|  |  |
| --- | --- |
| **Российский университет транспорта (МИИТ)**  **Институт транспортной техники и систем управления**  **Кафедра «Управления и защита информации»** | |
| **Курсовая работа**  **на тему**  **«Разработка программного обеспечения конвертации печатных форм»**  **по дисциплине**  **«Методы программирования»** | |
|  | Выполнил:  Студент группы ТКИ-342,  Потапов А.К. |
|  | Проверил:  Доцент кафедры УиЗИ, к.т.н.,  Сафронов А.И. |
|  | |
| Москва – 2024 | |

Оглавление

[Введение 3](#_Toc166444848)

[Цель работы 4](#_Toc166444849)

[Задачи работы 5](#_Toc166444850)

[1. Содержательная часть 6](#_Toc166444851)

[1.1. Анализ печатной формы 6](#_Toc166444852)

[1.2. Описание предназначения печатной формы 7](#_Toc166444853)

[1.3. Разнообразие печатных форм и выявление различий 8](#_Toc166444854)

[1.4. Унификация печатных форм для удобства обработки данных 9](#_Toc166444855)

[1.5. Разработка экранной формы для импорта сведений из печатной формы 10](#_Toc166444856)

[1.6. Разработка функций экспорта сведений из печатной формы в файл текстовый формата CSV 12](#_Toc166444857)

[1.7. Разработка функций импорта сведений из оригинального и изменённого текстовых файлов формата CSV с воссозданием Excel 14](#_Toc166444858)

[1.8. Разработка функций экспорта сведений из экранной формы в Word 16](#_Toc166444859)

[1.9. Графический анализ данных, содержащихся в печатной форме 18](#_Toc166444860)

[2. Инструкция пользователя 21](#_Toc166444861)

[3. Заключение 22](#_Toc166444862)

[3.1. Создание программы 22](#_Toc166444863)

[3.2. Листинг программы 22](#_Toc166444864)

[3.3. Тестирование программы 30](#_Toc166444865)

[4. Выводы 38](#_Toc166444866)

[Библиографический список 39](#_Toc166444867)

Введение

В данной работе рассмотрена разработка программного обеспечения, способного обрабатывать печатные формы установленного формата, содержащие расписание занятий. Программа позволит пользователю выполнить предпросмотр табличного расписания, его конвертацию в различные форматы, например, формат CSV (Comma-Separated Values), Word. Также будет реализовано построение расписание в виде таблицы, основанное на CSV файле. Программа будет содержать модуль для просмотра блоков наиболее загруженных дней и аудиторий в году и по семестрам.

Цель работы

Целью данной работы является разработка программного обеспечения, способного обрабатывать формы с расписанием, содержащимся в файлах .xls, уметь конвертировать их в .csv, .docx. по файлу .csv программа должна уметь воссоздать таблицу .xls, а также построить графики наиболее загруженных дней и аудиторий.

Задачи работы

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 Excel – Word.

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».

7. Показывать столбчатую диаграмму максимально насыщенных занятиями рабочих дней за весь период.

8. Показывать столбчатую диаграмму максимально насыщенных занятиями рабочих дней за каждый семестр.

9. Показывать столбчатую диаграмму популярности аудиторий за весь период.

10. Показывать столбчатую диаграмму популярности аудиторий за каждый семестр.

1. Содержательная часть
   1. Анализ печатной формы

Печатная форма представляет из себя расписание, заданное в таблице формата .xls. Общий вид расписания приведён на рисунке 1. Все печатные формы подобны друг другу. Различия – лишние переносы на новую строку в ячейках. От них стоит избавиться, чтобы избежать избыточности в представлении итогового документа. Стоит обратить внимание на запятые, содержащиеся в столбцах со временем, так как они будут мешать корректной обработке при конвертации в CSV файл.

|  |
| --- |
|  |
| 1. Общий вид формы |

* 1. Описание предназначения печатной формы

Данная печатная форма содержит расписание занятий по строго установленному формату. Первый столбец содержит фамилию, имя и отчество преподавателя, разделённых запятыми. Второй столбец обозначает номер семестра и год, соответствующий ему. Третий столбец отвечает за номер недели в семестре, тогда как четвёртый – за день недели. В пятом столбце находятся данные о типе занятия, проводимого по расписанию. Шестой столбец имеет название преподаваемого предмета, седьмой – время его проведения, восьмой – аудиторию, в которой проводится занятие. Наконец, в девятом столбце находятся данные о том, у какой группы идёт занятие.

Первая строка таблицы является заголовочной, поэтому для реализации некоторых пунктов нашей программы не будем учитывать её.

* 1. Разнообразие печатных форм и выявление различий

Как было сказано ранее, формы практически идентичны друг другу. Особое внимание стоит уделить тому, что некоторые ячейки содержат множество знаков переноса на следующую строку, что негативно скажется на внешнем виде при переносе в CSV файл или Word таблицу.

Отдельно необходимо рассмотреть вопрос запятых, так как CSV файл по умолчанию использует их, как разделитель. Стоит отметить, что при русской локализации CSV файл использует в качестве разделителя «;», но при этом запятые также негативно сказываются на построении файла.

* 1. Унификация печатных форм для удобства обработки данных

Для унификации печатных форм в нашей будущей программе будет реализовано две функции. Первая будет отвечать за удаление символов перехода на следующую строку и символов возврата каретки, которые часто используются в Windows файлах. Эта функция будет использоваться при любой работе с данными в составе других функций для улучшения читаемости.

Вторая функция – обрамление кавычками строк. Это позволит избежать проблемы с запятыми, так как текст, обрамлённый кавычками, воспринимается файлом CSV целиком, как обычный текст. Этот метод будет использован в составе функций, задействующих конвертацию в\из CSV файлов, так как в противном случает он вызовет избыточность.

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

В данной пункте рассмотрим написание части программы, которая позволит выполнить предпросмотр печатной формы в экранной форме Windows. Опишем методы будущего кода на следующих рисунках:

|  |
| --- |
|  |
| 1. Схема Петри формирования предпросмотра |

Опишем действия схемы на рисунке 3.

1

3

4

5

6

|  |
| --- |
| 2 |
| 1. Схемы методов предпросмотра |

По результату будет получена программа, в которой можно будет выполнить предпросмотр документа формата .xls или .xlsx. В заключительной части, когда будет написана программа, отобразим работу на примере трёх тестов: загрузка файла и его предпросмотр, переключение на другую строку, и попытка выйти за границу строки.

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

В этом этапе используем загрузку файла и добавим его конвертацию путём записи данных таблицы в файл. Не забудем обработать переносы строк и запятые. Попутно выполним запись данных в общий файл, который позднее будет использован для построения диаграмм. Отобразим данный пункт в нотации сетей Петри на следующем рисунке.

|  |
| --- |
| Нажата кнопка  Нажата кнопка  Нажата кнопка |
| 1. Сеть Петри конвертации в CSV |

Обрисуем схему нового метода на следующем рисунке:

|  |
| --- |
|  |
| 1. Схема метода btnExcelCSV\_Click |

Для теста выберем конвертацию файла в CSV.

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

В данной пункте разработаем обратное преобразование из CSV в Excel. Для этого реализуем ещё один метод, который позволит нам произвести подобную конвертацию. Опишем его на схеме Петри на Рисунке 6.

|  |
| --- |
| Нажата кнопка  «X»  Нажата кнопка  Нажата кнопка |
| 1. Сеть Петри обратной конвертации |

Опишем новый метод, который задействует при работе, на картинке ниже.

|  |
| --- |
|  |
| 1. Описание метода btnCSVExcel\_Click |

Как и в прошлом примере, в качестве теста будет выбрана конвертация файла, а конкретно, CSV файл в файл .xlsx.

* 1. Разработка функций экспорта сведений из экранной формы в Word

В данном пункте реализуем конвертацию из Excel в Word и из CSV в Word. Для удобной работы с данными будем позднее использовать библиотеку LINQ. Общую схему в нотациях сетей Петри опишем на рисунке 8.

|  |
| --- |
| Нажата кнопка  «X»  Нажата кнопка  Нажата кнопка  Нажата кнопка |
| 1. Схема Петри для конвертации в Word |

Как и в прошлых вариантах, опишем её методы с помощью схемы:

|  |
| --- |
|  |
| 1. Схемы конвертации в Word документ |

Для тестов также выберем два пункта: конвертацию из Excel и из CSV в Word документ.

* 1. Графический анализ данных, содержащихся в печатной форме

В данной части работы необходимо выполнить построение диаграмм, отражающих следующие пункты: наиболее задействованные дни недели за всё время, наиболее задействованные дни по семестрам, наиболее задействованные аудитории за всё время, наиболее задействованные аудитории по семестрам. Для этого напишем свой класс, который позволит нам строить диаграммы по заранее заданным методам. Опишем общую структуру в нотации Петри.

|  |
| --- |
|  |
| 1. Схема Петри для графического анализа |

Выведем реализацию методов, которые вызываются в данной программе, на рисунке 11, 12, 13.

|  |
| --- |
| 1  2  3  3  2  1  2  3  1  3  1  2 |
| 1. Основные методы для графического анализа |
| 2.9  2.8  2.7  2.6  2.5  2.4  2.3  2.2  2.1 |
| 1. Вспомогательный метод OpenChartForm(type) |
| 2.3.10  2.3.9  2.3.8  2.3.7  2.3.6  2.3.5  2.3.4  2.3.3  2.3.2  2.5.10  2.5.9  2.5.8  2.5.7  2.5.6  2.5.5  2.5.4  2.5.3  2.5.2 |
| 1. Вспомогательные методы отрисовки |

2.3.1

2.5.1

Для теста функции, разумеется, будет выбрано построение всех возможных четырёх графиков. Для построения будет использовать датасет, собранный из всех данных печатных форм.

1. Инструкция пользователя

Для начала работы с программой необходимо нажать кнопку «Choose File», после чего появится диалоговое окно, содержащее возможность выбора файлов .xls, .xlsx, .csv. В зависимости от выбранного типа файла будут предоставлены возможности по взаимодействию.

Если выбран файл электронных таблиц, то первым делом необходимо считать из него данные. Делается это путём нажатия на кнопку «Read Excel». После этого действия появится предпросмотр файла, равно как и возможность перемещения по его строкам. Дополнительно станут доступны кнопки, отвечающие за конвертацию файла Excel таблицы в CSV (Excel -> CSV) или Word (Excel->Word) формат.

Если исходно выбран файл .csv, то пользователю становятся доступны кнопки «CSV->Excel» и «CSV->Word» для соответствующих конвертаций.

При нажатии кнопки «Excel->CSV» данные из печатной формы также заносятся в общую базу, по которой будет производиться построение графиков. Если такая база существует, пользователю доступны кнопки «Day Total», «Day Sem», «Class Total», «Class Sem», отвечающие за построение графиков дней по популярности за всё время, дней по популярности по семестрам, аудиторий по популярности за всё время, дней по популярности по семестрам. График будет открыт в новом окне.

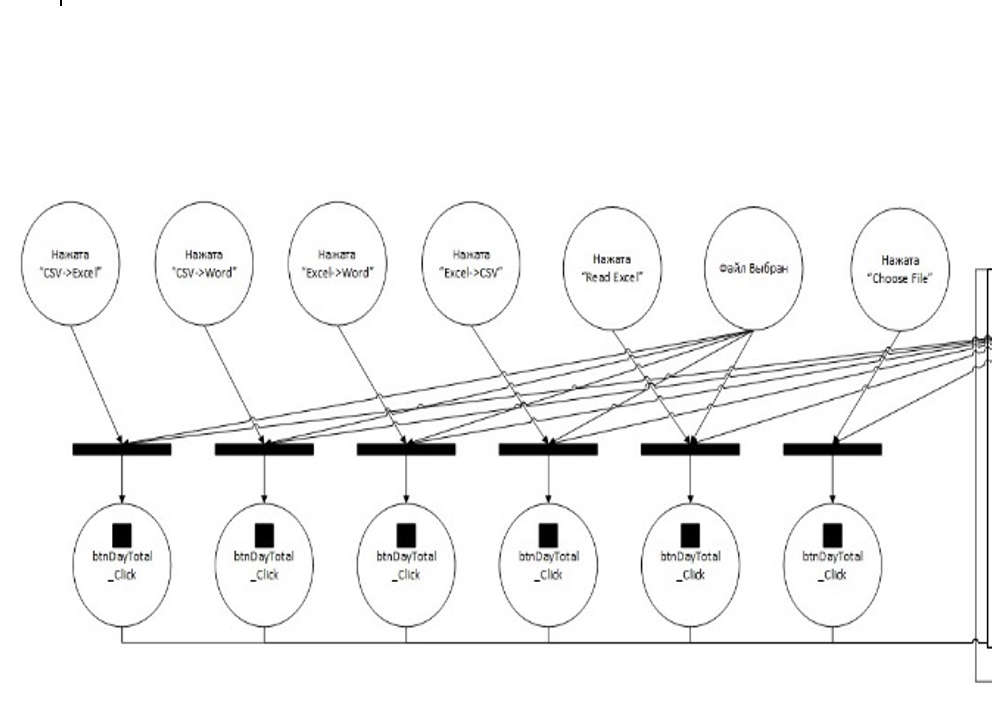
Также существуют две кнопки: «Info» – отображающая краткую версию данной справки, и кнопка «Delete D Pack», отвечающая за удаление общей базы данных.

1. Заключение

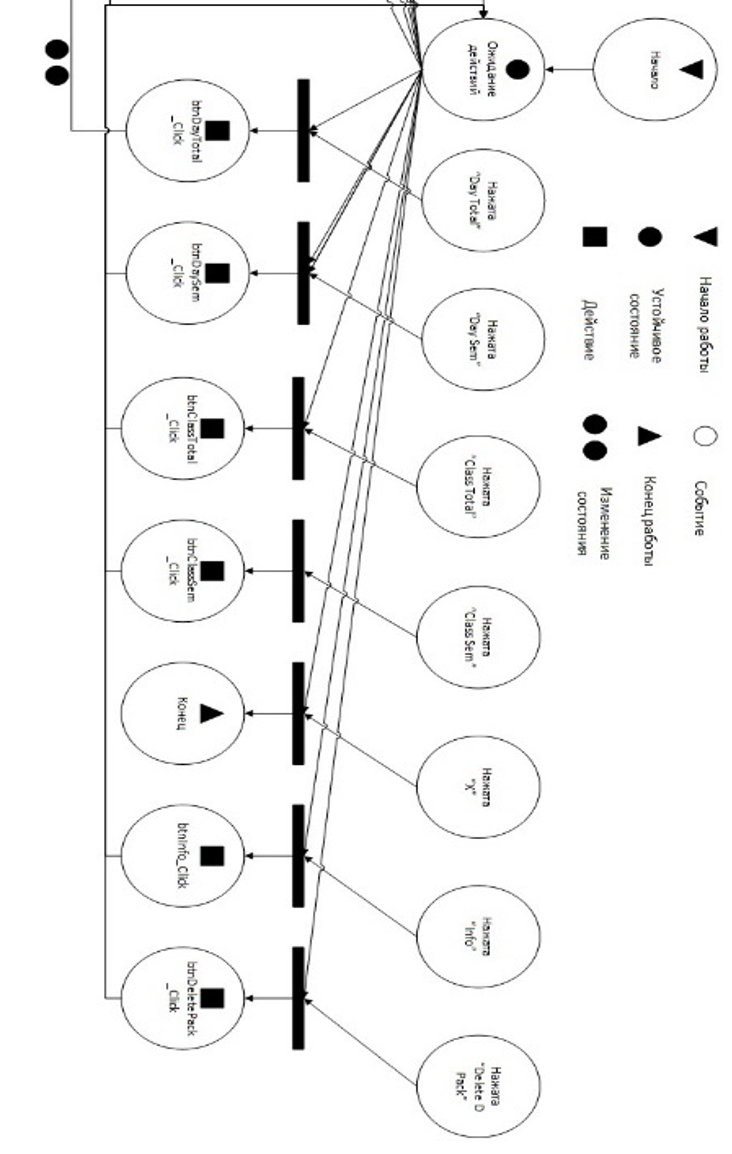
Произведём окончательную сборку того, что было реализовано ранее, напишем код программы, и произведём ранее заданные тесты.

* 1. Создание программы

Обрисуем общую схему Петри на следующей картинке:



|  |
| --- |
|  |
| 1. Схема Петри готового решения(Начало) |

  
Рисунок 14 - Схема Петри готового решения (Конец)

Все методы, используемые здесь, были обрисованы ранее в виде схем. Исключение составляют методы «btnInfo\_Click» и «btnDeletePack\_Click». Эти методы кратки и не требуют описания в виде схем. Первый выводит окно с краткой справкой, второй удаляет файл базы данных, если обнаруживает его.

* 1. Листинг программы

Далее будет приведён листинг кода программы, написанной по ранее построенным схемам.

C# .NetFramework v4.5

using Microsoft.Office.Interop.Excel;

using Microsoft.Office.Interop.Word;

using System;

using System.Collections.Generic;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Windows.Forms.DataVisualization.Charting;

using Ch = System.Windows.Forms.DataVisualization.Charting;

using Chart = System.Windows.Forms.DataVisualization.Charting.Chart;

using ChartArea = System.Windows.Forms.DataVisualization.Charting.ChartArea;

using E = Microsoft.Office.Interop.Excel;

using Series = System.Windows.Forms.DataVisualization.Charting.Series;

using W = Microsoft.Office.Interop.Word;

namespace O\_Course\_Work

{

public partial class frmMain : Form

{

bool isFirstFile = true;

string fileName;

int selectedRow;

string rowStr;

const char delimeter = '@';

E.Application excel;

Workbook wb;

Worksheet ws;

E.Range usedRange;

System.Windows.Forms.TextBox[] textBoxes;

public frmMain()

{

InitializeComponent();

}

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

{

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

{

fileName = ofdStart.FileName;

tbOpenedFile.Text = Path.GetFileName(fileName);

btnNext.Enabled = false;

btnPrev.Enabled = false;

btnExcelWord.Enabled = false;

btnExcelCsv.Enabled = false;

foreach (var textbox in textBoxes)

{

textbox.Text = "";

}

if (Path.GetExtension(fileName) == ".xls" || Path.GetExtension(fileName) == ".xlsx")

{

btnReadData.Enabled = true;

btnCSVWord.Enabled = false;

btnCSVExcel.Enabled = false;

}

else

{

btnReadData.Enabled = false;

btnNext.Enabled = false;

btnPrev.Enabled = false;

btnExcelWord.Enabled = false;

btnExcelCsv.Enabled = false;

btnCSVExcel.Enabled = true;

btnCSVWord.Enabled = true;

}

}

}

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

{

excel = new E.Application();

wb = excel.Workbooks.Open(fileName);

ws = (Worksheet)wb.ActiveSheet;

usedRange = ws.UsedRange;

FillTableWithRow(selectedRow);

btnPrev.Enabled = true;

btnNext.Enabled = true;

btnExcelCsv.Enabled = true;

btnExcelWord.Enabled = true;

}

void GetRowStr(int row)

{

rowStr = "";

int i = 1;

while (ws.Cells[row, i].Value != null)

{

rowStr += ws.Cells[row, i].Value;

rowStr += delimeter;

++i;

}

}

void FillTableWithRow(int row)

{

GetRowStr(row);

string[] rowStrArray = rowStr.Split(delimeter);

short i = 0;

foreach (System.Windows.Forms.TextBox t in textBoxes)

{

t.Text = rowStrArray[i++].ToString();

}

}

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

{

if (ws.Cells[selectedRow + 1, 1].Value != null)

{

selectedRow++;

FillTableWithRow(selectedRow);

}

else

{

MessageBox.Show("Reached last item!");

}

}

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

{

if (selectedRow > 2)

{

selectedRow--;

FillTableWithRow(selectedRow);

}

else

{

MessageBox.Show("Reached first item!");

}

}

private void frmMain\_Load(object sender, EventArgs e)

{

textBoxes = new System.Windows.Forms.TextBox[]

{

tbFIO, tbSemNum, tbWeekNum, tbWeekDay, tbLessonType, tbLessonName, tbTime, tbClassroom, tbGroup

};

selectedRow = 2;

btnReadData.Enabled = false;

btnCSVExcel.Enabled = false;

btnCSVWord.Enabled = false;

btnExcelWord.Enabled = false;

btnExcelCsv.Enabled = false;

btnNext.Enabled = false;

btnPrev.Enabled = false;

if (!File.Exists(System.Windows.Forms.Application.StartupPath + "\\" + "DataPack.csv"))

{

btnDayTotal.Enabled = false;

btnDaySem.Enabled = false;

btnClassTotal.Enabled = false;

btnClassSem.Enabled = false;

}

}

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

{

try

{

wb.Close();

excel.Quit();

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

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

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

}

catch (Exception ex)

{

}

finally

{

System.Windows.Forms.Application.Exit();

}

}

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

{

string csvFilePath = System.Windows.Forms.Application.StartupPath + "\\" + Path.GetFileNameWithoutExtension(fileName) + ".csv";

string csvDataPack = System.Windows.Forms.Application.StartupPath + "\\" + "DataPack.csv";

using (StreamWriter sw = new StreamWriter(csvFilePath, false, Encoding.UTF8))

{

int i = 1, j = 1;

while (ws.Cells[i, 1].Value != null)

{

while (ws.Cells[i, j].Value != null)

{

sw.Write("\"" + Convert.ToString(ws.Cells[i, j].Value).Replace("\n", "").Replace("\r", "") + "\"");

if (ws.Cells[i, j++ + 1].Value != null)

{

sw.Write(";");

}

}

++i;

j = 1;

sw.WriteLine();

}

}

using (StreamWriter sw = new StreamWriter(csvDataPack, true, Encoding.UTF8))

{

int i = 2, j = 1;

while (ws.Cells[i, 1].Value != null)

{

while (ws.Cells[i, j].Value != null)

{

sw.Write(Convert.ToString(ws.Cells[i, j].Value).Replace("\n", "").Replace("\r", ""));

if (ws.Cells[i, j++ + 1].Value != null)

{

sw.Write(";");

}

}

++i;

j = 1;

sw.WriteLine();

}

btnDayTotal.Enabled = true;

btnDaySem.Enabled = true;

btnClassTotal.Enabled = true;

btnClassSem.Enabled = true;

}

}

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

{

excel = new E.Application();

wb = excel.Workbooks.Add();

ws = wb.Sheets[1];

string excelPath = System.Windows.Forms.Application.StartupPath + "\\" + Path.GetFileNameWithoutExtension(fileName) + ".xlsx";

var rows = File.ReadAllLines(fileName).Select(row => row.Split(';')).ToList();

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

{

for (int j = 0; j < rows[i].Length; ++j)

{

ws.Cells[i + 1, j + 1] = Convert.ToString(rows[i][j]).Replace("\"", "");

}

}

if (File.Exists(excelPath))

{

File.Delete(excelPath);

}

wb.SaveAs(excelPath);

wb.Close();

excel.Quit();

}

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

{

string csvFilePath = System.Windows.Forms.Application.StartupPath + "\\" + Path.GetFileNameWithoutExtension(fileName) + ".csv";

string wordFilePath = System.Windows.Forms.Application.StartupPath + "\\" + Path.GetFileNameWithoutExtension(fileName) + ".docx";

W.Application word = new W.Application();

Document doc = word.Documents.Add();

var rows = File.ReadAllLines(csvFilePath)

.Select(line => line.Replace("\"", "")).Select(line => line.Split(';'))

.ToList();

Table = doc.Tables.Add(doc.Range(0, 0), rows.Count, rows[0].Length);

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

{

Row = table.Rows[i + 1];

row.AllowBreakAcrossPages = 0;

for (int j = 0; j < rows[i].Length; j++)

{

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

table.Cell(i + 1, j + 1).Range.Borders[WdBorderType.wdBorderLeft].LineStyle = WdLineStyle.wdLineStyleSingle;

table.Cell(i + 1, j + 1).Range.Borders[WdBorderType.wdBorderRight].LineStyle = WdLineStyle.wdLineStyleSingle;

table.Cell(i + 1, j + 1).Range.Borders[WdBorderType.wdBorderTop].LineStyle = WdLineStyle.wdLineStyleSingle;

table.Cell(i + 1, j + 1).Range.Borders[WdBorderType.wdBorderBottom].LineStyle = WdLineStyle.wdLineStyleSingle;

}

}

if (File.Exists(wordFilePath))

{

File.Delete(wordFilePath);

}

doc.SaveAs(wordFilePath);

doc.Close();

word.Quit();

}

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

{

string wordFilePath = System.Windows.Forms.Application.StartupPath + "\\" + Path.GetFileNameWithoutExtension(fileName) + ".docx";

W.Application word = new W.Application();

Document doc = word.Documents.Add();

Table = doc.Tables.Add(doc.Range(0, 0), usedRange.Rows.Count, usedRange.Columns.Count);

int i = 1, j = 1;

while (ws.Cells[i, 1].Value != null)

{

Row = table.Rows[i];

row.AllowBreakAcrossPages = 0;

while (ws.Cells[i, j].Value != null)

{

table.Cell(i, j).Range.Text = Convert.ToString(ws.Cells[i, j].Value).Replace("\n", "").Replace("\r", "");

table.Cell(i, j).Range.Borders[WdBorderType.wdBorderLeft].LineStyle = WdLineStyle.wdLineStyleSingle;

table.Cell(i, j).Range.Borders[WdBorderType.wdBorderRight].LineStyle = WdLineStyle.wdLineStyleSingle;

table.Cell(i, j).Range.Borders[WdBorderType.wdBorderTop].LineStyle = WdLineStyle.wdLineStyleSingle;

table.Cell(i, j).Range.Borders[WdBorderType.wdBorderBottom].LineStyle = WdLineStyle.wdLineStyleSingle;

++j;

}

++i;

j = 1;

}

if (File.Exists(wordFilePath))

{

File.Delete(wordFilePath);

}

doc.SaveAs(wordFilePath);

doc.Close();

word.Quit();

}

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

{

if (File.Exists(System.Windows.Forms.Application.StartupPath + "\\" + "DataPack.csv"))

{

File.Delete(System.Windows.Forms.Application.StartupPath + "\\" + "DataPack.csv");

btnDayTotal.Enabled = false;

btnDaySem.Enabled = false;

btnClassTotal.Enabled = false;

btnClassSem.Enabled = false;

}

}

private void OpenChartForm(int chartType)

{

ChartForm = new ChartForm(chartType);

chartForm.ShowDialog();

}

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

{

OpenChartForm(1);

}

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

{

OpenChartForm(2);

}

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

{

OpenChartForm(3);

}

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

{

OpenChartForm(4);

}

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

{

string message = "Сначала необходимо выбрать файл кнопкой \"Choose file\", ";

message += "Появятся возможности в зависимости от типа файла. Если файл ";

message += "Excel, то необходимо считать данные для работы с ним. Нажмите ";

message += "\"Read Excel\" для этого. Появятся возможности \"Excel -> CSV\" ";

message += "и \"Excel -> Word\", позволяющие выполнить соответствующие конвертации. ";

message += "При нажатии \"Excel -> CSV\" также происходит заполнение общей базы ";

message += "данных CSV, её удаление можно произвести по нажатию \"Delete D Pack\". ";

message += "По этой базе данных выполняется построение графиков кнопками ";

message += "\"Day Total\", \"Day Sem\", \"Class Total\", \"Class Sem\", которые ";

message += "выводят графики популярности дней за всё время и по семестрам, а также ";

message += "популярность аудиторий за всё время и по семестрам соответственно. ";

message += "Кнопка \"Info\" выводит эту подсказку. Нажатие \">>>\" и \"<<<\" ";

message += "позволит в предпросмотре файла переходить по строкам. Если исходно ";

message += "выбран CSV файл, то можно конвертировать его в Excel кнопкой ";

message += "\"CSV -> Excel\" или в Word кнопкой \"CSV -> Word\". ";

MessageBox.Show(message, "Информация", MessageBoxButtons.OK, MessageBoxIcon.Information,MessageBoxDefaultButton.Button1);

}

}

public partial class ChartForm : Form

{

public ChartForm(int chartType)

{

Size = new Size(820, 625);

if (chartType == 1)

{

DisplayChartTotal(3);

Text = "Наибольшая загруженность дней за всё время";

}

else if (chartType == 2)

{

DisplayChartSemester(3);

Text = "Наибольшая загруженность дней по семестрам";

}

else if (chartType == 3)

{

DisplayChartTotal(7);

Text = "Наибольшая загруженность аудиторий за всё время";

}

else if (chartType == 4)

{

DisplayChartSemester(7);

Text = "Наибольшая загруженность аудиторий по семестрам";

}

}

private void DisplayChartTotal(int criteria)

{

string filePath = System.Windows.Forms.Application.StartupPath + "\\DataPack.csv";

List<string> lines = File.ReadAllLines(filePath).ToList();

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

foreach (string line in lines)

{

string[] parts = line.Split(';');

string label = parts[criteria];

if (data.ContainsKey(label))

{

++data[label];

}

else

{

data.Add(label, 1);

}

}

Chart = new Chart();

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

chart.ChartAreas.Add(new ChartArea());

chart.ChartAreas[0].AxisX.LabelStyle.Angle = -90;

chart.ChartAreas[0].AxisX.IntervalAutoMode = IntervalAutoMode.VariableCount;

chart.ChartAreas[0].AxisX.IsLabelAutoFit = true;

Series = new Series(criteria == 3 ? "Дни недели" : "Аудитории");

series.ChartType = SeriesChartType.Column;

var sortedData = data.OrderByDescending(x => x.Value);

foreach (var kvp in sortedData)

{

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

}

chart.Series.Add(series);

Ch.Legend legend = new Ch.Legend();

legend.Enabled = true;

legend.Title = "Легенда диаграммы";

chart.Legends.Add(legend);

Controls.Add(chart);

}

private void DisplayChartSemester(int criteria)

{

string filePath = System.Windows.Forms.Application.StartupPath + "\\DataPack.csv";

List<string> lines = File.ReadAllLines(filePath).ToList();

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

foreach (string line in lines)

{

string[] parts = line.Split(';');

string label\_sem = parts[1];

string label\_day = parts[criteria];

if (!dictDict.ContainsKey(label\_sem))

{

dictDict[label\_sem] = new Dictionary<string, int>();

}

if (dictDict[label\_sem].ContainsKey(label\_day))

{

++dictDict[label\_sem][label\_day];

}

else

{

dictDict[label\_sem].Add(label\_day, 1);

}

}

Chart = new Chart();

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

chart.ChartAreas.Add(new ChartArea());

chart.ChartAreas[0].AxisX.LabelStyle.Angle = -90;

chart.ChartAreas[0].AxisX.IntervalAutoMode = IntervalAutoMode.VariableCount;

chart.ChartAreas[0].AxisX.IsLabelAutoFit = true;

foreach (var outerKey in dictDict.Keys)

{

Series = new Series(outerKey);

series.ChartType = SeriesChartType.Column;

var sortedValues = dictDict[outerKey].OrderByDescending(x => x.Value).ToList();

foreach (var entry in sortedValues)

{

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

}

chart.Series.Add(series);

}

Ch.Legend legend = new Ch.Legend();

legend.Enabled = true;

legend.Title = "Легенда диаграммы";

chart.Legends.Add(legend);

Controls.Add(chart);

}

}

}

* 1. Тестирование программы

Запустим нашу программу. На следующем рисунке отобразим главный интерфейс.

|  |
| --- |
|  |
| 1. Главное окно программы |

В начальном состоянии доступно только несколько кнопок. Сделаем тесты. Выберем файл и считаем его.

|  |
| --- |
|  |
| 1. Предварительный просмотр |

Теперь выполним переключение строки вперёд на рисунке 17.

|  |
| --- |
|  |
| 1. Переключение строки |

Наконец, попробуем выйти за границы допустимых строк на картинке:

|  |
| --- |
|  |
| 1. Выход за пределы границ |

Как видим, тесты первого блока успешно пройдены.

Теперь попробуем конвертировать файл из Excel формата в CSV на рисунках 19 и 20.

|  |
| --- |
|  |
| 1. Вид файла после конвертации |
|  |
| 1. Вид файла в таблице |

Тест пройден успешно. Теперь выполним обратную конвертацию из CSV в Excel и посмотрим полученный файл на рисунке 21.

|  |
| --- |
|  |
| 1. Конвертация из CSV в Excel |

Как видим, файл успешно сконвертировался.

Произведём две следующих конвертации: Из Excel в Word и из CSV в Word. Отобразим результаты на следующих изображениях.

|  |
| --- |
|  |
| 1. Результат конвертации Excel в Word |
|  |
| 1. Результат конвертации CSV в Word |

Как можно видеть, результаты идентичны.

Наконец, выполним построение всех четырёх графиков на рисунках 24, 25, 26, 27.

|  |
| --- |
|  |
| 1. График дней за всё время |
|  |
| 1. График дней по семестрам |
|  |
| 1. График аудиторий за всё время |
|  |
| 1. График аудиторий по семестрам |

Как видим, графики строятся корректно.

Также выведем содержимое краткой справки по нажатию на кнопку «Info» на рисунке 28.

|  |
| --- |
|  |
| 1. Содержимое краткой справки |

Как видим, все пункты выполняются так, как от них ожидается.

1. Выводы

В ходе написания данной работы были получены навыки работы с экранными формами Windows посредством языка программирования C#. Освоены основные библиотеки, взаимодействие классов, элементы объектно‑ориентированного программирования.

Результатом работы стал также приобретённый навык построения схем Петри процессов, отточен навык построения схем, также часто именуемых блок-схемами. Построены эти схемы как для составляющих будущего решения, так и для всего в целом. Описаны методы, использованные в данной реализации программного обеспечения.

Основным результатом работы стала программа, способная обрабатывать печатные формы с расписанием, конвертировать их в другие форматы, производить предварительный просмотр. Программа способна строить графики для анализа данных, собранных в результате её эксплуатации.

Помимо указанных результатов, важным аспектом изучения предметной области анализа документов является также освоение методов автоматического извлечения информации из текстовых и структурированных документов. Этот процесс включает в себя не только распознавание текста, но и анализ его структуры, выявление ключевых элементов и связей между ними. Достижения в этой области позволяют автоматизировать процессы обработки больших объемов документации, улучшая эффективность работы и сокращая временные затраты. Таким образом, изучение методов анализа документов, их структур и содержимого не только обогащает навыки программирования и разработки программного обеспечения, но и открывает новые возможности для оптимизации бизнес-процессов и повышения продуктивности труда.

Библиографический список

1. Алексей Васильев Программирование на C# для начинающих [Текст] / Алексей Васильев – М.: Эксмо. – 2019 – 528 c.
2. Джозеф Албахари C# 9.0. Карманный справочник [Текст] / Джозеф Албахари – Москва: Диалектика. – 2021 – 256 c.
3. Учебник по языку C# 12 и платформе .NET 8 / [Электронный ресурс] // METANIT.COM : [сайт]. – URL: https://metanit.com/sharp/tutorial/ (дата обращения: 12.05.2024).
4. Руководство по С# – Управляемый язык .NET | Microsoft Learn / [Электронный ресурс] // Microsoft Learn : [сайт]. – URL: https://learn.microsoft.com/ru-ru/dotnet/csharp/ (дата обращения: 12.05.2024).