Министерство транспорта Российской Федерации

Федеральное государственное автономное образовательное

учреждение высшего образования

«Российский университет транспорта»

(ФГАОУ ВО РУТ(МИИТ), РУТ (МИИТ)

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

Курсовая работа

по дисциплине: «Программирование и основы алгоритмизации»

на тему: «Электронный документооборот»

Выполнил: ст. гр. ТУУ-211

Воронов А.В.

Вариант №5

11.03.2025

(дата выполнения)

Проверил: к.т.н., доц. Сафронов А.И.

Москва – 2024 г.

# **Содержание**

[**Содержание** 2](#_Toc190540776)

[1. Введение 2](#_Toc190540777)

[**2. Цель работы** 4](#_Toc190540778)

[**3. Задачи работы** 4](#_Toc190540779)

[4. Содержательная часть работы 5](#_Toc190540780)

[4.1. Анализ печатной формы 5](#_Toc190540781)

[4.2. Описание предназначение печатной формы 5](#_Toc190540782)

[4.3. Разнообразие печатных форм и выявление отличий 5](#_Toc190540783)

[4.4. Унификация печатных форм для удобства обработки данных 6](#_Toc190540784)

[4.5. Разработка экранной формы для импорта сведений из печатной формы 6](#_Toc190540785)

[4.6. Разработка функций экспорта сведений из печатной формы в файл текстовый формата *CSV* 7](#_Toc190540786)

[4.6.1 Листинг 7](#_Toc190540787)

[4.6.2. Сеть Петри запрограммированного технологического процесса 9](#_Toc190540788)

[4.6.3. Блок-схема алгоритма 10](#_Toc190540789)

[4.7. Разработка функций импорта сведений из оригинального и изменённого текстовых файлов формата *CSV* с воссозданием *Word* 13](#_Toc190540790)

[4.7.1. Листинг 13](#_Toc190540791)

[4.7.3. Блок-схема алгоритма 18](#_Toc190540792)

[4.8. Разработка функций экспорта сведений из экранной формы в *Excel* 21](#_Toc190540793)

[4.8.1. Листинг 21](#_Toc190540794)

[4.8.2. Сеть Петри запрограммированного технологического процесса 28](#_Toc190540795)

[4.8.3. Блок-схема алгоритма 29](#_Toc190540796)

[4.9. Графический анализ данных, содержащихся в печатной форме 34](#_Toc190540797)

[5. Инструкция пользователя 38](#_Toc190540798)

[6. Заключение 39](#_Toc190540799)

[7. Выводы 40](#_Toc190540800)

[Библиографический список 41](#_Toc190540801)

1. Введение

Электронный документооборот (ЭДО) представляет собой комплексную систему, обеспечивающую полный цикл работы с документами в цифровом формате — от создания и редактирования до хранения и анализа. Электронный документооборот удобен: не нужно распечатывать документы, передавать их на подпись и отправлять по почте.

Под электронным документооборотом в сфере трудовых отношений (далее - электронный документооборот) понимается создание, подписание, использование и хранение работодателем, работником или лицом, поступающим на работу, документов, связанных с работой, оформленных в электронном виде без дублирования на бумажном носителе (далее - электронные документы), за исключением случаев, предусмотренных настоящей статьей и [статьями 22.2](https://www.consultant.ru/document/cons_doc_LAW_493279/f9c7ee89ea58b2910a7cfff23b04240bb094e871/#dst2947) и [22.3](https://www.consultant.ru/document/cons_doc_LAW_493279/00a8ce6ce116fc26361a0c9a23eebf37074acf6e/#dst2969) настоящего Кодекса.

Преимущества электронного документооборота:

* Ускорение бизнес-процессов. Сотрудники могут быстро отправлять и принимать документы, таким образом ускоряя поставки, выполнение работ, приёмку товара и др.
* Уменьшение затрат. Применение электронных документов позволяет сэкономить на печати, доставке и хранении
* Повышение контроля. Весь процесс легко проследить в личном кабинете сервиса ЭДО: проанализировать статусы, сроки, ошибки

Специализированное программное обеспечение, разработанное в рамках данной работы, обеспечивает преобразование данных печатных форм в удобные для анализа и обработки форматы (CSV, DOCX, XLS). Это упрощает совместную работу с документами, а инструменты визуализации позволяют оперативно оценивать и анализировать ключевые показатели.

# **2. Цель работы**

В среде *Microsoft Visual Studio* на языке *C#* разработать программное решение с графическим пользовательским интерфейсом, выполняющее функцию электронного документооборота. Проект подразумевает создание удобной экранной формы, реализацию экспорта данных из печатных форм в *CSV*-файлы, а также разработку механизма импорта из *CSV* с автоматическим восстановлением информации в документе *Word*. Дополнительно необходимо обеспечить экспорт данных из разработанного интерфейса в *Excel* и возможность графического анализа полученных сведений. В рамках работы предполагается освоить навыки программного взаимодействия с *Word* и *Excel*, а также методы визуального анализа данных.

# **3. Задачи работы**

1. Проанализировать печатные формы

2. Уметь переносить содержимое печатных форм в CSV

3. Уметь воссоздавать структуру печатной формы

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

5. Уметь конвертировать печатную форму Word - Excel

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

7. 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 на столбчатой диаграмме количество уникальных соавторов у каждого автора.

2. Показывать на столбчатой диаграмме для выбранного автора количество публикаций за каждый год.

3. Показывать на столбчатой диаграмме для выбранного автора количества работ различного объёма в печатных страницах (1 печатный лист (п.л.) равен 1 страницам).

4. Содержательная часть работы

4.1. Анализ печатной формы

Каждая печатная форма представляет собой *Word*-документ, который содержит в себе список трудов преподавателя. В данном списке содержится название работы, вид работы (печатный или рукописный), название издательства и журнала, год выпуска, номер авторского свидетельства, количество печатных страниц и фамилия соавторов. Все эти данные изменяются в зависимости от документа. Также, все печатные формы имеют информацию о авторе, которые могут изменяться, в зависимости от документа. После таблиц идут места для росписи преподавателя и заведующего кафедрой.

4.2. Описание предназначение печатной формы

Каждая печатная форма представляет собой **список научных трудов, составленный за отчетный период**. Её основное предназначение — систематизация и представление информации о публикациях, докладах и других научных работах автора в структурированном виде.

4.3. Разнообразие печатных форм и выявление отличий

Печатные формы могут различаться рассматриваемым преподавателем, тип, места и статус публикаций, формат представления, название работы и т.д.

4.4. Унификация печатных форм для удобства обработки данных

Унификация в данной работе не потребовалась.

4.5. Разработка экранной формы для импорта сведений из печатной формы

На Рисунке 1 показана главная экранная форма для импорта сведений из печатаной формы:

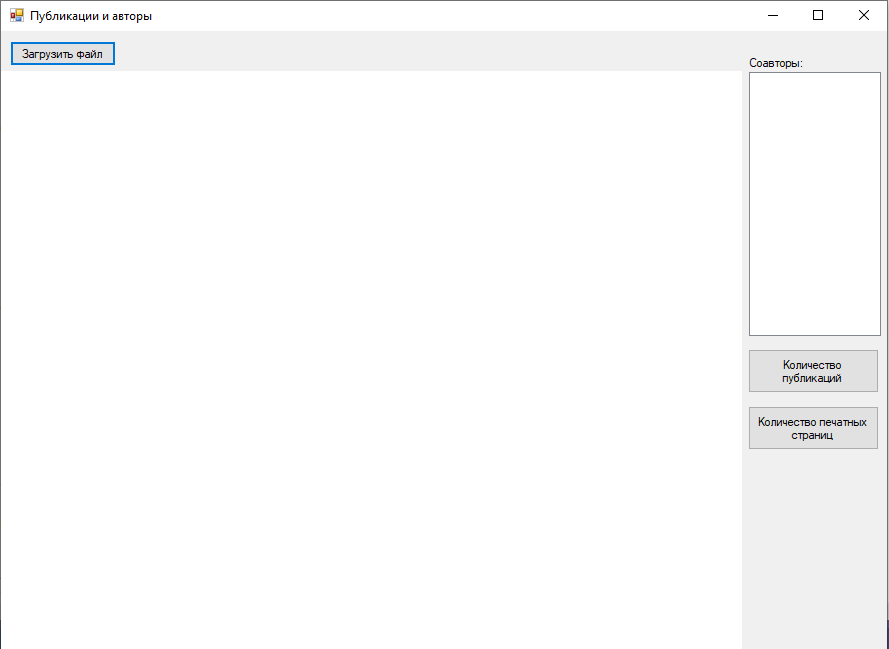


Рисунок 1 – Экранная форма для импорта сведений из печатной формы

Экранная форма для импорта сведений из печатной формы содержит в себе 3 кнопки. В левой части формы находиться кнопка «Загрузить файл», предназначенная для конвертации данных в разные типы: «Экспорт в *Word*-документ», запускающую процесс экспорта данных в *Word*-документ из *CSV* и «Экспорт в *Excel*-документ», запускающую процесс экспорта данных в *Excel*-документ из *CSV*.

В правой части формы находится кнопка «Количество публикаций», предназначенная для вывода количества публикаций в виде диаграмм за весь период, представленный в исходном файле. Также кнопка «Количество печатных страниц», предназначенная для вывода объёма печатных страниц и названия работ за весь период.

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

4.6.1 Листинг

public partial class kursovaya : Form

{

Year y = new Year();

Publish p = new Publish();

public kursovaya()

{

InitializeComponent();

this.StartPosition = FormStartPosition.CenterScreen;

}

private Dictionary<string, int> coauthorsCount = new Dictionary<string, int>();

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

{

// Открыть диалог для выбора файла Word

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Filter = "Word Documents|\*.doc;\*.docx";

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

{

// Вызываем метод конвертации

ConvertWordToExcel(openFileDialog.FileName);

}

}

private void ConvertWordToExcel(string wordFilePath)

{

Microsoft.Office.Interop.Word.Application wordApp = new Microsoft.Office.Interop.Word.Application();

wordApp.Visible = false; // Открываем Word в фоновом режиме

Microsoft.Office.Interop.Word.Document doc = wordApp.Documents.Open(wordFilePath, ReadOnly: true);

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

excelApp.Visible = false; // Открываем Excel в фоновом режиме

Microsoft.Office.Interop.Excel.Workbook workbook = excelApp.Workbooks.Add();

int row = 1; // Начальная строка для записи в Excel

// Перебираем все таблицы в документе

for (int tableIndex = 1; tableIndex <= doc.Tables.Count; tableIndex++)

{

Microsoft.Office.Interop.Word.Table table = doc.Tables[tableIndex];

// Проверяем, если в таблице более 4 строк (1 заголовок + 3 записи)

if (table.Rows.Count > 4)

{

// Обработка таблицы, если она удовлетворяет условию

row = WriteTableToExcel(table, workbook, row);

}

}

string excelFilePath = System.IO.Path.ChangeExtension(wordFilePath, ".xlsx");

// Сохранение Excel файла

workbook.SaveAs(excelFilePath);

workbook.Close(false); // Закрываем книгу без сохранения

excelApp.Quit();

doc.Close(false); // Закрываем документ без сохранения

wordApp.Quit();

// Освобождаем ресурсы

System.Runtime.InteropServices.Marshal.ReleaseComObject(workbook);

System.Runtime.InteropServices.Marshal.ReleaseComObject(excelApp);

System.Runtime.InteropServices.Marshal.ReleaseComObject(doc);

System.Runtime.InteropServices.Marshal.ReleaseComObject(wordApp);

MessageBox.Show($"Конвертация завершена! Файл сохранен как: {excelFilePath}", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

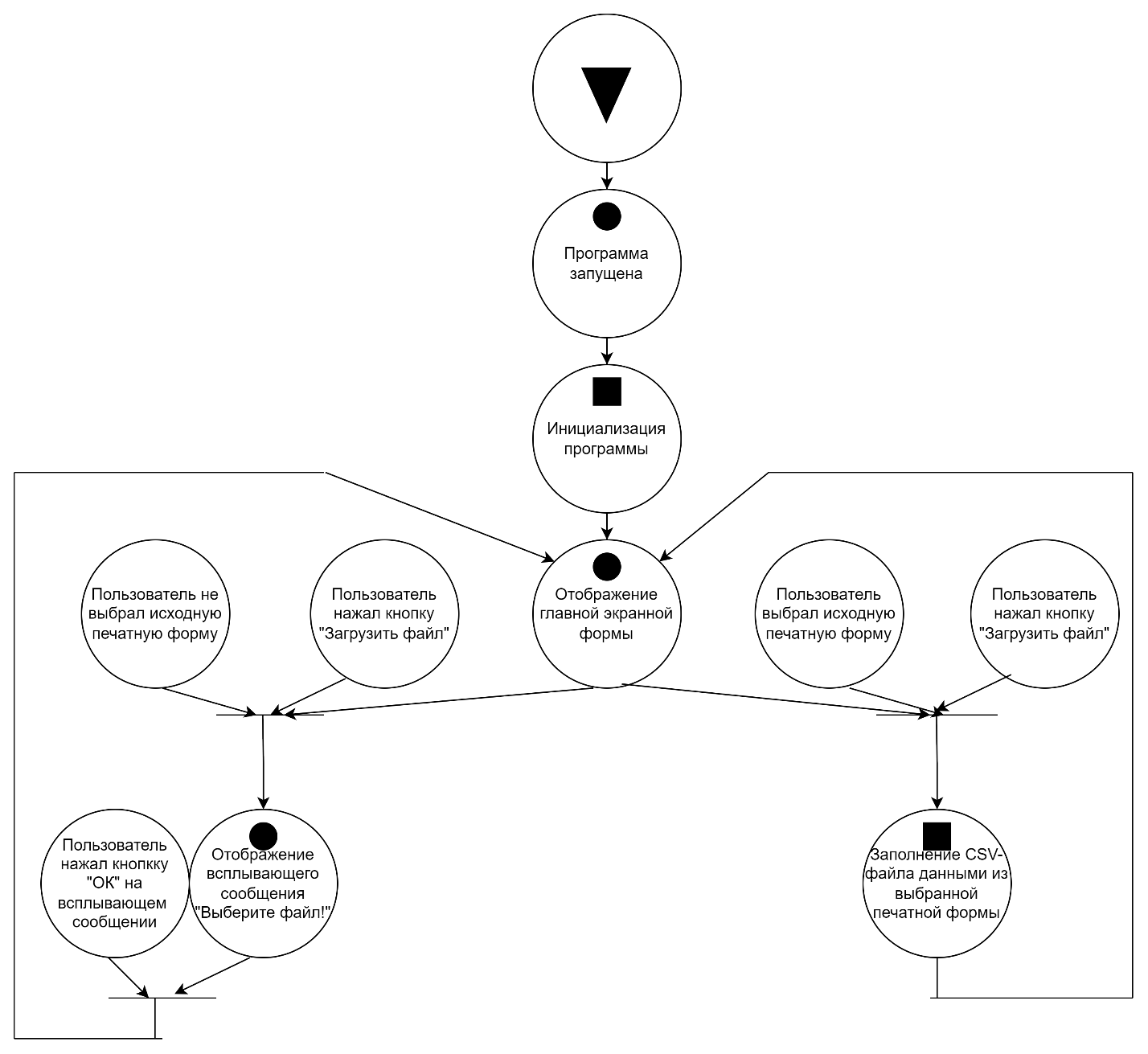
ConvertExcelToCsv(excelFilePath); //конвертация excel в csv

y.UpdatePublicationCounts(excelFilePath);

p.UpdateVolumeCounts(excelFilePath);

}

4.6.2. Сеть Петри запрограммированного технологического процесса

****

3CmxGraphModel%3E%3Croot%3E%3CmxCell%20id%3D%220%22%2F%3E%3CmxCell%20id%3D%221%22%20parent%3D%220%22%2F%3E%3CmxCell%20id%3D%222%22%20value%3D%22%22%20style%3D%22ellipse%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3Baspect%3Dfixed%3BfillColor%3D%23000000%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22390%22%20y%3D%22420%22%20width%3D%2220%22%20height%3D%2220%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3C%2Froot%3E%3C%2FmxGraphModel%3Eна

Рисунок 2 – Сеть Петри запрограммированного технологического процесса

4.6.3. Блок-схема алгоритма

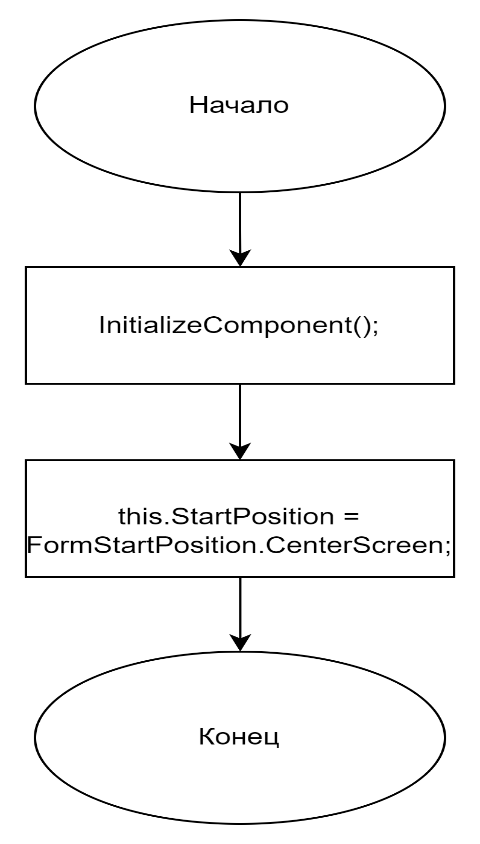
****

Рисунок 3 – Блок-схема метода kursovaya

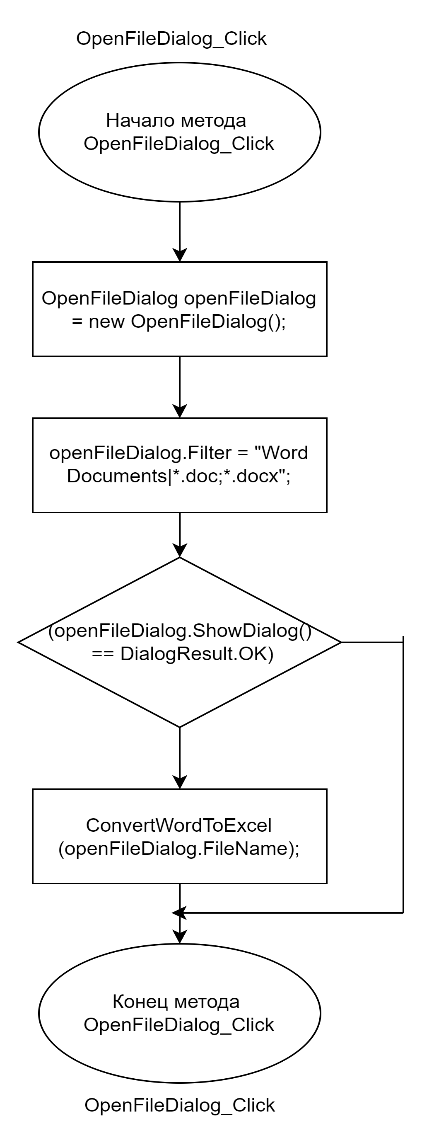
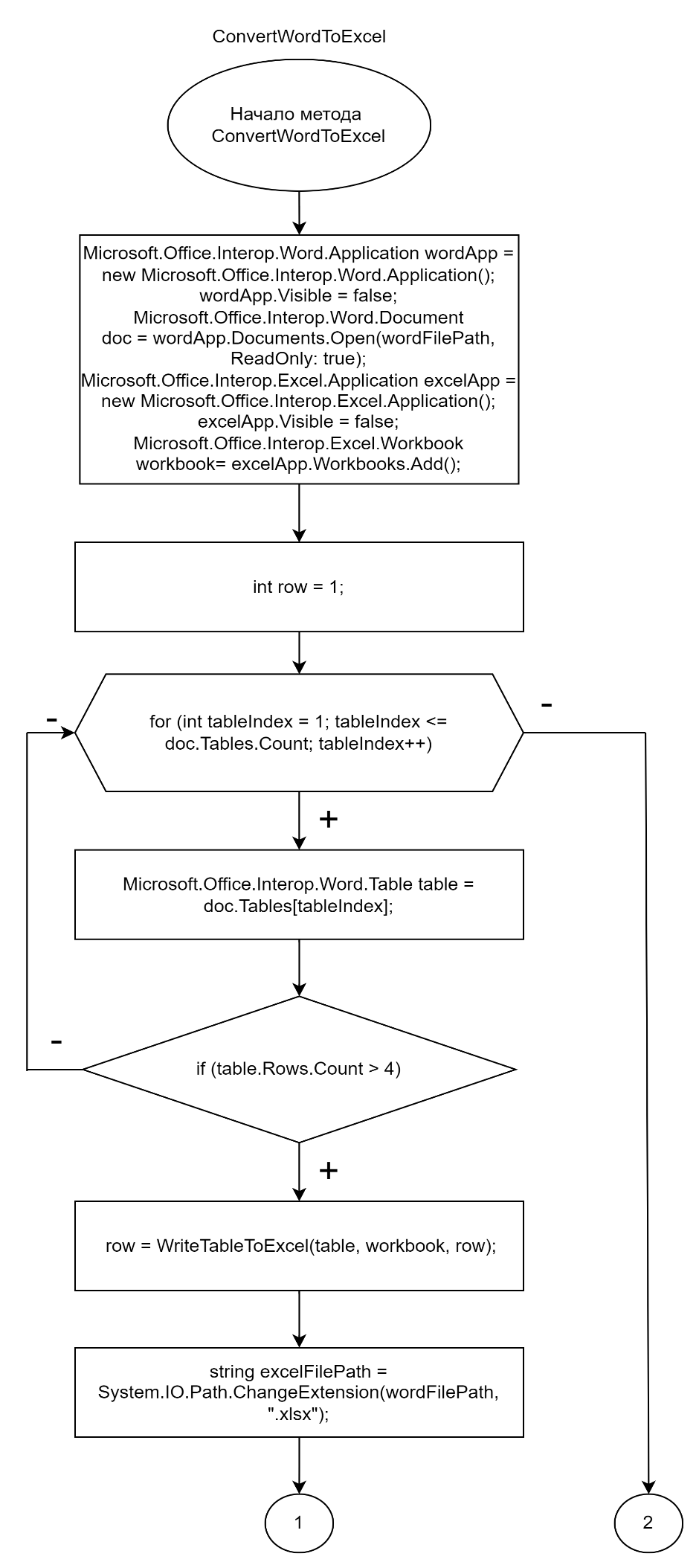


Рисунок 4 – Блок-схема метода OpenFileDialog\_Click



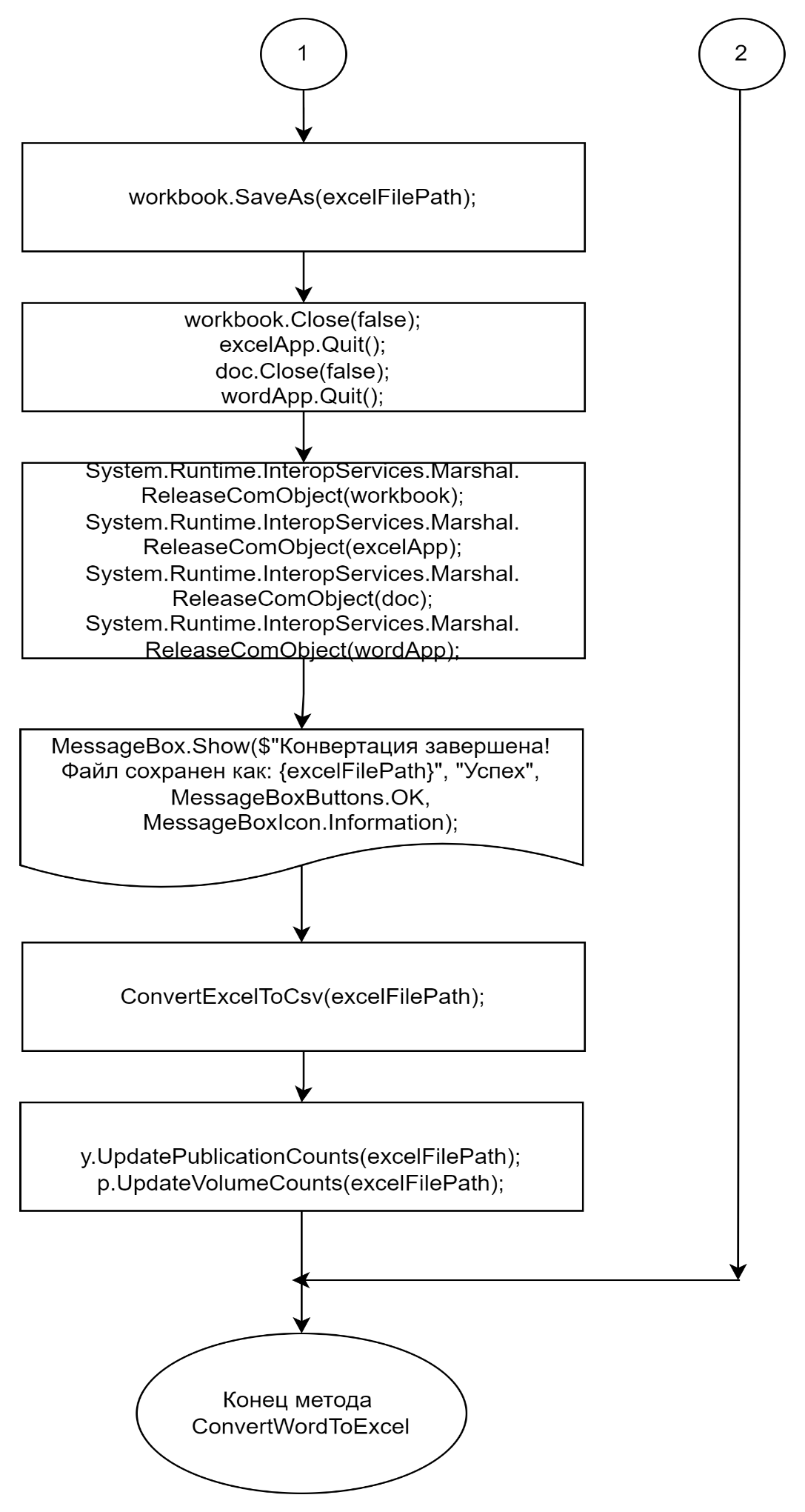


Рисунок 5 – Блок-схема метода ConvertWordToExcel

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

4.7.1. Листинг

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

{

Word.Document oDoc;

Word.Paragraph oPr;

Word.Application oWord = new Word.Application();

oDoc = oWord.Documents.Add();

oDoc.Paragraphs.Space1();

oPr = oDoc.Paragraphs.Add();

Word.Range range = oDoc.Content;

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

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

{

if (i >= lines.Length - 10)

{

oPr.Range.Font.Bold = 1;

SetParagraphText(oPr, lines[i], 12, "Times new roman", "Left");

}

else

{

if (i < 9)

{

oPr.Range.Font.Bold = 1;

}

else

{

oPr.Range.Font.Bold = 0;

}

if (i == 10 || i == 12)

{

SetParagraphText(oPr, lines[i], 10, "Times new roman", "Center");

}

else

{

SetParagraphText(oPr, lines[i], 12, "Times new roman", "Center");

}

}

if (lines[i].Contains("»") && i + 2 < lines.Length)

{

oPr.Range.Font.Bold = 0;

Word.Paragraph tablePr = oDoc.Paragraphs.Add();

Word.Range tableRange = tablePr.Range;

Word.Table table = oDoc.Tables.Add(tableRange, 1, 1);

table.Cell(1, 1).Range.Text = lines[i + 1];

Row row = table.Rows[1];

row.HeightRule = WdRowHeightRule.wdRowHeightExactly;

row.Height = 14;

table.Columns[1].Width = 510f;

table.Borders.Enable = 1;

table.Rows.Alignment = WdRowAlignment.wdAlignRowCenter;

i++;

continue;

}

if (lines[i].Contains("ФИО") && i + 2 < lines.Length)

{

oPr.Range.Font.Size = 12;

oPr.Alignment = Word.WdParagraphAlignment.wdAlignParagraphLeft;

Word.Paragraph tablePr = oDoc.Paragraphs.Add();

Word.Range tableRange = tablePr.Range;

Word.Table table = oDoc.Tables.Add(tableRange, 1, 1);

table.Cell(1, 1).Range.Text = lines[i + 1];

Row row = table.Rows[1];

row.HeightRule = WdRowHeightRule.wdRowHeightExactly;

row.Height = 14;

table.Columns[1].Width = 510f;

table.Borders.Enable = 1;

table.Rows.Alignment = WdRowAlignment.wdAlignRowCenter;

i++;

continue;

}

if (lines[i].Contains("1-й семестр") && i + 2 < lines.Length)

{

int startIndex = Array.FindIndex(lines, x => x.Contains("1-й семестр"));

int endIndex = Array.FindIndex(lines, x => x.Contains("2-й семестр"));

if (startIndex != -1 && endIndex != -1 && endIndex > startIndex)

{

int rowsCount = endIndex - startIndex - 1;

string[] slice = lines.Skip(startIndex + 1).Take(rowsCount).ToArray();

Word.Paragraph tablePr = oDoc.Paragraphs.Add();

Word.Range tableRange = tablePr.Range;

Word.Table table = oDoc.Tables.Add(tableRange, rowsCount, 6);

table.Rows.Alignment = WdRowAlignment.wdAlignRowCenter;

table.Columns[1].Width = 156f;

table.Columns[2].Width = 64f;

table.Columns[3].Width = 85f;

table.Columns[4].Width = 50f;

table.Columns[5].Width = 71f;

table.Columns[6].Width = 85f;

table.Borders.Enable = 1;

for (int j = 1; j <= table.Rows.Count; j++)

{

// Проверяем индекс перед обращением к slice

if (j - 1 < slice.Length)

{

string[] row = slice[j - 1].Split(';');

// Заполняем только существующие столбцы

for (int g = 1; g <= Math.Min(table.Columns.Count, row.Length); g++)

{

table.Cell(1, g).Range.Font.Bold = 1;

// Проверяем, что индекс не выходит за пределы массива

if (g - 1 < row.Length)

{

table.Cell(j, g).Range.Text = row[g - 1].Trim();

}

else

{

table.Cell(j, g).Range.Text = ""; // пустая ячейка если данных нет

}

}

}

}

i += rowsCount;

continue;

}

}

if (lines[i].Contains("2-й семестр") && i + 2 < lines.Length)

{

int startIndex = Array.FindIndex(lines, x => x.Contains("2-й семестр"));

int endIndex = Array.FindIndex(lines, x => x.Contains("Преподаватель")) - 2;

if (startIndex != -1 && endIndex != -1 && endIndex > startIndex)

{

int rowsCount = endIndex - startIndex - 1;

string[] slice = lines.Skip(startIndex + 1).Take(rowsCount).ToArray();

Word.Paragraph tablePr = oDoc.Paragraphs.Add();

Word.Range tableRange = tablePr.Range;

Word.Table table = oDoc.Tables.Add(tableRange, rowsCount, 6);

table.Rows.Alignment = WdRowAlignment.wdAlignRowCenter;

table.Columns[1].Width = 156f;

table.Columns[2].Width = 64f;

table.Columns[3].Width = 85f;

table.Columns[4].Width = 50f;

table.Columns[5].Width = 71f;

table.Columns[6].Width = 85f;

table.Borders.Enable = 1;

for (int j = 1; j <= table.Rows.Count; j++)

{

if (j - 1 < slice.Length)

{

string[] row = slice[j - 1].Split(';');

for (int g = 1; g <= Math.Min(table.Columns.Count, row.Length); g++)

{

table.Cell(1, g).Range.Font.Bold = 1;

table.Cell(rowsCount, g).Range.Font.Bold = 1;

if (g - 1 < row.Length)

{

table.Cell(j, g).Range.Text = row[g - 1].Trim();

table.Cell(j, g).TopPadding = 0;

table.Cell(j, g).BottomPadding = 0;

}

else

{

table.Cell(j, g).Range.Text = "";

}

}

}

}

i += rowsCount;

continue;

}

}

}

oDoc.SaveAs2(System.Windows.Forms.Application.StartupPath + "\\Импортированный Word-документ.docx");

oWord.Quit();

MessageBox.Show("Экспорт в Word-документ из CSV выполнен");

}

4.7.2. Сеть Петри запрограммированного технологического процесса

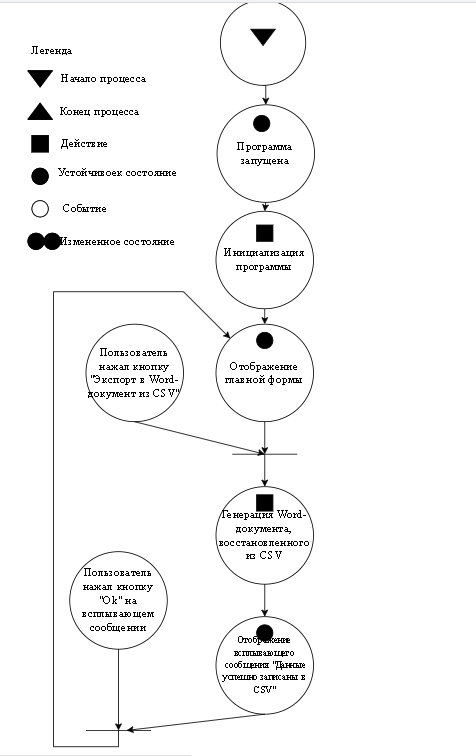


Рисунок 6 – Сеть Петри запрограммированного технологического процесса

4.7.3. Блок-схема алгоритма

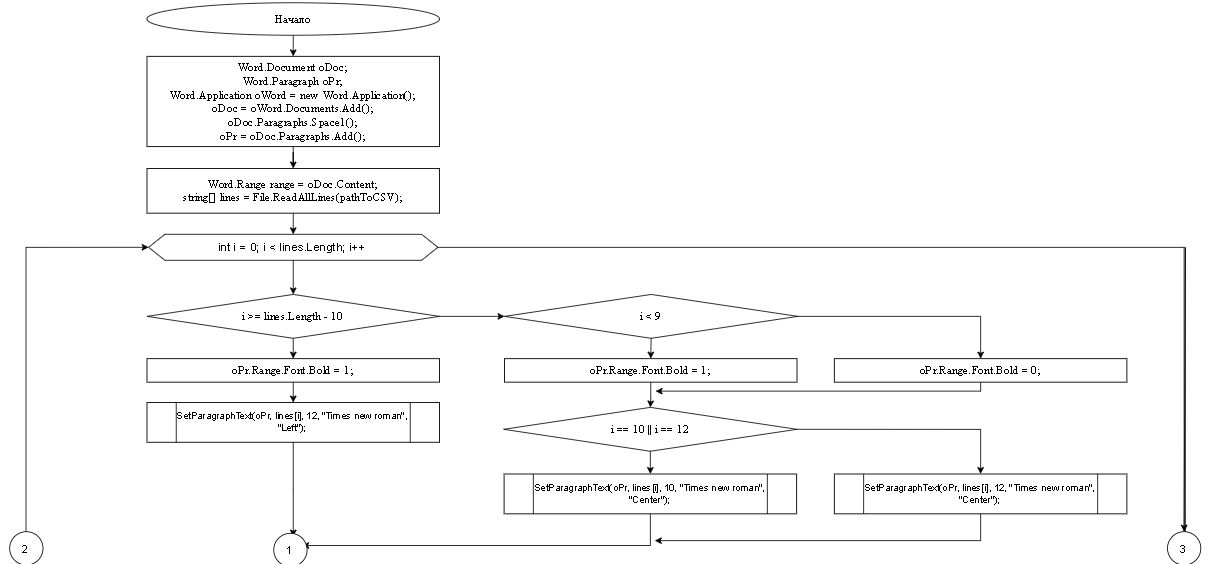


Рисунок 7 – Блок-схема метода *ExportToWord\_Click*

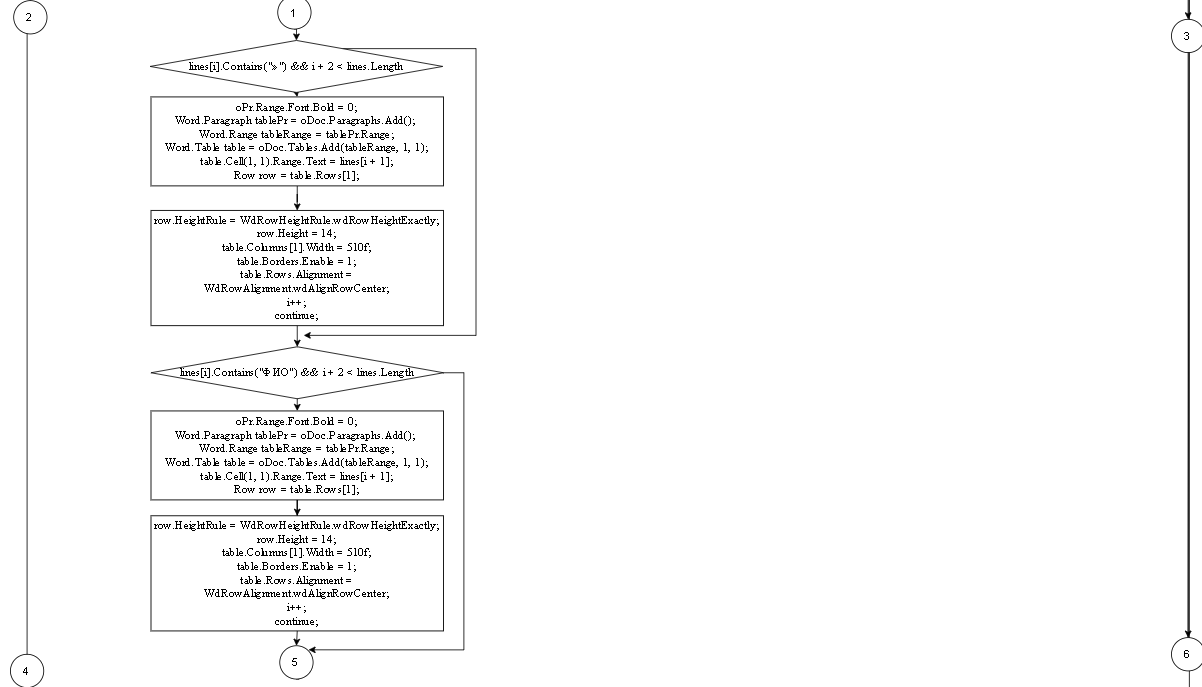


Рисунок 8 – Блок-схема метода *ExportToWord\_Click*

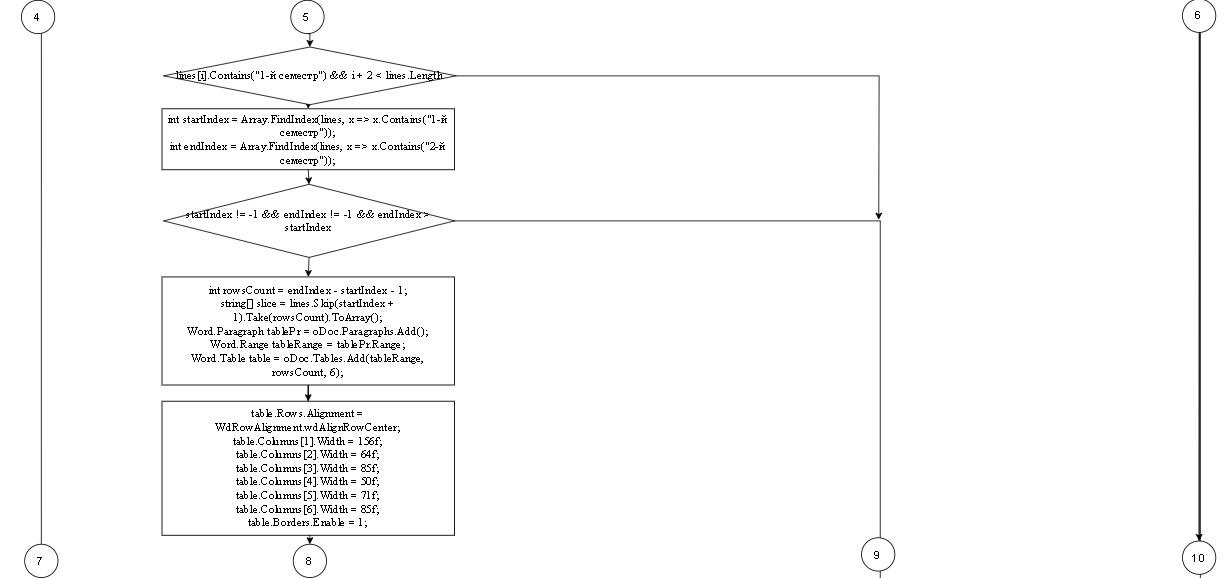


Рисунок 9 – Блок-схема метода *ExportToWord\_Click*

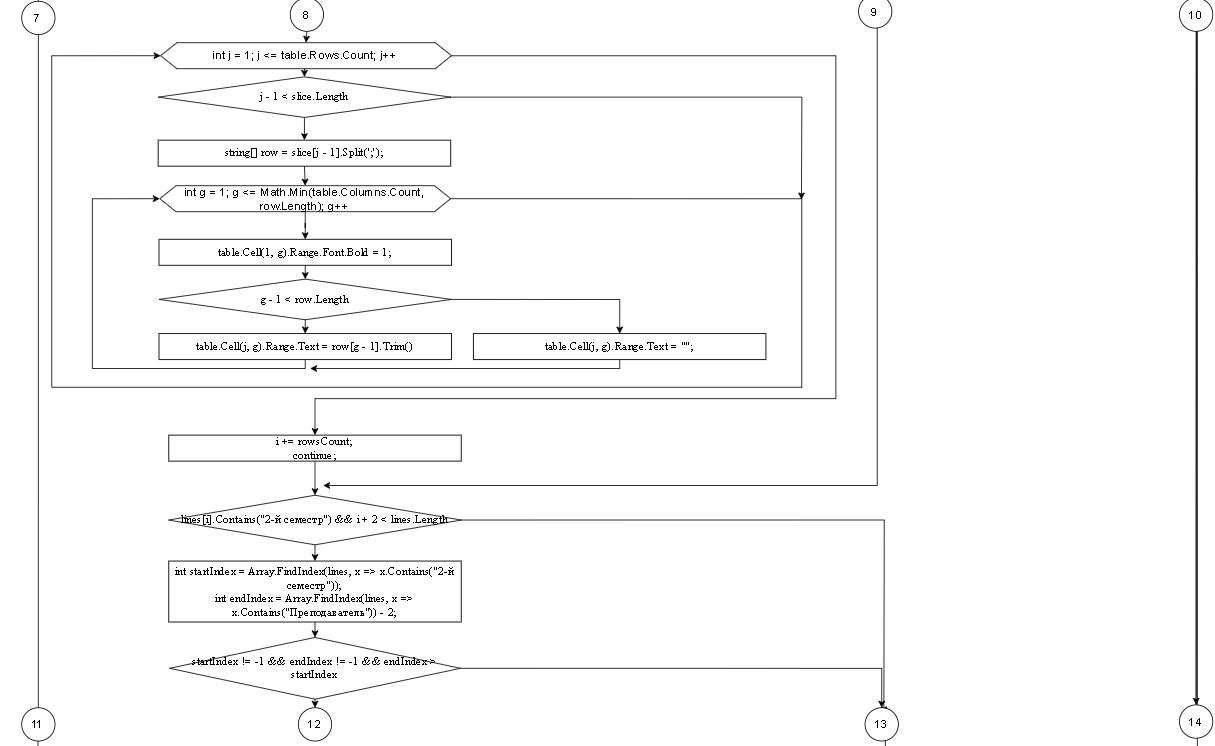
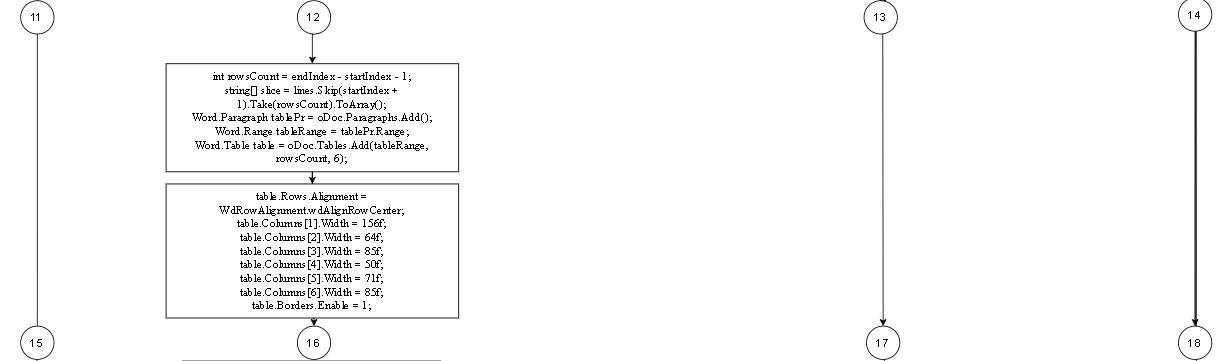
Рисунок 10 – Блок-схема метода *ExportToWord\_Click *

Рисунок 11 – Блок-схема метода *ExportToWord\_Click*

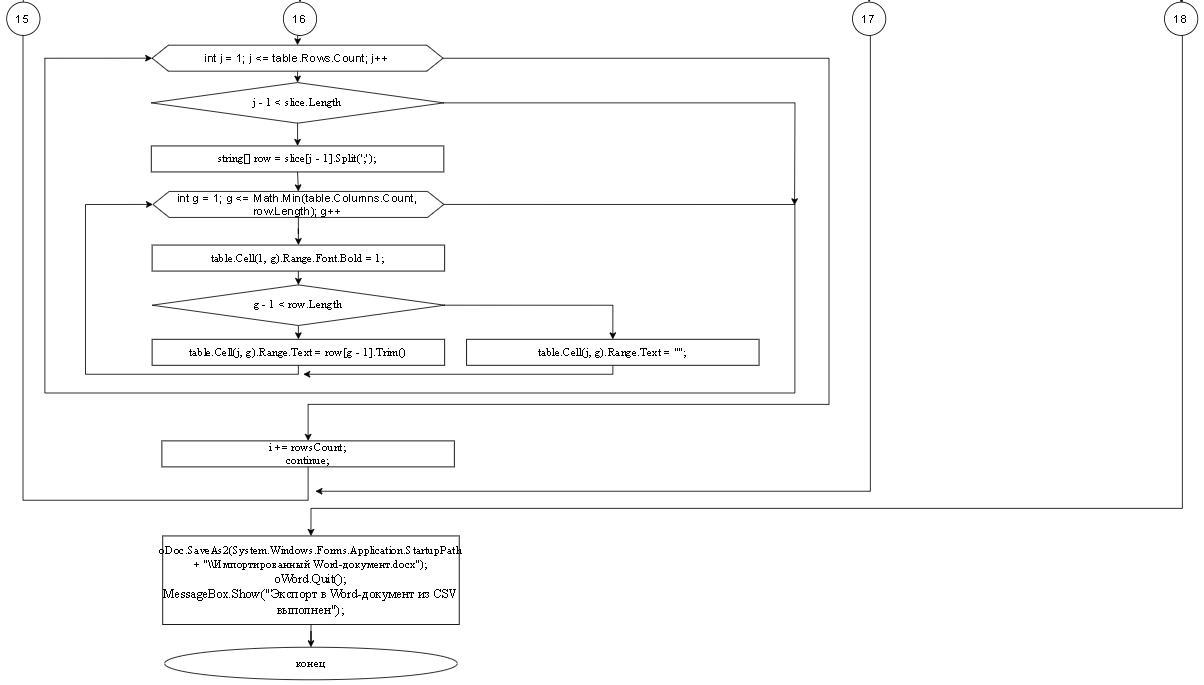
**

Рисунок 12 – Блок-схема метода *ExportToWord\_Click*

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

4.8.1. Листинг

private int WriteTableToExcel(Microsoft.Office.Interop.Word.Table table, Microsoft.Office.Interop.Excel.Workbook workbook, int startRow)

{

Microsoft.Office.Interop.Excel.Worksheet worksheet = (Microsoft.Office.Interop.Excel.Worksheet)workbook.Worksheets[1];

// Заполняем заголовки, если это первая таблица, которую мы обрабатываем

if (startRow == 1)

{

worksheet.Cells[1, 1] = "№ п/п";

worksheet.Cells[1, 2] = "Название";

worksheet.Cells[1, 3] = "Печатный или на правах рукописи";

worksheet.Cells[1, 4] = "Издательство, журнал (название, год, номер)";

worksheet.Cells[1, 5] = "Количество печатных листов или страниц";

worksheet.Cells[1, 6] = "Фамилия соавторов";

startRow++;

}

// Заполняем данные таблицы

for (int i = 2; i <= table.Rows.Count; i++)

{

// Получаем ячейки строки

string cell1 = table.Cell(i, 1).Range.Text.Trim('\r', '\a').Replace(".", ""); // Заменяем точки на пустую строку

var cell2 = table.Cell(i, 2).Range.Text.Trim('\r', '\a');

string cell3 = table.Cell(i, 3).Range.Text.Trim('\r', '\a');

string cell4 = table.Cell(i, 4).Range.Text.Trim('\r', '\a');

string cell5 = table.Cell(i, 5).Range.Text.Trim('\r', '\a');

string cell6 = table.Cell(i, 6).Range.Text.Trim('\r', '\a');

if (cell2 == "2")

{

continue; // Пропускаем запись для текущей строки

}

// Заполнение ячеек Excel

worksheet.Cells[startRow, 1] = cell1;

worksheet.Cells[startRow, 2] = cell2;

worksheet.Cells[startRow, 3] = cell3;

worksheet.Cells[startRow, 4] = cell4;

worksheet.Cells[startRow, 5] = cell5;

worksheet.Cells[startRow, 6] = cell6;

startRow++;

}

return startRow; // Возвращаем следующую строку для записи

}

private string NormalizeCoauthorName(string name)

{

// Удаляем пробелы и приводим к нижнему регистру

string normalized = name.ToLower().Trim();

// Разделяем имя на части

var nameParts = normalized.Split(new[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

// Проверяем, чтобы было хотя бы две части

if (nameParts.Length >= 2)

{

if (nameParts[0].Contains(".") && !nameParts[nameParts.Length - 1].Contains("."))

{

// Форматируем как "Фамилия Инициалы"

string initials = string.Join(" ", nameParts.Take(nameParts.Length - 1));

string surname = nameParts[nameParts.Length - 1];

return $"{surname} {initials}";

}

return normalized;

}

return normalized; // На случай, если имя состоит только из одной части

}

private Dictionary<string, List<string>> coauthorsDictionary = new Dictionary<string, List<string>>(); // Словарь для хранения соавторов

private Dictionary<string, int> GetUniqueCoauthorsCount(string excelFilePath)

{

HashSet<string> uniqueCoauthors = new HashSet<string>();

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

// Открываем Excel и получаем первый лист

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

Workbook workbook = excelApp.Workbooks.Open(excelFilePath);

Worksheet worksheet = (Worksheet)workbook.Worksheets[1];

// Получаем количество строк в листе

int lastRow = worksheet.Cells[worksheet.Rows.Count, 6].End[Microsoft.Office.Interop.Excel.XlDirection.xlUp].Row;

for (int i = 2; i <= lastRow; i++)

{

string coauthorsCell = ((Microsoft.Office.Interop.Excel.Range)worksheet.Cells[i, 6]).Text;

if (!string.IsNullOrWhiteSpace(coauthorsCell) && coauthorsCell.ToLower() != "нет")

{

// Разделяем имена по запятой и переносу строки

string[] coauthors = coauthorsCell.Split(new[] { ',', '\r' }, StringSplitOptions.RemoveEmptyEntries);

foreach (var coauthor in coauthors)

{

// Нормализуем имя соавтора

string coauthorName = NormalizeCoauthorName(coauthor.Trim());

// Добавляем нормализованное имя в HashSet

uniqueCoauthors.Add(coauthorName);

coauthorsList.Add(coauthorName); // Добавляем в общий список

}

}

}

// Закрываем Excel

workbook.Close(false);

excelApp.Quit();

// Освобождаем ресурсы

System.Runtime.InteropServices.Marshal.ReleaseComObject(worksheet);

System.Runtime.InteropServices.Marshal.ReleaseComObject(workbook);

System.Runtime.InteropServices.Marshal.ReleaseComObject(excelApp);

string fileNameWithoutExtension = System.IO.Path.GetFileNameWithoutExtension(excelFilePath);

// Добавляем соавторов в словарь

coauthorsDictionary[fileNameWithoutExtension] = coauthorsList;

return new Dictionary<string, int> { { fileNameWithoutExtension, uniqueCoauthors.Count } };

}

// Обновляем Chart

private void UpdateChart(Dictionary<string, int> newCoauthorsCount)

{

foreach (KeyValuePair<string, int> entry in newCoauthorsCount)

{

if (!coauthorsCount.ContainsKey(entry.Key))

{

coauthorsCount[entry.Key] = 0;

}

coauthorsCount[entry.Key] += entry.Value;

}

chart1.ChartAreas[0].AxisY.Interval = 1;

chart1.ChartAreas[0].AxisY.Title = "Количество уникальных соавторов";

chart1.ChartAreas[0].AxisX.Title = "";

foreach (KeyValuePair<string, int> entry in newCoauthorsCount)

{

System.Windows.Forms.DataVisualization.Charting.Series series = chart1.Series.FirstOrDefault(s => s.Name == entry.Key);

if (series == null)

{

series = new System.Windows.Forms.DataVisualization.Charting.Series(entry.Key)

{

ChartType = SeriesChartType.Column

};

chart1.Series.Add(series);

}

series.Points.AddXY(entry.Key, entry.Value);

}

chart1.ChartAreas[0].AxisX.Title = "Уникальные соавторы";

foreach (DataPoint point in chart1.Series.SelectMany(s => s.Points))

{

point.AxisLabel = "";

}

chart1.Invalidate();

}

// Обработчик клика по диаграмме

private void chart1\_Click(object sender, EventArgs e)

{

System.Drawing.Point mousePosition = PointToClient(MousePosition);

HitTestResult hitTestResult = chart1.HitTest(mousePosition.X, mousePosition.Y);

if (hitTestResult.ChartElementType == ChartElementType.DataPoint ||

hitTestResult.ChartElementType == ChartElementType.PlottingArea)

{

System.Windows.Forms.DataVisualization.Charting.Series series = hitTestResult.Series;

int pointIndex = hitTestResult.PointIndex;

if (series != null && pointIndex >= 0)

{

string seriesName = series.Name; // Получаем имя серии

if (coauthorsDictionary.TryGetValue(seriesName, out List<string> coauthors))

{

// Используем HashSet для уникальных соавторов

HashSet<string> uniqueCoauthors = new HashSet<string>(coauthors);

listBox1.Items.Clear(); // Очищаем список перед добавлением новых элементов (если это необходимо)

foreach (var coauthor in uniqueCoauthors)

{

if (coauthor != "")

{

listBox1.Items.Add(FormatCoauthor(coauthor)); // Добавляем каждого соавтора в ListBox

}

}

y.UpdateChartForYears(seriesName);

p.UpdateVolumeChart(seriesName);

}

}

else if (hitTestResult.ChartElementType == ChartElementType.PlottingArea)

{

foreach (System.Windows.Forms.DataVisualization.Charting.Series s in chart1.Series)

{

if (s.Points.Count > 0)

{

string seriesName = s.Name;

if (coauthorsDictionary.TryGetValue(seriesName, out List<string> coauthors))

{

// Используем HashSet для уникальных соавторов

HashSet<string> uniqueCoauthors = new HashSet<string>(coauthors);

listBox1.Items.Clear(); // Очищаем список перед добавлением новых элементов (если это необходимо)

foreach (string coauthor in uniqueCoauthors)

{

if (coauthor != "")

{

listBox1.Items.Add(FormatCoauthor(coauthor)); // Добавляем каждого соавтора в ListBox

}

}

y.UpdateChartForYears(seriesName);

p.UpdateVolumeChart(seriesName);

break;

}

}

}

}

}

}

private string FormatCoauthor(string name)

{

// Нормализуем имя

string normalized = NormalizeCoauthorName(name);

// Разбиваем на части

string[] parts = normalized.Split(new[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

if (parts.Length == 0)

return name; // Если нет частей, возвращаем исходное имя

// Предполагаем, что первая часть - фамилия, остальные части - инициалы

string surname = parts[0]; // Фамилия

string initials = string.Empty;

// Форматируем фамилию с заглавной буквы

string formattedSurname = char.ToUpper(surname[0]) + surname.Substring(1).ToLower(); // первая буква заглавная, остальные строчные

// Проверяем наличие инициала

if (parts.Length > 1) // Если есть инициалы

{

initials += char.ToUpper(parts[1][0]) + "."; // Первая буква инициала

// Проверка на наличие второй буквы

if (parts[1].Length > 2 && parts[1][1] == '.') // Проверяем, что после первой буквы есть точка

{

initials += char.ToUpper(parts[1][2]) + "."; // Вторая буква инициала

}

}

// Проверяем наличие третьего элемента

if (parts.Length > 2) // Если есть третий элемент

{

initials += char.ToUpper(parts[2][0]) + "."; // Используем первую букву третьего элемента в качестве инициала

}

// Возвращаем форматированную строку

return $"{formattedSurname} {initials.Trim()}".Trim();

}

private void ConvertExcelToCsv(string excelFilePath)

{

// Создаем экземпляр приложения Excel

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

excelApp.Visible = false; // Открываем Excel в фоновом режиме

// Открываем существующий Excel файл

Microsoft.Office.Interop.Excel.Workbook workbook = excelApp.Workbooks.Open(excelFilePath);

// Получаем первый лист в книге

Microsoft.Office.Interop.Excel.Worksheet worksheet = (Microsoft.Office.Interop.Excel.Worksheet)workbook.Worksheets[1];

// Определяем путь для сохранения CSV файла

string csvFilePath = System.IO.Path.ChangeExtension(excelFilePath, ".csv");

// Сохраняем текущий лист как CSV файл

worksheet.SaveAs(csvFilePath, Microsoft.Office.Interop.Excel.XlFileFormat.xlCSV);

// Закрываем работу с Excel

workbook.Close(false); // Закрываем workbook без сохранения

excelApp.Quit(); // Закрываем приложение Excel

// Освобождаем объекты

System.Runtime.InteropServices.Marshal.ReleaseComObject(worksheet);

System.Runtime.InteropServices.Marshal.ReleaseComObject(workbook);

System.Runtime.InteropServices.Marshal.ReleaseComObject(excelApp);

// Подтверждаем успешное создание CSV файла

MessageBox.Show($"Конвертация в CSV завершена! Файл сохранен как: {csvFilePath}", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

4.8.2. Сеть Петри запрограммированного технологического процесса

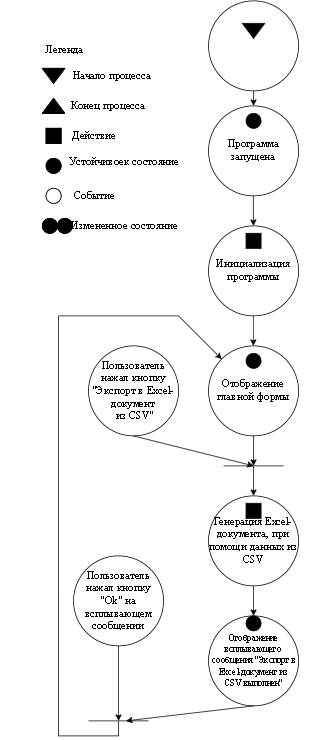


Рисунок 13 – Сеть Петри запрограммированного технологического процесса

4.8.3. Блок-схема алгоритма

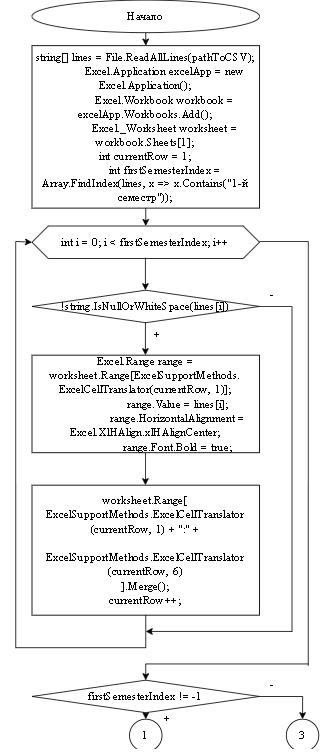


Рисунок 14 – Блок-схема метода *ExportToExcel\_Click*

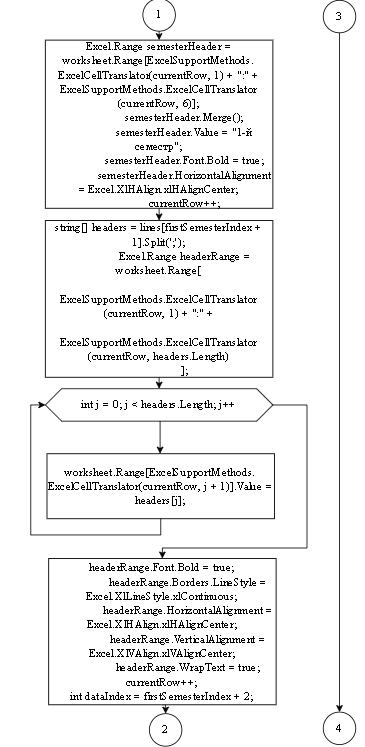


Рисунок 15 – Блок-схема метода *ExportToExcel\_Click*

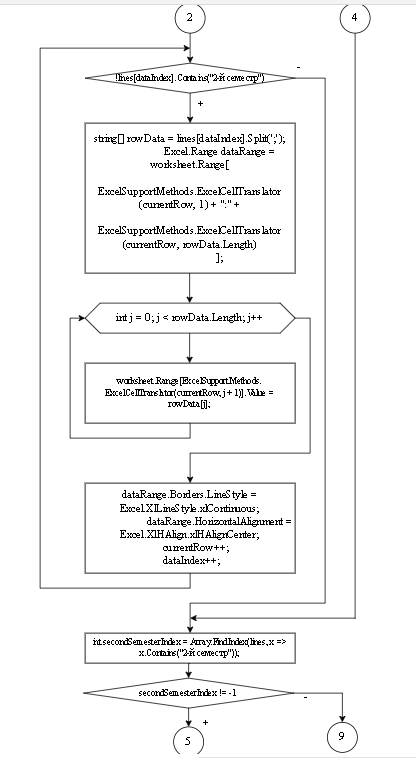
**

Рисунок 16 – Блок-схема метода *ExportToExcel\_Click*

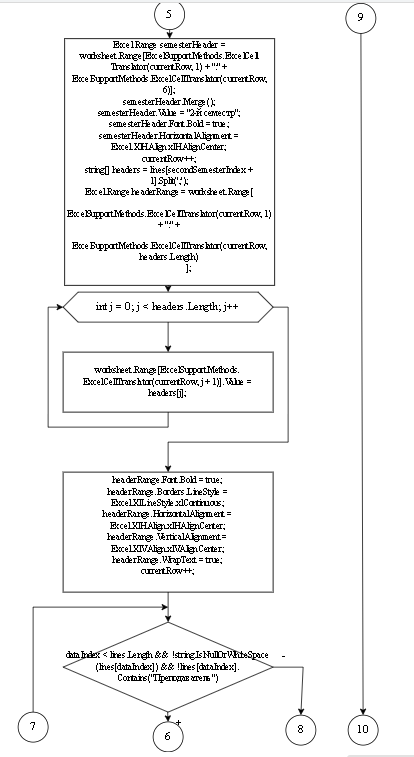
**

Рисунок 17 – Блок-схема метода *ExportToExcel\_Click*

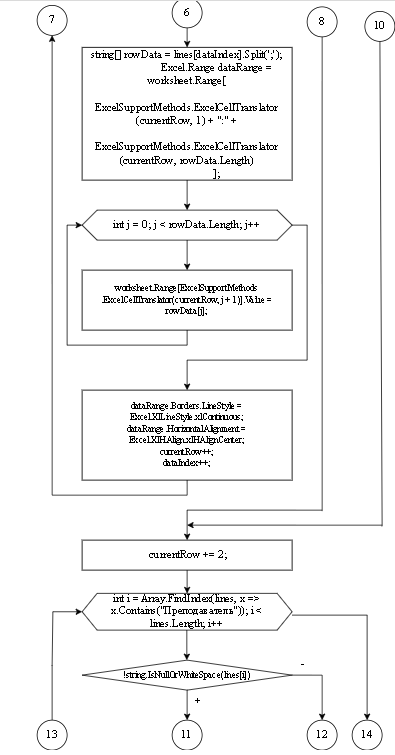
**

Рисунок 18 – Блок-схема метода *ExportToExcel\_Click*

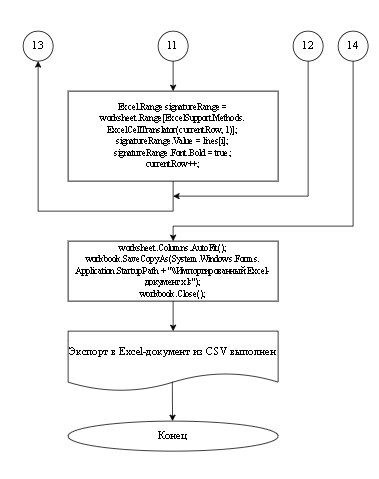
**

Рисунок 19 – Блок-схема метода *ExportToExcel\_Click*

4.9. Графический анализ данных, содержащихся в печатной форме

Режимы диаграмм, отображающихся на элементе экранной формы *Chart*, выбираются при помощи выпадающего списка (*ComboBox*), режимы диаграмм реализованы согласно рабочему заданию:

1. Показывать на столбчатой диаграмме количество уникальных соавторов у каждого автора.

2. Показывать на столбчатой диаграмме для выбранного автора количество публикаций за каждый год.

3. Показывать на столбчатой диаграмме для выбранного автора количества работ различного объёма в печатных страницах (1 печатный лист (п.л.) равен 1 страницам).

Данные диаграммы берутся из печатных форм, посредством циклической итерации по каждому документу.

4.9.1. Пример работы графического анализа данных, содержащихся в печатных формах

Пример работы режима диаграммы «1. Показывать на столбчатой диаграмме количество уникальных соавторов у каждого автора.» на Рисунке 20:

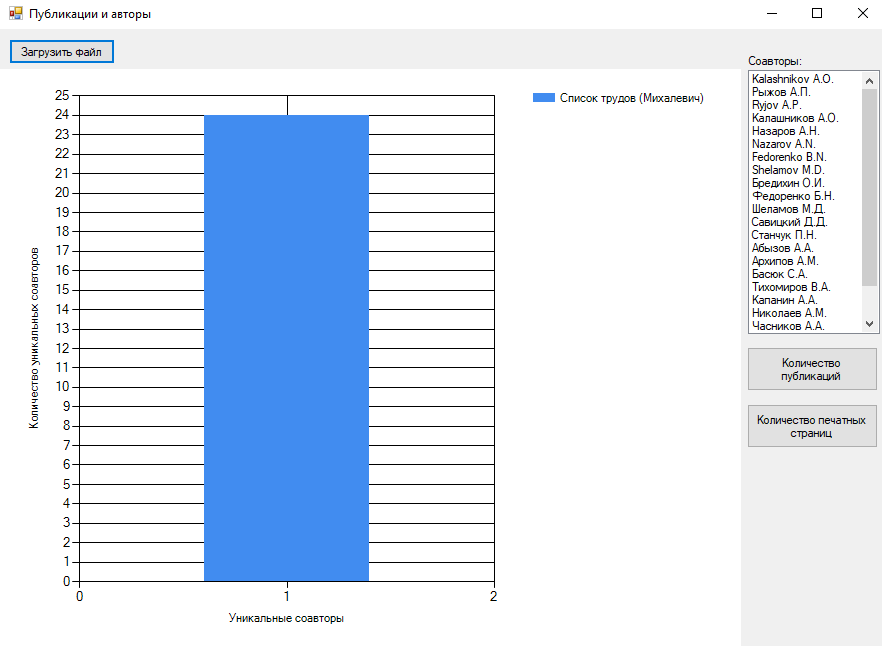


Рисунок 20 – Пример работы первого режима диаграммы

Пример работы режима диаграммы «2. Показывать на столбчатой диаграмме для выбранного автора количество публикаций за каждый год.» на Рисунке 21:

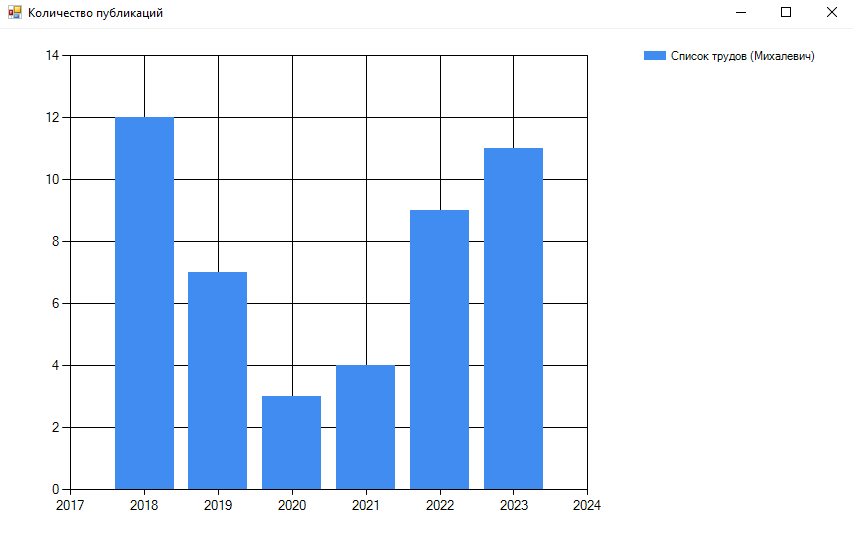


Рисунок 21 – Пример работы второго режима диаграммы

Пример работы режима диаграммы «3. Показывать на столбчатой диаграмме для выбранного автора количества работ различного объёма в печатных страницах (1 печатный лист (п.л.) равен 1 страницам).» на Рисунке 22:

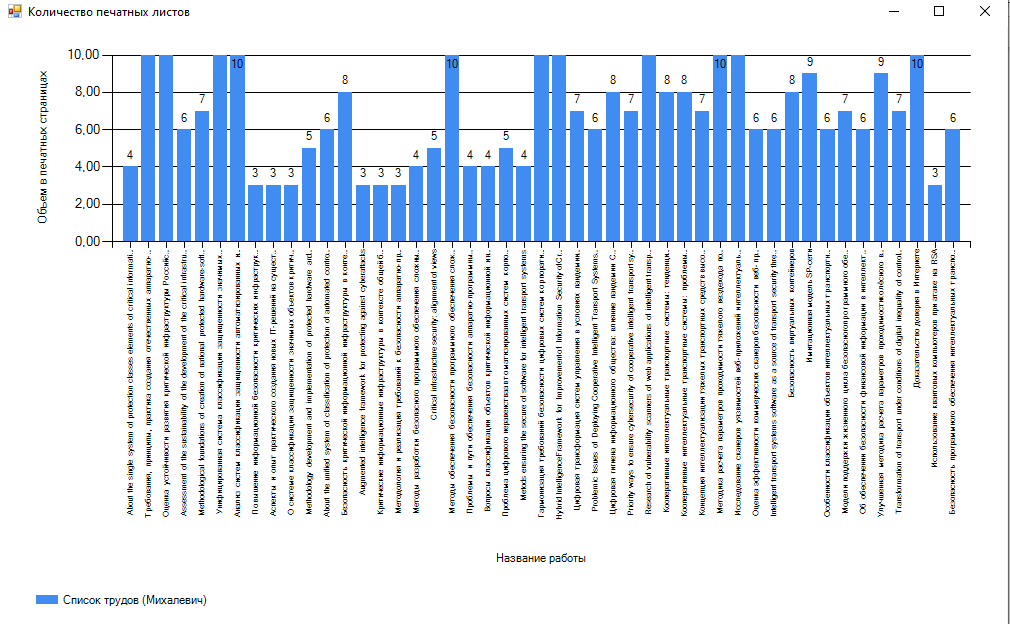


Рисунок 22 – Пример работы третьего режима диаграммы

5. Инструкция пользователя

Интерфейс главного окна программы на Рисунке 23:

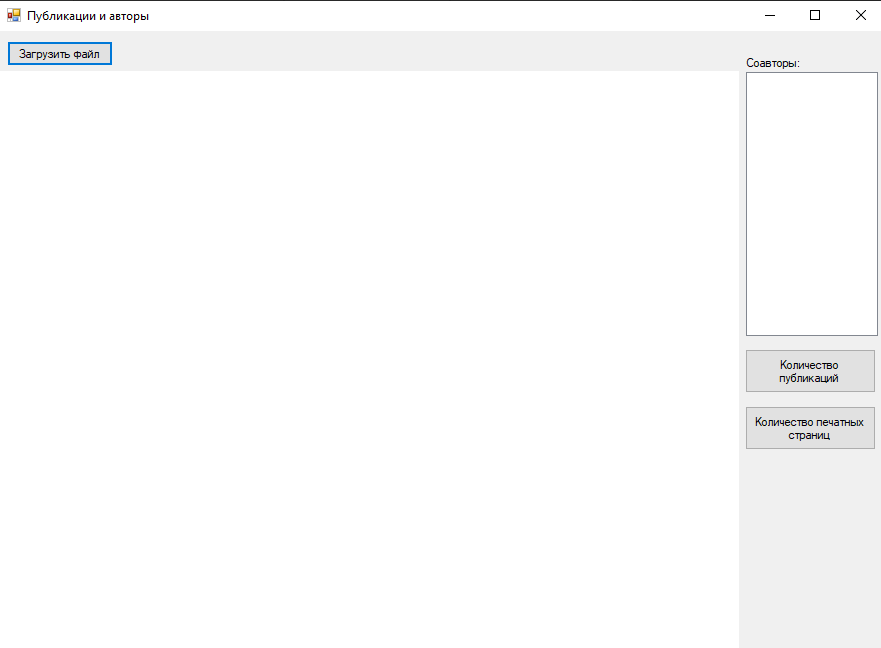


Рисунок 23 – Интерфейс главного окна программы

Группы элементов экранной формы, отвечающих за выбор и генерацию документов:

1. Функции импорта:
   1. Кнопка «Загрузить файл»: позволяет выбрать печатную форму из файловой системы для импорта данных в *CSV*
   2. Кнопка «ОК»: инициализирует процесс импорта сведений из выбранного файла в *CSV*, если файл не будет предварительно выбран, то будет показано всплывающее сообщение, свидетельствующее об этом (Рисунок 24):

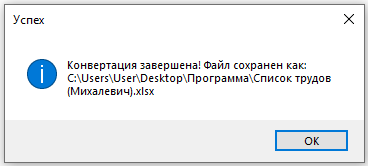


Рисунок 24 – Всплывающее сообщение «Конвертация завершена!»

1. Функции экспорта:
   1. Кнопка «ОК»: инициализирует процесс генерации *Word*-документа, основываясь на данных из *CSV­*-файла. По завершению этого процесса появится всплывающее сообщение о выполнении процесса экспорта (Рисунок 25):

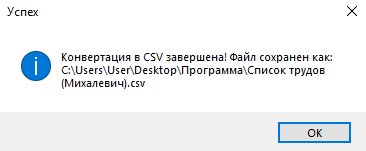


Рисунок 25 – Всплывающее сообщение «Конвертация в *CSV* завершена!»

6. Заключение

В результате выполненной работы было создано программное решение для электронного документооборота, включающее в себя функции конвертации данных в различные форматы (DOCX, XLS, CSV), а также поддержку графического анализа информации из печатных форм. Разработанная система обеспечивает эффективную обработку электронных документов, облегчает их анализ и экономит время пользователя за счет автоматизации рутинных процессов. Таким образом, повышается общая эффективность при работе с электронными документами.

7. Выводы

В процессе выполнения работы был разработан эргономичный пользовательский интерфейс, который позволяет пользователю легко пользоваться элементами и функциями программного обеспечения, предназначенными для электронного документооборота. В процессе выполнения работы был освоен навык работы с пространством имен *Microsoft.Office.Interop*, использование его для работы с *Excel,Word* программным путём, также был повышен уровень работы с *CSV* файлами. Ещё был освоен навык работы с *Chart*-элементом экранной формы (настройка осей диаграммы и серий данных диаграммы). Закреплен навык чтения и записи в файлы, а также улучшен навык написания алгоритмов.

Библиографический список

1. Электронный документооборот в сфере трудовых отношений

*URL:* [ТК РФ Статья 22.1. Электронный документооборот в сфере трудовых отношений \ КонсультантПлюс](https://www.consultant.ru/document/cons_doc_LAW_34683/6dda95f9b6acf89a431a0737944b9443695b9df9/) (дата обращения: 20.01.2025).

2. Электронный документооборот (ЭДО): для чего он нужен и как организовать его в компании [Электронный источник]: *skillbox.ru URL:https://skillbox.ru/media/management/elektronnyy-dokumentooborot-edo-dlya-chego-on-nuzhen-i-kak-organizovat-ego-v-kompanii/* (дата обращения: 10.01.2025).

3. Применение электронного документооборота. Для чего бизнесу нужен ЭДО? [Электронный источник]: *esphere.ru URL: https://www.esphere.ru/blog/chto-takoe-elektronnyj-dokumentooborot/* (дата обращения: 10.01.2025).