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-341

Кох Е.Е.

Насонова А.Н.

Папошина Л.С.

Вариант №10

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

Москва ­– 2024 г.

**Содержание**

[1. Введение 3](#_Toc168262618)

[2. Цель работы 3](#_Toc168262619)

[3. Задачи работы 3](#_Toc168262620)

[4. Содержательная часть работы 5](#_Toc168262621)

[4.1 Анализ печатной формы 5](#_Toc168262622)

[4.2 Описание предназначения печатной формы 5](#_Toc168262623)

[4.3 Разнообразие печатных форм и выявление различий 5](#_Toc168262624)

[4.4 Унификация печатных форм для удобства обработки данных 6](#_Toc168262625)

[4.5 Разработка экранной формы для импорта сведений из печатной формы 6](#_Toc168262631)

[4.6 Разработка функций экспорта сведений из печатной формы в файл текстовый формата CSV 11](#_Toc168262632)

[4.7 Разработка функций импорта сведений из оригинального и изменённого текстовых файлов формата CSV с воссозданием Word 15](#_Toc168262633)

[4.8 Разработка функций экспорта сведений из экранной формы в Excel 20](#_Toc168262634)

[4.9 Графический анализ данных, содержащихся в печатной форме (настройка Chart-элемента) 25](#_Toc168262635)

[4.9.1 Показывать на столбчатой диаграмме популярность заявок на аудитории без учёта "или" (первых из упомянутых) за весь известный период, за указанный год, за указанный семестр 29](#_Toc168262636)

[4.9.2 Показывать на столбчатой диаграмме популярность заявок на аудитории всех упомянутых за весь известный период, за указанный год, за указанный семестр 34](#_Toc168262637)

[4.9.3 Показывать на столбчатой диаграмме количество пожеланий указанным преподавателем аудиторий за весь известный период 39](#_Toc168262638)

[4.9.4 Показывать на столбчатой диаграмме количество пожеланий указанной аудитории преподавателями за весь известный период 42](#_Toc168262639)

[5. Инструкция пользователя 43](#_Toc168262640)

[6. Заключение 46](#_Toc168262641)

[7. Выводы 61](#_Toc168262642)

[8. Библиографический список 62](#_Toc168262643)

## **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 на столбчатой диаграмме популярность заявок на аудитории без учёта "или" (первых из упомянутых) за весь известный период;

2. Показывать на столбчатой диаграмме популярность заявок на аудитории без учёта "или" (первых из упомянутых) за указанный год;

3. Показывать на столбчатой диаграмме популярность заявок на аудитории без учёта "или" (первых из упомянутых) за указанный семестр;

4. Показывать на столбчатой диаграмме популярность заявок на аудитории всех упомянутых за весь известный период;

5. Показывать на столбчатой диаграмме популярность заявок на аудитории всех упомянутых за указанный год;

6. Показывать на столбчатой диаграмме популярность заявок на аудитории всех из упомянутых за указанный семестр;

7. Показывать на столбчатой диаграмме количество пожеланий указанным преподавателем аудиторий за весь известный период;

8. Показывать на столбчатой диаграмме количество пожеланий указанной аудитории преподавателями за весь известный период.

## **Содержательная часть работы**

## **Анализ печатной формы**

Печатные формы представляют собой документы WORD, в которых находится таблица с заявками Кафедры ''Управление и защита информации'' в диспетчерскую на проведение занятий семестрах с 2015 по 2024 учебные года.

## **4.2 Описание предназначения печатной формы**

Таблица включает в себя ФИО преподавателей и их пожелания к составлению расписания на семестр, то есть номера аудиторий для проведения их дисциплин, а также дни недели и время назначения занятий.

На рисунке 1 показан общий вид печатной формы.

Изображение выглядит как текст, число, Шрифт, чек

Автоматически созданное описание

Рисунок 1 – Общий вид печатной формы

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

Таблицы в файлах отличаются количеством и названием столбцов. Например, в файле под названием «2015-2016 I семестр» по сравнению с файлом «2023-2024 II семестр» меньшее количество столбцов и некоторые столбцы отличаются названием. 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 файлах «2023-2024 I семестр», «2023-2024 II семестр» уберем первый столбец «№ п/п», так как в других печатных формах его нет. В некоторых файлах разделим один столбец «Примечания» на два – «Примечания» и «Аудитория» для построения столбчатой диаграммы. Также объединенные ячейки с ФИО преподавателей разделим так, чтобы напротив каждой ячейки с номером аудитории стояло ФИО преподавателя. Это необходимо для достоверного построения столбчатой диаграммы в Chart-элементе. Для этих же целей во всех файлах очистим данные в столбцах ФИО от всего, кроме ФИО преподавателей.

1. 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. 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.

Изображение выглядит как круг, снимок экрана, черно-белый

Автоматически созданное описание

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.

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

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 «!». 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.

Изображение выглядит как круг, снимок экрана, черно-белый

Автоматически созданное описание

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, графический дизайн

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

if (File.Exists(выбранныйФайл))

{

ConvertWordToCSV(выбранныйФайл);

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 выбранныйФайл)

{

try

{

using (WordprocessingDocument wordDocument = WordprocessingDocument.Open(выбранныйФайл, 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 папкаCSV = Path.Combine(Environment.CurrentDirectory, "csv");

if (!Directory.Exists(папкаCSV))

{

Directory.CreateDirectory(папкаCSV);

}

csvFilePath = Path.Combine(папкаCSV, $"{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.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

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

**Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание**

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». 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.

Изображение выглядит как круг, снимок экрана, черно-белый

Автоматически созданное описание

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, Шрифт, 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.

Изображение выглядит как текст, Шрифт, программное обеспечение, снимок экрана

Автоматически созданное описание

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

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

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 InsideHorizontalBorder() { 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 (IOException ex)

{

MessageBox.Show($"Ошибка при попытке открыть файл Word: {ex.Message}", "Ошибка");

}

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». На рисунке 13 отобразим сеть Петри, отражающую разработку функций экспорта сведений из экранной формы в Excel. А на рисунке покажем 14 схему этого алгоритма.

Изображение выглядит как круг, снимок экрана, черно-белый

Автоматически созданное описание

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

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

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.

Изображение выглядит как текст, Шрифт, программное обеспечение, снимок экрана

Автоматически созданное описание

Рисунок 17 – Файл, экспортирующий сведения в EXCEL из файла формата CSV в папке «excel»

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 18 – 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 папкаExcel = Path.Combine(Environment.CurrentDirectory, "excel");

if (!Directory.Exists(папкаExcel))

{

Directory.CreateDirectory(папкаExcel);

}

string excelFilePath = Path.Combine(папкаExcel, $"{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 19 отобразим сеть Петри, отражающую настройку формы для Chart-элемента. А на рисунке покажем 14 схему этого алгоритма.

Изображение выглядит как круг, снимок экрана, черно-белый

Автоматически созданное описание

Рисунок 19 – Сеть Петри, отражающая настройку формы для Chart-элемента

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

Автоматически созданное описание

Рисунок 20 – Схема алгоритма, отражающая настройку формы для Chart-элемента

На рисунке 21 покажем дочернюю форму для построения столбчатых диаграмм.

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 21 – Дочернюю форму 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(); // Создаем форму frmAnalysis

frmAnalysis.Text = "Анализ данных";

frmAnalysis.Size = new Size(800, 600);

Button btnPopulAll = new Button();

btnPopulAll.Text = "Построить диаграмму 'Популярность заявок на аудитории всех упомянутых преподавателей'";

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

btnPopulAll.Location = new Point(30, 450);

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

frmAnalysis.Controls.Add(btnPopulAll);

Button btnPopul = new Button();

btnPopul.Text = "Построить диаграмму 'Популярность заявок на аудитории первых из упомянутых преподавателей'";

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

btnPopul.Location = new Point(220, 450);

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

frmAnalysis.Controls.Add(btnPopul);

TextBox txtTeacher = new TextBox();

txtTeacher.Location = new Point(505, 420);

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

frmAnalysis.Controls.Add(txtTeacher);

Button btnWish = new Button();

btnWish.Text = "Построить диаграмму 'Количество пожеланий указанным преподавателем аудиторий за весь известный период'";

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

btnWish.Location = new Point(410, 450);

btnWish.Click += (s, ev) => { UpdateChartByTeacherName(txtTeacher.Text); }; // Передаем значение из TextBox в метод

frmAnalysis.Controls.Add(btnWish);

Button btnAuditor = new Button();

btnAuditor.Text = "Построить диаграмму 'Количество пожеланий указанной аудитории преподавателями за весь известный период'";

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

btnAuditor.Location = new Point(600, 450);

btnAuditor.Click += (s, ev) => { UpdateChartByTeacherAuditor(txtTeacher.Text); }; // Передаем значение из TextBox в метод

frmAnalysis.Controls.Add(btnAuditor);

frmAnalysis.Show();

}

else

{

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

}

}

## **4.9.1 Показывать на столбчатой диаграмме популярность заявок на аудитории без учёта "или" (первых из упомянутых) за весь известный период, за указанный год, за указанный семестр**

В зависимости от промежутка построения диаграммы (за весь известный период, за указанный год, за указанный семестр), пользователю необходимо сконвертировать нужное количество файлов WORD в формат CSV. На рисунках 23, 25, 27 покажем столбчатые диаграммы «Популярность заявок на аудитории без учёта "или" (первых из упомянутых) за весь известный период, за указанный год, за указанный семестр».

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 22 – Файл, из которого берутся данные для диаграммы «Популярность заявок на аудитории без учёта "или" (первых из упомянутых) за указанный семестр»

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

Рисунок 23 – Диаграмма «Популярность заявок на аудитории без учёта "или" (первых из упомянутых) за указанный семестр»

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

Рисунок 24 – Файлы, из которых берутся данные для диаграммы «Популярность заявок на аудитории без учёта "или" (первых из упомянутых) за указанный год»

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 25 – Диаграмма «Популярность заявок на аудитории без учёта "или" (первых из упомянутых) за указанный год»

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 26 – Файлы, из которых берутся данные для диаграммы «Популярность заявок на аудитории без учёта "или" (первых из упомянутых) за весь известный период»

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

Рисунок 27 - Диаграмма «Популярность заявок на аудитории без учёта "или" (первых из упомянутых) за весь известный период»

**Листинг кода, реализующий построение диаграмм «Популярность заявок на аудитории без учёта "или" (первых из упомянутых) за весь известный период, за указанный год, за указанный семестр»**

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

{

string folderPath = @"C:\Users\Екатерина\source\repos\курсовая работа 2.0\bin\Debug\csv";

if (Directory.Exists(folderPath))

{

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

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

{

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

using (var reader = new StreamReader(filePath))

{

string line;

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

{

// Разделяем строку по "!", удаляя пустые значения

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

data.Add(parts);

}

}

var top5Aud = data.Skip(1).Take(24).ToList();

foreach (var row in top5Aud)

{

if (row.Length > 0) // Проверяем, есть ли хотя бы одно значение в строке

{

string audience = row[row.Length - 1].Trim(); // Получаем значение аудитории из последнего столбца

if (popularity.ContainsKey(audience))

{

popularity[audience]++;

}

else

{

popularity.Add(audience, 1);

}

}

}

}

Chart chart = new Chart();

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

chart.Parent = frmAnalysis;

chart.Dock = DockStyle.None;

chart.Location = new Point(50, 10);

chart.Titles.Add("Популярность аудиторий"); // Добавляем название диаграммы

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 popularity)

{

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

chart.Series["Кол-во"].Points[pointIndex].IsValueShownAsLabel = false; // Скрываем значение

if (!string.IsNullOrEmpty(item.Key))

{

chart.Series["Кол-во"].Points[pointIndex].AxisLabel = item.Key; // Добавляем метку к каждому столбцу, если ключ не пустой

}

}

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;

frmAnalysis.Controls.Add(chart); // Добавляем диаграмму на форму frmAnalysis

}

else

{

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

}

}

## **4.9.2 Показывать на столбчатой диаграмме популярность заявок на аудитории всех упомянутых за весь известный период, за указанный год, за указанный семестр**

В зависимости от промежутка построения диаграммы (за весь известный период, за указанный год, за указанный семестр), пользователю необходимо сконвертировать нужное количество файлов WORD в формат CSV. На рисунках 29, 31, 33 покажем столбчатые диаграммы «Популярность заявок на аудитории всех упомянутых за весь известный период, за указанный год, за указанный семестр».

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 28 – Файлы, из которых берутся данные для диаграммы «Популярность заявок на аудитории всех упомянутых за весь известный период»

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 29 – Диаграмма «Популярность заявок на аудитории всех упомянутых за весь известный период»

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

Рисунок 30 – Файл, из которого берутся данные для диаграммы «Популярность заявок на аудитории всех упомянутых за указанный семестр»

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 31 – Диаграмма «Популярность заявок на аудитории всех упомянутых за указанный семестр»

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

Рисунок 32 – Файлы, из которых берутся данные для диаграммы «Популярность заявок на аудитории всех упомянутых за указанный год»

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 33 – Диаграмма «Популярность заявок на аудитории всех упомянутых за указанный год»

**Листинг кода, реализующий построение диаграмм «Популярность заявок на аудитории всех упомянутых за весь известный период, за указанный год, за указанный семестр»**

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

{

string folderPath = @"C:\Users\Екатерина\source\repos\курсовая работа 2.0\bin\Debug\csv";

if (Directory.Exists(folderPath))

{

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

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

{

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);

}

}

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

{

if (row.Length > 0)

{

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

if (popularity.ContainsKey(audience))

{

popularity[audience]++;

}

else

{

popularity.Add(audience, 1);

}

}

}

}

Chart chart = new Chart();

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

chart.Parent = frmAnalysis;

chart.Dock = DockStyle.None;

chart.Location = new Point(50, 10);

chart.Titles.Add("Популярность аудиторий"); // Добавляем название диаграммы

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 popularity)

{

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

chart.Series["Кол-во"].Points[pointIndex].IsValueShownAsLabel = false; // Скрываем значение

if (!string.IsNullOrEmpty(item.Key))

{

chart.Series["Кол-во"].Points[pointIndex].AxisLabel = item.Key; // Добавляем метку к каждому столбцу, если ключ не пустой

}

}

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;

frmAnalysis.Controls.Add(chart); // Добавляем диаграмму на форму frmAnalysis

}

else

{

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

}

}

## **4.9.3 Показывать на столбчатой диаграмме количество пожеланий указанным преподавателем аудиторий за весь известный период**

Прежде чем строить данную диаграмму пользователю необходимо все файлы WORD преобразовать в формат CSV, а после ввести интересующего преподавателя в поле TextBox в frmAnalysis. Покажем диаграмму «Количество пожеланий указанным преподавателем аудиторий за весь известный период» на рисунке 34.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 34 – Диаграмма «Количество пожеланий указанным преподавателем аудиторий за весь известный период»

**Листинг кода, реализующий построение диаграмм «Количество пожеланий указанным преподавателем аудиторий за весь известный период»**

private void UpdateChartByTeacherName(string teacherName)

{

string folderPath = @"C:\Users\Екатерина\source\repos\курсовая работа 2.0\bin\Debug\csv";

if (Directory.Exists(folderPath))

{

// Инициализация словаря вне цикла по файлам

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

// Перебираем все файлы CSV в папке

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

{

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);

}

}

// Обрабатываем данные из каждого файла

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

{

if (row.Length > 0 && row[0].Trim() == teacherName) // Проверяем фамилию преподавателя

{

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

if (popularity.ContainsKey(audience))

{

popularity[audience]++;

}

else

{

popularity.Add(audience, 1);

}

}

}

}

// Создание и настройка диаграммы

Chart chart = new Chart();

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

chart.Parent = frmAnalysis;

chart.Dock = DockStyle.None;

chart.Location = new Point(50, 30);

chart.Titles.Add("Популярность аудиторий"); // Добавляем название диаграммы

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;

// Добавление данных на диаграмму

foreach (var item in popularity)

{

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

}

// Дополнительные настройки диаграммы

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;

frmAnalysis.Controls.Add(chart); // Добавляем диаграмму на форму frmAnalysis

}

else

{

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

}

}

## **4.9.4** **Показывать на столбчатой диаграмме количество пожеланий указанной аудитории преподавателями за весь известный период**

Прежде чем строить данную диаграмму пользователю необходимо все файлы WORD преобразовать в формат CSV, а после ввести интересующую аудиторию в поле TextBox в frmAnalysis. Покажем диаграмму «Количество пожеланий указанной аудитории преподавателями за весь известный период» на рисунке 35.

**Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание**

Рисунок 35 – Диаграмма «Количество пожеланий указанной аудитории преподавателями за весь известный период»

## **Инструкция пользователя**

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

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

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 4 кнопками для выбора нужной диаграммы и одним TextBox для ввода параметра построения диаграммы.

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 37 – Дочерняя форма «Анализ данных»

Для построения двух первых диаграмм (две кнопки слева) пользователю необходимо в зависимости от промежутка построения диаграммы (за весь известный период, за указанный год, за указанный семестр), пользователю необходимо сконвертировать нужное количество файлов WORD в формат CSV.

Для построения остальных диаграмм пользователю необходимо все файлы WORD преобразовать в формат CSV, а после ввести либо интересующего преподавателя (3 кнопка слева) либо интересующую аудиторию (4 кнопка слева) в поле TextBox.

## **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 курсовая\_работа\_2.\_0

{

public partial class frmMain : Form

{

DataGridView dataGridView;

private string csvFilePath;

private Form frmAnalysis;

public frmMain()

{

InitializeComponent();

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

if (Directory.Exists(папкаПечатныхФорм))

{

string[] файлыWord = Directory.GetFiles(папкаПечатныхФорм, "\*.docx");

foreach (string файл in файлыWord)

{

string имяФайла = Path.GetFileNameWithoutExtension(файл);

cBWord.Items.Add(имяФайла);

}

}

else

{

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

}

}

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

{

if (cBWord.SelectedItem != null)

{

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

if (File.Exists(выбранныйФайл))

{

ConvertWordToCSV(выбранныйФайл);

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 выбранныйФайл)

{

try

{

using (WordprocessingDocument wordDocument = WordprocessingDocument.Open(выбранныйФайл, 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 папкаCSV = Path.Combine(Environment.CurrentDirectory, "csv");

if (!Directory.Exists(папкаCSV))

{

Directory.CreateDirectory(папкаCSV);

}

csvFilePath = Path.Combine(папкаCSV, $"{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 папкаExcel = Path.Combine(Environment.CurrentDirectory, "excel");

if (!Directory.Exists(папкаExcel))

{

Directory.CreateDirectory(папкаExcel);

}

string excelFilePath = Path.Combine(папкаExcel, $"{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 InsideHorizontalBorder() { 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 (IOException ex)

{

MessageBox.Show($"Ошибка при попытке открыть файл Word: {ex.Message}", "Ошибка");

}

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(); // Создаем форму frmAnalysis

frmAnalysis.Text = "Анализ данных";

frmAnalysis.Size = new Size(800, 600);

Button btnPopulAll = new Button();

btnPopulAll.Text = "Построить диаграмму 'Популярность заявок на аудитории всех упомянутых преподавателей'";

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

btnPopulAll.Location = new Point(30, 450);

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

frmAnalysis.Controls.Add(btnPopulAll);

Button btnPopul = new Button();

btnPopul.Text = "Построить диаграмму 'Популярность заявок на аудитории первых из упомянутых преподавателей'";

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

btnPopul.Location = new Point(220, 450);

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

frmAnalysis.Controls.Add(btnPopul);

TextBox txtTeacher = new TextBox();

txtTeacher.Location = new Point(505, 420);

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

frmAnalysis.Controls.Add(txtTeacher);

Button btnWish = new Button();

btnWish.Text = "Построить диаграмму 'Количество пожеланий указанным преподавателем аудиторий за весь известный период'";

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

btnWish.Location = new Point(410, 450);

btnWish.Click += (s, ev) => { UpdateChartByTeacherName(txtTeacher.Text); }; // Передаем значение из TextBox в метод

frmAnalysis.Controls.Add(btnWish);

Button btnAuditor = new Button();

btnAuditor.Text = "Построить диаграмму 'Количество пожеланий указанной аудитории преподавателями за весь известный период'";

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

btnAuditor.Location = new Point(600, 450);

btnAuditor.Click += (s, ev) => { UpdateChartByTeacherAuditor(txtTeacher.Text); }; // Передаем значение из TextBox в метод

frmAnalysis.Controls.Add(btnAuditor);

frmAnalysis.Show();

}

else

{

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

}

}

private void UpdateChartByTeacherAuditor(string teacherAuditor)

{

string folderPath = @"C:\Users\Екатерина\source\repos\курсовая работа 2.0\bin\Debug\csv";

if (Directory.Exists(folderPath))

{

// Инициализация словаря вне цикла по файлам

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

// Перебираем все файлы CSV в папке

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

{

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);

}

}

// Обрабатываем данные из каждого файла

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

{

if (row.Length > 0 && row[row.Length - 1].Trim() == teacherAuditor) // Проверяем аудиторию

{

string teacherName = row[0].Trim(); // Получаем имя преподавателя

if (popularity.ContainsKey(teacherName))

{

popularity[teacherName]++;

}

else

{

popularity.Add(teacherName, 1);

}

}

}

}

// Создание и настройка диаграммы

Chart chart = new Chart();

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

chart.Parent = frmAnalysis;

chart.Dock = DockStyle.None;

chart.Location = new Point(50, 30);

chart.Titles.Add("Популярность аудиторий"); // Добавляем название диаграммы

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;

// Добавление данных на диаграмму

foreach (var item in popularity)

{

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

}

// Дополнительные настройки диаграммы

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;

frmAnalysis.Controls.Add(chart); // Добавляем диаграмму на форму frmAnalysis

}

else

{

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

}

}

private void UpdateChartByTeacherName(string teacherName)

{

string folderPath = @"C:\Users\Екатерина\source\repos\курсовая работа 2.0\bin\Debug\csv";

if (Directory.Exists(folderPath))

{

// Инициализация словаря вне цикла по файлам

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

// Перебираем все файлы CSV в папке

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

{

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);

}

}

// Обрабатываем данные из каждого файла

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

{

if (row.Length > 0 && row[0].Trim() == teacherName) // Проверяем фамилию преподавателя

{

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

if (popularity.ContainsKey(audience))

{

popularity[audience]++;

}

else

{

popularity.Add(audience, 1);

}

}

}

}

// Создание и настройка диаграммы

Chart chart = new Chart();

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

chart.Parent = frmAnalysis;

chart.Dock = DockStyle.None;

chart.Location = new Point(50, 30);

chart.Titles.Add("Популярность аудиторий"); // Добавляем название диаграммы

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;

// Добавление данных на диаграмму

foreach (var item in popularity)

{

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

}

// Дополнительные настройки диаграммы

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;

frmAnalysis.Controls.Add(chart); // Добавляем диаграмму на форму frmAnalysis

}

else

{

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

}

}

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

{

string folderPath = @"C:\Users\Екатерина\source\repos\курсовая работа 2.0\bin\Debug\csv";

if (Directory.Exists(folderPath))

{

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

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

{

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);

}

}

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

{

if (row.Length > 0)

{

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

if (popularity.ContainsKey(audience))

{

popularity[audience]++;

}

else

{

popularity.Add(audience, 1);

}

}

}

}

Chart chart = new Chart();

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

chart.Parent = frmAnalysis;

chart.Dock = DockStyle.None;

chart.Location = new Point(50, 10);

chart.Titles.Add("Популярность аудиторий"); // Добавляем название диаграммы

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 popularity)

{

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

chart.Series["Кол-во"].Points[pointIndex].IsValueShownAsLabel = false; // Скрываем значение

if (!string.IsNullOrEmpty(item.Key))

{

chart.Series["Кол-во"].Points[pointIndex].AxisLabel = item.Key; // Добавляем метку к каждому столбцу, если ключ не пустой

}

}

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;

frmAnalysis.Controls.Add(chart); // Добавляем диаграмму на форму frmAnalysis

}

else

{

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

}

}

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

{

string folderPath = @"C:\Users\Екатерина\source\repos\курсовая работа 2.0\bin\Debug\csv";

if (Directory.Exists(folderPath))

{

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

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

{

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

using (var reader = new StreamReader(filePath))

{

string line;

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

{

// Разделяем строку по "!", удаляя пустые значения

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

data.Add(parts);

}

}

var top5Aud = data.Skip(1).Take(24).ToList();

foreach (var row in top5Aud)

{

if (row.Length > 0) // Проверяем, есть ли хотя бы одно значение в строке

{

string audience = row[row.Length - 1].Trim(); // Получаем значение аудитории из последнего столбца

if (popularity.ContainsKey(audience))

{

popularity[audience]++;

}

else

{

popularity.Add(audience, 1);

}

}

}

}

Chart chart = new Chart();

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

chart.Parent = frmAnalysis;

chart.Dock = DockStyle.None;

chart.Location = new Point(50, 10);

chart.Titles.Add("Популярность аудиторий"); // Добавляем название диаграммы

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 popularity)

{

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

chart.Series["Кол-во"].Points[pointIndex].IsValueShownAsLabel = false; // Скрываем значение

if (!string.IsNullOrEmpty(item.Key))

{

chart.Series["Кол-во"].Points[pointIndex].AxisLabel = item.Key; // Добавляем метку к каждому столбцу, если ключ не пустой

}

}

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;

frmAnalysis.Controls.Add(chart); // Добавляем диаграмму на форму frmAnalysis

}

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 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.