Министерство транспорта Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Российский университет транспорта» (МИИТ)

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

Курсовая работа

по дисциплине

**«**Методы программирования**»**

Выполнили: ст. гр. ТКИ-342

Павлючков Д.Д.

Рудов Л.А.

Проверил: доцент кафедры УиЗИ, к.т.н. Сафронов А.И.

Москва ­– 2024 г.

**Содержание**

[1. Введение 3](#_Toc179829550)

[2. Цель работы 3](#_Toc179829551)

[3. Задачи работы 3](#_Toc179829552)

[4. Содержательная часть работы 5](#_Toc179829553)

[4.1 Анализ печатной формы 5](#_Toc179829554)

[4.2 Описание предназначения печатной формы 5](#_Toc179829555)

[4.3 Разнообразие печатных форм и выявление различий 6](#_Toc179829556)

[4.4 Унификация печатных форм для удобства обработки данных 6](#_Toc179829557)

[4.5 Разработка экранной формы для импорта сведений из печатной формы 6](#_Toc179829562)

[4.6 Разработка функций экспорта сведений из печатной формы в файл текстовый формата CSV 10](#_Toc179829564)

[4.7 Разработка функций импорта сведений из оригинального и изменённого текстовых файлов формата CSV с воссозданием Word 14](#_Toc179829565)

[4.8 Разработка функций экспорта сведений из экранной формы в Excel 19](#_Toc179829567)

[4.9 Графический анализ данных, содержащихся в печатной форме (настройка Chart-элемента) 24](#_Toc179829568)

[4.9.1 Показывать столбчатую диаграмму количеств исполнителей, приходящихся на разделы 28](#_Toc179829570)

[4.9.2 Показывать столбчатую диаграмму количеств разделов с указанным ключевым словом за всё время с разбивкой по отчётным периодам 31](#_Toc179829572)

[4.9.3 Показывать столбчатую диаграмму количеств разделов, исполняемых сольно, за всё время с разбивкой по отчётным периодам 35](#_Toc179829573)

[4.9.4 Показывать столбчатую диаграмму количеств упоминаний ключевых слов, разделённых ";", для выбранного отчётного периода 38](#_Toc179829575)

[4.9.5 Показывать столбчатую диаграмму количеств разделов внутри отчётных периодов за всё время 40](#_Toc179829576)

[4.9.6 Показывать столбчатую диаграмму количеств исполнителей без степеней и званий за всё время с разбивкой по отчётным периодам 42](#_Toc179829578)

[5. Инструкция пользователя 46](#_Toc179829579)

[6. Заключение 49](#_Toc179829580)

[7. Выводы 67](#_Toc179829581)

[8. Библиографический список 68](#_Toc179829582)

## **Введение**

В современном мире способность быстро и точно обрабатывать информацию становится критически важной. Особенно это касается печатных форм, которые часто служат основой для сбора и анализа данных в различных сферах деятельности. В связи с этим, данная работа направлена на разработку приложения, которое позволит автоматизировать процесс обработки данных из печатных форм, повышая эффективность работы с документами и предоставляя удобный инструмент для анализа данных. Программа позволит пользователю просматривать файлы .docx в экранной форме приложения, конвертировать их в форматы .csv и .xlsx, а также анализировать данные с помощью диаграмм.

## **Цель работы**

Цель работы заключается в создании удобного и функционального приложения в режиме *Windows Forms Application* на языке *Visual C#*, для конвертации печатных форм из Word в СSV и Excel. Также необходимо уметь воссоздавать структуру печатной формы. В ходе работы освоить элемент Chart для графического анализа данных из печатных форм.

## **Задачи работы**

**Рабочее задание:**

1. Проанализировать печатные формы;

2. Уметь воссоздавать структуру печатной формы;

3. Уметь читать печатную форму и представлять информацию из неё на экранной форме;

4. Уметь конвертировать печатную форму Word – Excel;

5. Уметь анализировать данные, получаемые из нескольких печатных форм за различные периоды (и представлять результаты анализа в Chart-элементе);

6. Технологический процесс работы приложения отразить на карте, выполненной в нотации сетей Петри, в сочетании с классическими схемами алгоритма для этапов, отмеченных как "эффекты".

**Индивидуальное задание:**

1. Показывать столбчатую диаграмму количеств исполнителей, приходящихся на разделы, за всё время;

2. Показывать столбчатую диаграмму количеств исполнителей, приходящихся на разделы, за указанный отчётный период;

3. Показывать столбчатую диаграмму количеств разделов с указанным ключевым словом за всё время с разбивкой по отчётным периодам;

4. Показывать столбчатую диаграмму количеств разделов, исполняемых сольно, за всё время с разбивкой по отчётным периодам;

5. Показывать столбчатую диаграмму количеств упоминаний ключевых слов, разделённых ";", для выбранного отчётного периода;

6. Показывать столбчатую диаграмму количеств разделов внутри отчётных периодов за всё время;

7. Показывать столбчатую диаграмму количеств исполнителей без степеней и званий за всё время с разбивкой по отчётным периодам;

8. Показывать столбчатую диаграмму количеств исполнителей без степеней за всё время с разбивкой по отчётным периодам;

## **Содержательная часть работы**

## **Анализ печатной формы**

Печатные формы представляют собой документы WORD, в которых находится таблица, представляющая собой тематический план бюджетных исследований кафедры ''Управление и защита информации'' с 2015 по 2020 года.

## **4.2 Описание предназначения печатной формы**

Таблица включает в себя наименования тем и ФИО исполнителей ответственных за данную тему, цель работы и ожидаемый результат.

На рисунке 1 показан общий вид печатной формы.

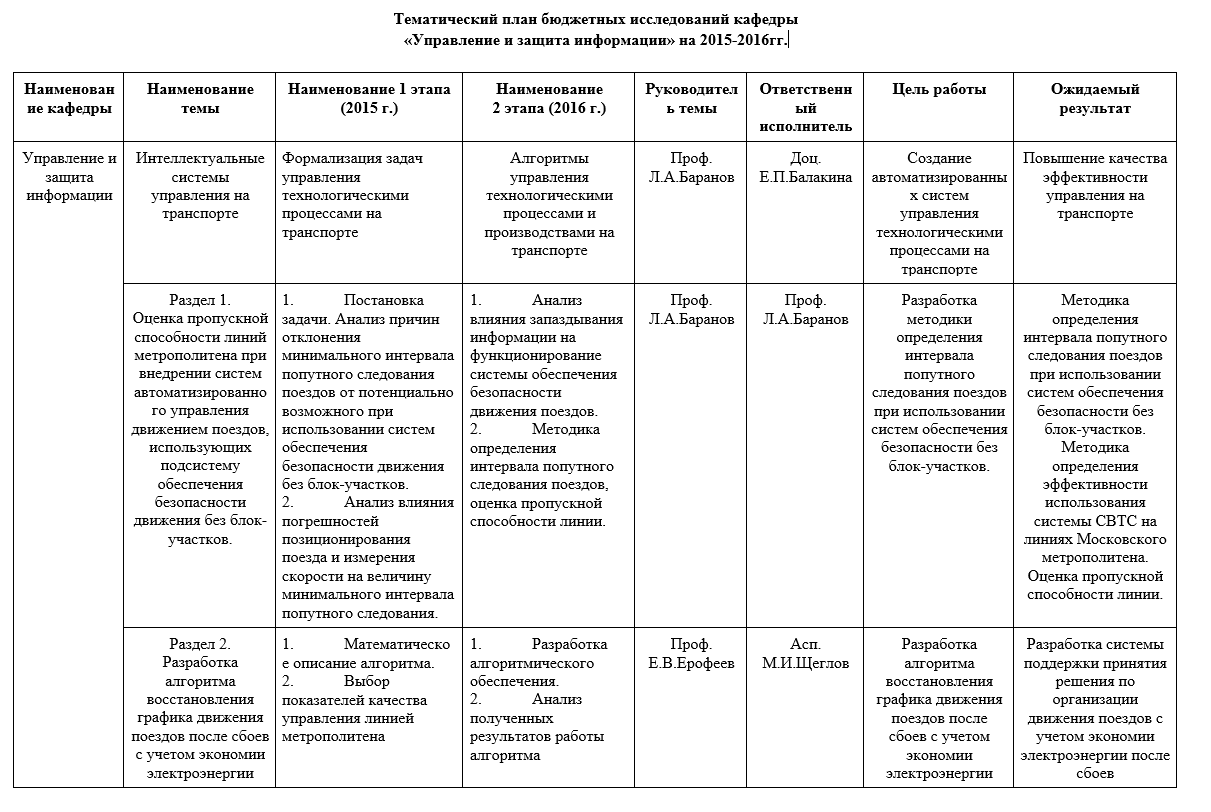


Рисунок 1 – Общий вид печатной формы

## **4.3 Разнообразие печатных форм и выявление различий**

Таблицы в файлах отличаются количеством столбцов. Например, в файле под названием «НИР 2015-2016 1» по сравнению с файлом «НИР 2017-2018 1» большее количество столбцов. Данное различие может помешать анализу печатных форм.

## **4.4 Унификация печатных форм для удобства обработки данных**

Для удобства обработки данных уберем в таблицах текст перед и после таблиц, лишние пустые строки и столбцы, в файле «НИР 2015-2016 1» уберем первый столбец «Наименование кафедры», так как в других печатных формах его нет. Также объединенные ячейки с ФИО исполнителей разделим так, чтобы напротив каждой ячейки с наименованием темы стояло ФИО исполнителя. Это необходимо для достоверного построения столбчатой диаграммы в Chart-элементе.

* 5. **Разработка экранной формы для импорта сведений из печатной формы**

Для импорта сведений из печатной формы подойдет элемент управления *DataGridView,* так как он лучше всего отражает данные, хранящиеся в таблицах. Создадим дочернюю форму «Предпросмотр печатной формы», которая будет показывать в элементе *DataGridView* данные, хранящиеся в печатной форме [1]. Данные в *DataGridView* переносятся уже после того, как печатная форма конвертировалась в формат csv. На рисунке 2 отобразим сеть Петри, отражающую процесс разработки экранной формы для импорта сведений из печатной формы. А на рисунке 3 покажем схемы этого алгоритма.

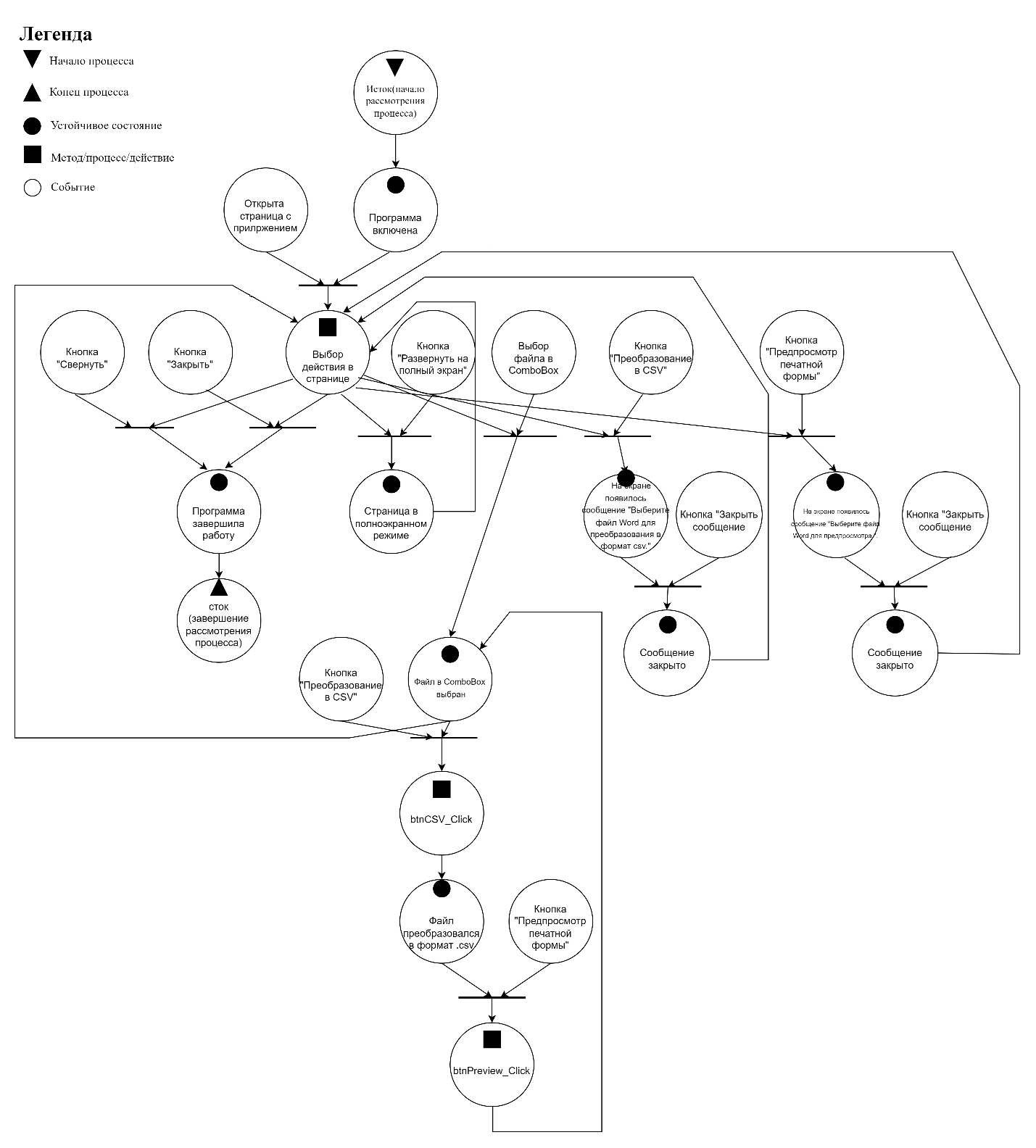


Рисунок 2 – Сеть Петри, отражающая процесс разработки экранной формы для импорта сведений из печатной формы

Изображение выглядит как текст, снимок экрана, Шрифт, графический дизайн

Автоматически созданное описание

Рисунок 3 – Схемы алгоритмов, отражающие процесс разработки экранной формы для импорта сведений из печатной формы

На рисунке 4 покажем, как выглядит экранная форма с элементом DataGridView после импорта данных из печатной формы.

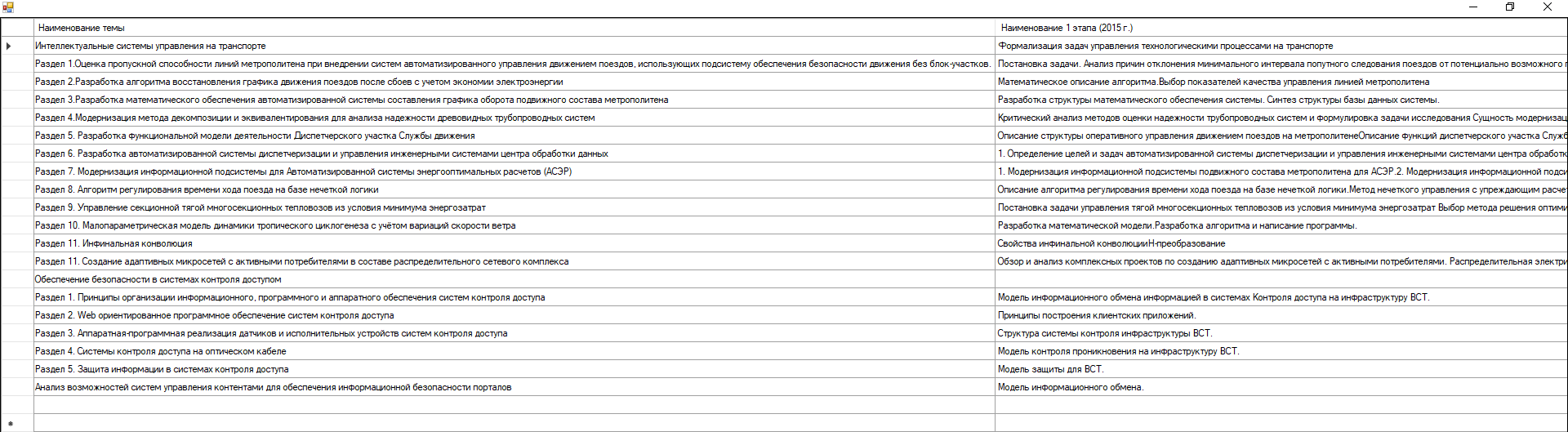


Рисунок 4 – Экранная форма с элементом DataGridView после импорта данных из печатной формы.

**Листинг кода, реализующий разработку экранной формы для импорта сведений из печатной формы**

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

{

if (cBWord.SelectedItem != null)

{

if (!string.IsNullOrEmpty(csvFilePath))

{

Form frmPreview = new Form();

dataGridView = new DataGridView();

dataGridView.Dock = DockStyle.Fill;

frmPreview.Controls.Add(dataGridView);

frmPreview.Show();

LoadCSVDataToDataGridView(csvFilePath);

}

else

{

MessageBox.Show("Сначала необходимо преобразовать файл Word в CSV.", "Предупреждение");

}

}

else

{

MessageBox.Show("Выберите файл Word для предпросмотра.", "Предупреждение");

}

}

private void LoadCSVDataToDataGridView(string csvFilePath)

{

if (File.Exists(csvFilePath))

{

try

{

System.Data.DataTable dataTable = new System.Data.DataTable();

string[] lines = File.ReadAllLines(csvFilePath);

if (lines.Length > 0)

{

string[] headers = lines[0].Split('|');

foreach (var header in headers)

{

dataTable.Columns.Add(header);

}

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

{

string[] values = lines[i].Split('|');

DataRow dataRow = dataTable.NewRow();

for (int j = 0; j < values.Length && j < dataTable.Columns.Count; j++)

{

dataRow[j] = values[j].Trim();

}

dataTable.Rows.Add(dataRow);

}

dataGridView.DataSource = dataTable;

dataGridView.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;

}

else

{

MessageBox.Show("CSV файл пустой.", "Предупреждение");

}

}

catch (Exception ex)

{

MessageBox.Show($"Произошла ошибка при загрузке данных: {ex.Message}", "Ошибка");

}

}

else

{

MessageBox.Show($"CSV файл не найден: {csvFilePath}", "Ошибка");

}

## }

## **Разработка функций экспорта сведений из печатной формы в файл текстовый формата CSV**

Для того, чтобы произвести экспорт сведений из WORD в Excel изначально необходимо разработать функцию экспорта сведений из печатной формы в файл текстовый формата CSV. CSV- файлы – это файлы особого типа, которые можно создавать и редактировать в Excel. В CSV-файлах данные хранятся не в столбцах, а разделенные запятыми. Текст и числа, сохраненные в CSV-файле, можно легко переносить из одной программы в другую. Так как в наших печатных файлах в данных часто встречается запятая, изменим знак разделителя на «|». Используем класс [StringBuilder](https://learn.microsoft.com/ru-ru/dotnet/api/system.text.stringbuilder) для повышения производительности при соединении большого количества строк в цикле [2]. После экспорта сведений из печатной формы текстовый файл формата CSV сохраняется в папке «csv». На рисунке 5 отобразим сеть Петри, отражающую процесс разработки экранной формы для импорта сведений из печатной формы. А на рисунке 6 покажем схемы этого алгоритма.

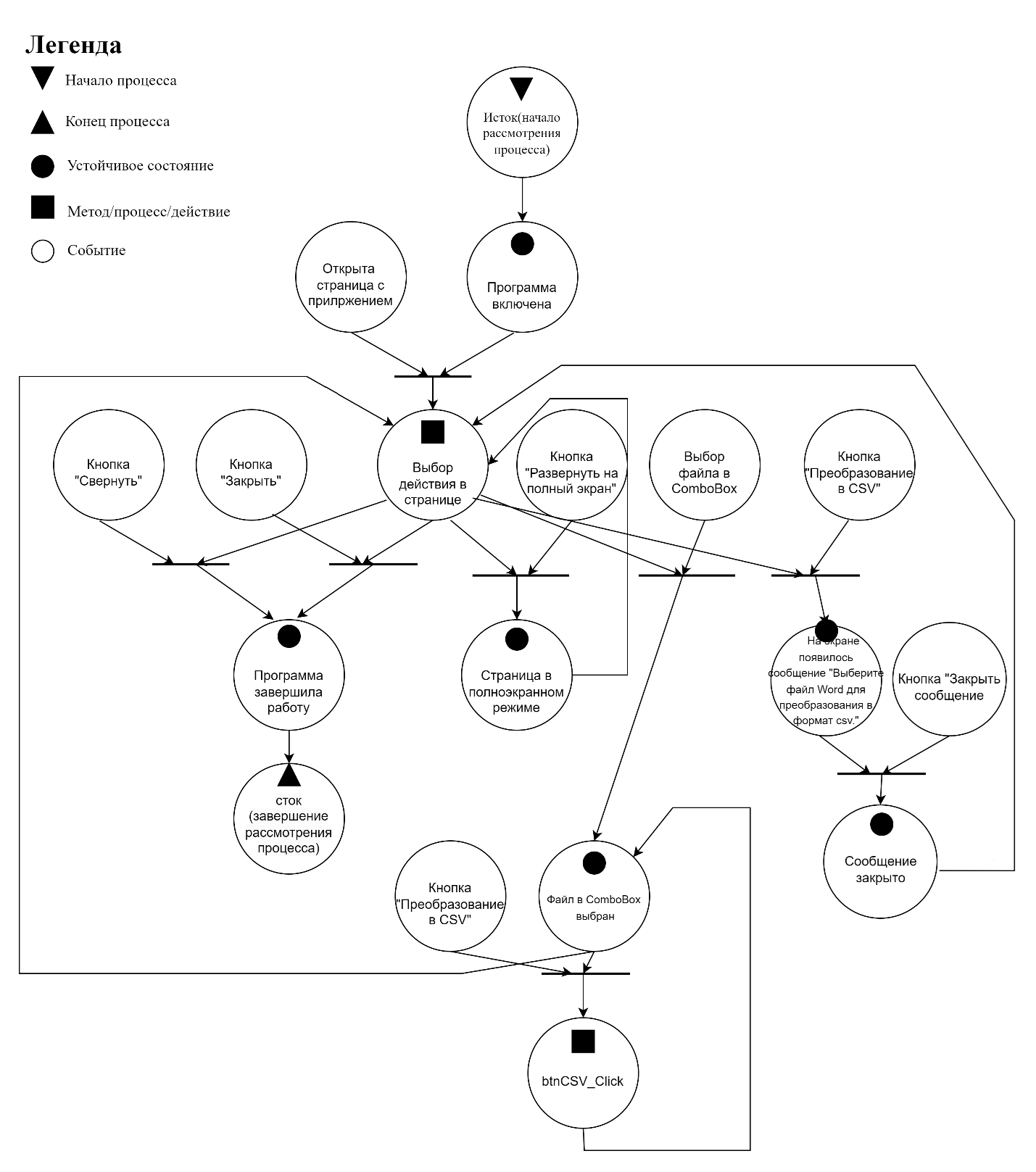


Рисунок 5 – Сеть Петри, отражающая процесс разработки функции экспорта сведений из печатной формы в файл текстовый формата CSV

Изображение выглядит как текст, снимок экрана, Шрифт, черно-белый

Автоматически созданное описание

Рисунок 6 – Схемы алгоритмов, отражающие процесс разработки функции экспорта сведений из печатной формы в файл текстовый формата CSV

**Листинг кода, реализующий разработку функций экспорта сведений из печатной формы в файл текстовый формата CSV**

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

{

if (cBWord.SelectedItem != null)

{

string selectedFile = Path.Combine(Environment.CurrentDirectory, "печатные формы", $"{cBWord.SelectedItem}.docx");

if (File.Exists(selectedFile))

{

ConvertWordToCSV(selectedFile);

MessageBox.Show($"Файл успешно сохранен в папке csv как {cBWord.SelectedItem}.csv", "Успех");

}

else

{

MessageBox.Show("Выбранный файл Word не найден.", "Ошибка");

}

}

else

{

MessageBox.Show("Выберите файл Word для преобразования в формат csv.", "Предупреждение");

}

}

private void ConvertWordToCSV(string selectedFile)

{

try

{

using (WordprocessingDocument wordDocument = WordprocessingDocument.Open(selectedFile, true))

{

MainDocumentPart mainPart = wordDocument.MainDocumentPart;

var tables = mainPart.Document.Descendants<DocumentFormat.OpenXml.Wordprocessing.Table>();

StringBuilder csvBuilder = new StringBuilder();

foreach (var table in tables)

{

csvBuilder.AppendLine(string.Join("|", table.Descendants<DocumentFormat.OpenXml.Wordprocessing.TableRow>().First().Descendants<DocumentFormat.OpenXml.Wordprocessing.TableCell>().Select(c => c.InnerText.Trim())));

foreach (var row in table.Descendants<DocumentFormat.OpenXml.Wordprocessing.TableRow>().Skip(1))

{

csvBuilder.AppendLine(string.Join("|", row.Descendants<DocumentFormat.OpenXml.Wordprocessing.TableCell>().Select(c => c.InnerText.Trim())));

}

csvBuilder.AppendLine();

}

string folderCSV = Path.Combine(Environment.CurrentDirectory, "csv");

if (!Directory.Exists(folderCSV))

{

Directory.CreateDirectory(folderCSV);

}

csvFilePath = Path.Combine(folderCSV, $"{cBWord.SelectedItem}.csv");

File.WriteAllText(csvFilePath, csvBuilder.ToString());

}

}

catch (Exception ex)

{

MessageBox.Show($"Произошла ошибка при преобразовании файла: {ex.Message}", "Ошибка");

}

}

На рисунках 7 и 8 отобразим текстовый файл формата CSV после экспорта сведений из печатной формы.

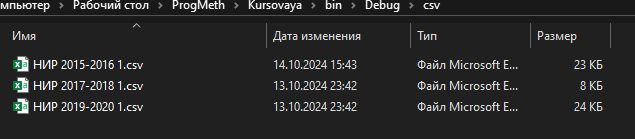


Рисунок 7 – Текстовый файл формата CSV после экспорта сведений из печатной формы в папке «csv»

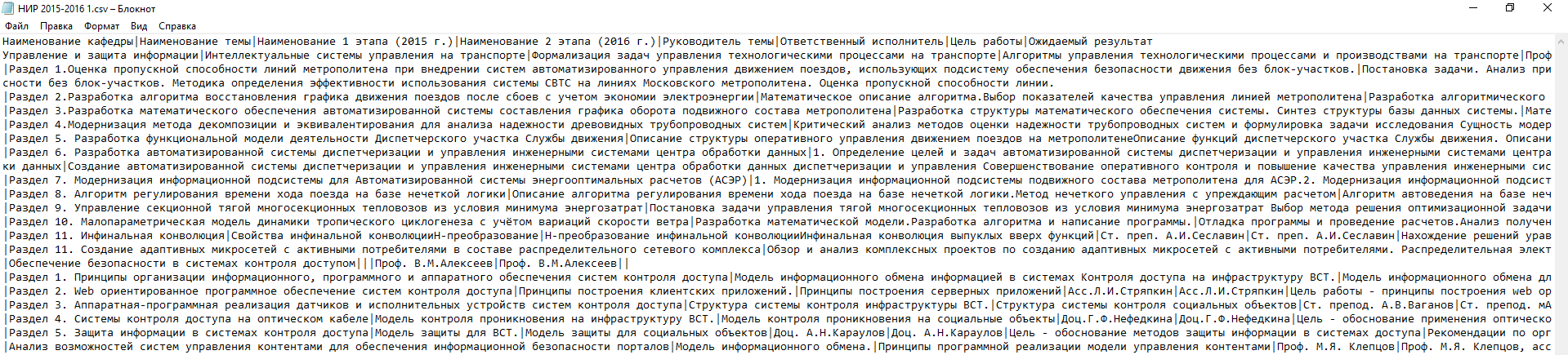
****

Рисунок 8 – Текстовый файл формата CSV после экспорта сведений из печатной формы

## **4.7** **Разработка функций импорта сведений из оригинального и изменённого текстовых файлов формата CSV с воссозданием Word**

Из папки под названием «csv» файл CSV построчно переносит данные в файл WORD. Новая строка в файле CSV означает новую строку в таблице WORD. А «|» в файле CSV означает новый столбец в файле WORD. После импорта сведений из текстового файла формата CSV файл формата WORD сохраняется в папке «word». Используем пакет SDK Open XML для создания Word структуры документа и содержимого с помощью строго типизированных классов, соответствующих элементам WordprocessingML [3]. На рисунке 9 отобразим сеть Петри, отражающую процесс разработки функций импорта сведений из оригинального и изменённого текстовых файлов формата CSV с воссозданием Word. А на рисунке покажем 10 схему этого алгоритма.

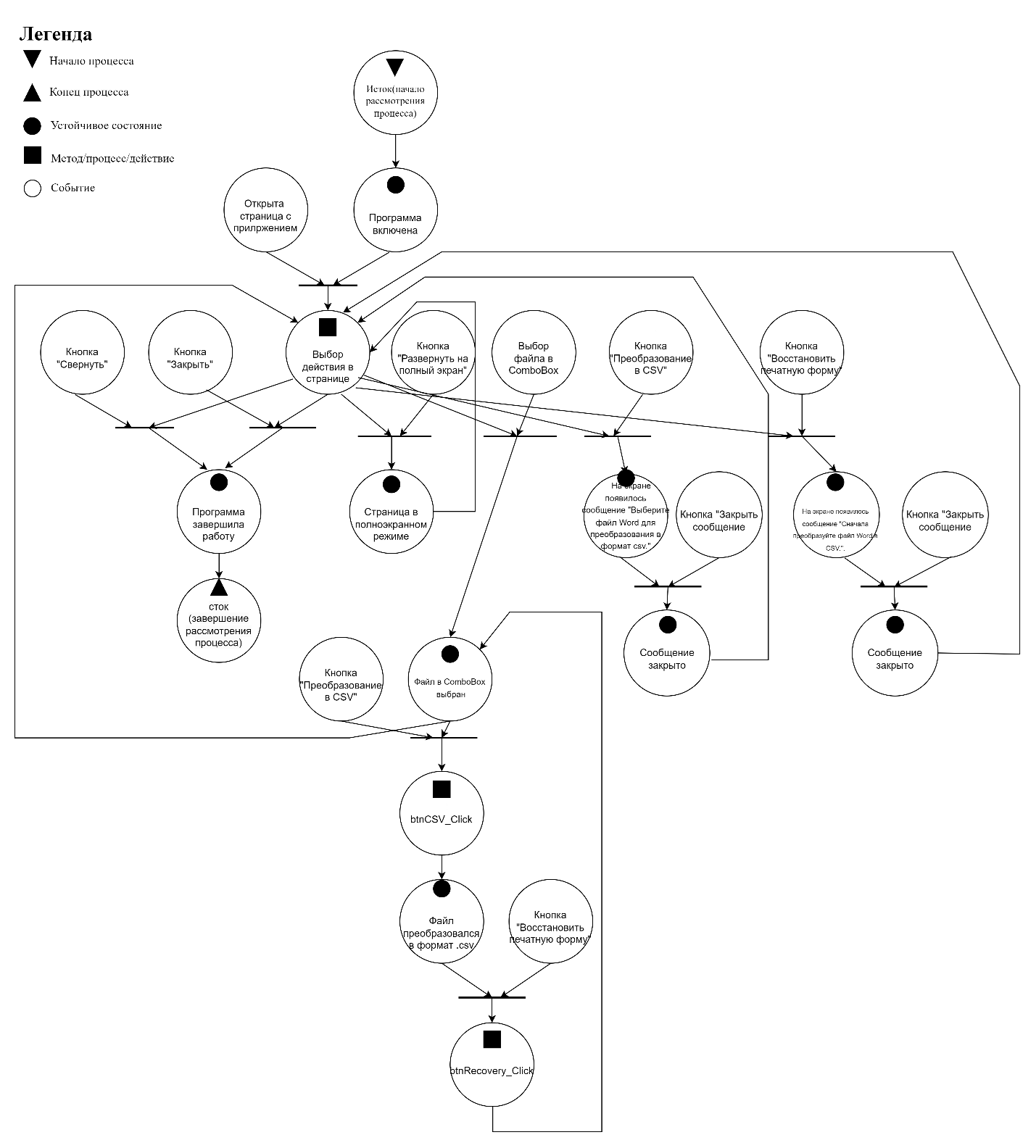


Рисунок 9 – Сеть Петри, отражающая процесс разработки функций импорта сведений из оригинального и изменённого текстовых файлов формата CSV с воссозданием Word

Изображение выглядит как текст, снимок экрана, черно-белый, графический дизайн

Автоматически созданное описание

Рисунок 10 – Схема алгоритма, отражающая процесс разработки функций импорта сведений из оригинального и изменённого текстовых файлов формата CSV с воссозданием Word

На рисунках 11 и 12 отобразим файл, воссозданный в WORD из файла формата CSV.

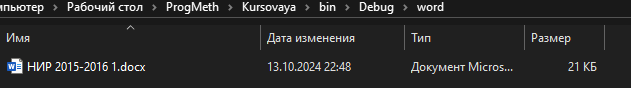


Рисунок 11 – Файл, воссозданный в WORD из файла формата CSV в папке «word»



Рисунок 12 – Файл, воссозданный в WORD из файла формата CSV

**Листинг кода, реализующий разработку функций импорта сведений из оригинального и изменённого текстовых файлов формата CSV с воссозданием Word**

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

{

if (!string.IsNullOrEmpty(csvFilePath))

{

try

{

string[] lines = File.ReadAllLines(csvFilePath);

string wordFilePath = Path.Combine(Directory.GetCurrentDirectory(), "word", $"{cBWord.SelectedItem}.docx");

using (WordprocessingDocument wordDocument = WordprocessingDocument.Create(wordFilePath, WordprocessingDocumentType.Document))

{

MainDocumentPart mainPart = wordDocument.AddMainDocumentPart();

mainPart.Document = new Document();

Body body = new Body();

DocumentFormat.OpenXml.Wordprocessing.Table table = new DocumentFormat.OpenXml.Wordprocessing.Table();

table.AppendChild(new TableProperties(

new TableBorders(

new DocumentFormat.OpenXml.Wordprocessing.TopBorder() { Val = new EnumValue<BorderValues>(BorderValues.Single), Size = 6 },

new DocumentFormat.OpenXml.Wordprocessing.BottomBorder() { Val = new EnumValue<BorderValues>(BorderValues.Single), Size = 6 },

new DocumentFormat.OpenXml.Wordprocessing.LeftBorder() { Val = new EnumValue<BorderValues>(BorderValues.Single), Size = 6 },

new DocumentFormat.OpenXml.Wordprocessing.RightBorder() { Val = new EnumValue<BorderValues>(BorderValues.Single), Size = 6 },

new InsideVerticalBorder() { Val = new EnumValue<BorderValues>(BorderValues.Single), Size = 6 }

)

));

foreach (var line in lines)

{

if (!string.IsNullOrEmpty(line))

{

TableRow row = new TableRow();

string[] values = line.Split('|'); int currentColumnsCount = 0;

foreach (var value in values)

{

TableCell cell = new TableCell(new Paragraph(new DocumentFormat.OpenXml.Wordprocessing.Run(new DocumentFormat.OpenXml.Wordprocessing.Text(value))));

row.Append(cell);

currentColumnsCount++;

}

while (currentColumnsCount < row.ChildElements.Count)

{

row.Append(new TableCell(new Paragraph(new DocumentFormat.OpenXml.Wordprocessing.Run())));

currentColumnsCount++;

}

table.Append(row);

}

}

body.Append(table);

mainPart.Document.Append(body);

}

MessageBox.Show($"Файл успешно сохранен в папке word как {cBWord.SelectedItem}.docx", "Успех");

}

catch (Exception ex)

{

MessageBox.Show($"Произошла ошибка при восстановлении файла: {ex.Message}", "Ошибка");

}

}

else

{

MessageBox.Show("Сначала преобразуйте файл Word в CSV.", "Предупреждение");

}

## }

## **4.8 Разработка функций экспорта сведений из экранной формы в Excel**

Из папки под названием «csv» файл CSV построчно переносит данные в файл EXCEL. Новая строка в файле CSV означает новую строку в таблице EXCEL. А «!» в файле CSV означает новый столбец в файле EXCEL. После импорта сведений из текстового файла формата CSV файл формата EXCEL сохраняется в папке «excel». Используем пакет Open XML SDK, где класс [SpreadsheetDocument](https://msdn.microsoft.com/library/office/documentformat.openxml.packaging.spreadsheetdocument.aspx) представляет пакет документа Excel [4]. На рисунке 13 отобразим сеть Петри, отражающую разработку функций экспорта сведений из экранной формы в Excel. А на рисунке покажем 14 схему этого алгоритма.

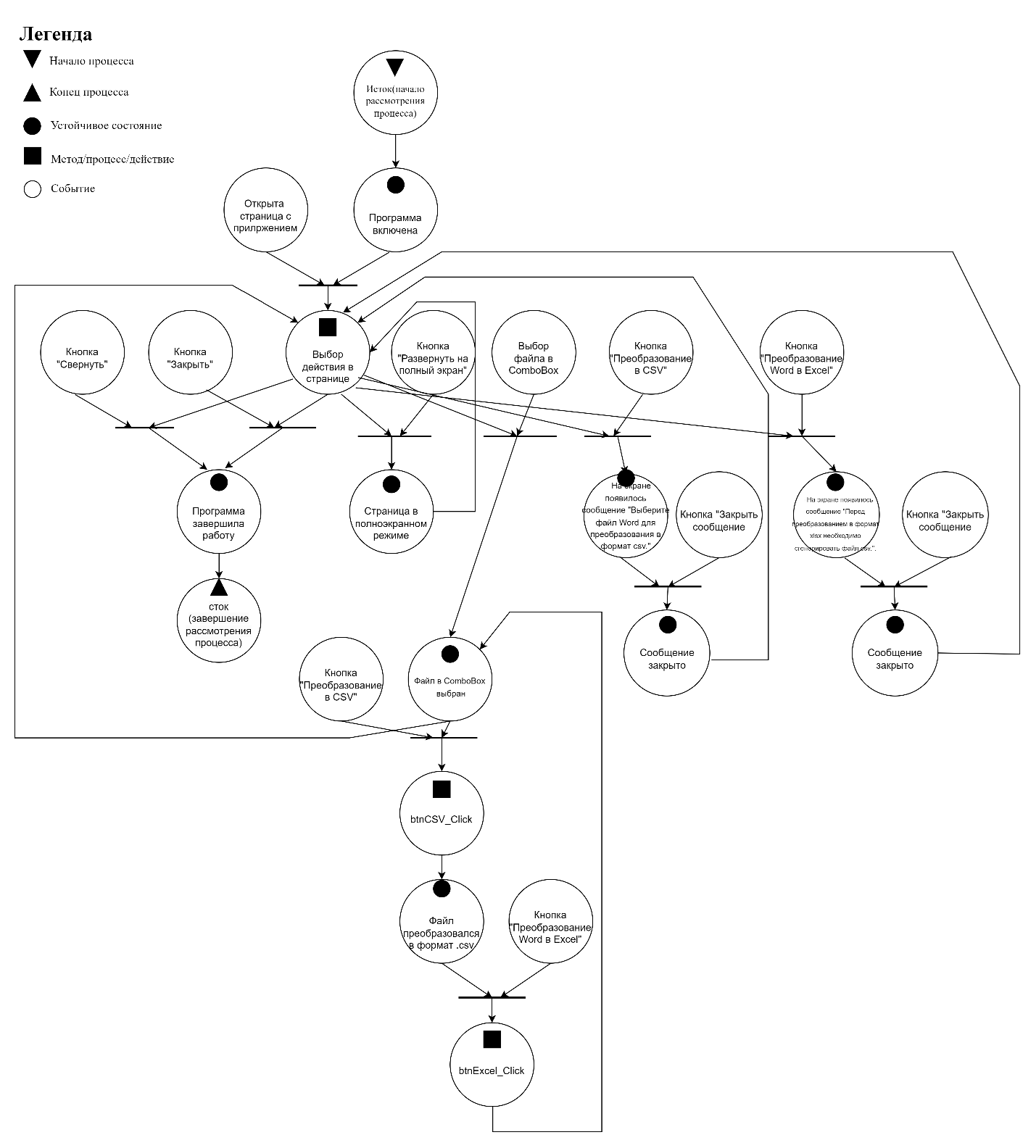


Рисунок 13 – Сеть Петри, отражающая разработку функций экспорта сведений из экранной формы в Excel

Изображение выглядит как снимок экрана, текст, черно-белый, черный

Автоматически созданное описание

Рисунок 14 – Схема алгоритма, отражающая разработку функций экспорта сведений из экранной формы в Excel

На рисунках 15 и 16 отобразим файл, экспортирующий сведения в EXCEL из файла формата CSV.

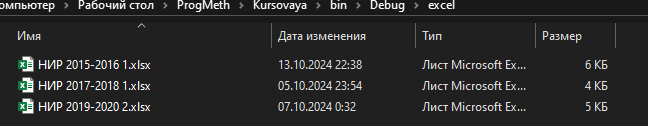


Рисунок 15 – Файл, экспортирующий сведения в EXCEL из файла формата CSV в папке «excel»

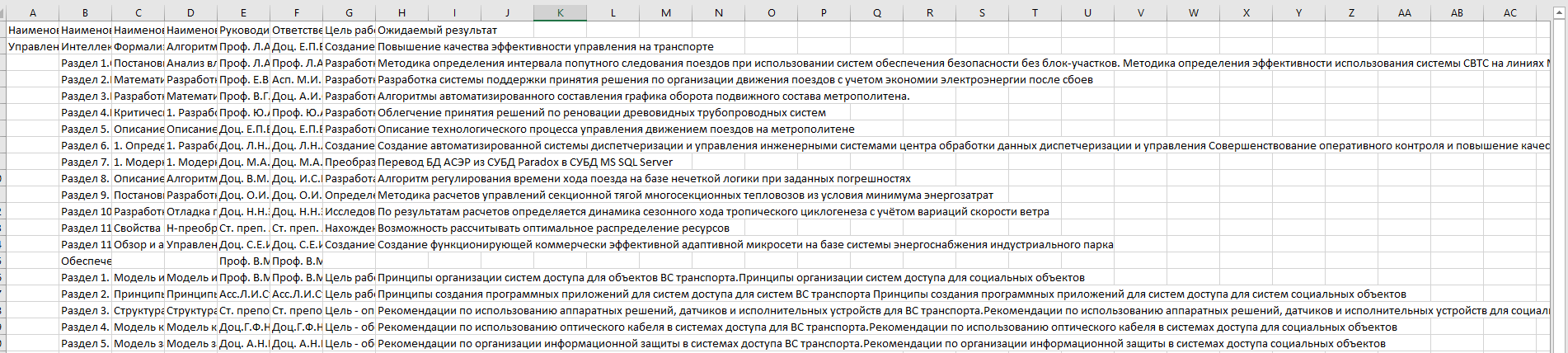


Рисунок 16 – Файл, экспортирующий сведения в EXCEL из файла формата CSV

**Листинг кода, реализующий разработку функций экспорта сведений из экранной формы в Excel**

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

{

if (File.Exists(csvFilePath))

{

string folderExcel = Path.Combine(Environment.CurrentDirectory, "excel");

if (!Directory.Exists(folderExcel))

{

Directory.CreateDirectory(folderExcel);

}

string excelFilePath = Path.Combine(folderExcel, $"{cBWord.SelectedItem}.xlsx");

System.Data.DataTable dataTable = new System.Data.DataTable();

using (StreamReader reader = new StreamReader(csvFilePath))

{

string headersLine = reader.ReadLine();

string[] headers = headersLine.Split('|');

foreach (string header in headers)

{

dataTable.Columns.Add(header);

}

dataTable.Rows.Add(headers);

while (!reader.EndOfStream)

{

string[] values = reader.ReadLine().Split('|');

if (values.Length > dataTable.Columns.Count)

{

for (int i = dataTable.Columns.Count; i < values.Length; i++)

{

dataTable.Columns.Add($"Столбец{i + 1}");

}

}

dataTable.Rows.Add(values);

}

}

using (var document = SpreadsheetDocument.Create(excelFilePath, SpreadsheetDocumentType.Workbook))

{

var workbookPart = document.AddWorkbookPart();

workbookPart.Workbook = new Workbook();

var worksheetPart = workbookPart.AddNewPart<WorksheetPart>();

worksheetPart.Worksheet = new Worksheet();

var sheets = workbookPart.Workbook.AppendChild(new Sheets());

sheets.AppendChild(new Sheet()

{

Id = workbookPart.GetIdOfPart(worksheetPart),

SheetId = 1,

Name = "Лист1"

});

var sheetData = worksheetPart.Worksheet.AppendChild(new SheetData());

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

{

var row = sheetData.AppendChild(new Row());

for (int j = 0; j < dataTable.Columns.Count; j++)

{

var cell = row.AppendChild(new Cell());

cell.DataType = new EnumValue<CellValues>(CellValues.String);

cell.AppendChild(new CellValue($"{dataTable.Rows[i][j]}"));

}

}

}

MessageBox.Show($"Файл успешно сохранен в папке excel как {cBWord.SelectedItem}.xlsx", "Успех");

}

else

{

MessageBox.Show("Перед преобразованием в формат xlsx необходимо сгенерировать файл csv.", "Предупреждение");

}

}

## **4.9 Графический анализ данных, содержащихся в печатной форме (настройка Chart-элемента)**

Необходимо проанализировать данные, получаемые из нескольких печатных форм за различные периоды и представить результаты анализа в Chart-элементе. На рисунке 17 отобразим сеть Петри, отражающую настройку формы для Chart-элемента. А на рисунке 18 покажем схему этого алгоритма.

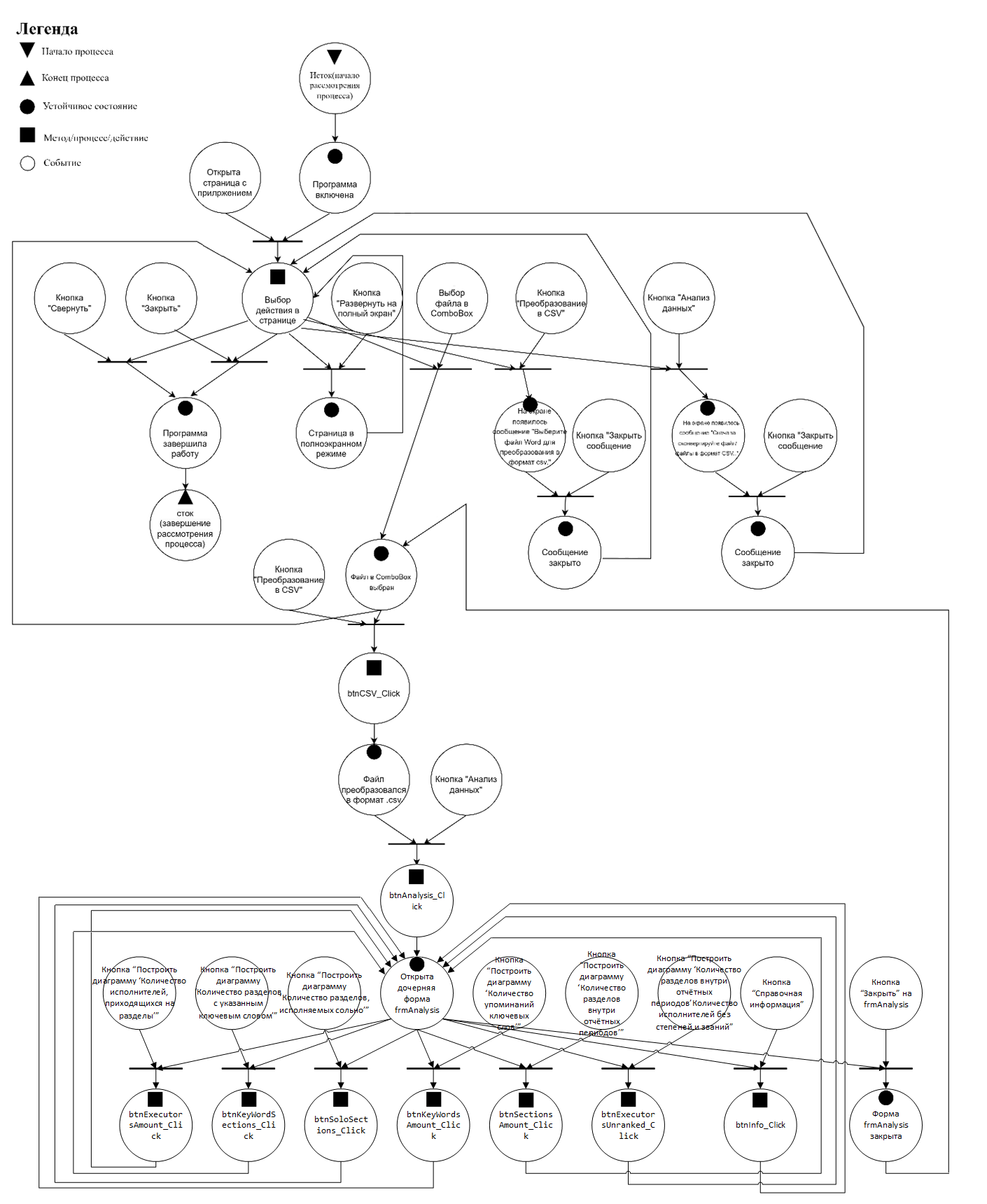


Рисунок 17 – Сеть Петри, отражающая настройку формы для Chart-элемента

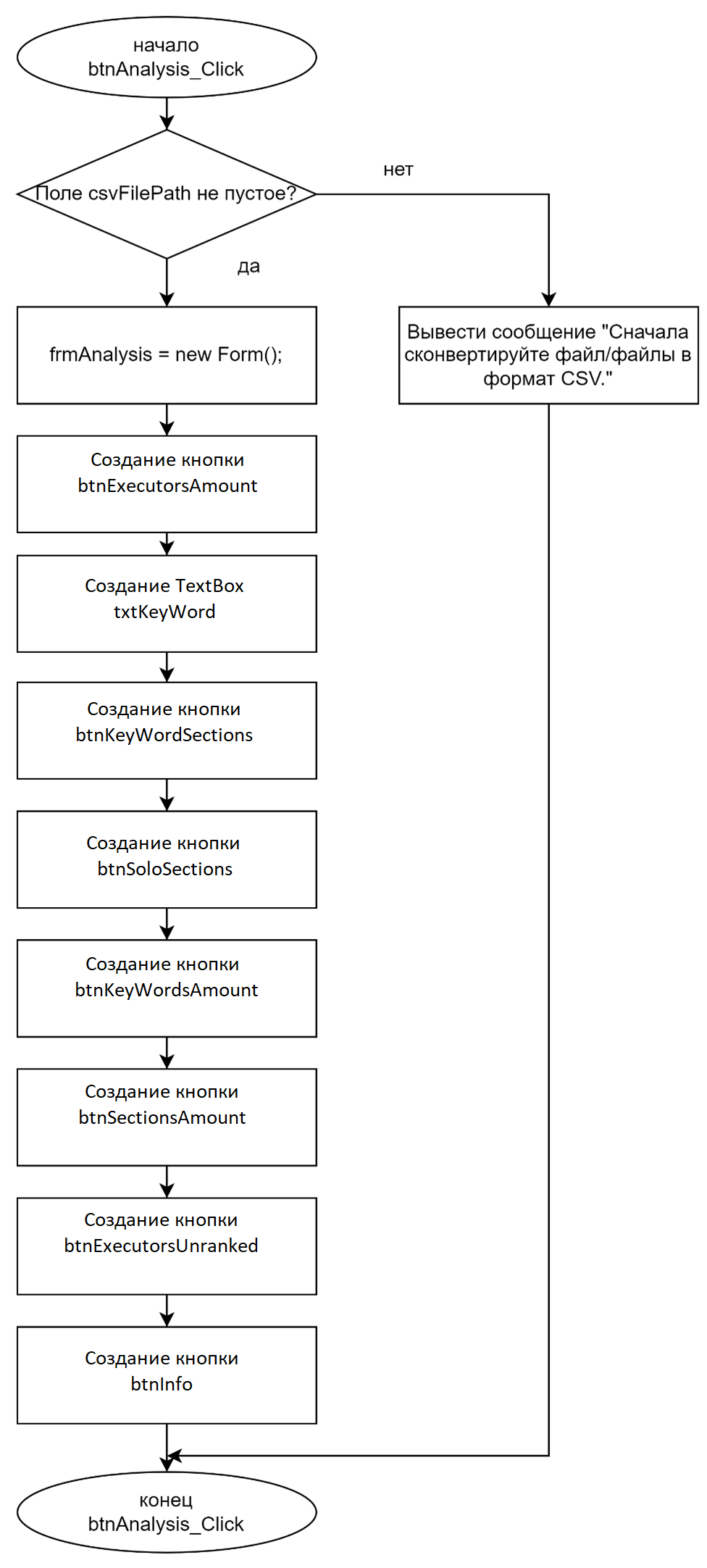


Рисунок 18 – Схема алгоритма, отражающая настройку формы для Chart-элемента

На рисунке 19 покажем дочернюю форму для построения столбчатых диаграмм.

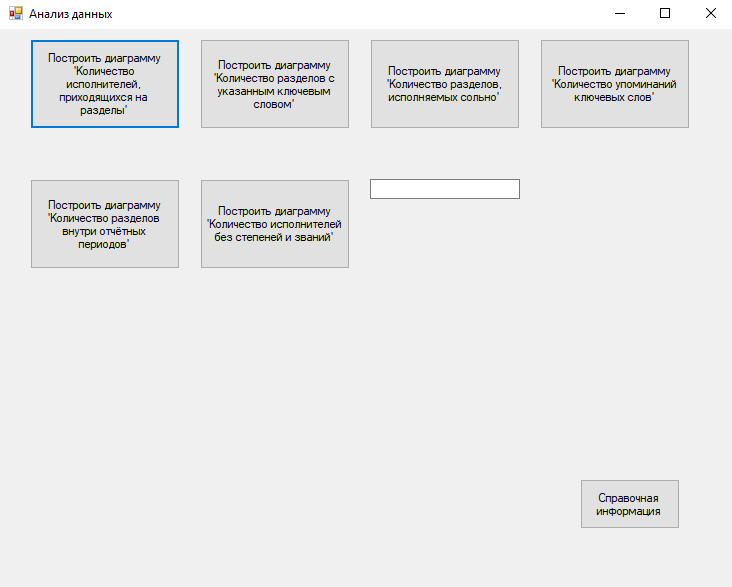


Рисунок 19 – Дочерняя форма для построения столбчатых диаграмм

**Листинг кода, реализующего настройку формы для Chart-элемента**

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

{

if (!string.IsNullOrEmpty(csvFilePath))

{

frmAnalysis = new Form();

frmAnalysis.Text = "Анализ данных";

frmAnalysis.Size = new Size(750, 600);

Button btnExecutorsAmount = new Button();

btnExecutorsAmount.Text = "Построить диаграмму 'Количество исполнителей, приходящихся на разделы'";

btnExecutorsAmount.Size = new Size(150, 90);

btnExecutorsAmount.Location = new Point(30, 10);

btnExecutorsAmount.Click += new EventHandler(btnExecutorsAmount\_Click);

frmAnalysis.Controls.Add(btnExecutorsAmount);

TextBox txtKeyWord = new TextBox();

txtKeyWord.Location = new Point(370, 150);

txtKeyWord.Size = new Size(150, 100);

frmAnalysis.Controls.Add(txtKeyWord);

Button btnKeyWordSections = new Button();

btnKeyWordSections.Text = "Построить диаграмму 'Количество разделов с указанным ключевым словом'";

btnKeyWordSections.Size = new Size(150, 90);

btnKeyWordSections.Location = new Point(200, 10);

btnKeyWordSections.Click += (s, ev) => { btnKeyWordSections\_Click(txtKeyWord.Text); };

frmAnalysis.Controls.Add(btnKeyWordSections);

Button btnSoloSections = new Button();

btnSoloSections.Text = "Построить диаграмму 'Количество разделов, исполняемых сольно'";

btnSoloSections.Size = new Size(150, 90);

btnSoloSections.Location = new Point(370, 10);

btnSoloSections.Click += new EventHandler(btnSoloSections\_Click);

frmAnalysis.Controls.Add(btnSoloSections);

Button btnKeyWordsAmount = new Button();

btnKeyWordsAmount.Text = "Построить диаграмму 'Количество упоминаний ключевых слов'";

btnKeyWordsAmount.Size = new Size(150, 90);

btnKeyWordsAmount.Location = new Point(540, 10);

btnKeyWordsAmount.Click += (s, ev) => { btnKeyWordsAmount\_Click(txtKeyWord.Text); };

frmAnalysis.Controls.Add(btnKeyWordsAmount);

Button btnSectionsAmount = new Button();

btnSectionsAmount.Text = "Построить диаграмму 'Количество разделов внутри отчётных периодов'";

btnSectionsAmount.Size = new Size(150, 90);

btnSectionsAmount.Location = new Point(30, 150);

btnSectionsAmount.Click += new EventHandler(btnSectionsAmount\_Click);

frmAnalysis.Controls.Add(btnSectionsAmount);

Button btnExecutorsUnranked = new Button();

btnExecutorsUnranked.Text = "Построить диаграмму 'Количество исполнителей без степеней и званий'";

btnExecutorsUnranked.Size = new Size(150, 90);

btnExecutorsUnranked.Location = new Point(200, 150);

btnExecutorsUnranked.Click += new EventHandler(btnExecutorsUnranked\_Click);

frmAnalysis.Controls.Add(btnExecutorsUnranked);

Button btnInfo = new Button();

btnInfo.Text = "Справочная информация";

btnInfo.Size = new Size(100, 50);

btnInfo.Location = new Point(580, 450);

btnInfo.Click += new EventHandler(btnInfo\_Click);

frmAnalysis.Controls.Add(btnInfo);

frmAnalysis.Show();

}

else

{

MessageBox.Show("Сначала сконвертируйте файл/файлы в формат CSV.", "Предупреждение");

}

}

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

{

frmInfo = new Form();

frmInfo.Text = "Справочная информация";

frmInfo.Size = new Size(450, 410);

Label lblInfo = new Label();

lblInfo.AutoSize = false;

lblInfo.Size = new Size(frmInfo.ClientRectangle.Width, frmInfo.ClientRectangle.Height - 40);

lblInfo.Location = new Point(0, 10);

lblInfo.Anchor = AnchorStyles.Top | AnchorStyles.Bottom | AnchorStyles.Left | AnchorStyles.Right;

lblInfo.Text = File.ReadAllText("info.txt");

frmInfo.SizeChanged += (s, ev) =>

{

lblInfo.Size = new Size(frmInfo.ClientRectangle.Width, frmInfo.ClientRectangle.Height - 40);

};

frmInfo.Controls.Add(lblInfo);

frmInfo.ShowDialog();

## }

## **4.9.1 Показывать столбчатую диаграмму количеств исполнителей, приходящихся на разделы**

В зависимости от промежутка построения диаграммы, пользователю необходимо сконвертировать нужное количество файлов WORD в формат CSV. На рисунках 21, 23 покажем столбчатые диаграммы «Количество исполнителей, приходящихся на разделы».

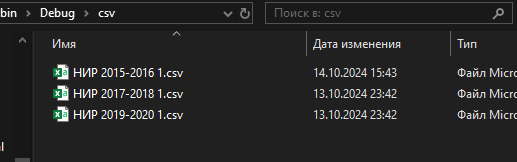


Рисунок 20 – Файлы, из которых берутся данные для диаграммы «Количество исполнителей, приходящихся на разделы»

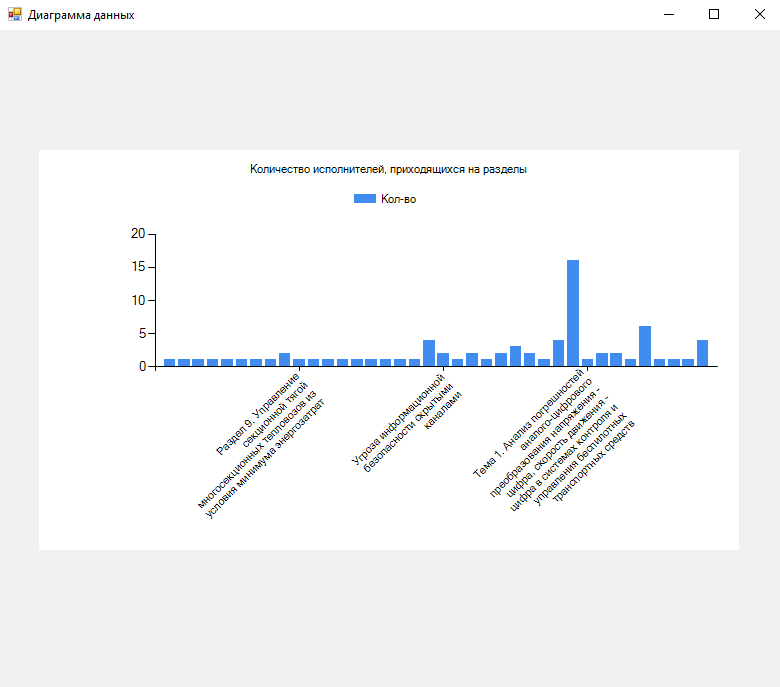


Рисунок 21 – Диаграмма «Количество исполнителей, приходящихся на разделы»

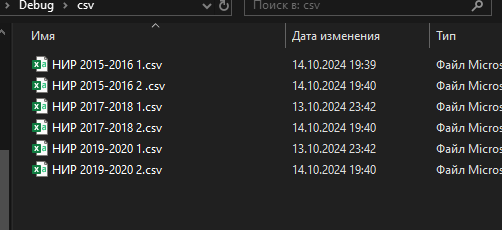


Рисунок 22 – Файлы, из которых берутся данные для диаграммы «Количество исполнителей, приходящихся на разделы»

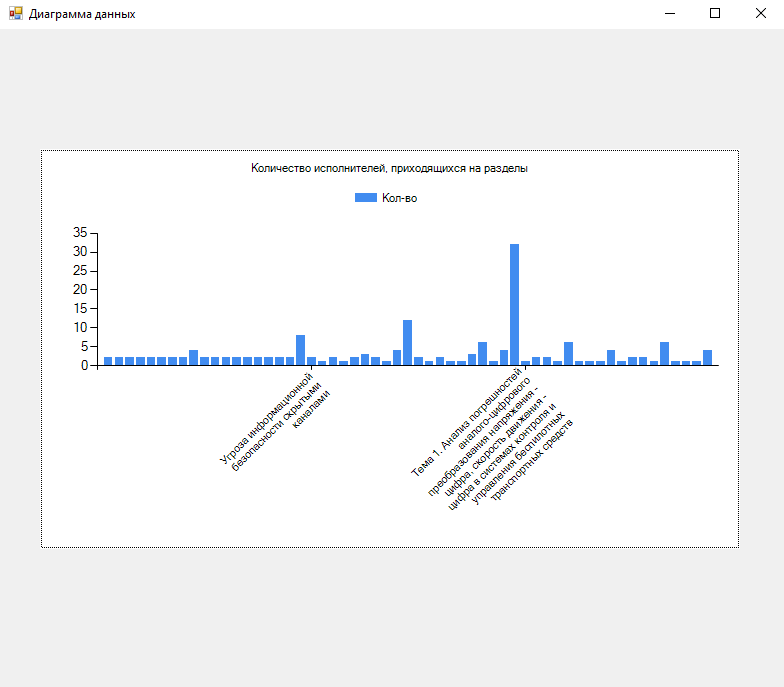


Рисунок 23 – Диаграмма «Количество исполнителей, приходящихся на разделы»

**Листинг кода, реализующий построение диаграмм «Количество исполнителей, приходящихся на разделы»**

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

{

string folderPath = @"csv";

if (Directory.Exists(folderPath))

{

Dictionary<string, int> amount = new Dictionary<string, int>();

foreach (string filePath in Directory.GetFiles(folderPath, "\*.csv"))

{

List<string[]> data = csvReader(filePath);

int ExecutorsCounter(string executors)

{

int count = 1;

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

{

if (executors[i] == ',')

{

count++;

}

}

return (count);

}

foreach (var row in data.Skip(1))

{

if (row.Length > 4)

{

string executors = row[row.Length - 3].Trim();

string section = row[0].Trim();

if (amount.ContainsKey(section))

{

int count = ExecutorsCounter(executors);

amount[section] += count;

}

else

{

int count = ExecutorsCounter(executors);

amount.Add(section, count);

}

}

else if (row.Length == 4)

{

string executors = row[row.Length - 1].Trim();

string section = row[0].Trim();

if (amount.ContainsKey(section))

{

int count = ExecutorsCounter(executors);

amount[section] += count;

}

else

{

int count = ExecutorsCounter(executors);

amount.Add(section, count);

}

}

}

}

diagramBuilder(amount, "Количество исполнителей, приходящихся на разделы");

}

else

{

MessageBox.Show("Сначала сконвертируйте файл в формат CSV.", "Предупреждение");

}

## }

## **4.9.2 Показывать столбчатую диаграмму количеств разделов с указанным ключевым словом за всё время с разбивкой по отчётным периодам**

В зависимости от промежутка построения диаграммы пользователю необходимо сконвертировать нужное количество файлов WORD в формат CSV, после чего вписать в текстовое поле ключевое слово для поиска. На рисунках 25, 27 покажем столбчатые диаграммы «Количество разделов с указанным ключевым словом».

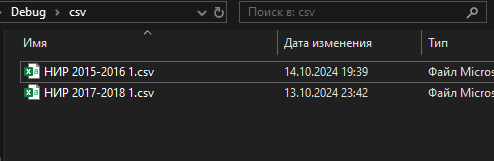


Рисунок 24 – Файлы, из которых берутся данные для диаграммы «Количество разделов с указанным ключевым словом»

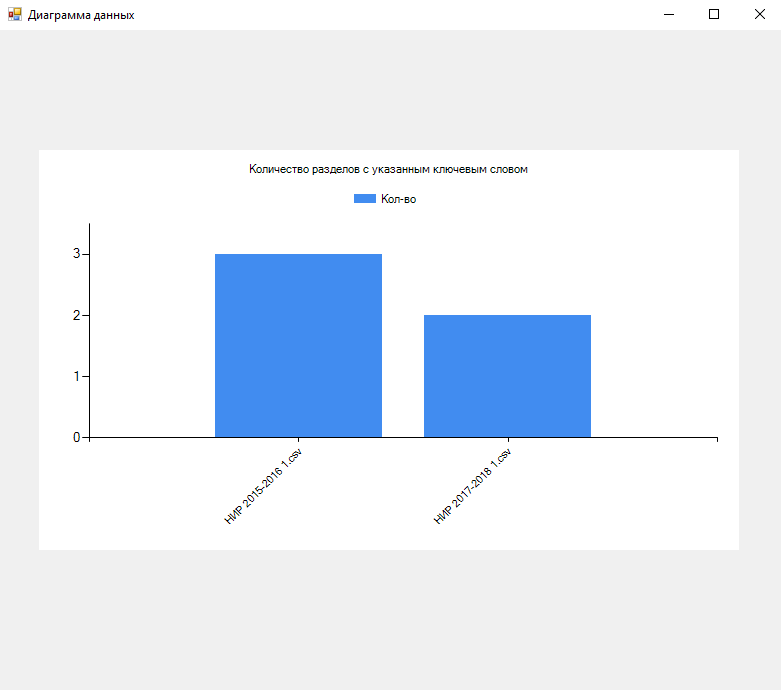


Рисунок 25 – Диаграмма «Количество разделов с указанным ключевым словом» с ключевым словом «безопасности»

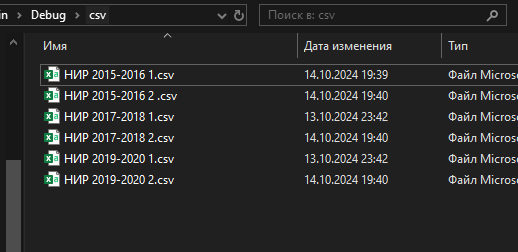


Рисунок 26 – Файлы, из которых берутся данные для диаграммы «Количество разделов с указанным ключевым словом»

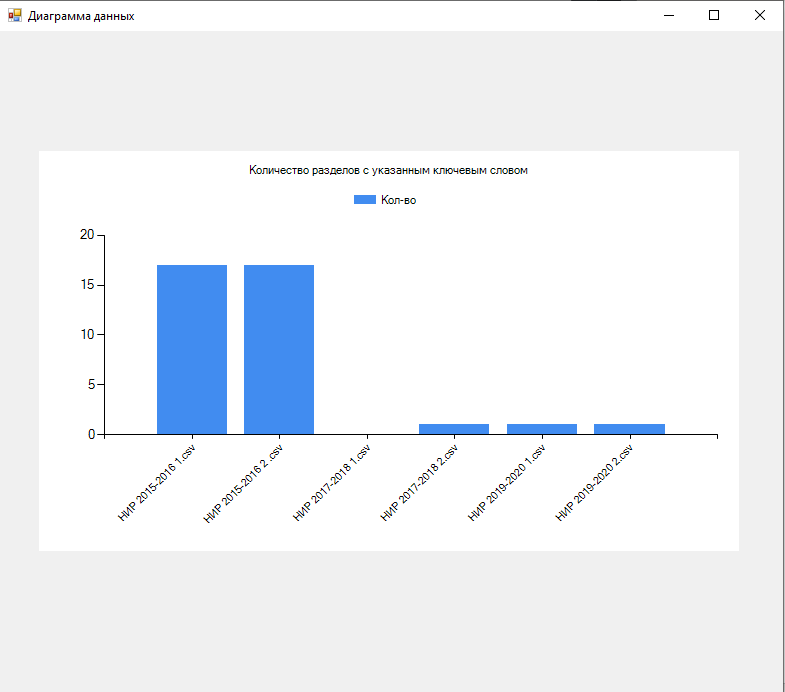


Рисунок 27 – Диаграмма «Количество разделов с указанным ключевым словом» с ключевым словом «Раздел»

**Листинг кода, реализующий построение диаграмм «Количество разделов с указанным ключевым словом»**

private void btnKeyWordSections\_Click(string keyword)

{

string folderPath = @"csv";

if (Directory.Exists(folderPath))

{

Dictionary<string, int> amount = new Dictionary<string, int>();

foreach (string filePath in Directory.GetFiles(folderPath, "\*.csv"))

{

List<string[]> data = csvReader(filePath);

string fileName = Path.GetFileName(filePath);

foreach (var row in data.Skip(1))

{

if (row.Length > 0)

{

string executors = row[0].Trim();

if (amount.ContainsKey(fileName))

{

int count = 0;

if (executors.Contains(keyword))

{

count++;

}

amount[fileName] += count;

}

else

{

int count = 0;

if (executors.Contains(keyword))

{

count++;

}

amount.Add(fileName, count);

}

}

}

}

diagramBuilder(amount, "Количество разделов с указанным ключевым словом");

}

else

{

MessageBox.Show("Сначала сконвертируйте файл в формат CSV.", "Предупреждение");

}

## **4.9.3 Показывать столбчатую диаграмму количеств разделов, исполняемых сольно, за всё время с разбивкой по отчётным периодам**

В зависимости от промежутка построения диаграммы пользователю необходимо сконвертировать нужное количество файлов WORD в формат CSV. Покажем диаграмму «Количество разделов, исполняемых сольно» на рисунке 29.

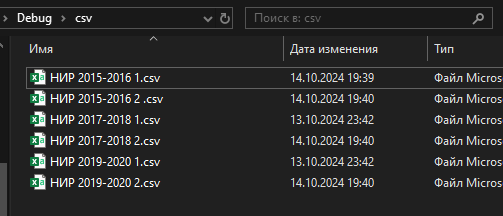


Рисунок 28 – Файлы, из которых берутся данные для диаграммы «Количество разделов, исполняемых сольно»

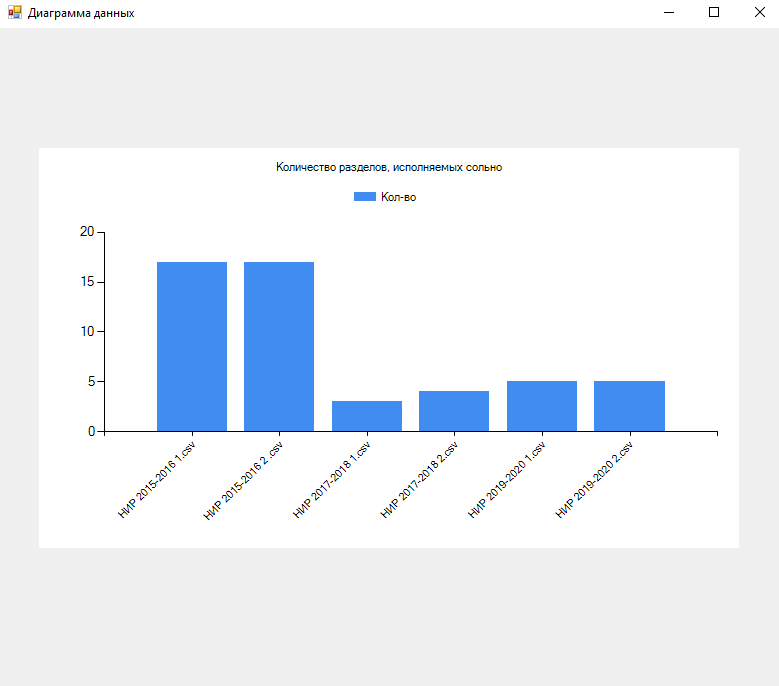


Рисунок 29 – Диаграмма «Количество разделов, исполняемых сольно»

**Листинг кода, реализующий построение диаграмм «Количество разделов, исполняемых сольно»**

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

{

string folderPath = @"csv";

if (Directory.Exists(folderPath))

{

Dictionary<string, int> amount = new Dictionary<string, int>();

int ExecutersCounter(string executors)

{

int count = 1;

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

{

if (executors[i] == ',')

{

count++;

}

}

return count;

}

foreach (string filePath in Directory.GetFiles(folderPath, "\*.csv"))

{

List<string[]> data = csvReader(filePath);

string fileName = Path.GetFileName(filePath);

foreach (var row in data.Skip(1))

{

if (row.Length > 4)

{

string executors = row[row.Length - 3].Trim();

string section = row[0].Trim();

if (amount.ContainsKey(fileName))

{

int count = ExecutersCounter(executors);

if (count == 1)

{

amount[fileName] += count;

}

}

else

{

int count = ExecutersCounter(executors);

if (count == 1)

{

amount.Add(fileName, count);

}

}

}

else if (row.Length == 4)

{

string executors = row[row.Length - 1].Trim();

if (amount.ContainsKey(fileName))

{

int count = ExecutersCounter(executors);

if (count == 1)

{

amount[fileName] += count;

}

}

else

{

int count = ExecutersCounter(executors);

if (count == 1)

{

amount.Add(fileName, count);

}

}

}

}

}

diagramBuilder(amount, "Количество разделов, исполняемых сольно");

}

else

{

MessageBox.Show("Сначала сконвертируйте файл в формат CSV.", "Предупреждение");

}

## }

## **4.9.4** **Показывать столбчатую диаграмму количеств упоминаний ключевых слов, разделённых ";", для выбранного отчётного периода**

В зависимости от промежутка построения диаграммы пользователю необходимо сконвертировать нужное количество файлов WORD в формат CSV, после чего вписать ключевые слова для поиска, разделяя их знаком «;» без пробелов. Покажем диаграмму «Количество упоминаний ключевых слов» на рисунке 31.

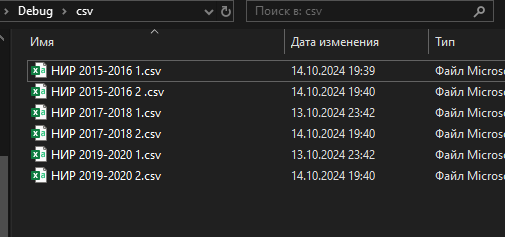


Рисунок 30 – Файлы, из которых берутся данные для диаграммы «Количество упоминаний ключевых слов»

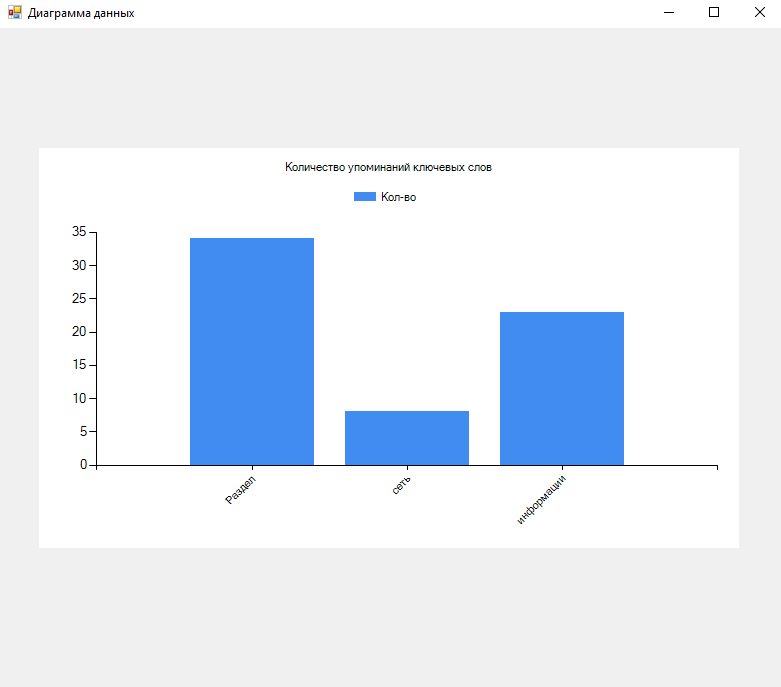
****

Рисунок 31 – Диаграмма «Количество упоминаний ключевых слов» с ключевыми словами «Раздел;сеть;информации»

**Листинг кода, реализующий построение диаграмм «Количество упоминаний ключевых слов»**

private void btnKeyWordsAmount\_Click(string keyWords)

{

string folderPath = @"csv";

int ExecutorsCounter(string executors, string keyWord)

{

String[] words = executors.Split(new char[] { ' ', ',', '.' });

int count = 0;

foreach (var word in words)

{

if (word.ToLower() == keyWord.ToLower())

{

count++;

}

}

return (count);

}

if (Directory.Exists(folderPath))

{

Dictionary<string, int> amount = new Dictionary<string, int>();

foreach (string filePath in Directory.GetFiles(folderPath, "\*.csv"))

{

List<string[]> data = csvReader(filePath);

foreach (var row in data.Skip(1))

{

if (row.Length > 0)

{

String[] keyWords1 = keyWords.Split(new char[] { ';' });

foreach (var keyWord in keyWords1)

{

for (int j = 0; j < row.Length; j++)

{

string executors = row[j].Trim();

if (amount.ContainsKey(keyWord))

{

amount[keyWord] += ExecutorsCounter(executors, keyWord);

}

else

{

amount.Add(keyWord, ExecutorsCounter(executors, keyWord));

}

}

}

}

}

}

diagramBuilder(amount, "Количество упоминаний ключевых слов");

}

else

{

MessageBox.Show("Сначала сконвертируйте файл в формат CSV.", "Предупреждение");

}

}

## **4.9.5** **Показывать столбчатую диаграмму количеств разделов внутри отчётных периодов за всё время**

В зависимости от промежутка построения диаграммы пользователю необходимо сконвертировать нужное количество файлов WORD в формат CSV. Покажем диаграмму «Количество разделов внутри отчётных периодов» на рисунке 33.

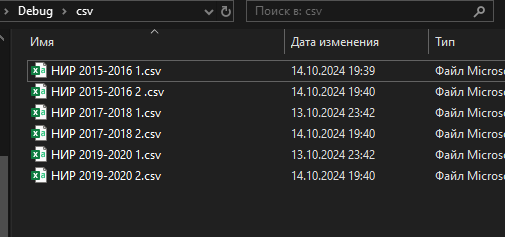


Рисунок 32 – Файлы, из которых берутся данные для диаграммы «Количество разделов внутри отчётных периодов»

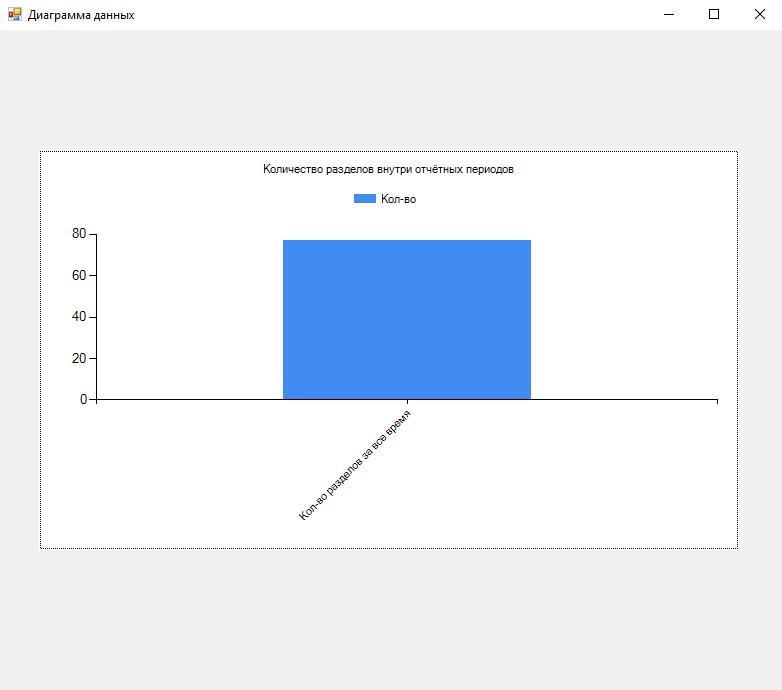
****

Рисунок 33 – Диаграмма «Количество разделов внутри отчётных периодов»

**Листинг кода, реализующий построение диаграмм «Количество разделов внутри отчётных периодов»**

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

{

string folderPath = @"csv";

if (Directory.Exists(folderPath))

{

Dictionary<string, int> amount = new Dictionary<string, int>();

foreach (string filePath in Directory.GetFiles(folderPath, "\*.csv"))

{

List<string[]> data = csvReader(filePath);

string fileName = "Кол-во разделов за все время" ;

foreach (var row in data.Skip(1))

{

if (row.Length >= 4)

{

if (amount.ContainsKey(fileName))

{

amount[fileName] += 1;

}

else

{

amount.Add(fileName, 1);

}

}

}

}

diagramBuilder(amount, "Количество разделов внутри отчётных периодов");

}

else

{

MessageBox.Show("Сначала сконвертируйте файл в формат CSV.", "Предупреждение");

}

## }

## **4.9.6** **Показывать столбчатую диаграмму количеств исполнителей без степеней и званий за всё время с разбивкой по отчётным периодам**

В зависимости от промежутка построения диаграммы пользователю необходимо сконвертировать нужное количество файлов WORD в формат CSV. Покажем диаграмму «Количество исполнителей без степеней и званий» на рисунке 35.

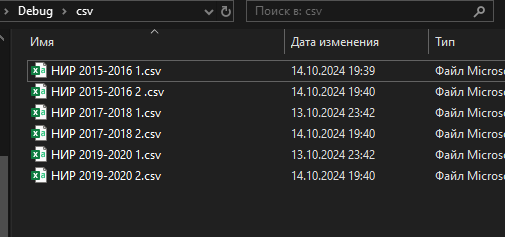


Рисунок 34 – Файлы, из которых берутся данные для диаграммы «Количество исполнителей без степеней и званий»

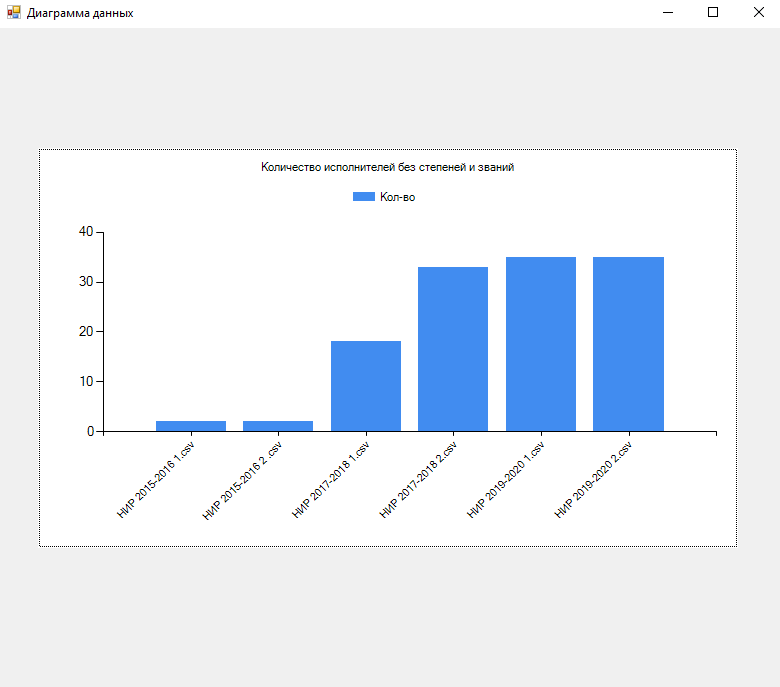
****

Рисунок 35 – Диаграмма «Количество исполнителей без степеней и званий»

**Листинг кода, реализующий построение диаграмм «Количество исполнителей без степеней и званий»**

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

{

string folderPath = @"csv";

void KeyWordsCounter(string executors, String[] keyWords, ref int count)

{

String[] words = executors.Split(new char[] { ' ', ',', '.' });

foreach (var keyWord in keyWords)

{

foreach (var word in words)

{

if (word.ToLower() == keyWord.ToLower())

{

count--;

}

}

}

}

void ExecutorsCounter(string executors, ref int count)

{

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

{

if (executors[i] == ',')

{

count++;

}

}

}

if (Directory.Exists(folderPath))

{

Dictionary<string, int> amount = new Dictionary<string, int>();

foreach (string filePath in Directory.GetFiles(folderPath, "\*.csv"))

{

List<string[]> data = csvReader(filePath);

string fileName = Path.GetFileName(filePath);

string keyWords = "Доц;Проф;Ст;Асс";

String[] keyWords1 = keyWords.Split(new char[] { ';' });

foreach (var row in data.Skip(1))

{

int count = 1;

if (row.Length > 4)

{

string executors = row[row.Length - 3].Trim();

KeyWordsCounter(executors, keyWords1, ref count);

ExecutorsCounter(executors, ref count);

if (amount.ContainsKey(fileName) & count > 0)

{

amount[fileName] += count;

}

else

{

if (count > 0)

{

amount.Add(fileName, count);

}

}

}

else if (row.Length == 4)

{

string executors = row[row.Length - 1].Trim();

KeyWordsCounter(executors, keyWords1, ref count);

ExecutorsCounter(executors, ref count);

if (amount.ContainsKey(fileName) & count > 0)

{

amount[fileName] += count;

}

else

{

if (count > 0)

{

amount.Add(fileName, count);

}

}

}

}

}

diagramBuilder(amount, "Количество исполнителей без степеней и званий");

}

else

{

MessageBox.Show("Сначала сконвертируйте файл в формат CSV.", "Предупреждение");

}

}

## **Инструкция пользователя**

Запуская программу, пользователь попадает в главное меню, в котором находится ComboBox для выбора печатной формы и 5 кнопок, выполняющих действия над файлом.

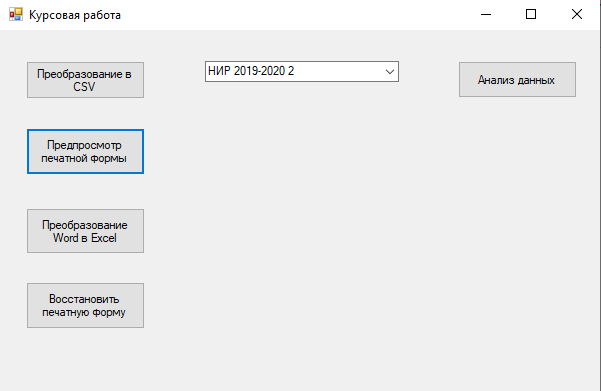


Рисунок 36 – Главное меню приложения

Кнопка «Преобразование в CSV» преобразует печатную форму, выбранную в ComboBox в формат CSV. Пользователь не сможет сделать другие действия с печатной формой, пока не преобразует ее в формат CSV. Получившийся файл формата CSV сохраняется в папке «csv». Все печатные формы изначально хранятся в папке «печатные формы».

Кнопка «Предпросмотр печатной формы» открывает дочернюю форму с элементом DataGridView, который показывает в виде таблицы данные, хранящиеся в печатной форме.

Кнопка «Преобразование Word в Excel» преобразует данные из печатной формы в формат Excel и сохраняет получившийся файл в папке «excel».

Кнопка «Восстановить печатную форму» восстанавливает формат WORD у печатной формы, после всех ее преобразований и сохраняет получившийся файл в папке «word».

Кнопка «Анализ данных» открывает дочернюю форму «Анализ данных» с 6 кнопками для выбора нужной диаграммы, TextBox для ввода параметра построения диаграммы и кнопку «Справочная информация».

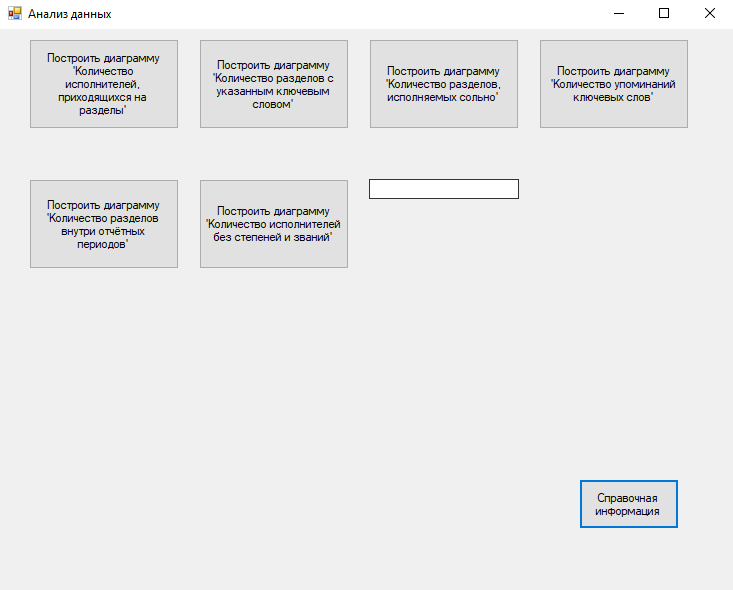


Рисунок 37 – Дочерняя форма «Анализ данных»

Для построения всех диаграмм пользователю необходимо в зависимости от промежутка построения диаграммы сконвертировать нужное количество файлов WORD в формат CSV. Для построения диаграммы «Количество разделов с указанным ключевым словом» пользователю также необходимо ввести ключевое слово в поле TextBox. Для построения диаграммы «Количество упоминаний ключевых слов» пользователю также необходимо ввести ключевые слова в поле TextBox, разделяя их знаком «;» без пробелов.

При нажатии на кнопку «Справочная информация» открывается дочерняя форма со справкой для пользователя по построению диаграмм.

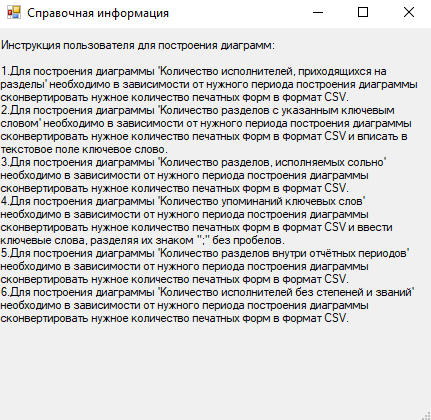


Рисунок 38 – Дочерняя форма «Справочная информация»

## **Заключение**

В рамках данной курсовой работы была разработана программа, предназначенная для автоматизации процесса обработки данных из печатных форм. Разработанное приложение позволяет пользователю просматривать файлы .docx в экранной форме, конвертировать их в форматы .csv и .xlsx, а также анализировать данные с помощью диаграмм.

При разработке приложения были реализованы следующие возможности:

* Импорт файлов .docx: приложение позволяет загружать файлы .docx и отображать их содержимое в удобном формате;
* Конвертация в .csv и .xlsx: приложение предоставляет возможность конвертировать файлы .docx в форматы .csv и .xlsx для удобного анализа и обработки данных;
* Создание структуры печатной формы: приложение позволяет воссоздать структуру печатной формы, что упрощает процесс анализа данных и позволяет создавать более точные отчёты;
* Визуализация данных с помощью диаграмм: приложение позволяет создавать различные типы диаграмм для наглядной визуализации данных.

**Полный листинг разработанной программы**

using System;

using System.Collections.Generic;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using DocumentFormat.OpenXml.Wordprocessing;

using DocumentFormat.OpenXml;

using DocumentFormat.OpenXml.Packaging;

using DocumentFormat.OpenXml.Spreadsheet;

using System.Windows.Forms.DataVisualization.Charting;

namespace Kursovaya

{

public partial class frmMain : Form

{

DataGridView dataGridView;

private string csvFilePath;

private Form frmAnalysis;

private Form frmInfo;

private Form frmDiagram;

public frmMain()

{

InitializeComponent();

string printedForms = Path.Combine(Environment.CurrentDirectory, "печатные формы");

if (Directory.Exists(printedForms))

{

string[] filesWord = Directory.GetFiles(printedForms, "\*.docx");

foreach (string file in filesWord)

{

string nameFile = Path.GetFileNameWithoutExtension(file);

cBWord.Items.Add(nameFile);

}

}

else

{

MessageBox.Show("Папка 'печатные формы' не найдена.", "Ошибка");

}

}

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

{

if (cBWord.SelectedItem != null)

{

string selectedFile = Path.Combine(Environment.CurrentDirectory, "печатные формы", $"{cBWord.SelectedItem}.docx");

if (File.Exists(selectedFile))

{

ConvertWordToCSV(selectedFile);

MessageBox.Show($"Файл успешно сохранен в папке csv как {cBWord.SelectedItem}.csv", "Успех");

}

else

{

MessageBox.Show("Выбранный файл Word не найден.", "Ошибка");

}

}

else

{

MessageBox.Show("Выберите файл Word для преобразования в формат csv.", "Предупреждение");

}

}

private void ConvertWordToCSV(string selectedFile)

{

try

{

using (WordprocessingDocument wordDocument = WordprocessingDocument.Open(selectedFile, true))

{

MainDocumentPart mainPart = wordDocument.MainDocumentPart;

var tables = mainPart.Document.Descendants<DocumentFormat.OpenXml.Wordprocessing.Table>();

StringBuilder csvBuilder = new StringBuilder();

foreach (var table in tables)

{

csvBuilder.AppendLine(string.Join("|", table.Descendants<DocumentFormat.OpenXml.Wordprocessing.TableRow>().First().Descendants<DocumentFormat.OpenXml.Wordprocessing.TableCell>().Select(c => c.InnerText.Trim())));

foreach (var row in table.Descendants<DocumentFormat.OpenXml.Wordprocessing.TableRow>().Skip(1))

{

csvBuilder.AppendLine(string.Join("|", row.Descendants<DocumentFormat.OpenXml.Wordprocessing.TableCell>().Select(c => c.InnerText.Trim())));

}

csvBuilder.AppendLine();

}

string folderCSV = Path.Combine(Environment.CurrentDirectory, "csv");

if (!Directory.Exists(folderCSV))

{

Directory.CreateDirectory(folderCSV);

}

csvFilePath = Path.Combine(folderCSV, $"{cBWord.SelectedItem}.csv");

File.WriteAllText(csvFilePath, csvBuilder.ToString());

}

}

catch (Exception ex)

{

MessageBox.Show($"Произошла ошибка при преобразовании файла: {ex.Message}", "Ошибка");

}

}

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

{

if (cBWord.SelectedItem != null)

{

if (!string.IsNullOrEmpty(csvFilePath))

{

Form frmPreview = new Form();

dataGridView = new DataGridView();

dataGridView.Dock = DockStyle.Fill;

frmPreview.Controls.Add(dataGridView);

frmPreview.Show();

LoadCSVDataToDataGridView(csvFilePath);

}

else

{

MessageBox.Show("Сначала необходимо преобразовать файл Word в CSV.", "Предупреждение");

}

}

else

{

MessageBox.Show("Выберите файл Word для предпросмотра.", "Предупреждение");

}

}

private void LoadCSVDataToDataGridView(string csvFilePath)

{

if (File.Exists(csvFilePath))

{

try

{

System.Data.DataTable dataTable = new System.Data.DataTable();

string[] lines = File.ReadAllLines(csvFilePath);

if (lines.Length > 0)

{

string[] headers = lines[0].Split('|');

foreach (var header in headers)

{

dataTable.Columns.Add(header);

}

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

{

string[] values = lines[i].Split('|');

DataRow dataRow = dataTable.NewRow();

for (int j = 0; j < values.Length && j < dataTable.Columns.Count; j++)

{

dataRow[j] = values[j].Trim();

}

dataTable.Rows.Add(dataRow);

}

dataGridView.DataSource = dataTable;

dataGridView.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;

}

else

{

MessageBox.Show("CSV файл пустой.", "Предупреждение");

}

}

catch (Exception ex)

{

MessageBox.Show($"Произошла ошибка при загрузке данных: {ex.Message}", "Ошибка");

}

}

else

{

MessageBox.Show($"CSV файл не найден: {csvFilePath}", "Ошибка");

}

}

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

{

if (File.Exists(csvFilePath))

{

string folderExcel = Path.Combine(Environment.CurrentDirectory, "excel");

if (!Directory.Exists(folderExcel))

{

Directory.CreateDirectory(folderExcel);

}

string excelFilePath = Path.Combine(folderExcel, $"{cBWord.SelectedItem}.xlsx");

System.Data.DataTable dataTable = new System.Data.DataTable();

using (StreamReader reader = new StreamReader(csvFilePath))

{

string headersLine = reader.ReadLine();

string[] headers = headersLine.Split('|');

foreach (string header in headers)

{

dataTable.Columns.Add(header);

}

dataTable.Rows.Add(headers);

while (!reader.EndOfStream)

{

string[] values = reader.ReadLine().Split('|');

if (values.Length > dataTable.Columns.Count)

{

for (int i = dataTable.Columns.Count; i < values.Length; i++)

{

dataTable.Columns.Add($"Столбец{i + 1}");

}

}

dataTable.Rows.Add(values);

}

}

using (var document = SpreadsheetDocument.Create(excelFilePath, SpreadsheetDocumentType.Workbook))

{

var workbookPart = document.AddWorkbookPart();

workbookPart.Workbook = new Workbook();

var worksheetPart = workbookPart.AddNewPart<WorksheetPart>();

worksheetPart.Worksheet = new Worksheet();

var sheets = workbookPart.Workbook.AppendChild(new Sheets());

sheets.AppendChild(new Sheet()

{

Id = workbookPart.GetIdOfPart(worksheetPart),

SheetId = 1,

Name = "Лист1"

});

var sheetData = worksheetPart.Worksheet.AppendChild(new SheetData());

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

{

var row = sheetData.AppendChild(new Row());

for (int j = 0; j < dataTable.Columns.Count; j++)

{

var cell = row.AppendChild(new Cell());

cell.DataType = new EnumValue<CellValues>(CellValues.String);

cell.AppendChild(new CellValue($"{dataTable.Rows[i][j]}"));

}

}

}

MessageBox.Show($"Файл успешно сохранен в папке excel как {cBWord.SelectedItem}.xlsx", "Успех");

}

else

{

MessageBox.Show("Перед преобразованием в формат xlsx необходимо сгенерировать файл csv.", "Предупреждение");

}

}

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

{

if (!string.IsNullOrEmpty(csvFilePath))

{

try

{

string[] lines = File.ReadAllLines(csvFilePath);

string wordFilePath = Path.Combine(Directory.GetCurrentDirectory(), "word", $"{cBWord.SelectedItem}.docx");

using (WordprocessingDocument wordDocument = WordprocessingDocument.Create(wordFilePath, WordprocessingDocumentType.Document))

{

MainDocumentPart mainPart = wordDocument.AddMainDocumentPart();

mainPart.Document = new Document();

Body body = new Body();

DocumentFormat.OpenXml.Wordprocessing.Table table = new DocumentFormat.OpenXml.Wordprocessing.Table();

table.AppendChild(new TableProperties(

new TableBorders(

new DocumentFormat.OpenXml.Wordprocessing.TopBorder() { Val = new EnumValue<BorderValues>(BorderValues.Single), Size = 6 },

new DocumentFormat.OpenXml.Wordprocessing.BottomBorder() { Val = new EnumValue<BorderValues>(BorderValues.Single), Size = 6 },

new DocumentFormat.OpenXml.Wordprocessing.LeftBorder() { Val = new EnumValue<BorderValues>(BorderValues.Single), Size = 6 },

new DocumentFormat.OpenXml.Wordprocessing.RightBorder() { Val = new EnumValue<BorderValues>(BorderValues.Single), Size = 6 },

new InsideVerticalBorder() { Val = new EnumValue<BorderValues>(BorderValues.Single), Size = 6 }

)

));

foreach (var line in lines)

{

if (!string.IsNullOrEmpty(line))

{

TableRow row = new TableRow();

string[] values = line.Split('|'); int currentColumnsCount = 0;

foreach (var value in values)

{

TableCell cell = new TableCell(new Paragraph(new DocumentFormat.OpenXml.Wordprocessing.Run(new DocumentFormat.OpenXml.Wordprocessing.Text(value))));

row.Append(cell);

currentColumnsCount++;

}

while (currentColumnsCount < row.ChildElements.Count)

{

row.Append(new TableCell(new Paragraph(new DocumentFormat.OpenXml.Wordprocessing.Run())));

currentColumnsCount++;

}

table.Append(row);

}

}

body.Append(table);

mainPart.Document.Append(body);

}

MessageBox.Show($"Файл успешно сохранен в папке word как {cBWord.SelectedItem}.docx", "Успех");

}

catch (Exception ex)

{

MessageBox.Show($"Произошла ошибка при восстановлении файла: {ex.Message}", "Ошибка");

}

}

else

{

MessageBox.Show("Сначала преобразуйте файл Word в CSV.", "Предупреждение");

}

}

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

{

if (!string.IsNullOrEmpty(csvFilePath))

{

frmAnalysis = new Form();

frmAnalysis.Text = "Анализ данных";

frmAnalysis.Size = new Size(750, 600);

Button btnExecutorsAmount = new Button();

btnExecutorsAmount.Text = "Построить диаграмму 'Количество исполнителей, приходящихся на разделы'";

btnExecutorsAmount.Size = new Size(150, 90);

btnExecutorsAmount.Location = new Point(30, 10);

btnExecutorsAmount.Click += new EventHandler(btnExecutorsAmount\_Click);

frmAnalysis.Controls.Add(btnExecutorsAmount);

TextBox txtKeyWord = new TextBox();

txtKeyWord.Location = new Point(370, 150);

txtKeyWord.Size = new Size(150, 100);

frmAnalysis.Controls.Add(txtKeyWord);

Button btnKeyWordSections = new Button();

btnKeyWordSections.Text = "Построить диаграмму 'Количество разделов с указанным ключевым словом'";

btnKeyWordSections.Size = new Size(150, 90);

btnKeyWordSections.Location = new Point(200, 10);

btnKeyWordSections.Click += (s, ev) => { btnKeyWordSections\_Click(txtKeyWord.Text); };

frmAnalysis.Controls.Add(btnKeyWordSections);

Button btnSoloSections = new Button();

btnSoloSections.Text = "Построить диаграмму 'Количество разделов, исполняемых сольно'";

btnSoloSections.Size = new Size(150, 90);

btnSoloSections.Location = new Point(370, 10);

btnSoloSections.Click += new EventHandler(btnSoloSections\_Click);

frmAnalysis.Controls.Add(btnSoloSections);

Button btnKeyWordsAmount = new Button();

btnKeyWordsAmount.Text = "Построить диаграмму 'Количество упоминаний ключевых слов'";

btnKeyWordsAmount.Size = new Size(150, 90);

btnKeyWordsAmount.Location = new Point(540, 10);

btnKeyWordsAmount.Click += (s, ev) => { btnKeyWordsAmount\_Click(txtKeyWord.Text); };

frmAnalysis.Controls.Add(btnKeyWordsAmount);

Button btnSectionsAmount = new Button();

btnSectionsAmount.Text = "Построить диаграмму 'Количество разделов внутри отчётных периодов'";

btnSectionsAmount.Size = new Size(150, 90);

btnSectionsAmount.Location = new Point(30, 150);

btnSectionsAmount.Click += new EventHandler(btnSectionsAmount\_Click);

frmAnalysis.Controls.Add(btnSectionsAmount);

Button btnExecutorsUnranked = new Button();

btnExecutorsUnranked.Text = "Построить диаграмму 'Количество исполнителей без степеней и званий'";

btnExecutorsUnranked.Size = new Size(150, 90);

btnExecutorsUnranked.Location = new Point(200, 150);

btnExecutorsUnranked.Click += new EventHandler(btnExecutorsUnranked\_Click);

frmAnalysis.Controls.Add(btnExecutorsUnranked);

Button btnInfo = new Button();

btnInfo.Text = "Справочная информация";

btnInfo.Size = new Size(100, 50);

btnInfo.Location = new Point(580, 450);

btnInfo.Click += new EventHandler(btnInfo\_Click);

frmAnalysis.Controls.Add(btnInfo);

frmAnalysis.Show();

}

else

{

MessageBox.Show("Сначала сконвертируйте файл/файлы в формат CSV.", "Предупреждение");

}

}

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

{

frmInfo = new Form();

frmInfo.Text = "Справочная информация";

frmInfo.Size = new Size(450, 410);

Label lblInfo = new Label();

lblInfo.AutoSize = false;

lblInfo.Size = new Size(frmInfo.ClientRectangle.Width, frmInfo.ClientRectangle.Height - 40);

lblInfo.Location = new Point(0, 10);

lblInfo.Anchor = AnchorStyles.Top | AnchorStyles.Bottom | AnchorStyles.Left | AnchorStyles.Right;

lblInfo.Text = File.ReadAllText("info.txt");

frmInfo.SizeChanged += (s, ev) =>

{

lblInfo.Size = new Size(frmInfo.ClientRectangle.Width, frmInfo.ClientRectangle.Height - 40);

};

frmInfo.Controls.Add(lblInfo);

frmInfo.ShowDialog();

}

private void diagramBuilder(Dictionary<string, int> amount, string title)

{

frmDiagram = new Form();

frmDiagram.Text = "Диаграмма данных";

frmDiagram.Size = new Size(800, 700);

Chart chart = new Chart();

chart.Size = new Size(700, 400);

chart.Parent = frmDiagram;

chart.Dock = DockStyle.None;

chart.Location = new Point(40, 120);

chart.Titles.Add(title);

chart.ChartAreas.Add("ChartArea1");

chart.Series.Add("Кол-во");

chart.Series["Кол-во"].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Column;

chart.Series["Кол-во"].XValueType = System.Windows.Forms.DataVisualization.Charting.ChartValueType.String;

chart.Series["Кол-во"].YValueType = System.Windows.Forms.DataVisualization.Charting.ChartValueType.Int32;

foreach (var item in amount)

{

chart.Series["Кол-во"].Points.AddXY(item.Key, item.Value);

}

chart.ChartAreas["ChartArea1"].AxisX.MajorGrid.Enabled = false;

chart.ChartAreas["ChartArea1"].AxisY.MajorGrid.Enabled = false;

chart.ChartAreas["ChartArea1"].AxisX.LabelStyle.Angle = -45;

chart.ChartAreas["ChartArea1"].AxisY.LabelStyle.Format = "{0:N0}";

chart.Legends.Add("Legend1");

chart.Legends["Legend1"].Docking = System.Windows.Forms.DataVisualization.Charting.Docking.Top;

chart.Legends["Legend1"].Alignment = StringAlignment.Center;

frmDiagram.Controls.Add(chart);

frmDiagram.Show();

}

private List<string[]> csvReader(string filePath)

{

List<string[]> data = new List<string[]>();

using (var reader = new StreamReader(filePath))

{

string line;

while ((line = reader.ReadLine()) != null)

{

string[] parts = line.Split('|').Where(s => !string.IsNullOrWhiteSpace(s)).ToArray();

data.Add(parts);

}

}

return (data);

}

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

{

string folderPath = @"csv";

if (Directory.Exists(folderPath))

{

Dictionary<string, int> amount = new Dictionary<string, int>();

foreach (string filePath in Directory.GetFiles(folderPath, "\*.csv"))

{

List<string[]> data = csvReader(filePath);

int ExecutorsCounter(string executors)

{

int count = 1;

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

{

if (executors[i] == ',')

{

count++;

}

}

return (count);

}

foreach (var row in data.Skip(1))

{

if (row.Length > 4)

{

string executors = row[row.Length - 3].Trim();

string section = row[0].Trim();

if (amount.ContainsKey(section))

{

int count = ExecutorsCounter(executors);

amount[section] += count;

}

else

{

int count = ExecutorsCounter(executors);

amount.Add(section, count);

}

}

else if (row.Length == 4)

{

string executors = row[row.Length - 1].Trim();

string section = row[0].Trim();

if (amount.ContainsKey(section))

{

int count = ExecutorsCounter(executors);

amount[section] += count;

}

else

{

int count = ExecutorsCounter(executors);

amount.Add(section, count);

}

}

}

}

diagramBuilder(amount, "Количество исполнителей, приходящихся на разделы");

}

else

{

MessageBox.Show("Сначала сконвертируйте файл в формат CSV.", "Предупреждение");

}

}

private void btnKeyWordSections\_Click(string keyword)

{

string folderPath = @"csv";

if (Directory.Exists(folderPath))

{

Dictionary<string, int> amount = new Dictionary<string, int>();

foreach (string filePath in Directory.GetFiles(folderPath, "\*.csv"))

{

List<string[]> data = csvReader(filePath);

string fileName = Path.GetFileName(filePath);

foreach (var row in data.Skip(1))

{

if (row.Length > 0)

{

string executors = row[0].Trim();

if (amount.ContainsKey(fileName))

{

int count = 0;

if (executors.Contains(keyword))

{

count++;

}

amount[fileName] += count;

}

else

{

int count = 0;

if (executors.Contains(keyword))

{

count++;

}

amount.Add(fileName, count);

}

}

}

}

diagramBuilder(amount, "Количество разделов с указанным ключевым словом");

}

else

{

MessageBox.Show("Сначала сконвертируйте файл в формат CSV.", "Предупреждение");

}

}

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

{

string folderPath = @"csv";

if (Directory.Exists(folderPath))

{

Dictionary<string, int> amount = new Dictionary<string, int>();

int ExecutersCounter(string executors)

{

int count = 1;

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

{

if (executors[i] == ',')

{

count++;

}

}

return count;

}

foreach (string filePath in Directory.GetFiles(folderPath, "\*.csv"))

{

List<string[]> data = csvReader(filePath);

string fileName = Path.GetFileName(filePath);

foreach (var row in data.Skip(1))

{

if (row.Length > 4)

{

string executors = row[row.Length - 3].Trim();

string section = row[0].Trim();

if (amount.ContainsKey(fileName))

{

int count = ExecutersCounter(executors);

if (count == 1)

{

amount[fileName] += count;

}

}

else

{

int count = ExecutersCounter(executors);

if (count == 1)

{

amount.Add(fileName, count);

}

}

}

else if (row.Length == 4)

{

string executors = row[row.Length - 1].Trim();

if (amount.ContainsKey(fileName))

{

int count = ExecutersCounter(executors);

if (count == 1)

{

amount[fileName] += count;

}

}

else

{

int count = ExecutersCounter(executors);

if (count == 1)

{

amount.Add(fileName, count);

}

}

}

}

}

diagramBuilder(amount, "Количество разделов, исполняемых сольно");

}

else

{

MessageBox.Show("Сначала сконвертируйте файл в формат CSV.", "Предупреждение");

}

}

private void btnKeyWordsAmount\_Click(string keyWords)

{

string folderPath = @"csv";

int ExecutorsCounter(string executors, string keyWord)

{

String[] words = executors.Split(new char[] { ' ', ',', '.' });

int count = 0;

foreach (var word in words)

{

if (word.ToLower() == keyWord.ToLower())

{

count++;

}

}

return (count);

}

if (Directory.Exists(folderPath))

{

Dictionary<string, int> amount = new Dictionary<string, int>();

foreach (string filePath in Directory.GetFiles(folderPath, "\*.csv"))

{

List<string[]> data = csvReader(filePath);

foreach (var row in data.Skip(1))

{

if (row.Length > 0)

{

String[] keyWords1 = keyWords.Split(new char[] { ';' });

foreach (var keyWord in keyWords1)

{

for (int j = 0; j < row.Length; j++)

{

string executors = row[j].Trim();

if (amount.ContainsKey(keyWord))

{

amount[keyWord] += ExecutorsCounter(executors, keyWord);

}

else

{

amount.Add(keyWord, ExecutorsCounter(executors, keyWord));

}

}

}

}

}

}

diagramBuilder(amount, "Количество упоминаний ключевых слов");

}

else

{

MessageBox.Show("Сначала сконвертируйте файл в формат CSV.", "Предупреждение");

}

}

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

{

string folderPath = @"csv";

if (Directory.Exists(folderPath))

{

Dictionary<string, int> amount = new Dictionary<string, int>();

foreach (string filePath in Directory.GetFiles(folderPath, "\*.csv"))

{

List<string[]> data = csvReader(filePath);

string fileName = "Кол-во разделов за все время" ;

foreach (var row in data.Skip(1))

{

if (row.Length >= 4)

{

if (amount.ContainsKey(fileName))

{

amount[fileName] += 1;

}

else

{

amount.Add(fileName, 1);

}

}

}

}

diagramBuilder(amount, "Количество разделов внутри отчётных периодов");

}

else

{

MessageBox.Show("Сначала сконвертируйте файл в формат CSV.", "Предупреждение");

}

}

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

{

string folderPath = @"csv";

void KeyWordsCounter(string executors, String[] keyWords, ref int count)

{

String[] words = executors.Split(new char[] { ' ', ',', '.' });

foreach (var keyWord in keyWords)

{

foreach (var word in words)

{

if (word.ToLower() == keyWord.ToLower())

{

count--;

}

}

}

}

void ExecutorsCounter(string executors, ref int count)

{

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

{

if (executors[i] == ',')

{

count++;

}

}

}

if (Directory.Exists(folderPath))

{

Dictionary<string, int> amount = new Dictionary<string, int>();

foreach (string filePath in Directory.GetFiles(folderPath, "\*.csv"))

{

List<string[]> data = csvReader(filePath);

string fileName = Path.GetFileName(filePath);

string keyWords = "Доц;Проф;Ст;Асс";

String[] keyWords1 = keyWords.Split(new char[] { ';' });

foreach (var row in data.Skip(1))

{

int count = 1;

if (row.Length > 4)

{

string executors = row[row.Length - 3].Trim();

KeyWordsCounter(executors, keyWords1, ref count);

ExecutorsCounter(executors, ref count);

if (amount.ContainsKey(fileName) & count > 0)

{

amount[fileName] += count;

}

else

{

if (count > 0)

{

amount.Add(fileName, count);

}

}

}

else if (row.Length == 4)

{

string executors = row[row.Length - 1].Trim();

KeyWordsCounter(executors, keyWords1, ref count);

ExecutorsCounter(executors, ref count);

if (amount.ContainsKey(fileName) & count > 0)

{

amount[fileName] += count;

}

else

{

if (count > 0)

{

amount.Add(fileName, count);

}

}

}

}

}

diagramBuilder(amount, "Количество исполнителей без степеней и званий");

}

else

{

MessageBox.Show("Сначала сконвертируйте файл в формат CSV.", "Предупреждение");

}

}

}

}

## **Выводы**

Разработанное приложение решает актуальную проблему автоматизации обработки данных из печатных форм. Оно позволяет повысить эффективность работы с документами, сократить время на обработку данных и предоставить удобный инструмент для анализа данных.

В ходе выполнения курсовой работы нами была освоена работа с различными форматами данных, чтобы эффективно обрабатывать информацию из печатных форм. Мы научились работать с файлами формата CSV, изучили технологию Open XML для программирования взаимодействия с документами Word и Excel, а также познакомились с классом StringBuilder для удобной обработки строк. В итоге мы получили возможность не только извлекать данные из печатных форм, но и анализировать их, используя элемент Chart для создания диаграмм. Это позволило визуализировать информацию, делая ее более понятной и наглядной.

Таким образом, в ходе работы были закреплены ключевые технологии языка Visual C# и принципы разработки приложений в среде Windows Forms Application.

## **Библиографический список**

1. Лекционный материал по дисциплине «Методы программирования» Сафронов А. И., 2024.
2. Использование класса StringBuilder в .NET [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/standard/base-types/stringbuilder, свободный – (15.05.2024).
3. Откройте и добавьте текст в текстовый документ [Электронный ресурс]. – Режим доступа: https://video2.skills-academy.com/ru-ru/office/open-xml/word/how-to-open-and-add-text-to-a-word-processing-document?tabs=cs-0%2Ccs-1%2Ccs-2%2Ccs-3%2Ccs, свободный – (15.05.2024).
4. Создайте документ электронной таблицы, указав имя файла [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/office/open-xml/spreadsheet/how-to-create-a-spreadsheet-document-by-providing-a-file-name?tabs=cs-0%2Ccs-100%2Ccs-1%2Ccs, свободный – (15.05.2024).