МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники

Кафедра «Программного обеспечения информационных систем и технологий»

**КУРСОВАЯ РАБОТА**

по дисциплине: «Разработка приложений в визуальных средах»

на тему: **«Разработка приложения решения ДУ методом Рунге-Кутта четвертого порядка»**

Выполнил Д.В. Бородкин

Студент гр. 10701322

Принял доц. Н.Н. Гурский

Минск 2024

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники

Кафедра «Программного обеспечения информационных систем и технологий»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

по курсовой работе

по дисциплине «Разработка приложений в визуальных средах»

тема: «Разработка приложения решения ДУ методом Рунге-Кутта четвертого порядка»

Исполнитель Д.В. Бородкин

Студент гр. 10701322

Руководитель доц. Н.Н. Гурский

Минск 2024

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 4](#_Toc168086374)

[1 Математическая формулировка задачи 6](#_Toc168086375)

[2 Описание программы 7](#_Toc168086376)

[2.1 Структурная схема программы 7](#_Toc168086377)

[2.2 Описание разработанного класса 8](#_Toc168086378)

[2.3 Описание и реализации сохранения данных в Word и Excel 9](#_Toc168086379)

[2.4 Описание реализации технологии сокетного сервера 11](#_Toc168086380)

[2.5 Описание динамических библиотек 13](#_Toc168086381)

[2.6 Основные возможности программы 14](#_Toc168086382)

[3 Руководство Пользователя 16](#_Toc168086383)

[3 Методика испытаний 22](#_Toc168086384)

[Заключение 25](#_Toc168086385)

[Литература 26](#_Toc168086386)

[ПРИЛОЖЕНИЕ 27](#_Toc168086387)

[Файл Graphic.cs 27](#_Toc168086388)

[Файл MatLabSolver.cs 27](#_Toc168086389)

[Файл Saver.cs 28](#_Toc168086390)

[Файл TCPClient.cs 29](#_Toc168086391)

[Файл AboutAuthor.cs 30](#_Toc168086392)

[Файл AboutProgram.cs 30](#_Toc168086393)

[Файл MainForm.cs 31](#_Toc168086394)

[Файл Settings.cs 35](#_Toc168086395)

[Файл Algorithm.cs 36](#_Toc168086396)

[Файл Equation.cs 37](#_Toc168086397)

[Файл Program.cs (Консольный сервер) 38](#_Toc168086398)

[Файл TCPServer.cs (Консольный сервер) 38](#_Toc168086399)

[Файл Logger.cs 40](#_Toc168086400)

[Файл TCPServer.cs (Оконный сервер) 40](#_Toc168086401)

[Файл Server.cs 42](#_Toc168086402)

# ВВЕДЕНИЕ

C# — объектно-ориентированный язык программирования общего назначения. C# позволяет разработчикам создавать разные типы безопасных и надежных приложений, выполняющихся в .NET. Относится к широко известному семейству языков C.

Разработан в 1998–2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота.

Язык имеет следующие особенности:

— статическую типизацию;

— поддержку полиморфизма;

— перегрузку операторов;

— делегаты, атрибуты, события, переменные, свойства, обобщённые типы и методы;

— итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

**Цель** данной курсовой работы: написать приложение для решения ДУ методов Рунге-Кутта четвертого порядка с парадигмой объектно-ориентированного программирования.

1. Задачи, решаемые в курсовой работе:

Основной задачей курсовой работы является формирование навыков разработки программных продуктов с заданным уровнем качества, а также разработка алгоритма для эффективного и точного решения и построения графика ДУ.

2. Создание графического интерфейса пользователя (GUI):

Разработать удобный и интуитивно понятный интерфейс для пользователей в соответствии с заданной бизнес-логикой переключения графических окон, позволяющий пользоваться приложением, строить и просматривать графики, сохранять их в файл, а также просматривать таблицы точек.

3. Тестирование и оптимизация приложения:

Провести тестирование приложения, выявить и устранить возможные ошибки и недоработки, а также оптимизировать его производительность для обеспечения комфортной работы пользователей.

Обоснование актуальности темы:

Актуальность данной темы обусловлена не только теоретическими аспектами, связанными с разработкой структуры данных и алгоритмов, но и практическими потребностями обучающихся. Приложение позволит значительно повысить эффективность при работе с данной темой и автоматизирует построение графиков. Таким образом, разработка приложения решения ДУ методом Рунге-Кутта четвертого порядка имеет как теоретическую, так и практическую значимость, исследуя которую, можно сделать важный вклад в развитие области образования и информационных технологий.

# Математическая формулировка задачи

Тема курсовой работы «Решение ДУ методом Рунге-Кутта четвертого порядка». Исходя из этой темы:

Дифференциальные уравнения — это уравнения, которые связывают функцию и ее производные.

График дифференциального уравнения (ДУ) представляет собой визуальное представление решений данного уравнения. Поскольку дифференциальное уравнение связывает функцию и ее производные, график решений такого уравнения показывает, как эта функция изменяется в зависимости от независимой переменной.

Рисунок 1 – Пример дифференциального уравнения, где – искомая функция, – независимая переменная, – заданная функция, – начальное значение независимой переменной, – начальное значение искомой функции.

Метод Рунге-Кутта четвертого порядка для решения ОДУ определяется следующими формулами:

Здесь - шаг интегрирования, ​ - промежуточные значения, используемые для вычисления следующего значения ​ искомой функции.

# Описание программы

## Структурная схема программы

Структурно программа состоит из клиента, оконного сервера, консольного сервера, COM-сервера и динамической библиотеки RungeKutta4. При этом сервера содержат классы, реализующие основную логику программы. Оконный сервер, консольный сервер и динамическая библиотека RungeKutta4 содержат одинаковые классы для решения ДУ методом Рунге-Кутта 4-го порядка, однако различается способ получения начальных параметров от клиента. Консольный сервер представляет из себя TCP-сервер с классами Algorithm и Equation, для решения ДУ и распознавания уравнения, ведённого пользователем. Оконный сервер содержит доработанный функционал консольного сервера. Он работает через форму и позволяет задавать порт на котором он будет работать. Также есть возможность сохранения информации о работе сервера в текстовый файл. Оконный сервер гораздо удобнее для пользователя, чем консольный, однако консольный сервер требует меньше вычислительных мощностей и может использоваться на реальных устройствах без поддержки графики.

Пользователь клиента может вводить начальные значения и вычислять ДУ. Также есть окно настроек Settings.cs в котором можно выбрать способ вычисления ДУ, а также IP-адрес и порт по которому будет происходить подключение к серверам. Пользователь может сохранить результаты вычисления в Word, Excel или просто в текстовый документ.

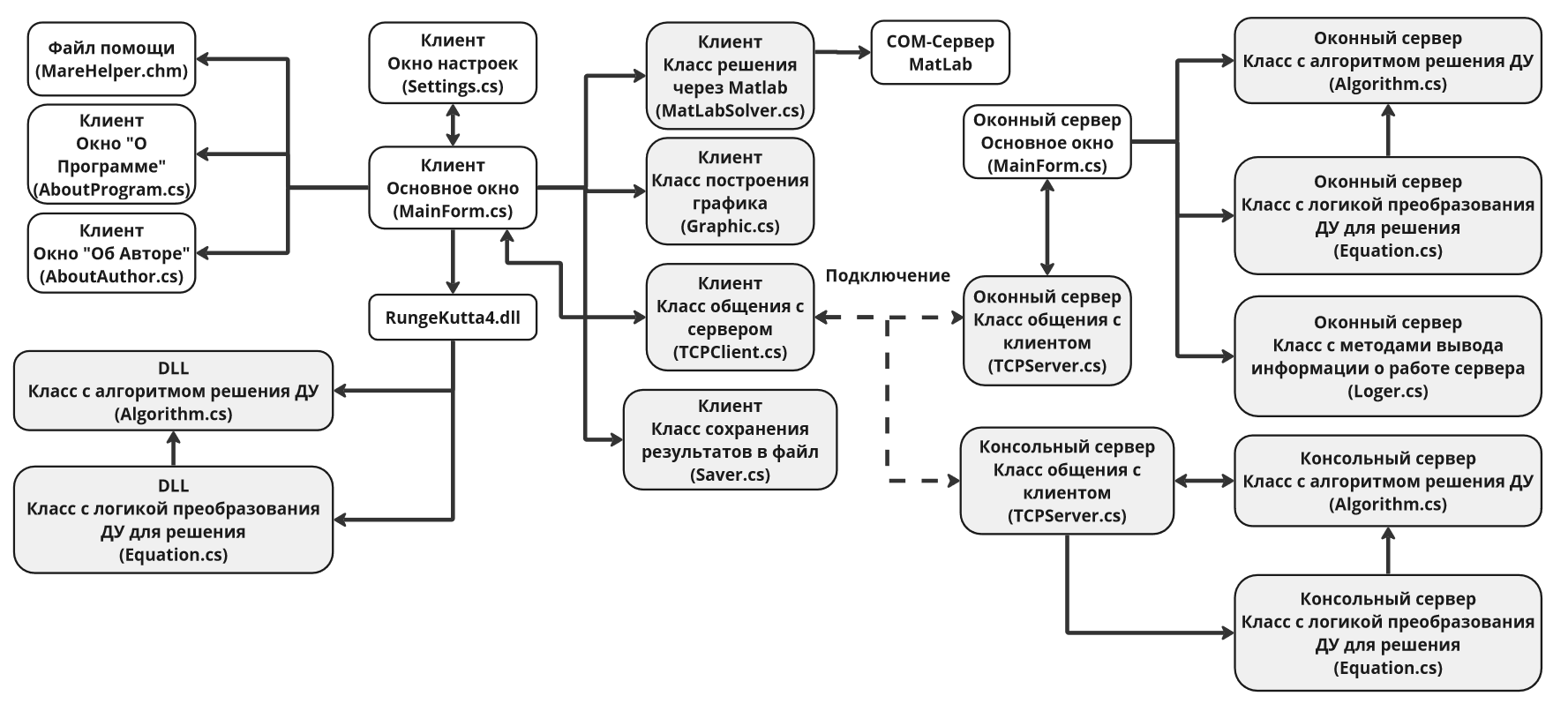


Рисунок 2.1 – Структурная схема связи модулей программы

## Описание разработанного класса

В результате анализа предметной области для ее описания и моделирования был разработан класс Algorithm, содержащий алгоритм решения ДУ методом Рунге-Кутта четвертого порядка.

public class Algorithm

{

private double t0, y0, h, tf;

private string usereq;

private List<(double x, double y)> points = new List<(double x, double y)>();

public double T0 { get; set; }

public double Y0 { get; set; }

public double H { get; set; }

public double Tf { get; set; }

public string Usereq { get; set; }

public List<(double x, double y)> Points

{

get

{

return points;

}

set

{

points = value;

}

}

public void SetValues(double t0, double y0, double h, double tf, string eq)

{

T0 = t0;

Y0 = y0;

H = h;

Tf = tf;

Usereq = eq;

Points.Clear();

}

public string[] Solve()

{

Equation eq = new Equation(Usereq);

double t = T0;

double y = Y0;

double k1, k2, k3, k4;

(double x, double y) dot = (t, y);

Points.Add(dot);

while (t < Tf)

{

k1 = eq.f(t, y);

k2 = eq.f(t + H / 2, y + (H \* k1 / 2));

k3 = eq.f(t + H / 2, y + (H \* k2 / 2));

k4 = eq.f(t + H, y + (H \* k3));

y = y + ((H / 6) \* (k1 + 2 \* k2 + 2 \* k3 + k4));

t = t + H;

dot = (t, y);

Points.Add(dot);

}

return ParseTuplesToStrings(Points);

}

public string[] ParseTuplesToStrings(List<(double x, double y)> tuples)

{

List<string> result = new List<string>();

foreach (var tuple in tuples)

{

result.Add(Math.Round(tuple.Item1, 5).ToString());

result.Add(Math.Round(tuple.Item2, 5).ToString());

}

return result.ToArray();

}

}

Данный класс имеет метод Solve, который, используя поля класса, рассчитывает список точек представляющих пары значений X и Y. Его работа начинается с инициализации объекта класса Equation, в котором происходит подстановка значений в уравнение и вычисление результата.

Метод SetValues служит для инициализации начальных условий и параметров вычислений.

Метод ParseTuplesToString предназначен для преобразования списка кортежей, состоящих из пар значений (x,y), в массив строк, где каждое значение в кортеже округляется до пяти знаков после запятой и преобразуется в строку. Этот метод полезен для подготовки готовых значений к отправке клиенту.

## Описание и реализации сохранения данных в Word и Excel

Исходя их требований на разработку данной курсовой работы была реализована возможность экспорта результатов вычислений в документы Word и Excel. С этой целью был разработан класс Saver.

public class Saver

{

public void SaveToExcel(string filename, string[] data, string usereq, double t0, double y0, double h, double tf)

{

FileInfo fileInfo = new FileInfo(filename);

using (ExcelPackage package = new ExcelPackage(fileInfo))

{

ExcelWorksheet worksheet = package.Workbook.Worksheets.Add("Equation");

worksheet.Cells[1, 1].Value = "Уравнение";

worksheet.Cells[1, 2].Value = usereq;

worksheet.Cells[2, 1].Value = "x0";

worksheet.Cells[2, 2].Value = t0;

worksheet.Cells[3, 1].Value = "y0";

worksheet.Cells[3, 2].Value = y0;

worksheet.Cells[4, 1].Value = "h";

worksheet.Cells[4, 2].Value = h;

worksheet.Cells[5, 1].Value = "tf";

worksheet.Cells[5, 2].Value = tf;

worksheet.Cells[7, 1].Value = "X";

worksheet.Cells[7, 2].Value = "Y";

int cell = 7;

for (int i = 1; i < data.Length; i += 2)

{

worksheet.Cells[cell, 1].Value = double.Parse(data[i - 1]);

worksheet.Cells[cell, 2].Value = double.Parse(data[i]);

cell++;

}

cell--;

var chart = worksheet.Drawings.AddChart("График", eChartType.LineMarkers);

var series = chart.Series.Add(worksheet.Cells["B7" + ":B" + cell.ToString()], worksheet.Cells["A7" + ":A" + cell.ToString()]);

series.Header = "График ДУ";

chart.Title.Text = "График уравнения " + usereq;

chart.XAxis.Title.Text = "X";

chart.YAxis.Title.Text = "Y";

chart.SetPosition(0, 0, 3, 0);

chart.SetSize(600, 400);

package.Save();

}

}

public void SaveToWord(Chart chart, string filePath)

{

string tempImagePath = Path.GetTempFileName() + ".png";

SaveChartAsImage(chart, tempImagePath);

var wordApp = new Word.Application();

var document = wordApp.Documents.Add();

try

{

var paragraph = document.Paragraphs.Add();

paragraph.Range.InlineShapes.AddPicture(tempImagePath);

document.SaveAs2(filePath);

}

finally

{

document.Close();

wordApp.Quit();

File.Delete(tempImagePath);

}

}

private void SaveChartAsImage(Chart chart, string filePath)

{

using (var stream = new MemoryStream())

{

chart.SaveImage(stream, ChartImageFormat.Png);

var bitmap = new Bitmap(stream);

bitmap.Save(filePath);

}

}

}

Этот класс имеет три метода SaveChartAsImage, SaveToWord и SaveToExcel.

SaveChartAsImage реализует сохранение графика в качестве изображения на компьютере. Сохранённое изображение необходимо для следующего метода SaveToWord.

SaveToWord вызывает метод SaveChartAsImage для создания изображения графика, создаёт новый документ Word и импортирует в него сохранённое изображение графика. После чего удаляет изображение с компьютера и сохраняет документ.

Метод SaveToExcel реализует алгоритм сохранения вычисленных точек в таблицу Excel и построение графика в таблице по диапазону сохранённых точек. Итоговый Excel документ сохраняет на компьютере.

## Описание реализации технологии сокетного сервера

В соответствии с требованиями на разработку данной курсовой работы был также реализован потоковый сокетный сервер. Приложение может подключиться к серверу сети с помощью IPv4 адреса. Было создано две версии сервера: оконный сервер и консольный сервер. Оконный сервер представляет из себя удобное отдельное приложение, которое запускает прослушивание сокета на выбранном порту, принимает входящие подключения от клиента, решает ДУ методом Рунге-Кутта четвертого порядка и отправляет клиенту поток из вычисленных значений. Параллельно с вычислениями, он асинхронно записывает в информационное окно всю информацию о работе, которую можно сохранить в отдельный файл. Консольный же сервер создан для работы на устройствах, без поддержки визуальных сред. Это может быть полезно, для запуска сервера на реальных серверных стойках, к которым не подключены мониторы или устройства визуального отображения.

Ниже показано реализация потокового сокетного оконного сервера.

public class TCPServer

{

TcpListener server = null;

NetworkStream stream = null;

private Algorithm algo = new Algorithm(); // создаём объект класса Algorithm

private Loger loger; // создаём объект класса Loger

public TCPServer(Loger loger)

{

this.loger = loger;

}

public void StartServer(int port, CancellationToken cancellationToken)

{

try

{

InitializeServer(port);

while (true)

{

ReceiveAndSend();

}

}

catch (SocketException e)

{

Console.WriteLine("SocketException: {0}", e);

}

finally

{

StopServer();

}

}

public void StopServer()

{

server.Stop();

}

private void InitializeServer(int port)

{

IPAddress localAddr = IPAddress.Parse("127.0.0.1");

server = new TcpListener(localAddr, port);

server.Start();

loger.Log("Сервер запущен. Ожидание подключений...");

}

private void ReceiveAndSend()

{

TcpClient client = server.AcceptTcpClient();

loger.Log("Клиент подключен.");

stream = client.GetStream();

byte[] data = new byte[256];

int bytes = stream.Read(data, 0, data.Length);

string requestData = Encoding.UTF8.GetString(data, 0, bytes);

loger.Log("Полученные данные: " + requestData);

string[] splittedData = requestData.Split('~');

algo.SetValues(

double.Parse(splittedData[0]),

double.Parse(splittedData[1]),

double.Parse(splittedData[2]),

double.Parse(splittedData[3]),

splittedData[4]

);

string responseData = "";

string[] points = new string[1];

try

{

points = algo.Solve();

}

catch (Exception e)

{

loger.LogError(e);

}

for (int i = 1; i < points.Length; i += 2)

{

responseData += points[i - 1];

responseData += '~';

responseData += points[i];

responseData += '~';

loger.Log(points[i - 1] + " " + points[i]);

}

try

{

responseData = responseData.Remove(responseData.Length - 1);

} catch (Exception e)

{

loger.LogError(e);

}

byte[] responseBytes = Encoding.UTF8.GetBytes(responseData);

stream.Write(responseBytes, 0, responseBytes.Length);

loger.Log("Ответ отправлен клиенту.");

client.Close();

}

}

При инициализации объекта класса TCPServer, в поле logger записывает объект класса Logger. Таким образом к серверу привязывается отдельный класс, который отвечает за информирование пользователя о работе сервера.

Класс TCPServer реализует четыре метода: StartServer, StopServer, InitializeServer, ReceiveAndSend.

Метод StartServer отвечает за запуск сервера. Он пытается вызвать метод InitializeServer и после чего циклически вызывать метод ReceiveAndSend.

Метод StopServer останавливает прослушивание сокета.

Метод InitializeServer привязывает сокет к определенному порту и запускает прослушивание

Метод ReceiveAndSend выполняет основную работу по передаче данных между клиентом и сервером. При вызове этого метода, он ожидает подключение клиента. Как только клиент подключится, метод примет и запишет переданные данные в массив строк. Массив строк разделяется на части и передаётся в объект класса Algorithm, где будут происходить вычисления по начальным параметрам, которые нам передал клиент. Список строк со значениями вычисленных точек записывается в массив байт и отправляется клиенту. После чего сервер разрывает соединение.

Оконный сервер работает по такому же принципу, но не использует класс Logger, а выводит всю информацию прямо в консоль.

## Описание динамических библиотек

При разработке приложения была создана динамическая библиотека.

Динамическая библиотека RungeKutta4.dll была разработана следующим обазом:

* В новом проекте были созданы два класса «Algorithm» и «Equation» для решения ДУ методом Рунге-Кутта четвертого порядка.
* Проект был скомпилирован, в папке bin появилась библиотека RungeKutta4.dll
* Данный проект был подключен к основному проект через ссылки

Динамическая библиотека RungeKutta4 содержит два класса идентичные классам описанным в главе 2.2 “Описание разработанного класса”. Библиотека будет полезна для работы с будущими проектами, где может потребоваться этот алгоритм. Также компиляция алгоритма в DLL позволяет другим разработчикам легко и быстро использовать алгоритм.

## Основные возможности программы

Программа начинается с запуска основного окна (форма MainForm.cs).

Главное окно управляет работой всей программы в соответствии с запросами пользователя.

Оно содержит:

* Вызов окна «Справка».
* Передача и сохранения данных работы программы в Word.
* Передача и сохранения данных работы программы в Excel.
* Ввод дифференциального уравнения, которое пользователь хочет решить.
* Ввод границ начальных условий X0 и Y0.
* Ввод шага h.
* Ввод конечного времени tf (верхняя граница диапазона вычисления)
* Меню для удобства пользователя
* Возможность сохранить данные из меню
* Возможность переходы между окнами с помощью меню
* Возможность закрытия программы из меню
* Использование возможностей потокового сокетного сервера.
* Использование средств, предоставляемых Microsoft Office.
* Вызов окна “О Программе” с информацией о программе.
* Вызов окна “Об Авторе” с информацией об авторе.
* Использование DLL для решения ДУ
* Использование среды Matlab для решения ДУ

Диалог с пользователем поддерживается с помощью главного меню, кнопок и других интерфейсных элементов.

Окно «Справка» содержит основную информацию о программе. Содержание окна помощь:

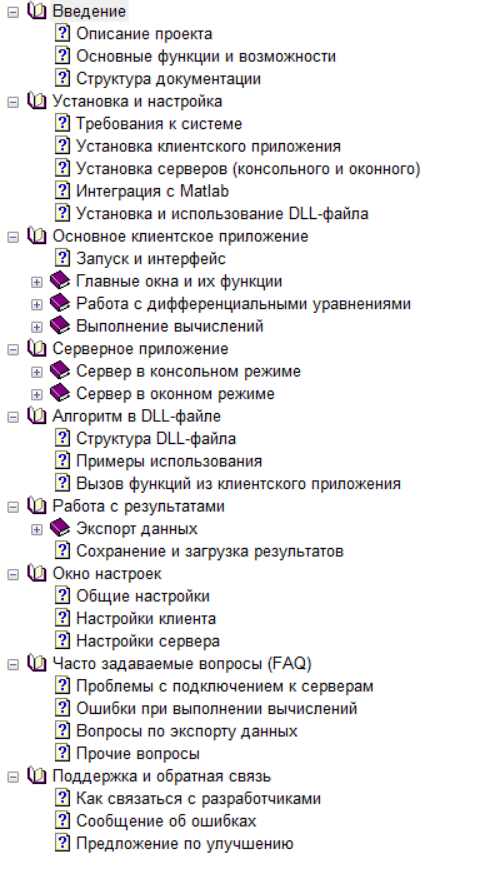


Рисунок 2.5.1 – Окно «Помощь»

Вопросы непосредственной программной реализации конкретных модулей приведены в приложении.

# 3 Руководство Пользователя

Для активации программы необходимо запустить файл Сlient.exe.

После этого на экране появится главное окно (рис 3.2):

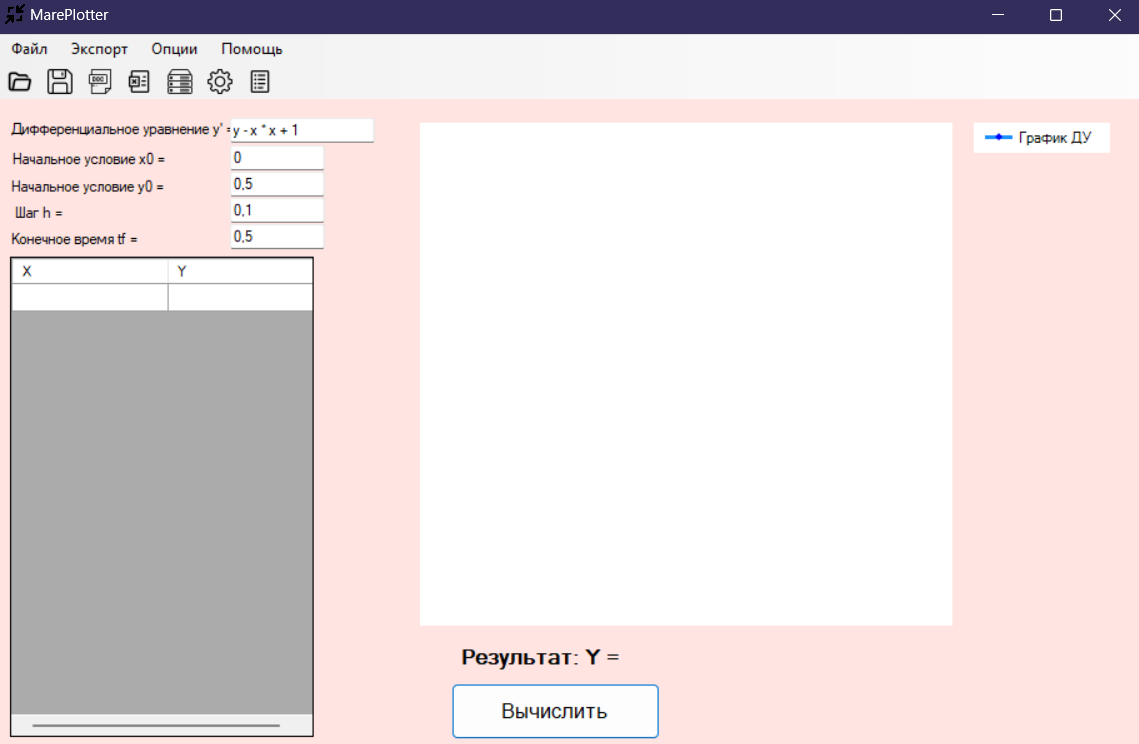


Рисунок 3.1 – Главное окно

Основное окно содержит главное меню, которое помогает пользователю ориентироваться в программе и содержит основные ее функции (рис. 3.8).

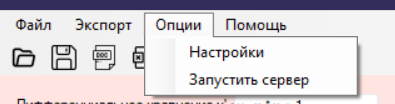
****

Рисунок 3.2 – Главное меню

Пользователь имеет возможность изменить начальные параметры и ввести свое уравнение:

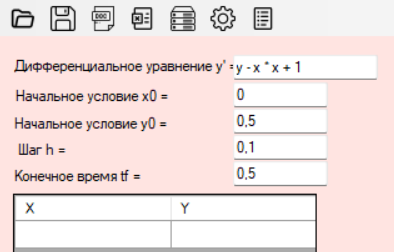


Рисунок 3.3 – Список уравнений

Также в главном окне есть место для построения графика и список точек. И график, и список точек при запуске программы пусты. Если мы ввели все параметры, ввели уравнение и нажали на кнопку “Вычислить”, то список точек заполнится, а график построится.

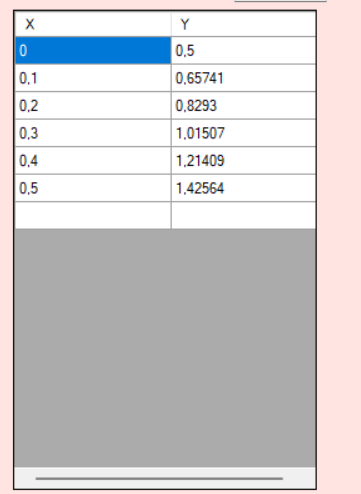


Рисунок 3.4 – Список точек

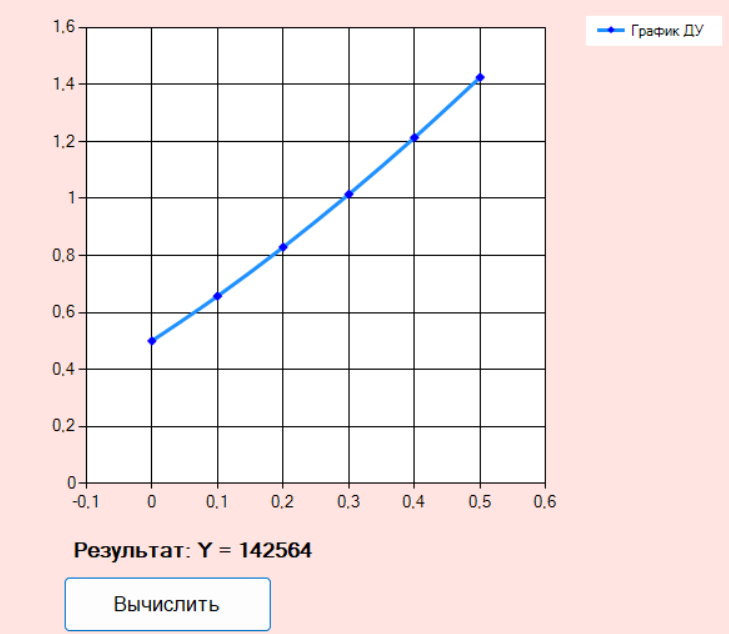


Рисунок 3.5 – График ДУ

Пользователь может открыть окно настроек, щелкнув по иконке в меню.

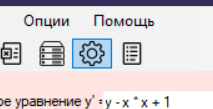


Рисунок 3.6 – Иконка окна “Настройки”

Окно настроек представляет из себя новую форму, в которой можно настроить необходимые пользователю параметры в трех категориях: “Для вычисления использовать:”, “Настройки сервера” и “Настройки клиента”.

В категории “Для вычисления использовать:” пользователь выбирает один из трех источников вычисления ДУ, а именно DLL, MatLab или сервер.

Категория “Настройки сервера” предоставляет всего одну настройку, в которой пользователь выбирает какой вид сервера он желает использовать: оконный сервер или консольный сервер.

Категория “Настройки клиента” содержит настройки самого клиента. Тут можно ввести IP-адрес и порт по которому клиент будет пытаться подключиться, размер потока получения данных необходимый для расширения или ограничения потока передачи данных с сервера на клиент и стандартный путь сохранения, в котором можно указать куда сохранять файл при нажатии на кнопку “Сохранить” из главного меню.

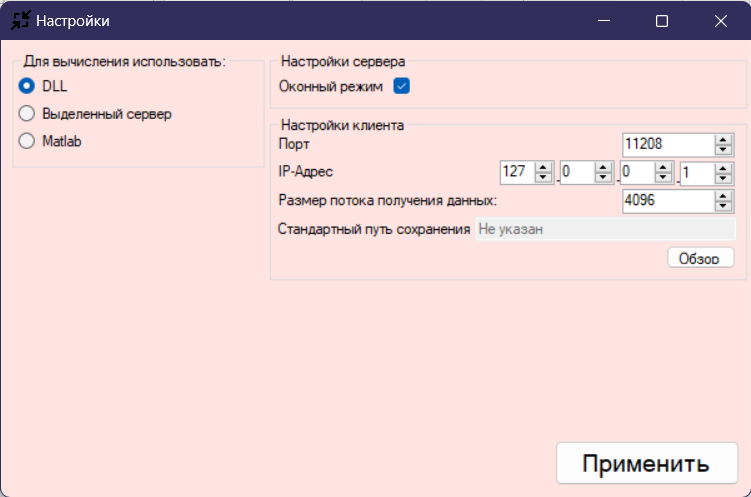


Рисунок 3.7 – Окно настроек

Возвращаясь к главному окну, пользователь может нажать на кнопку сервера, что запустит или консольный, или оконный сервер в зависимости от выбора в настройках.

Консольный сервер автоматически запускается и прослушивает порт 11208.

Оконный сервер даёт пользователю возможность ввести порт, на котором будет происходить прослушивание. Также он имеет информационное окно и кнопку “Сохранить лог” для его сохранения в текстовый документ. По нажатию на кнопку запуск сервер запустится и начнёт прослушивание. Пользователь в любой момент может повторно нажать на кнопку и остановить сервер для смены порта или для сохранения информации в текстовый документ.

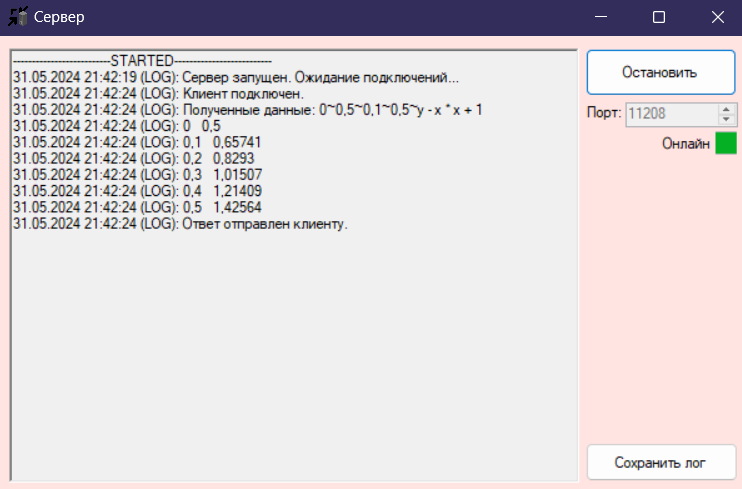


Рисунок 3.8 – Окно оконного сервера

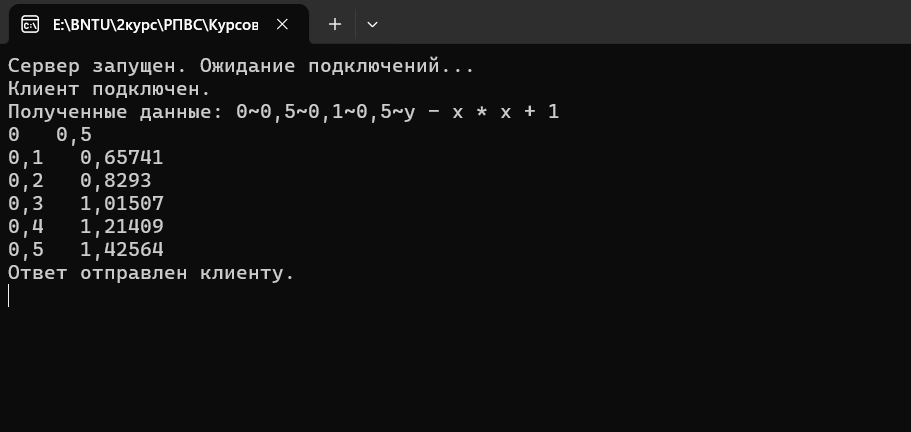


Рисунок 3.9 – Консольный сервер

Однако если пользователь хочет использовать среду решения Matlab, то при нажатии на кнопку “Вычислить” в главном окне будет построен график в отдельном окне среды Matlab.

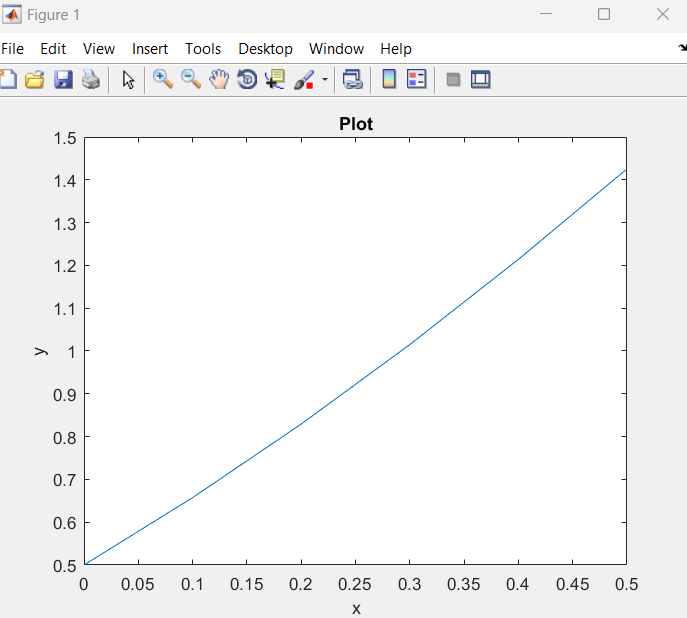


Рисунок 3.10 – График в среде Matlab

# Методика испытаний

Главной целью проведения испытаний, то есть тестирование программы, является проверка работоспособности программы, анализ ее поведения в различных условиях и выявление ошибок в работе для их дальнейшего исправления.

Программа должна обеспечивать корректность ввода исходных данных (путем осуществления соответствующих проверок и информирования пользователя о возникших неточностях в работе), а также получение непротиворечивого результата.

Для проверки корректности работы программы необходимо провести ряд испытаний (тестов) с различными начальными условиями.

Для обеспечение нормальной работы программы требуется наличие соответствующих DLL, а также приложения Word.

Тестовые примеры выполнялись в среде Windows 11 при использовании процесора Intel(R) Core(TM) i7-11370H @ 3.30GHz и разрешении экрана 1920\*1080. Для работы с программой необходимо также наличие клавиатуры и манипулятора типа «мышь».

Первое испытание проведем, вычисляя уравнение с некорректными параметрами.

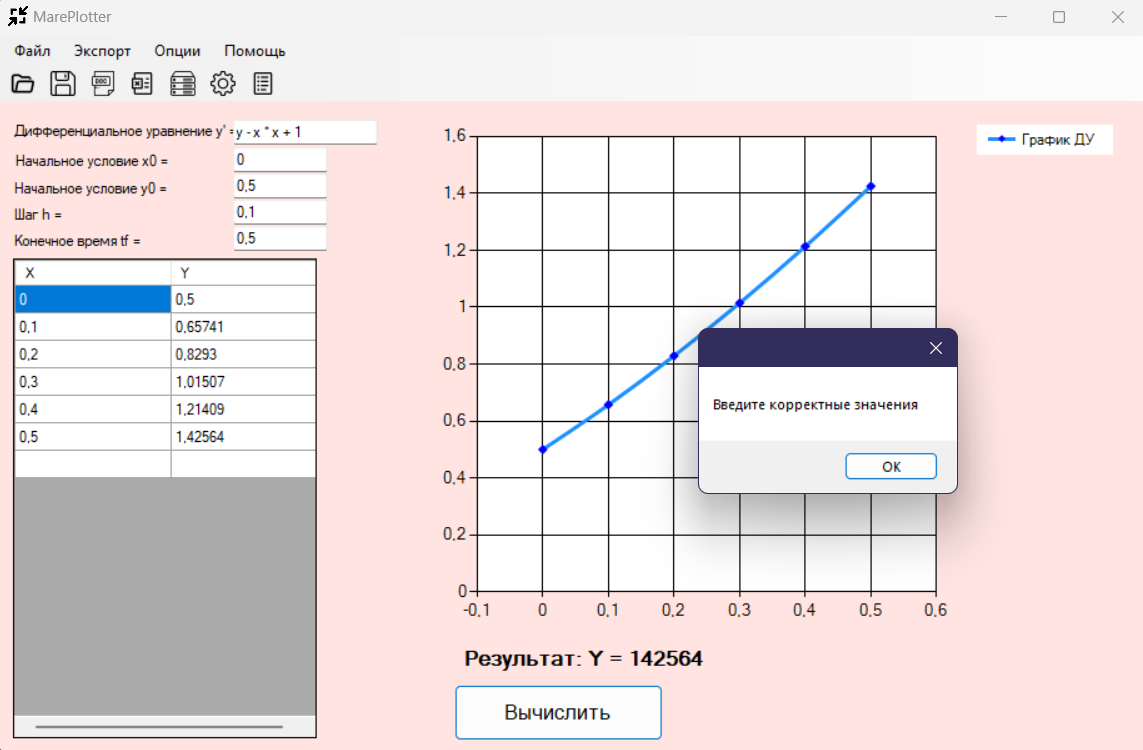


Рисунок 4.1 – Предупреждение об ошибке

Первое испытание завершилось с ожидаемым результатом, приложение вернуло стандартные параметры и вывело сообщение об ошибке. Такой же результат достигается, если оставить поле для ввода уравнения пустым или ввести в любое из полей некорректное значение.

Следующее испытание проведем с корректными начальными данными. Ожидается нормальное поведение программы (рис 4.2).

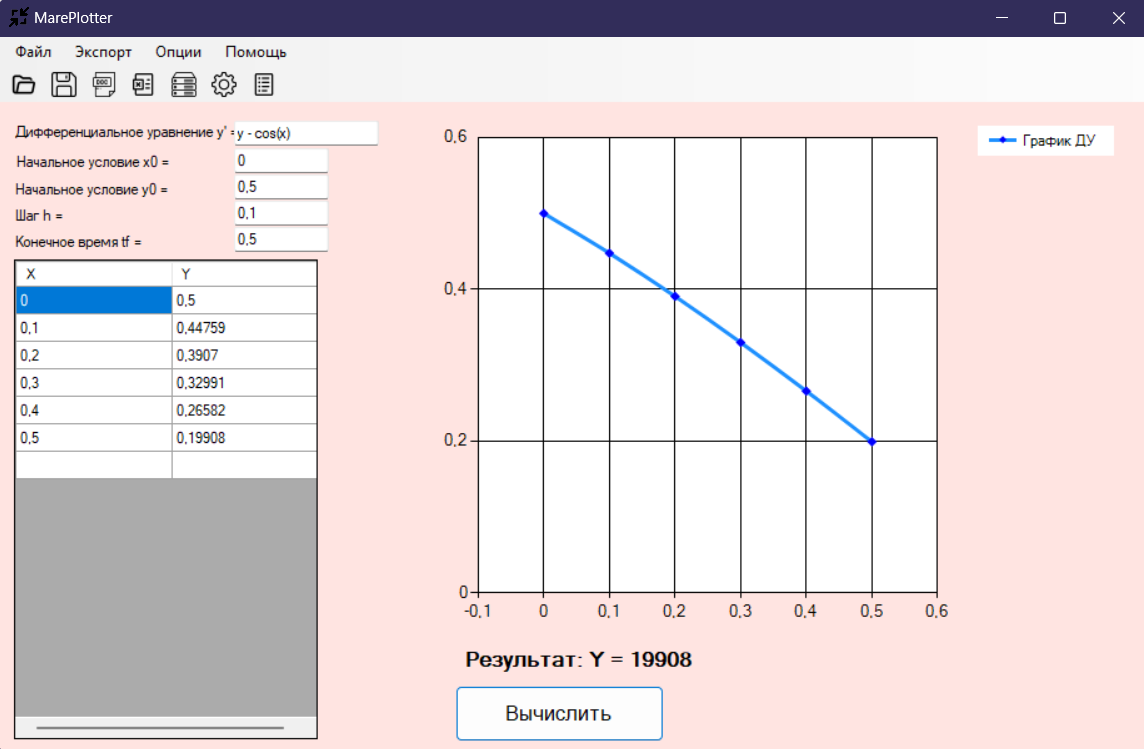


Рисунок 4.2 – Корректная работы программы

Испытание завершилось с ожидаемым результатом.

Следующее испытание проведем используя подключение к серверу, не запустив сервера. (Рис. 4.3)

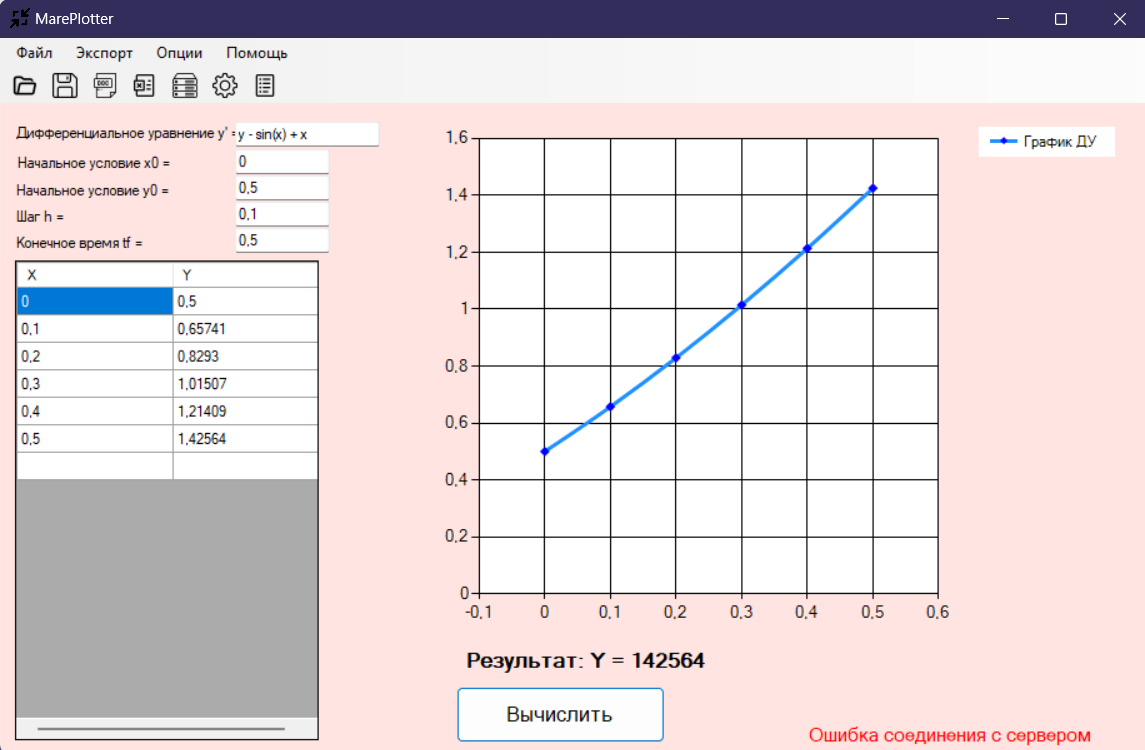


Рисунок 4.3 – Попытка вычисления с незапущенным сервером

Как и ожидалось, программа вывела сообщение об ошибке.

# Заключение

В ходе выполнения курсовой работы:

* были закреплены знания по курсу «Разработка приложений в визуальных средах»;
* приобретен опыт при разработке объектно-ориентированных программ;
* изучены принципы создания динамических библиотек DLL;
* изучены и реализованы способы создания клиент-серверных приложений на базе технологий создания потоковых сокетных клиент-серверов;
* освоены методы передачи данных между приложениями;
* изучены возможности создания справочной системой высокой степени сложности и различных форматов.

В процессе разработки программы был использован модульный принцип программирования. Данная структура программы позволяет в дальнейшем модифицировать отдельные её части без нарушения работоспособности и потери функциональности приложения.

Разработанная программа имеет достаточно ясный и понятый пользовательский интерфейс, обеспечивающий удобство в работе и обеспечивающий получение необходимой пользователю информации с наименьшими затратами времени.

Таким образом, разработанное в ходе выполнения курсовой работы приложение является законченным программным продуктом.

Программа может быть использована для решения ДУ методом Рунге-Кутта четвертого порядка.

# Литература

1. Л. Аммерал Принципы программирования в машинной графике. Пер.с англ.-М.:"Сол Систем", 1992.-224с.
2. Биллиг В. А. Основы объектного программирования на C# (C# 3.0, Visual Studio 2008) [Текст]: учебное пособие / В. А. Биллиг. — Москва: Интернет-университет информационных технологий, Бином. Лаборатория знаний, 2016. — 584 c.
3. Котов О. М. Язык C#. Краткое описание и введение в технологии программирования [Текст]: учебное пособие / О. М. Котов — Екатеринбург: Урал. ун-та, 2014. — 208 с.
4. Вагнер, Билл С# Эффективное программирование / Билл Вагнер. - М.: ЛОРИ, 2021. - 320 c.
5. Смоленцев, Н. К. MATLAB. Программирование на Visual С#, Borland JBuilder, VBA (+ CD-ROM) / Н.К. Смоленцев. - М.: ДМК Пресс, 2018. - 456 c.
6. Албахари, Джозеф C# 3.0. Справочник / Джозеф Албахари , Бен Албахари. - М.: БХВ-Петербург, 2021. - 944 c.

# ПРИЛОЖЕНИЕ

## Файл Graphic.cs

using System.Windows.Forms;

using System.Windows.Forms.DataVisualization.Charting;

namespace Client

{

internal class Graphic

{

private double[] points = new double[1];

private Chart chart;

private DataGridView dataGridView;

public Graphic(string[] data, Chart ch, DataGridView dgv)

{

chart = ch;

dataGridView = dgv;

points = new double[data.Length];

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

{

points[i] = double.Parse(data[i]);

}

}

public void Visualise()

{

dataGridView.Rows.Clear();

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

for (int i = 1; i < points.Length; i += 2)

{

chart.Series[0].Points.AddXY(points[i - 1], points[i]);

dataGridView.Rows.Add(points[i - 1], points[i]);

}

}

}

}

## Файл MatLabSolver.cs

using System.Globalization;

namespace Client

{

internal class MatLabSolver

{

public string RunCalculation(MLApp.MLApp matlab, double input\_y0, double input\_t0, double input\_tf, double input\_h, string usereq)

{

string t0 = input\_t0.ToString(CultureInfo.InvariantCulture);

string y0 = input\_y0.ToString(CultureInfo.InvariantCulture);

string h = input\_h.ToString(CultureInfo.InvariantCulture);

string tf = input\_tf.ToString(CultureInfo.InvariantCulture);

matlab.Execute($"t0 = {t0}");

matlab.Execute($"y0 = {y0}");

matlab.Execute("tf = " + tf);

matlab.Execute($"h = {h}");

matlab.Execute($"f = @(x,y) {usereq}");

matlab.Execute("x = t0:h:tf");

matlab.Execute("n = length(x)");

matlab.Execute("y = zeros(1, n)");

matlab.Execute("y(1) = y0");

matlab.Execute("for i = 1:n-1\r\n" +

" k1 = h \* feval(f, x(i), y(i));\r\n" +

" k2 = h \* feval(f, x(i) + 0.5\*h, y(i) + 0.5\*k1);\r\n" +

" k3 = h \* feval(f, x(i) + 0.5\*h, y(i) + 0.5\*k2);\r\n" +

" k4 = h \* feval(f, x(i) + h, y(i) + k3);\r\n" +

" y(i+1) = y(i) + (k1 + 2\*k2 + 2\*k3 + k4) / 6;\r\n" +

" end");

matlab.Execute("result = [x' y']");

matlab.Execute("plot(x, y);\r\n" +

" xlabel('x');\r\n" +

" ylabel('y');\r\n" +

" title('Plot');");

string result = matlab.Execute("result(n, 2)").Replace("\n", "").Replace("ans =", "").Replace(" ", "");

//"\nans =\n\n 1.4256\n\n" - without

//"ans =\n\n 1.4256" - Trim

return result;

}

}

}

## Файл Saver.cs

using OfficeOpenXml.Drawing.Chart;

using OfficeOpenXml;

using System.IO;

using System.Drawing;

using System.Windows.Forms.DataVisualization.Charting;

using Word = Microsoft.Office.Interop.Word;

namespace Client

{

internal class Saver

{

public void SaveToExcel(string filename, string[] data, string usereq, double t0, double y0, double h, double tf)

{

FileInfo fileInfo = new FileInfo(filename);

using (ExcelPackage package = new ExcelPackage(fileInfo))

{

ExcelWorksheet worksheet = package.Workbook.Worksheets.Add("Equation");

worksheet.Cells[1, 1].Value = "Уравнение";

worksheet.Cells[1, 2].Value = usereq;

worksheet.Cells[2, 1].Value = "x0";

worksheet.Cells[2, 2].Value = t0;

worksheet.Cells[3, 1].Value = "y0";

worksheet.Cells[3, 2].Value = y0;

worksheet.Cells[4, 1].Value = "h";

worksheet.Cells[4, 2].Value = h;

worksheet.Cells[5, 1].Value = "tf";

worksheet.Cells[5, 2].Value = tf;

worksheet.Cells[7, 1].Value = "X";

worksheet.Cells[7, 2].Value = "Y";

int cell = 7;

for (int i = 1; i < data.Length; i += 2)

{

worksheet.Cells[cell, 1].Value = double.Parse(data[i - 1]);

worksheet.Cells[cell, 2].Value = double.Parse(data[i]);

cell++;

}

cell--;

var chart = worksheet.Drawings.AddChart("График", eChartType.LineMarkers);

var series = chart.Series.Add(worksheet.Cells["B7" + ":B" + cell.ToString()], worksheet.Cells["A7" + ":A" + cell.ToString()]);

series.Header = "График ДУ";

chart.Title.Text = "График уравнения " + usereq;

chart.XAxis.Title.Text = "X";

chart.YAxis.Title.Text = "Y";

chart.SetPosition(0, 0, 3, 0);

chart.SetSize(600, 400);

package.Save();

}

}

public void SaveToWord(Chart chart, string filePath)

{

string tempImagePath = Path.GetTempFileName() + ".png";

SaveChartAsImage(chart, tempImagePath);

var wordApp = new Word.Application();

var document = wordApp.Documents.Add();

try

{

var paragraph = document.Paragraphs.Add();

paragraph.Range.InlineShapes.AddPicture(tempImagePath);

document.SaveAs2(filePath);

}

finally

{

document.Close();

wordApp.Quit();

File.Delete(tempImagePath);

}

}

private void SaveChartAsImage(Chart chart, string filePath)

{

using (var stream = new MemoryStream())

{

chart.SaveImage(stream, ChartImageFormat.Png);

var bitmap = new Bitmap(stream);

bitmap.Save(filePath);

}

}

}

}

## Файл TCPClient.cs

using System;

using System.Text;

using System.Net.Sockets;

namespace Client

{

internal class TCPClient

{

private TcpClient tcpClient = new TcpClient();

private NetworkStream stream;

private void Connect(string address, int port)

{

tcpClient.Connect(address, port);

}

private void Disconnect()

{

if (tcpClient != null)

{

tcpClient.Close();

}

}

private void Send(string msg)

{

stream = tcpClient.GetStream();

byte[] convertedMsg = Encoding.UTF8.GetBytes(msg);

stream.Write(convertedMsg, 0, convertedMsg.Length);

}

private string Receive(int streamsize)

{

stream = tcpClient.GetStream();

byte[] responseData = new byte[streamsize];

StringBuilder response = new StringBuilder();

int bytes = stream.Read(responseData, 0, responseData.Length);

response.Append(Encoding.UTF8.GetString(responseData, 0, bytes));

return response.ToString();

}

public string[] Handshake(string msg, string address, int port, int streamsize)

{

try

{

Connect(address, port);

Send(msg);

string receivedMsg = Receive(streamsize);

Disconnect();

string[] completedata = receivedMsg.Split('~');

return completedata;

}

catch (Exception e)

{

string[] errmsg = new string[1];

errmsg[0] = e.GetType().ToString();

return errmsg;

}

}

}

}

## Файл AboutAuthor.cs

using System.Windows.Forms;

namespace Client

{

public partial class AboutAuthor : Form

{

public AboutAuthor()

{

InitializeComponent();

GithubLinkLabel.LinkVisited = false;

}

private void GithubLinkLabel\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

GithubLinkLabel.LinkVisited = true;

System.Diagnostics.Process.Start("https://github.com/arkain123");

}

}

}

## Файл AboutProgram.cs

using System.Windows.Forms;

namespace Client

{

public partial class AboutProgram : Form

{

public AboutProgram()

{

InitializeComponent();

}

}

}

## Файл MainForm.cs

using System;

using System.Diagnostics;

using System.Globalization;

using System.Windows.Forms;

using DLL;

namespace Client

{

public partial class MainForm : Form

{

private double t0 = 0;

private double y0 = 0.5;

private double h = 0.1;

private double tf = 0.5;

private string usereq = "y - x \* x + 1";

private string address = "127.0.0.1";

private int port = 11208;

private string savedirectory = "Не указан";

private int streamsize = 4096;

private bool windowedServer = true;

private string[] data = new string[1];

private string solver = "dll";

public string Solver

{

get { return solver; }

set { solver = value; }

}

public bool WindowedServer

{

get { return windowedServer; }

set { windowedServer = value; }

}

public string Address

{

get { return address; }

set { address = value; }

}

public int Port

{

get { return port; }

set { port = value; }

}

public string Savedirectory

{

get { return savedirectory; }

set { savedirectory = value; }

}

public int Streamsize

{

get { return streamsize; }

set { streamsize = value; }

}

public MainForm()

{

InitializeComponent();

dataGridView1.RowHeadersVisible = false;

data[0] = "solving";

}

public void GetValues()

{

try

{

t0 = Convert.ToDouble(t0TextBox.Text);

y0 = Convert.ToDouble(y0TextBox.Text);

h = Convert.ToDouble(hTextBox.Text);

tf = Convert.ToDouble(tfTextBox.Text);

usereq = eqTextBox.Text;

if (eqTextBox.Text == "")

{

throw new Exception("Incorrect equation");

}

}

catch (Exception ex)

{

t0 = 0;

y0 = 0.5;

h = 0.1;

tf = 0.5;

usereq = "y - x \* x + 1";

tfTextBox.Text = "0,5";

hTextBox.Text = "0,1";

y0TextBox.Text = "0,5";

t0TextBox.Text = "0";

eqTextBox.Text = usereq;

MessageBox.Show("Введите корректные значения");

}

}

private void CheckSolver()

{

if (solver == "server")

{

TCPClient client = new TCPClient();

string msg = t0 + "~" + y0 + "~" + h + "~" + tf + "~" + usereq;

data = client.Handshake(msg, Address, Port, Streamsize);

}

else if (solver == "dll")

{

Algorithm algo = new Algorithm();

algo.SetValues(t0, y0, h, tf, usereq);

data = algo.Solve();

}

else if (solver == "matlab")

{

MLApp.MLApp matlab = new MLApp.MLApp();

matlab.Visible = 0;

MatLabSolver matlabsolver = new MatLabSolver();

data = new string[1];

data[0] = matlabsolver.RunCalculation(matlab, y0, t0, tf, h, usereq);

}

else

{

data[0] = "System.Net.Sockets.SocketException";

}

}

private void Solve()

{

try

{

double result = double.Parse(data[data.Length - 1], CultureInfo.InvariantCulture);

ResultLabel.Text = "Результат: Y = " + result;

try

{

Graphic graph = new Graphic(data, mainChart, dataGridView1);

graph.Visualise();

}

catch (Exception ex) { }

}

catch (Exception ex)

{

ResultLabel.Text = "Результат не определен";

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

dataGridView1.Rows.Clear();

}

}

private bool ErrHandler()

{

if (data[0] == "System.Net.Sockets.SocketException")

{

errlabel.Text = "Ошибка соединения с сервером";

return true;

}

if (data[0] == "System.IO.IOException")

{

errlabel.Text = "Удалённый хост разорвал подключение";

return true;

}

return false;

}

private void Save()

{

string text = null;

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

{

text += data[i] + "\n";

}

if (Savedirectory != "Не указан")

System.IO.File.WriteAllText(Savedirectory + "\\saved.txt", text);

}

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

{

data = null;

errlabel.Visible = false;

GetValues();

CheckSolver();

if (ErrHandler())

{

errlabel.Visible = true;

}

else

{

Solve();

}

}

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

{

AboutAuthor aboutAuthor = new AboutAuthor();

aboutAuthor.Show();

}

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

{

AboutProgram aboutProgram = new AboutProgram();

aboutProgram.Show();

}

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

{

Settings settings = new Settings(this);

settings.Show();

}

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

{

Close();

}

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

{

Process startServer = new Process();

if (windowedServer)

startServer = Process.Start("ServerForm.exe");

else

startServer = Process.Start("ServerConsole.exe");

}

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

{

if (Savedirectory == "" || Savedirectory == "Не указан")

{

if (folderBrowserDialog1.ShowDialog() == DialogResult.OK)

{

Savedirectory = folderBrowserDialog1.SelectedPath;

} else

{

return;

}

}

Save();

}

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

{

if (folderBrowserDialog1.ShowDialog() == DialogResult.OK)

{

Savedirectory = folderBrowserDialog1.SelectedPath;

}

Save();

}

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

{

if (openFileDialog1.ShowDialog() == DialogResult.Cancel)

return;

string filename = openFileDialog1.FileName;

string fileText = System.IO.File.ReadAllText(filename);

string[] tempdata = fileText.Split('\n');

data = new string[tempdata.Length - 1];

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

{

data[i] = tempdata[i];

}

Solve();

}

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

{

открытьКакToolStripMenuItem\_Click(sender, e);

}

private void сохранитьToolStripMenuItem1\_Click(object sender, EventArgs e)

{

сохранитьКакToolStripMenuItem\_Click(sender, e);

}

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

{

запуститьСерверToolStripMenuItem\_Click(sender, e);

}

private void настройкиToolStripMenuItem1\_Click(object sender, EventArgs e)

{

настройкиToolStripMenuItem\_Click(sender, e);

}

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

{

Saver excelSaver = new Saver();

SaveFileDialog saveFileDialog = new SaveFileDialog();

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

saveFileDialog.Title = "Экспорт в Excel";

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

{

try

{

excelSaver.SaveToExcel(saveFileDialog.FileName, data, usereq, t0, y0, h, tf);

MessageBox.Show("Файл сохранён успешно!", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при сохранении файла: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

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

{

Saver wordSaver = new Saver();

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "Word Documents (\*.docx)|\*.docx";

saveFileDialog.Title = "Экспорт в Word";

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

{

try

{

wordSaver.SaveToWord(mainChart, saveFileDialog.FileName);

MessageBox.Show("Файл сохранён успешно!", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при сохранении файла: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

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

{

экспортВWordToolStripMenuItem\_Click(sender, e);

}

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

{

экспортВExcelToolStripMenuItem\_Click(sender, e);

}

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

{

HelpProvider helpProvider1 = new HelpProvider();

helpProvider1.HelpNamespace = "MareHelper.chm";

helpProvider1.SetHelpNavigator(this, HelpNavigator.Topic);

helpProvider1.SetHelpKeyword(this, "");

Help.ShowHelp(this, helpProvider1.HelpNamespace, helpProvider1.GetHelpNavigator(this), helpProvider1.GetHelpKeyword(this));

}

private void справкаToolStripMenuItem1\_Click(object sender, EventArgs e)

{

справкаToolStripMenuItem\_Click(sender, e);

}

}

}

## Файл Settings.cs

using System;

using System.Windows.Forms;

namespace Client

{

public partial class Settings : Form

{

private MainForm mainform;

private string solver = "dll";

public Settings(MainForm form)

{

InitializeComponent();

mainform = form;

directoryTBox.Text = mainform.Savedirectory;

streamTBox.Text = mainform.Streamsize.ToString();

portTBox.Text = mainform.Port.ToString();

string[] splittedAddress = mainform.Address.Split('.');

ip1TBox.Text = splittedAddress[0];

ip2TBox.Text = splittedAddress[1];

ip3TBox.Text = splittedAddress[2];

ip4TBox.Text = splittedAddress[3];

solver = mainform.Solver;

if (mainform.WindowedServer)

windowCheckBox.Checked = true;

else

windowCheckBox.Checked = false;

if (solver == "server")

ServerRButton.Checked = true;

else if (solver == "dll")

DLLRButton.Checked = true;

else

MatLabRButton.Checked = true;

}

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

{

if (folderBrowserDialog1.ShowDialog() == DialogResult.OK)

{

directoryTBox.Text = folderBrowserDialog1.SelectedPath;

}

}

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

{

mainform.Address = ip1TBox.Text + "." + ip2TBox.Text + "." + ip3TBox.Text + "." + ip4TBox.Text;

mainform.Port = int.Parse(portTBox.Text);

mainform.Streamsize = int.Parse(streamTBox.Text);

mainform.Savedirectory = directoryTBox.Text;

mainform.Solver = solver;

if (windowCheckBox.Checked)

mainform.WindowedServer = true;

else

mainform.WindowedServer = false;

Close();

}

private void DLLRButton\_CheckedChanged(object sender, EventArgs e)

{

solver = "dll";

}

private void ServerRButton\_CheckedChanged(object sender, EventArgs e)

{

solver = "server";

}

private void MatLabRButton\_CheckedChanged(object sender, EventArgs e)

{

solver = "matlab";

}

}

}

## Файл Algorithm.cs

using System;

using System.Collections.Generic;

namespace Server

{

internal class Algorithm

{

private double t0, y0, h, tf;

private string usereq;

private List<(double x, double y)> points = new List<(double x, double y)>();

public double T0 { get; set; }

public double Y0 { get; set; }

public double H { get; set; }

public double Tf { get; set; }

public string Usereq { get; set; }

public List<(double x, double y)> Points

{

get

{

return points;

}

set

{

points = value;

}

}

public void SetValues(double t0, double y0, double h, double tf, string eq)

{

T0 = t0;

Y0 = y0;

H = h;

Tf = tf;

Usereq = eq;

Points.Clear();

}

public string[] Solve()

{

Equation eq = new Equation(Usereq);

double t = T0;

double y = Y0;

double k1, k2, k3, k4;

(double x, double y) dot = (t, y);

Points.Add(dot);

while (t < Tf)

{

k1 = eq.f(t, y);

k2 = eq.f(t + H / 2, y + (H \* k1 / 2));

k3 = eq.f(t + H / 2, y + (H \* k2 / 2));

k4 = eq.f(t + H, y + (H \* k3));

y = y + ((H / 6) \* (k1 + 2 \* k2 + 2 \* k3 + k4));

t = t + H;

dot = (t, y);

Points.Add(dot);

}

return ParseTuplesToStrings(Points);

}

public string[] ParseTuplesToStrings(List<(double x, double y)> tuples)

{

List<string> result = new List<string>();

foreach (var tuple in tuples)

{

result.Add(Math.Round(tuple.Item1, 5).ToString());

result.Add(Math.Round(tuple.Item2, 5).ToString());

}

return result.ToArray();

}

}

}

## Файл Equation.cs

using NCalc;

using System;

namespace Server

{

internal class Equation

{

public string eq = "y - x \* x + 1";

public Equation(string usereq)

{

eq = usereq;

}

public double f(double x, double y)

{

Parse();

Expression expression = new Expression(eq);

expression.Parameters["x"] = x;

expression.Parameters["y"] = y;

try

{

double result = (double)expression.Evaluate();

return result;

}

catch (Exception ex)

{

return 0.0;

}

}

public void Parse()

{

eq = eq.Replace("abs", "Abs");

eq = eq.Replace("exp", "Exp");

eq = eq.Replace("log", "Log");

eq = eq.Replace("ln", "Log10");

eq = eq.Replace("pow", "Pow");

eq = eq.Replace("sqrt", "Sqrt");

eq = eq.Replace("tan", "Tan");

eq = eq.Replace("cos", "Cos");

eq = eq.Replace("sin", "Sin");

}

}

}

## Файл Program.cs (Консольный сервер)

namespace Server

{

internal class Program

{

static void Main(string[] args)

{

const int PORT = 11208;

TCPServer server = new TCPServer();

server.StartServer(PORT);

}

}

}

## Файл TCPServer.cs (Консольный сервер)

using System;

using System.Net.Sockets;

using System.Net;

using System.Text;

namespace Server

{

internal class TCPServer

{

TcpListener server = null;

NetworkStream stream = null;

private Algorithm algo = new Algorithm();

public void StartServer(int port)

{

try

{

InitializeServer(port);

while (true)

{

ReceiveAndSend();

}

}

catch (SocketException e)

{

Console.WriteLine("SocketException: {0}", e);

}

finally

{

server.Stop();

}

}

private void InitializeServer(int port)

{

IPAddress localAddr = IPAddress.Parse("127.0.0.1");

server = new TcpListener(localAddr, port);

server.Start();

Console.WriteLine("Сервер запущен. Ожидание подключений...");

}

private void ReceiveAndSend()

{

TcpClient client = server.AcceptTcpClient();

Console.WriteLine("Клиент подключен.");

stream = client.GetStream();

byte[] data = new byte[256];

int bytes = stream.Read(data, 0, data.Length);

string requestData = Encoding.UTF8.GetString(data, 0, bytes);

Console.WriteLine("Полученные данные: {0}", requestData);

string[] splittedData = requestData.Split('~');

algo.SetValues(

double.Parse(splittedData[0]),

double.Parse(splittedData[1]),

double.Parse(splittedData[2]),

double.Parse(splittedData[3]),

splittedData[4]

);

string responseData = "";

string[] points = new string[1];

try

{

points = algo.Solve();

}

catch (Exception e) { }

for (int i = 1; i < points.Length; i += 2)

{

responseData += points[i - 1];

responseData += '~';

responseData += points[i];

responseData += '~';

Console.WriteLine(points[i - 1] + " " + points[i]);

}

try

{

responseData = responseData.Remove(responseData.Length - 1);

} catch (Exception e) { }

byte[] responseBytes = Encoding.UTF8.GetBytes(responseData);

stream.Write(responseBytes, 0, responseBytes.Length);

Console.WriteLine("Ответ отправлен клиенту.");

client.Close();

}

}

}

## Файл Logger.cs

using System;

using System.Windows.Forms;

namespace ServerForm

{

internal class Loger

{

private RichTextBox logTbox;

public Loger(RichTextBox logtextbox)

{

logTbox = logtextbox;

}

public void LogError(Exception e)

{

string log = DateTime.Now.ToString() + " (ERROR): " + e.ToString() + "\n";

logTbox.Text += log;

}

public void Log(string msg)

{

string log = DateTime.Now.ToString() + " (LOG): " + msg + "\n";

logTbox.Text += log;

}

public void Clear()

{

logTbox.Text = "";

}

public void LogStop()

{

string log = "--------------------------STOPPED--------------------------";

logTbox.Text += log;

}

public void LogStart()

{

string log = "--------------------------STARTED--------------------------" + "\n";

logTbox.Text += log;

}

}

}

## Файл TCPServer.cs (Оконный сервер)

using System;

using System.Net.Sockets;

using System.Net;

using System.Text;

using System.Threading;

using ServerForm;

namespace Server

{

internal class TCPServer

{

TcpListener server = null;

NetworkStream stream = null;

private Algorithm algo = new Algorithm();

private Loger loger;

public TCPServer(Loger loger)

{

this.loger = loger;

}

public void StartServer(int port, CancellationToken cancellationToken)

{

try

{

InitializeServer(port);

while (true)

{

ReceiveAndSend();

}

}

catch (SocketException e)

{

Console.WriteLine("SocketException: {0}", e);

}

finally

{

StopServer();

}

}

public void StopServer()

{

server.Stop();

}

private void InitializeServer(int port)

{

IPAddress localAddr = IPAddress.Parse("127.0.0.1");

server = new TcpListener(localAddr, port);

server.Start();

loger.Log("Сервер запущен. Ожидание подключений...");

}

private void ReceiveAndSend()

{

TcpClient client = server.AcceptTcpClient();

loger.Log("Клиент подключен.");

stream = client.GetStream();

byte[] data = new byte[256];

int bytes = stream.Read(data, 0, data.Length);

string requestData = Encoding.UTF8.GetString(data, 0, bytes);

loger.Log("Полученные данные: " + requestData);

string[] splittedData = requestData.Split('~');

algo.SetValues(

double.Parse(splittedData[0]),

double.Parse(splittedData[1]),

double.Parse(splittedData[2]),

double.Parse(splittedData[3]),

splittedData[4]

);

string responseData = "";

string[] points = new string[1];

try

{

points = algo.Solve();

}

catch (Exception e)

{

loger.LogError(e);

}

for (int i = 1; i < points.Length; i += 2)

{

responseData += points[i - 1];

responseData += '~';

responseData += points[i];

responseData += '~';

loger.Log(points[i - 1] + " " + points[i]);

}

try

{

responseData = responseData.Remove(responseData.Length - 1);

} catch (Exception e)

{

loger.LogError(e);

}

byte[] responseBytes = Encoding.UTF8.GetBytes(responseData);

stream.Write(responseBytes, 0, responseBytes.Length);

loger.Log("Ответ отправлен клиенту.");

client.Close();

}

}

}

## Файл Server.cs

using Server;

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.Threading;

using System.Windows.Forms;

using System.IO;

namespace ServerForm

{

public partial class Server : Form

{

private string address = "127.0.0.1";

private int port = 11208;

private bool started = false;

private TCPServer server;

private CancellationTokenSource cancellationTokenSource;

private Loger loger;

public string Address

{

get { return address; }

}

public int Port

{

get { return port; }

set { port = value; }

}

public Server()

{

InitializeComponent();

InitializeBackgroundWorker();

loger = new Loger(richTextBox1);

server = new TCPServer(loger);

}

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

{

if (started)

{

OnlineBar.Value = 0;

portTBox.Enabled = true;

StartButton.Text = "Запустить";

started = false;

loger.LogStop();

cancellationTokenSource.Cancel();

server.StopServer();

}

else

{

OnlineBar.Value = 1;

portTBox.Enabled = false;

StartButton.Text = "Остановить";

started = true;

richTextBox1.Clear();

loger.LogStart();

Port = int.Parse(portTBox.Text);

cancellationTokenSource = new CancellationTokenSource();

backgroundWorker1.RunWorkerAsync(cancellationTokenSource.Token);

}

}

private void InitializeBackgroundWorker()

{

backgroundWorker1 = new BackgroundWorker();

backgroundWorker1.DoWork += new DoWorkEventHandler(backgroundWorker1\_DoWork);

backgroundWorker1.RunWorkerCompleted += new RunWorkerCompletedEventHandler(backgroundWorker1\_RunWorkerCompleted);

}

private void backgroundWorker1\_DoWork(object sender, DoWorkEventArgs e)

{

BackgroundWorker worker = sender as BackgroundWorker;

CancellationToken cancellationToken = (CancellationToken)e.Argument;

server.StartServer(Port, cancellationToken);

}

private void backgroundWorker1\_RunWorkerCompleted(object sender, RunWorkerCompletedEventArgs e)

{

if (e.Error != null)

{

MessageBox.Show("Error: " + e.Error.Message);

server.StopServer();

}

else

{

server.StopServer();

}

}

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

{

saveFileDialog1.Filter = "Text Files (\*.txt)|\*.txt|All Files (\*.\*)|\*.\*";

saveFileDialog1.DefaultExt = "txt";

saveFileDialog1.AddExtension = true;

saveFileDialog1.Title = "Save text to file";

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

{

string filePath = saveFileDialog1.FileName;

try

{

File.WriteAllText(filePath, richTextBox1.Text);

MessageBox.Show("Лог успешно сохранен!", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка сохранения файла: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

}