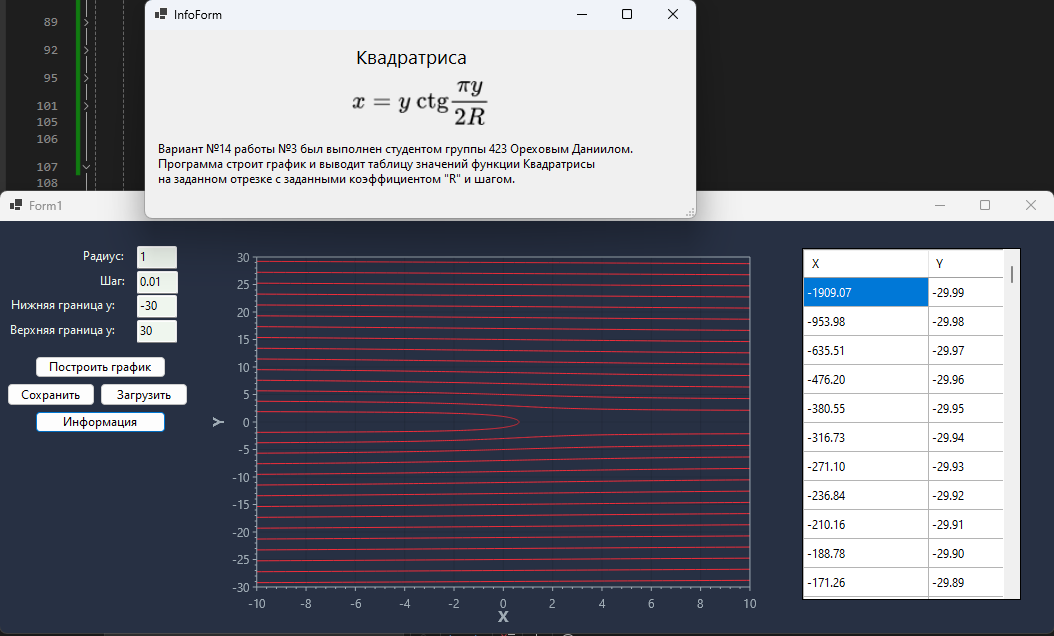


Рисунок 3 – Экранная копия результата работы разработанной программы

## 10 Выводы по заданию

В ходе выполнения задания были изучены методы работы с языком С# и Windows Forms в частности, с внешними библиотеками и менеджером библиотек NuGet, а также устранены недостатки в знаниях и получены новые навыки программирования.

## 11 Исходный код полученного программного решения

//Form1.cs

using ScottPlot;

using ScottPlot.AxisLimitCalculators;

using System.Globalization;

namespace WinFormsApp1 {

public partial class Form1 : Form {

readonly ScottPlot.WinForms.FormsPlot fp;

public Form1() {

CultureInfo.DefaultThreadCurrentCulture = new CultureInfo("en-US");

InitializeComponent();

fp = new() { Dock = DockStyle.Fill };

fp.Plot.XLabel("X");

fp.Plot.YLabel("Y");

fp.Plot.Style.Background(figure: ScottPlot.Color.FromHex("#273043"),

data: ScottPlot.Color.FromHex("#273043"));

fp.Plot.Style.ColorAxes(ScottPlot.Color.FromHex("#a0acb5"));

panel1.Controls.Add(fp);

}

private static int CountDigitsAfterDecimalPoint(double number) {

string numberString =

number.ToString("0.#############################");

int decimalPointIndex = numberString.IndexOf('.');

int digitsAfterDecimal = 0;

if (decimalPointIndex != -1) {

digitsAfterDecimal = numberString.Length - decimalPointIndex - 1;

}

return digitsAfterDecimal;

}

private void TextBoxGeneral\_KeyPress(object sender, KeyPressEventArgs e) {

if (!double.TryParse(y1Box.Text + e.KeyChar.ToString(), out double a)

&& e.KeyChar != 8 && e.KeyChar != '.' &&

e.KeyChar != ',' && e.KeyChar != '-') {

e.Handled = true;

}

}

private void TextBoxPos\_KeyPress(object sender, KeyPressEventArgs e) {

if (!double.TryParse(y1Box.Text + e.KeyChar.ToString(), out double a) &&

e.KeyChar != 8 && e.KeyChar != '.' && e.KeyChar != ',') {

e.Handled = true;

}

}

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

List<TextBox> textBoxes = new() { radiusBox, y1Box, y2Box, stepBox };

bool errFlag = false;

foreach (TextBox textBox in textBoxes) {

if (!IsValidNumber(textBox.Text)) {

errFlag = true;

MarkError(textBox);

}

else {

UnmarkError(textBox);

}

}

if (!AreBordersCorrect(y1Box.Text, y2Box.Text)) {

errFlag = true;

MarkError(y1Box);

MarkError(y2Box);

}

else {

UnmarkError(y1Box);

UnmarkError(y2Box);

}

if (!IsPositive(stepBox)) {

errFlag = true;

MarkError(stepBox);

}

else {

UnmarkError(stepBox);

buildBtn.Enabled = true;

}

if (!IsPositive(radiusBox)) {

errFlag = true;

MarkError(radiusBox);

}

else {

UnmarkError(radiusBox);

buildBtn.Enabled = true;

}

buildBtn.Enabled = !errFlag;

}

private static bool IsValidNumber(string input) {

return !string.IsNullOrWhiteSpace(input) && double.TryParse(input, out \_);

}

private static void MarkError(TextBox textBox) {

textBox.ForeColor = System.Drawing.Color.Red;

}

private static void UnmarkError(TextBox textBox) {

textBox.ForeColor = System.Drawing.Color.Black;

}

private static bool AreBordersCorrect(string y1S, string y2S) {

if (string.IsNullOrEmpty(y1S) || string.IsNullOrEmpty(y2S)) return

true;

if (!double.TryParse(y1S.Replace(',', '.'), out double y1)

|| !double.TryParse(y2S.Replace(',', '.'), out double y2))

return false;

return y1 < y2;

}

private static bool IsPositive(TextBox textBox) {

if (!double.TryParse(textBox.Text.Replace(',', '.'), out double

number)) return false;

return number > 0;

}

private void BuildGraph() {

ScottPlot.Plot myPlot = fp.Plot;

panel1.Controls.Remove(fp);

myPlot.Clear();

panel1.Controls.Add(fp);

resultsTable.Rows.Clear();

var Ctg = new Func<double, double>((x) => 1 / Math.Tan(x));

List<double> XList = new();

List<double> YList = new();

double R = double.Parse(radiusBox.Text.Replace(',', '.'));

double step = double.Parse(stepBox.Text.Replace(',', '.'));

double y1 = double.Parse(y1Box.Text.Replace(',', '.'));

double y2 = double.Parse(y2Box.Text.Replace(',', '.'));

//int R = 2;

myPlot.Axes.SetLimitsY(y1, y2);

for (double y = y1; y <= y2; y += step) {

double x = y \* Ctg((Math.PI \* y) / (2 \* R));

if (XList.Count > 0) {

if ((y > 0 && XList.Last() < x) || (y < 0 && XList.Last() >

x)) {

var scatter = myPlot.Add.Scatter(XList.ToArray(),

YList.ToArray());

scatter.MarkerSize = 0;

scatter.Color = ScottPlot.Color.FromHex("#F02D3A");

XList.Clear();

YList.Clear();

myPlot = fp.Plot;

}

}

if (Math.Abs(Math.Tan((Math.PI \* y) / (2 \* R))) > 1e-10 &&

Math.Abs(Math.Tan((Math.PI \* y) / (2 \* R))) < 1e10) {

YList.Add(y);

XList.Add(x);

int digitsAfterDecimal = CountDigitsAfterDecimalPoint(step);

resultsTable.Rows.Add(x.ToString($"F{digitsAfterDecimal}"),

y.ToString($"F{digitsAfterDecimal}"));

}

}

if (XList.Count > 0) {

var scatter = myPlot.Add.Scatter(XList.ToArray(),

YList.ToArray());

scatter.MarkerSize = 0;

scatter.Color = ScottPlot.Color.FromHex("#F02D3A");

}

}

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

saveBtn.Enabled = true;

BuildGraph();

}

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

openFileDialog1.Filter = "Text File|\*.txt";

openFileDialog1.Title = "Open Text File";

openFileDialog1.FileName = "";

if (openFileDialog1.ShowDialog() == DialogResult.OK) {

string filePath = openFileDialog1.FileName;

using StreamReader reader = new(filePath);

radiusBox.Text = reader.ReadLine();

stepBox.Text = reader.ReadLine();

y1Box.Text = reader.ReadLine();

y2Box.Text = reader.ReadLine();

reader.Close();

MessageBox.Show("Данные успешно загружены из файла: " +

filePath);

BuildGraph();

}

}

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

saveFileDialog1.Filter = "Text File|\*.txt";

saveFileDialog1.Title = "Save to Text File";

saveFileDialog1.CheckFileExists = false;

saveFileDialog1.CheckPathExists = true;

if (saveFileDialog1.ShowDialog() == DialogResult.OK) {

string filePath = saveFileDialog1.FileName;

using StreamWriter writer = new(filePath);

writer.WriteLine(radiusBox.Text);

writer.WriteLine(stepBox.Text);

writer.WriteLine(y1Box.Text);

writer.WriteLine(y2Box.Text);

writer.WriteLine("//");

writer.WriteLine("Результаты:");

foreach (DataGridViewRow row in resultsTable.Rows) {

foreach (DataGridViewCell cell in row.Cells) {

writer.Write($"{cell.Value} ");

}

writer.WriteLine();

}

writer.Close();

MessageBox.Show("Данные из таблицы успешно сохранены в файл: " +

filePath);

}

}

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

InfoForm info = new();

info.ShowDialog();

}

}

}