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

Поваляева А.В.

Топталова С.Д.

Вариант №2

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

Москва ­– 2024 г.

**Содержание**

[1. Введение 3](#_Toc168914775)

[2. Цель работы 3](#_Toc168914776)

[3. Задачи работы 3](#_Toc168914777)

[4. Содержательная часть 4](#_Toc168914778)

[4.1 Анализ печатной формы 4](#_Toc168914779)

[4.2 Описание предназначения печатной формы 4](#_Toc168914780)

[4.3 Разнообразие печатных форм и выявление различий 5](#_Toc168914781)

[4.4 Унификация печатных форм для удобства обработки данных 5](#_Toc168914782)

[4.5 Разработка экранной формы для импорта сведений из печатной формы 6](#_Toc168914787)

[4.6 Разработка функций экспорта сведений из печатной формы в файл текстовый формата CSV 8](#_Toc168914788)

[4.7 Разработка функций импорта сведений из оригинального и изменённого текстовых файлов формата CSV с воссозданием Word 11](#_Toc168914789)

[4.8 Разработка функций экспорта сведений из экранной формы в Excel 13](#_Toc168914790)

[4.9 Графический анализ данных, содержащихся в печатной форме 17](#_Toc168914792)

[5. Инструкция пользователя 21](#_Toc168914793)

[6. Заключение 45](#_Toc168914794)

[7. Выводы 62](#_Toc168914795)

[8. Библиографический список 63](#_Toc168914796)

## **Введение**

В рамках данной работы рассмотрена автоматизация чтения данных из документа Word, его последующее экспортирование в Excel, возможность изменения документа в Excel и обратное экспортирование в Word со всеми изменениями. Так же по документам строится статистика в виде столбчатых диаграмм. Автоматизация подобных задач является актуальной в любой сфере деятельности, так как ведение документации является обязательной частью любой деятельности.

## **Цель работы**

Закрепить навыки, полученные в рамках курса «Методы программирования», в том числе: подключение внешних модулей (библиотек классов) и их использования в разрабатываемых программах, программируемая настройка электронных таблиц Excel и документов Word. Освоить навыки работы с элементом Chart и построением графиков и диаграмм.

## **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 суммарное количество отработанной почасовой нагрузки преподавателями за весь рассматриваемый период.

2. Показывать на столбчатой диаграмме суммарное количество отработанной почасовой нагрузки преподавателями в первых семестрах за весь рассматриваемый период.

3. Показывать на столбчатой диаграмме суммарное количество отработанной почасовой нагрузки преподавателями во вторых семестрах за весь рассматриваемый период.

4. Показывать на столбчатой диаграмме количество лекций.

5. Показывать на столбчатой диаграмме количество часов, которые отработал указанный преподаватель в различных учебных группах за весь рассматриваемый период.

6. Показывать на столбчатой диаграмме количество часов дисциплин, попавших в почасовую нагрузку за весь рассматриваемый период.

# **Содержательная часть**

* 1. Анализ печатной формы

Печатные формы представляют собой документы WORD для каждого преподавателя отдельно, в которых находятся таблицы с информацией об их почасовой нагрузке.

* 1. Описание предназначения печатной формы

Таблица включает в себя дисциплины, которые ведет преподаватель, группы, виды занятий, а также количество часов.

4.3 Разнообразие печатных форм и выявление различий

Таблицы в файлах в большей степени отличаются видами дисциплин и количеством часов.

4.4 Унификация печатных форм для удобства обработки данных

Для унификации печатных форм в документах Word для каждого преподавателя отдельно с таблицами почасовой нагрузки, необходимо учитывать ряд важных аспектов. В первую очередь, следует стандартизировать формат и расположение таблиц на документе, чтобы обеспечить удобство чтения и анализа данных. Кроме того, важно разработать единый шаблон документа, который будет содержать все необходимые разделы и информацию, чтобы исключить возможность упущения какой-либо важной детали.

Каждая таблица с почасовой нагрузкой преподавателя должна быть структурирована таким образом, чтобы в ней была представлена вся необходимая информация о занятиях, их продолжительности, даты и места проведения. Также важно предусмотреть возможность внесения корректировок и обновлений в таблицу, например, при изменении расписания занятий или изменении структуры учебного процесса.

Отдельное внимание следует уделить оформлению таблиц и их соответствию общему стилю документа, чтобы обеспечить единообразие внешнего вида. Необходимо следить за тем, чтобы вся информация была представлена четко, лаконично и удобно для восприятия. Важно также предусмотреть возможность автоматической обработки данных из таблиц для дальнейшего анализа и отчетности.

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



Рисунок 1 – Просмотр файла

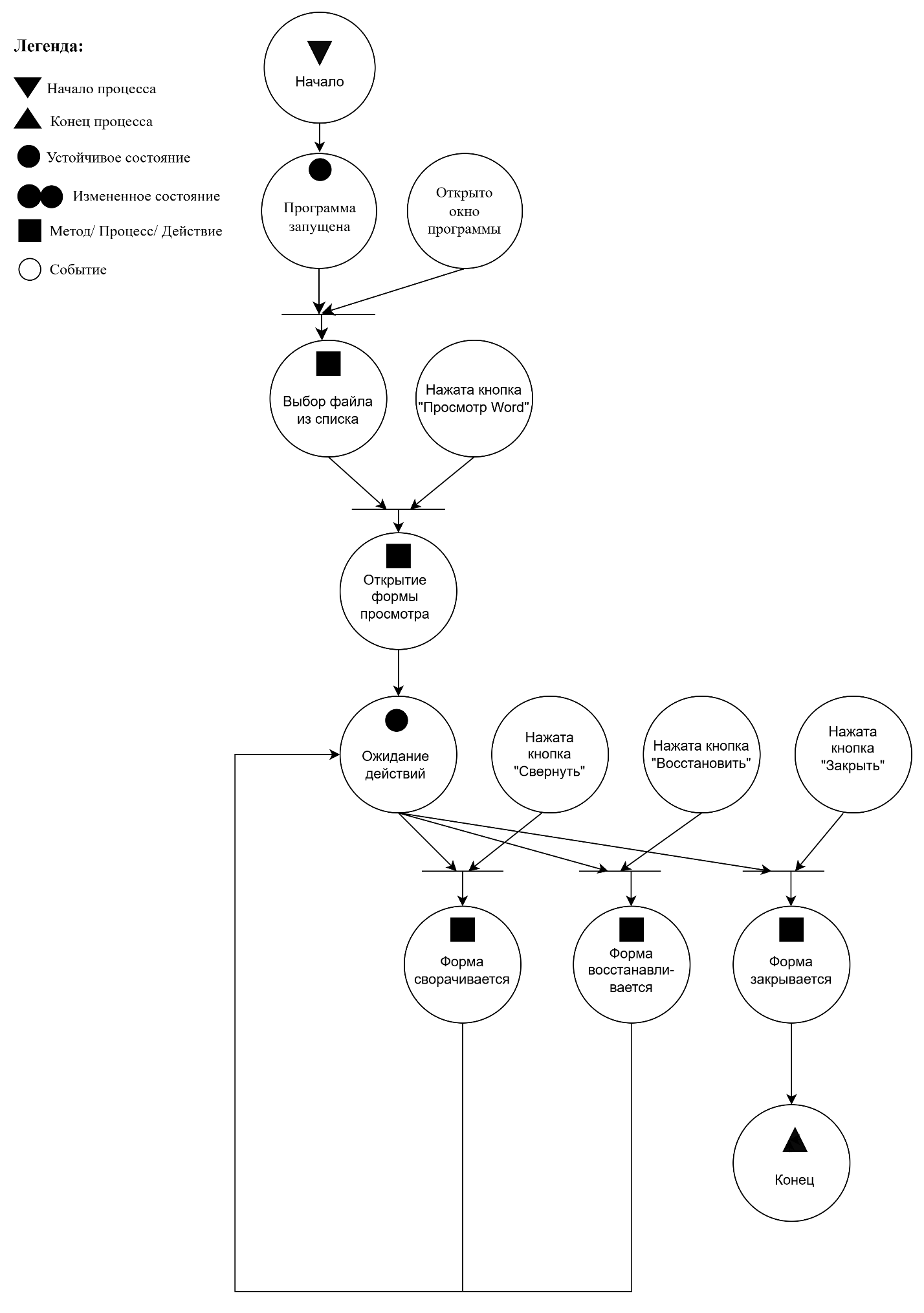


Рисунок 2 – Схема Петри формирования просмотра файла Word

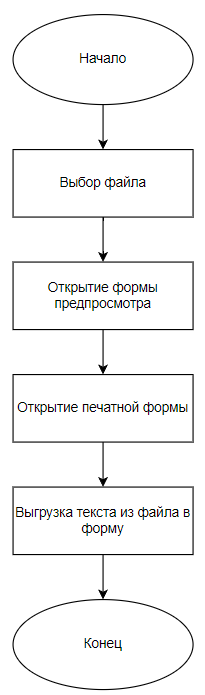


Рисунок 3 – Схема алгоритма метода формирования просмотра

**Листинг кода экранной формы**

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

{

wordapp = new Word.Application();

wd = wordapp.Documents.Open(this.files[this.cmbFiles.SelectedIndex], ReadOnly: true);

wordapp.Visible = true;

}

## **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. На рисунке 4 отобразим сеть Петри, отражающую процесс импорта. А на рисунке 5 покажем схемы этого алгоритма.

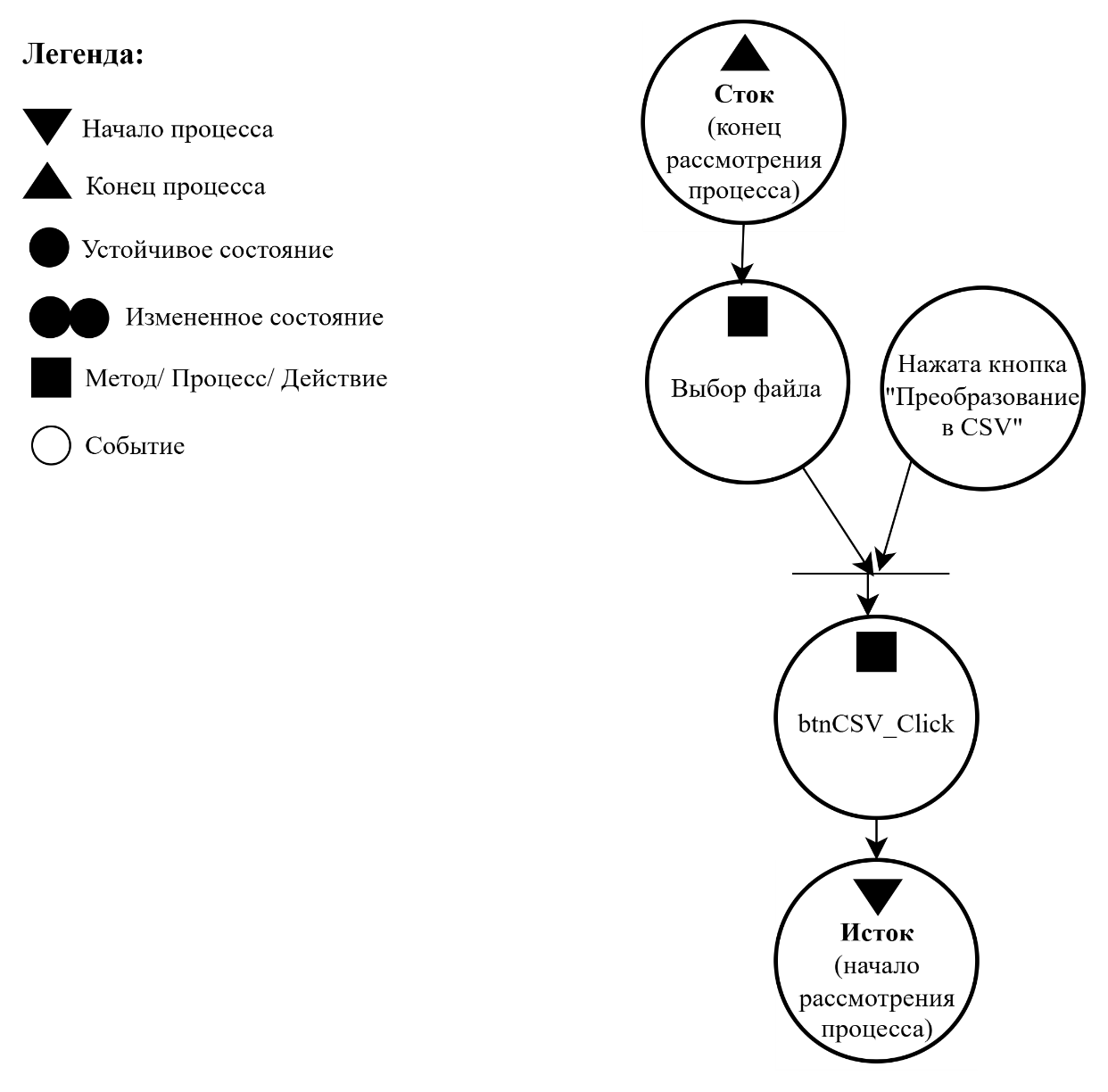


Рисунок 4 – Сеть Петри, отражающая процесс импорта в CSV

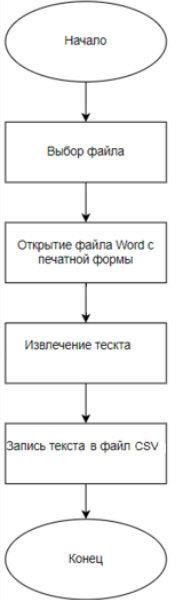


Рисунок 5 – Схема алгоритма, отражающая процесс импорта в CSV

**Листинг кода функции экспорта в CSV файл**

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

{

wordapp = new Word.Application();

wd = wordapp.Documents.Open(this.files[this.cmbFiles.SelectedIndex], ReadOnly: true);

StreamWriter sw = new StreamWriter(Application.StartupPath + "\\csv\\" + ChngExtension(this.cmbFiles.SelectedItem.ToString(), ".txt"));

// Перебираем все таблицы в документе

foreach (Word.Table table in wd.Tables)

{

// Перебираем все строки в таблице

foreach (Word.Row row in table.Rows)

{

// Перебираем все ячейки в строке

foreach (Word.Cell cl in row.Cells)

{

string temp = cl.Range.Text;

temp = temp.Replace("\a", string.Empty).Replace("\n", string.Empty).Remove(temp.Length - 2);

Console.WriteLine(temp);

sw.Write(temp);

sw.Write(";");

}

Console.WriteLine("N");

sw.Write("\n");

}

sw.Write("\n\n"); // Добавляем отступ после каждой таблицы

}

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

Из папки под названием «csv» файл построчно переносит данные в файл WORD. На рисунке 6 отобразим сеть Петри, отражающую процесс разработки функций импорта сведений из оригинального и изменённого текстовых файлов формата CSV с воссозданием Word. А на рисунке покажем 7 схему этого алгоритма.

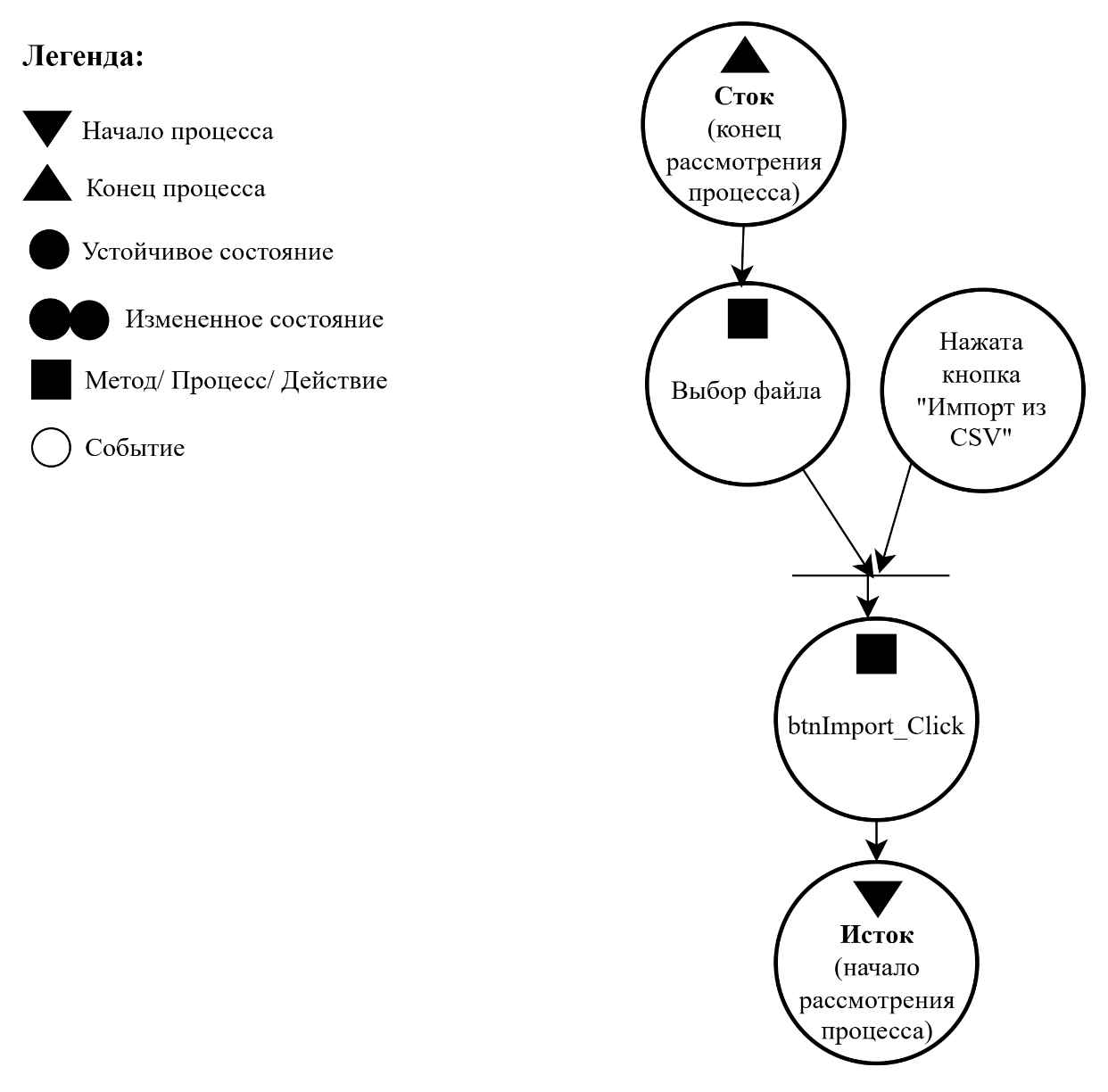


Рисунок 6 – Сеть Петри, отражающая процесс импорта из CSV



Рисунок 7 – Схема алгоритма, отражающая процесс импорта из CSV

**Листинг кода функции импорта из CSV**

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

{

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

Word.Document wd = wordapp.Documents.Add();

Word.Paragraph wp = wd.Paragraphs[1];

Word.Table wt = wd.Tables.Add(wp.Range, 1, 6);

wt.Borders.Enable = 1; // Включаем отображение границ

wt.Borders.OutsideLineStyle = Word.WdLineStyle.wdLineStyleSingle; // Стиль линии для внешних границ

wt.Borders.InsideLineStyle = Word.WdLineStyle.wdLineStyleSingle; // Стиль линии для внутренних границ

StreamReader sr = new StreamReader(Application.StartupPath + "\\csv\\" + ChngExtension(this.cmbFiles.SelectedItem.ToString(), ".txt"));

int counter1 = 1;

int counter2 = 1;

string csv;

string[] atemp;

string[] top = { "Дисциплина", "Группа", "Вид занятий", "Кол-во часов", "Стоимость часа(руб.)", "Общая стоимость оплаты" };

bool isNewTable = true; // Флаг для определения начала новой таблицы

while ((csv = sr.ReadLine()) != null)

{

atemp = csv.Split(';');

if (isNewTable)

{

isNewTable = false;

}

counter2 = 1;

if (counter1 > 1) // Добавляем новые строки только в текущую таблицу

{

wt.Rows.Add();

}

foreach (string line in atemp)

{

if (counter2 > 6) // Если достигли конца текущей строки таблицы

{

isNewTable = true; // Начинаем новую таблицу

break;

}

wt.Cell(wt.Rows.Count, counter2).Range.Text = line; // Добавляем данные в последнюю строку таблицы

wt.Cell(wt.Rows.Count, counter2).Borders.OutsideLineStyle = Word.WdLineStyle.wdLineStyleSingle; // Границы для каждой ячейки

counter2++;

}

counter1++;

}

sr.Close();

string outputFileName = Application.StartupPath + "\\output\\" + this.cmbFiles.SelectedItem.ToString();

wd.SaveAs2(outputFileName);

wordapp.Quit();

MessageBox.Show("Границы в таблицах успешно добавлены. Файл сохранен как " + outputFileName, "Успех");

}

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

Из папки под названием «csv» файл построчно переносит данные в файл EXCEL. На рисунке 8 отобразим сеть Петри, отражающую экспорт в Excel. А на рисунке покажем 9 схему этого алгоритма.

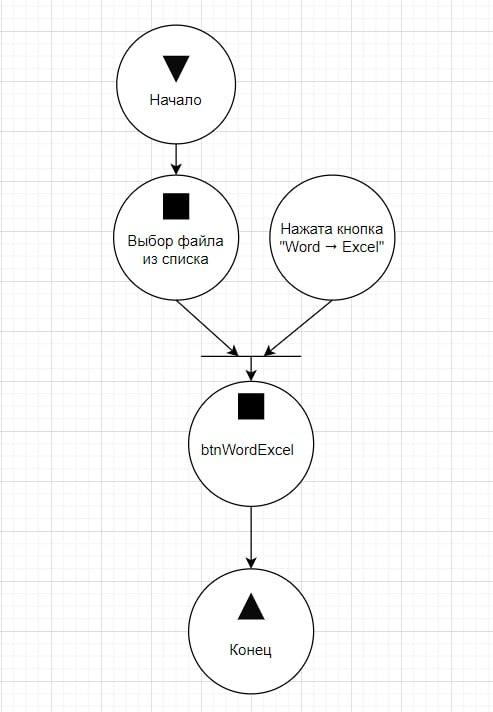


Рисунок 8 – Сеть Петри, отражающая экспорт в Excel



Рисунок 9 – Схема алгоритма, отражающая экспорт в Excel

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

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

{

wordapp = new Word.Application();

exapp = new Excel.Application();

if (rbtnWordExcelOne.Checked)

{

wd = wordapp.Documents.Open(this.files[this.cmbFiles.SelectedIndex], ReadOnly: true);

ex = exapp.Workbooks.Add();

Excel.Worksheet pg = (Excel.Worksheet)ex.ActiveSheet;

string[] filename\_parts = this.cmbFiles.SelectedItem.ToString().Split('.');

//В зависимости от семестра номер таблицы-источника данных может меняться

string[] semeter\_parse\_parts = filename\_parts[filename\_parts.Length - 2].Split('\_');

Word.Table wt;

if (semeter\_parse\_parts[semeter\_parse\_parts.Length - 1] == "II")

{

wt = wd.Tables[wd.Tables.Count];

}

else

{

wt = wd.Tables[wd.Tables.Count - 1];

}

string filename = "";

for (int i = 0; i < filename\_parts.Count() - 1; i++) { filename = filename + filename\_parts[i] + "."; }

filename += "xlsx";

string temp;

foreach (Word.Cell cl in wt.Range.Cells)

{

temp = wt.Cell(cl.RowIndex, cl.ColumnIndex).Range.Text;

temp = temp.Remove(temp.Length - 2);

pg.Cells[cl.RowIndex, cl.ColumnIndex] = temp;

}

exapp.DisplayAlerts = false;

ex.SaveAs(System.Windows.Forms.Application.StartupPath + "\\excel\\" + filename);

exapp.DisplayAlerts = true;

ex.Close();

wd.Close();

}

else

{

for (int file\_i = 0; file\_i < this.cmbFiles.Items.Count - 1; file\_i++)

{

wd = wordapp.Documents.Open(this.files[file\_i], ReadOnly: true);

ex = exapp.Workbooks.Add();

Excel.Worksheet pg = (Excel.Worksheet)ex.ActiveSheet;

string[] filename\_parts = this.cmbFiles.Items[file\_i].ToString().Split('.');

//В зависимости от семестра номер таблицы может меняться

string[] semeter\_parse\_parts = filename\_parts[filename\_parts.Length - 2].Split('\_');

Word.Table wt;

if (semeter\_parse\_parts[semeter\_parse\_parts.Length - 1] == "II")

{

wt = wd.Tables[wd.Tables.Count];

}

else

{

wt = wd.Tables[wd.Tables.Count - 1];

}

string filename = "";

for (int i = 0; i < filename\_parts.Count() - 1; i++) { filename = filename + filename\_parts[i] + "."; }

filename += "xlsx";

string temp;

foreach (Word.Cell cl in wt.Range.Cells)

{

temp = wt.Cell(cl.RowIndex, cl.ColumnIndex).Range.Text;

temp = temp.Remove(temp.Length - 2);

pg.Cells[cl.RowIndex, cl.ColumnIndex] = temp;

}

exapp.DisplayAlerts = false;

ex.SaveAs(System.Windows.Forms.Application.StartupPath + "\\excel\\" + filename);

exapp.DisplayAlerts = true;

ex.Close();

wd.Close();

}

## }

## **4.9 Графический анализ данных, содержащихся в печатной форме**

В данной части работы необходимо выполнить построение диаграмм, отражающих следующие пункты: суммарная нагрузка, нагрузка в 1 семестр, нагрузка в 2 семестр, лекции, лабораторные работы, практики. Ниже на рисунках 10 – 16 представлены диаграммы, выводимые программой.

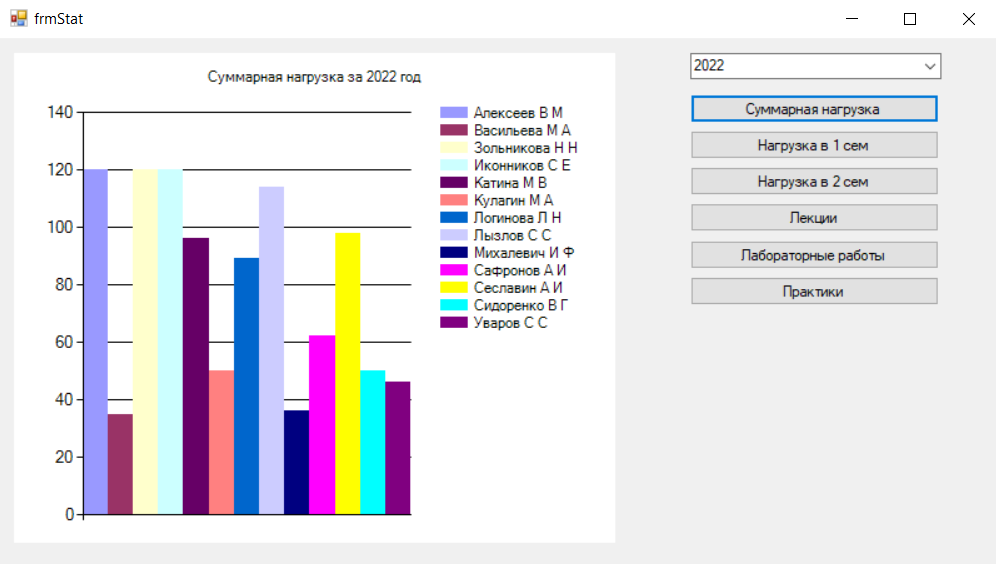
****

Рисунок 10 – Суммарная нагрузка за 2022 год

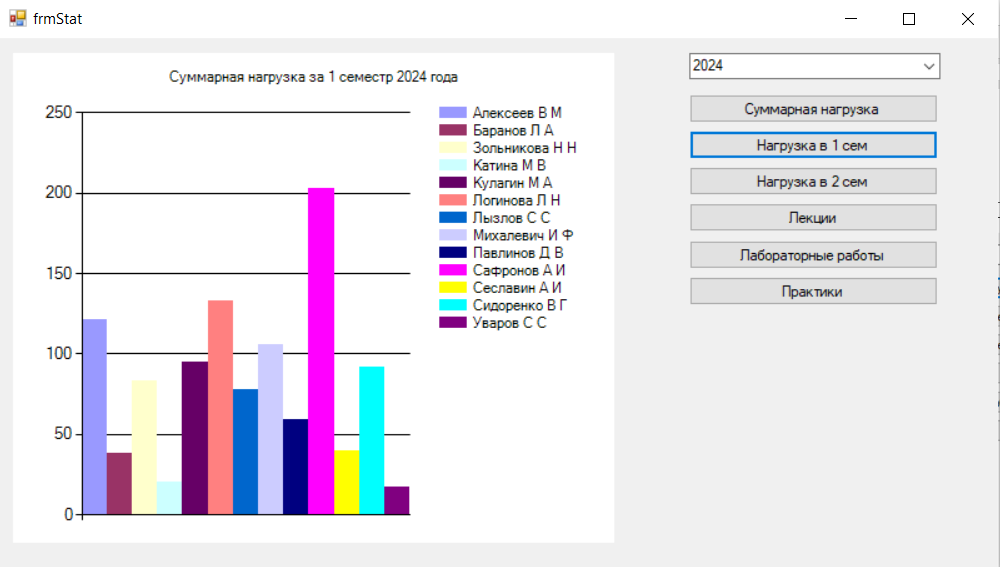
****

Рисунок 11 – Суммарная нагрузка за 1 семестр 2024 года

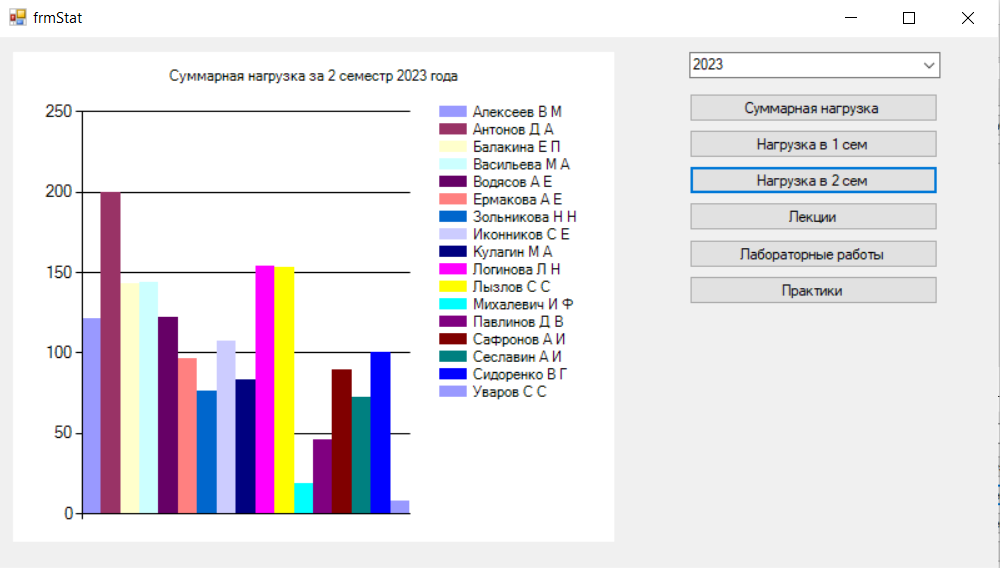
****

Рисунок 12 – Суммарная нагрузка за 2 семестр 2023 года

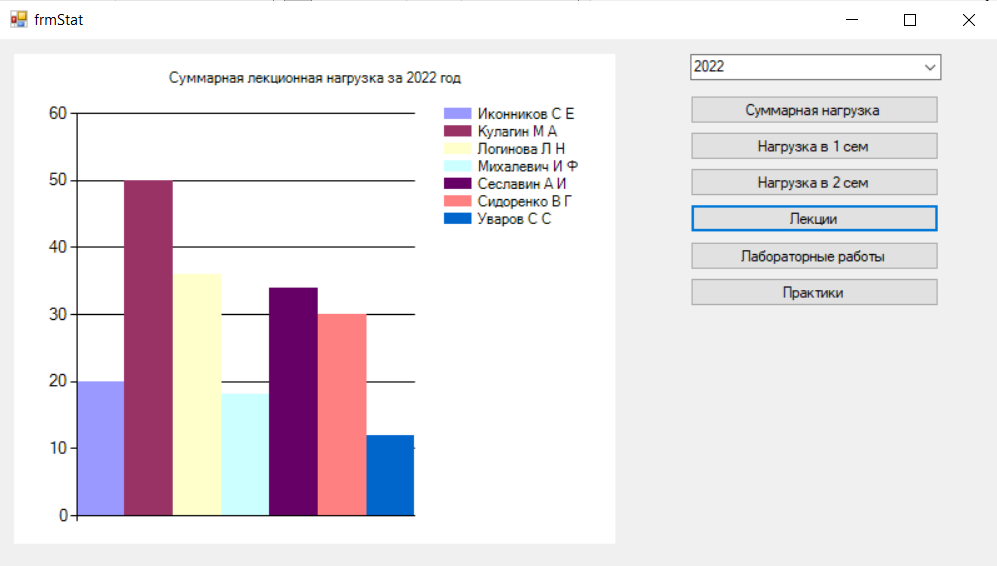
****

Рисунок 3 – Суммарная лекционная нагрузка за 2022 год

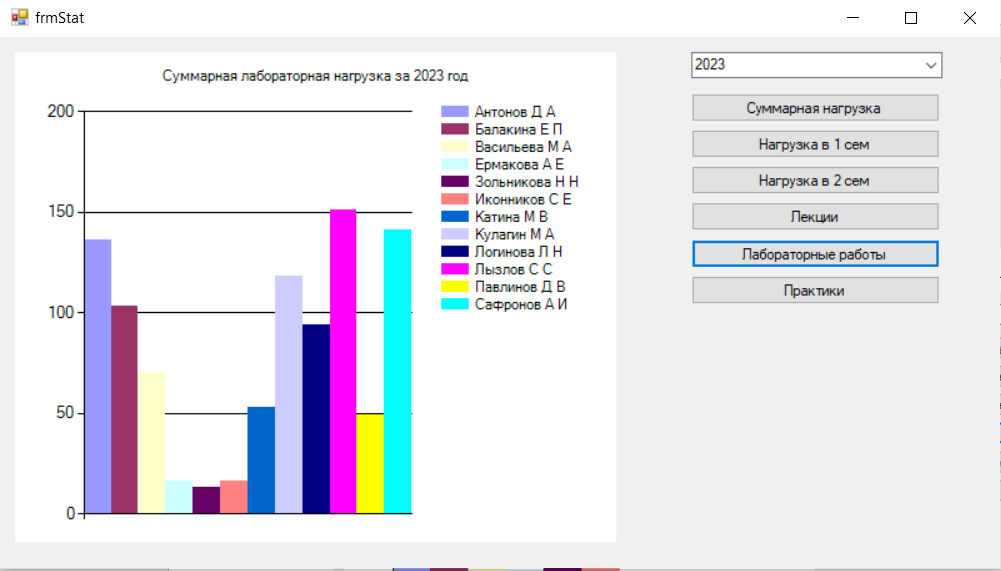
****

Рисунок 14 – Суммарная лабораторная нагрузка за 2023 год

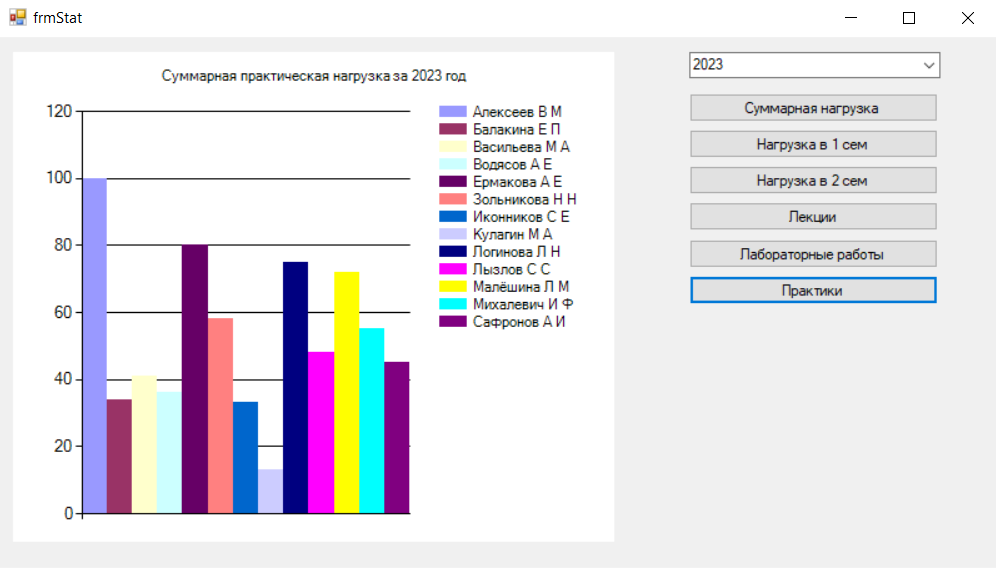
****

Рисунок 15 – Суммарная практическая нагрузка за 2023 год

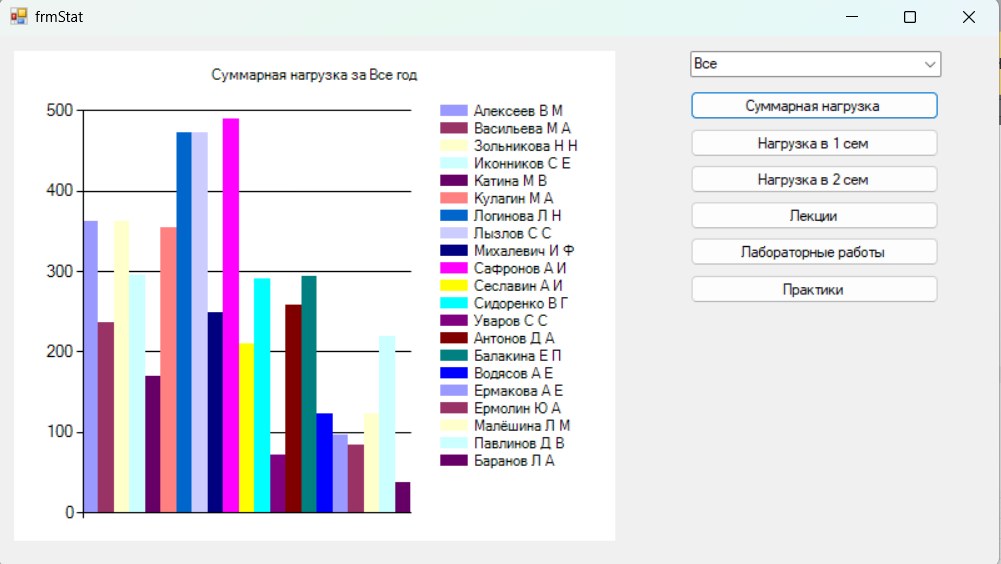


Рисунок 16 – Суммарная нагрузка за все года

Опишем общую структуру в сети Петри.

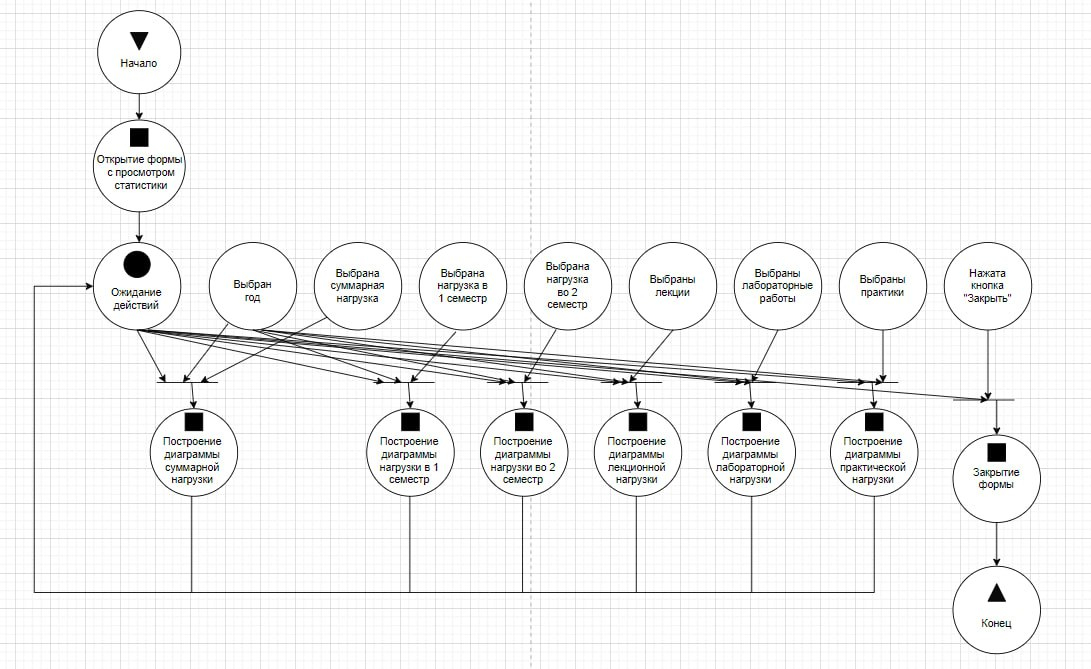


Рисунок 17 – Сеть Петри, отражающая графический анализ столбчатых диаграмм



Рисунок 18 – Схема алгоритма, отражающая графический анализ столбчатых диаграмм

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

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

{

this.MinimumSize = this.Size;

files = Directory.GetFiles(System.Windows.Forms.Application.StartupPath + "\\excel");

Console.WriteLine(files.Length);

string[] temp;

cmbYear.Items.Add("Все");

foreach (string file in files)

{

temp = file.Split('\\');

string filename = temp[temp.Length - 1];

string another\_year = filename.Split('\_')[0];

//Заполняем комбо-бокс с годами

if(!cmbYear.Items.Contains(another\_year))

{

cmbYear.Items.Add(filename.Split('\_')[0]);

}

}

}

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

{

if (cmbYear.SelectedIndex == -1)

{

MessageBox.Show("Рассматриваемый период не выбран!");

return;

}

string year = this.cmbYear.SelectedItem.ToString();

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

List<int> dats = new List<int>();

//Добыча данных из Exel при помощи его инструментов

exapp = new Excel.Application();

string[] temp;

foreach (string file in files)

{

temp = file.Split('\\');

string filename = temp[temp.Length - 1];

if (filename.Split('\_')[0] == year | year == "Все")

{

ex = exapp.Workbooks.Open(System.Windows.Forms.Application.StartupPath + "\\excel\\" + filename, ReadOnly: false);

string[] name\_parts = filename.Split('\_');

string name = "";

for (int i = 1; i < name\_parts.Length - 2; i++) { name += name\_parts[i] + " "; }

name += name\_parts[name\_parts.Length - 2];

Excel.Worksheet pg = ex.Sheets[1];

//Поиск столбца с названием "Количество часов"

Excel.Range cl = pg.get\_Range("1:1").Find("Кол-во часов");

//Поиск строки с названием "Всего за семестр"

Excel.Range row = pg.get\_Range("A:A").Find("Всего за семестр");

//Данные на их пересечении

int data;

bool parced = int.TryParse(pg.get\_Range(Excel\_cell(row.Row, cl.Column)).Text, out data);

Console.WriteLine(parced);

if (names.Contains(name) & parced)

{

dats[names.IndexOf(name)] += data;

}

else

{

if (parced)

{

names.Add(name);

dats.Add(data);

}

}

ex.Close();

}

}

exapp.Quit();

chrtStat.Series.Clear();

chrtStat.Titles.Clear();

chrtStat.Titles.Add("Суммарная нагрузка за " + year + " год");

chrtStat.ChartAreas[0].AxisX.Maximum = 1 + 0.4;

chrtStat.ChartAreas[0].AxisX.Minimum = 1 - 0.4;

chrtStat.ChartAreas[0].AxisX.LabelStyle.Enabled = false;

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

{

Series series = this.chrtStat.Series.Add(names[i]);

series.Points.Add(dats[i]);

series.ChartType = SeriesChartType.Column;

}

}

private void btn1semTime\_Click(object sender, EventArgs e)

{

if (cmbYear.SelectedIndex == -1)

{

MessageBox.Show("Рассматриваемый период не выбран!");

return;

}

string year = this.cmbYear.SelectedItem.ToString();

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

List<int> dats = new List<int>();

//Добыча данных из Exel при помощи его инструментов

exapp = new Excel.Application();

string[] temp;

foreach (string file in files)

{

temp = file.Split('\\');

string filename = temp[temp.Length - 1];

string[] name\_parts = filename.Split('\_');

string sem = name\_parts[name\_parts.Length - 1].Split('.')[0];

if ((filename.Split('\_')[0] == year | year == "Все") & sem == "II")

{

Console.WriteLine(filename);

ex = exapp.Workbooks.Open(System.Windows.Forms.Application.StartupPath + "\\excel\\" + filename, ReadOnly: false);

string name = "";

for (int i = 1; i < name\_parts.Length - 2; i++) { name += name\_parts[i] + " "; }

name += name\_parts[name\_parts.Length - 2];

Excel.Worksheet pg = ex.Sheets[1];

//Поиск столбца с названием "Количество часов"

Excel.Range cl = pg.get\_Range("1:1").Find("Кол-во часов");

//Поиск строки с названием "Всего за семестр"

Excel.Range row = pg.get\_Range("A:A").Find("Всего за семестр");

//Данные на их пересечении

int data = 0;

bool parced = int.TryParse(pg.get\_Range(Excel\_cell(row.Row, cl.Column)).Text, out data);

if (names.Contains(name) & parced)

{

dats[names.IndexOf(name)] += data;

}

else

{

if (parced)

{

names.Add(name);

dats.Add(data);

}

}

ex.Close();

}

}

exapp.Quit();

chrtStat.Series.Clear();

chrtStat.Titles.Clear();

chrtStat.Titles.Add("Суммарная нагрузка за 1 семестр " + year + " года");

chrtStat.ChartAreas[0].AxisX.Maximum = 1 + 0.4;

chrtStat.ChartAreas[0].AxisX.Minimum = 1 - 0.4;

chrtStat.ChartAreas[0].AxisX.LabelStyle.Enabled = false;

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

{

Series series = this.chrtStat.Series.Add(names[i]);

series.Points.Add(dats[i]);

series.ChartType = SeriesChartType.Column;

}

}

private void btn2semTime\_Click(object sender, EventArgs e)

{

if (cmbYear.SelectedIndex == -1)

{

MessageBox.Show("Рассматриваемый период не выбран!");

return;

}

string year = this.cmbYear.SelectedItem.ToString();

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

List<int> dats = new List<int>();

//Добыча данных из Exel при помощи его инструментов

exapp = new Excel.Application();

string[] temp;

foreach (string file in files)

{

temp = file.Split('\\');

string filename = temp[temp.Length - 1];

string[] name\_parts = filename.Split('\_');

string sem = name\_parts[name\_parts.Length - 1].Split('.')[0];

if ((filename.Split('\_')[0] == year | year == "Все") & sem == "I")

{

Console.WriteLine(filename);

ex = exapp.Workbooks.Open(System.Windows.Forms.Application.StartupPath + "\\excel\\" + filename, ReadOnly: false);

string name = "";

for (int i = 1; i < name\_parts.Length - 2; i++) { name += name\_parts[i] + " "; }

name += name\_parts[name\_parts.Length - 2];

Excel.Worksheet pg = ex.Sheets[1];

//Поиск столбца с названием "Количество часов"

Excel.Range cl = pg.get\_Range("1:1").Find("Кол-во часов");

//Поиск строки с названием "Всего за семестр"

Excel.Range row = pg.get\_Range("A:A").Find("Всего за семестр");

//Данные на их пересечении

int data = 0;

bool parced = int.TryParse(pg.get\_Range(Excel\_cell(row.Row, cl.Column)).Text, out data);

if (names.Contains(name) & parced)

{

dats[names.IndexOf(name)] += data;

}

else

{

if (parced)

{

names.Add(name);

dats.Add(data);

}

}

ex.Close();

}

}

exapp.Quit();

chrtStat.Series.Clear();

chrtStat.Titles.Clear();

chrtStat.Titles.Add("Суммарная нагрузка за 2 семестр " + year + " года");

chrtStat.ChartAreas[0].AxisX.Maximum = 1 + 0.4;

chrtStat.ChartAreas[0].AxisX.Minimum = 1 - 0.4;

chrtStat.ChartAreas[0].AxisX.LabelStyle.Enabled = false;

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

{

Series series = this.chrtStat.Series.Add(names[i]);

series.Points.Add(dats[i]);

series.ChartType = SeriesChartType.Column;

}

}

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

{

if (cmbYear.SelectedIndex == -1)

{

MessageBox.Show("Рассматриваемый период не выбран!");

return;

}

string year = this.cmbYear.SelectedItem.ToString();

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

List<int> dats = new List<int>();

//Добыча данных из Exel при помощи его инструментов

exapp = new Excel.Application();

string[] temp;

foreach (string file in files)

{

temp = file.Split('\\');

string filename = temp[temp.Length - 1];

string[] name\_parts = filename.Split('\_');

if (filename.Split('\_')[0] == year | year == "Все")

{

ex = exapp.Workbooks.Open(System.Windows.Forms.Application.StartupPath + "\\excel\\" + filename, ReadOnly: false);

string name = "";

for (int i = 1; i < name\_parts.Length - 2; i++) { name += name\_parts[i] + " "; }

name += name\_parts[name\_parts.Length - 2];

Excel.Worksheet pg = ex.Sheets[1];

//Поиск строки с названием "Лекции"

Excel.Range cl ;

cl = pg.get\_Range("C1:C65536").Find("Лекции");

while(cl != null)

{

//Смещение выделения в клетку с количеством часов

Excel.Range data\_cl = cl.Offset[0,1];

int data = 0;

bool parced = int.TryParse(data\_cl.Text, out data);

if (names.Contains(name) & parced)

{

dats[names.IndexOf(name)] += data;

}

else

{

if (parced)

{

names.Add(name);

dats.Add(data);

}

}

cl = pg.get\_Range("C" + (cl.Row + 1).ToString() + ":C65536").Find("Лекции");

}

ex.Close();

}

}

exapp.Quit();

chrtStat.Series.Clear();

chrtStat.Titles.Clear();

chrtStat.Titles.Add("Суммарная лекционная нагрузка за " + year + " год");

chrtStat.ChartAreas[0].AxisX.Maximum = 1 + 0.4;

chrtStat.ChartAreas[0].AxisX.Minimum = 1 - 0.4;

chrtStat.ChartAreas[0].AxisX.LabelStyle.Enabled = false;

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

{

Series series = this.chrtStat.Series.Add(names[i]);

series.Points.Add(dats[i]);

series.ChartType = SeriesChartType.Column;

}

}

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

{

if (cmbYear.SelectedIndex == -1)

{

MessageBox.Show("Рассматриваемый период не выбран!");

return;

}

string year = this.cmbYear.SelectedItem.ToString();

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

List<int> dats = new List<int>();

//Добыча данных из Exel при помощи его инструментов

exapp = new Excel.Application();

string[] temp;

foreach (string file in files)

{

temp = file.Split('\\');

string filename = temp[temp.Length - 1];

string[] name\_parts = filename.Split('\_');

if (filename.Split('\_')[0] == year | year == "Все")

{

ex = exapp.Workbooks.Open(System.Windows.Forms.Application.StartupPath + "\\excel\\" + filename, ReadOnly: false);

string name = "";

for (int i = 1; i < name\_parts.Length - 2; i++) { name += name\_parts[i] + " "; }

name += name\_parts[name\_parts.Length - 2];

Excel.Worksheet pg = ex.Sheets[1];

//Поиск строки с названием "Лабораторные работы"

Excel.Range cl;

cl = pg.get\_Range("C1:C65536").Find("Лабораторные работы");

while (cl != null)

{

//Смещение выделения в клетку с количеством часов

Excel.Range data\_cl = cl.Offset[0, 1];

int data = 0;

bool parced = int.TryParse(data\_cl.Text, out data);

if (names.Contains(name) & parced)

{

dats[names.IndexOf(name)] += data;

}

else

{

if (parced)

{

names.Add(name);

dats.Add(data);

}

}

cl = pg.get\_Range("C" + (cl.Row + 1).ToString() + ":C65536").Find("Лабораторные работы");

}

ex.Close();

}

}

exapp.Quit();

chrtStat.Series.Clear();

chrtStat.Titles.Clear();

chrtStat.Titles.Add("Суммарная лабораторная нагрузка за " + year + " год");

chrtStat.ChartAreas[0].AxisX.Maximum = 1 + 0.4;

chrtStat.ChartAreas[0].AxisX.Minimum = 1 - 0.4;

chrtStat.ChartAreas[0].AxisX.LabelStyle.Enabled = false;

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

{

Series series = this.chrtStat.Series.Add(names[i]);

series.Points.Add(dats[i]);

series.ChartType = SeriesChartType.Column;

}

}

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

{

if (cmbYear.SelectedIndex == -1)

{

MessageBox.Show("Рассматриваемый период не выбран!");

return;

}

string year = this.cmbYear.SelectedItem.ToString();

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

List<int> dats = new List<int>();

//Добыча данных из Exel при помощи его инструментов

exapp = new Excel.Application();

string[] temp;

foreach (string file in files)

{

temp = file.Split('\\');

string filename = temp[temp.Length - 1];

string[] name\_parts = filename.Split('\_');

if (filename.Split('\_')[0] == year | year == "Все")

{

ex = exapp.Workbooks.Open(System.Windows.Forms.Application.StartupPath + "\\excel\\" + filename, ReadOnly: false);

string name = "";

for (int i = 1; i < name\_parts.Length - 2; i++) { name += name\_parts[i] + " "; }

name += name\_parts[name\_parts.Length - 2];

Excel.Worksheet pg = ex.Sheets[1];

//Поиск строки с названием "Практические занятия"

Excel.Range cl;

cl = pg.get\_Range("C1:C65536").Find("Практические занятия");

while (cl != null)

{

//Смещение выделения в клетку с количеством часов

Excel.Range data\_cl = cl.Offset[0, 1];

int data = 0;

bool parced = int.TryParse(data\_cl.Text, out data);

if (names.Contains(name) & parced)

{

dats[names.IndexOf(name)] += data;

}

else

{

if (parced)

{

names.Add(name);

dats.Add(data);

}

}

cl = pg.get\_Range("C" + (cl.Row + 1).ToString() + ":C65536").Find("Практические занятия");

}

ex.Close();

}

}

exapp.Quit();

chrtStat.Series.Clear();

chrtStat.Titles.Clear();

chrtStat.Titles.Add("Суммарная практическая нагрузка за " + year + " год");

chrtStat.ChartAreas[0].AxisX.Maximum = 1 + 0.4;

chrtStat.ChartAreas[0].AxisX.Minimum = 1 - 0.4;

chrtStat.ChartAreas[0].AxisX.LabelStyle.Enabled = false;

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

{

Series series = this.chrtStat.Series.Add(names[i]);

series.Points.Add(dats[i]);

series.ChartType = SeriesChartType.Column;

}

}

1. **Инструкция пользователя**

Чтобы начать работу с программой необходимо выбрать файл из предложенного списка. В главной форме имеется 7 кнопок: «Просмотр Word», «Word ⭢ Excel», «Просмотр Excel», «Excel ⭢ Word », «Просмотр статистики», «Преобразование в CSV», «Импорт из CSV».

* Запуск программы и отображение главного интерфейса

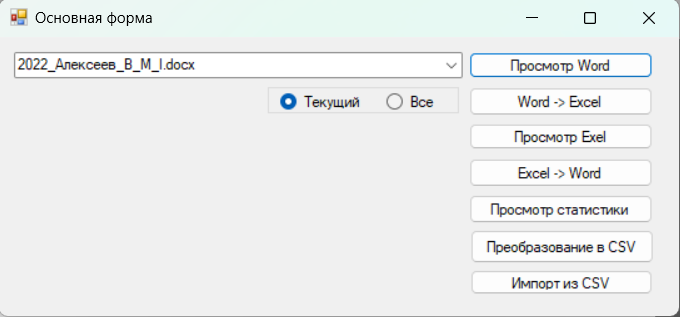


Рисунок 19 – Главное окно программы

* Выбор файла для дальнейшей работы с ним

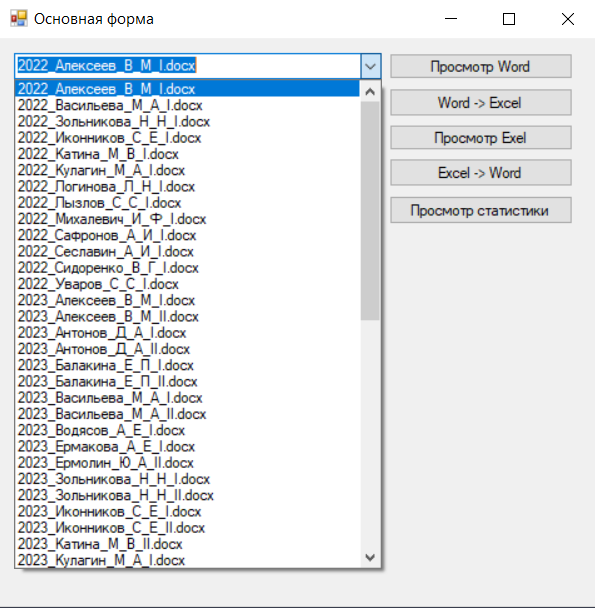


Рисунок 20 – Выбор файла

* Нажатие кнопки «Просмотр Word» на главном интерфейсе

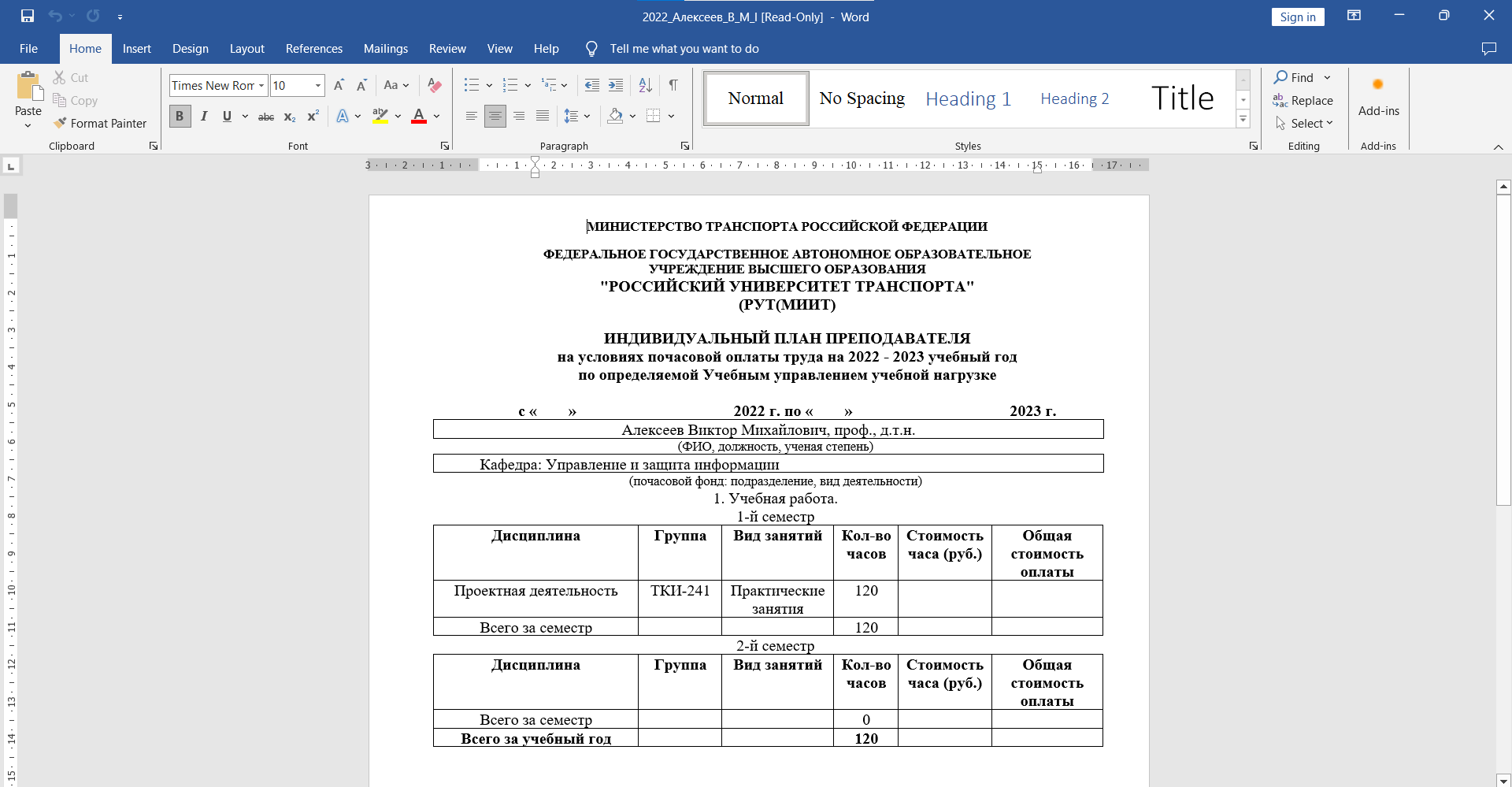


Рисунок 21 – Результат нажатия кнопки «Просмотр Word»

* Нажатие на кнопку «Word ⭢ Excel», далее нажатие на кнопку «Просмотр Excel»

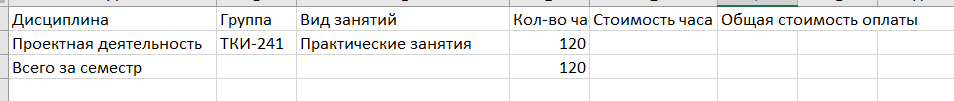


Рисунок 22 – Результат нажатия кнопки «Просмотр Excel»

* Изменение данных в Excel, затем импорт из Excel в Word, и сохранение данных, которые были изменены

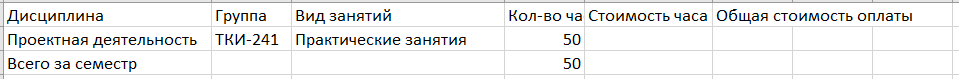


Рисунок 23 – Измененные данные в Excel

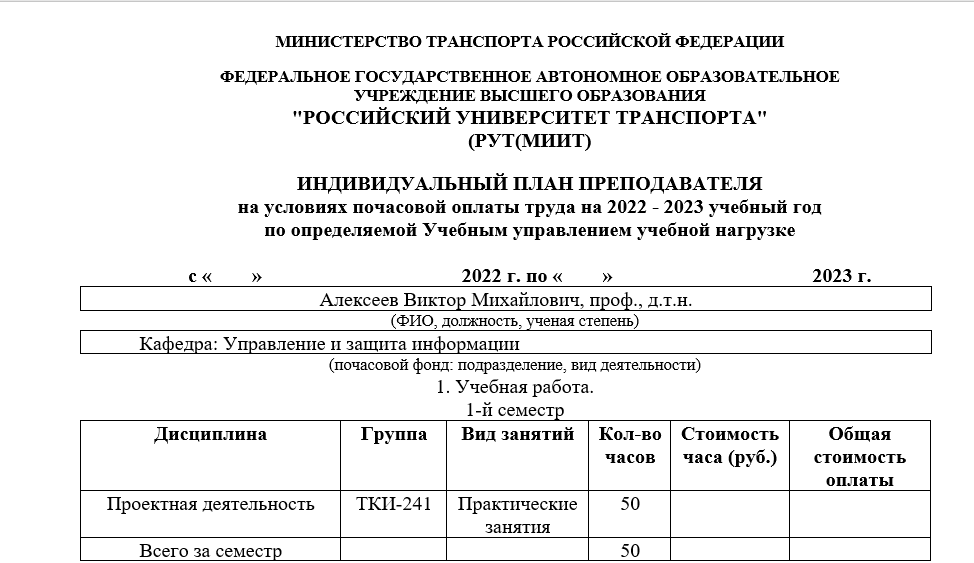


Рисунок 24 – Данные сохранились в Word

Для просмотра кнопок «Word ⭢ Excel» и «Excel ⭢ Word» соответственно есть дополнение, мы можем выбрать все файлы для импорта. Если нам надо посмотреть статистику некоторых файлов, то их мы импортируем отдельно, выбирая опцию для каждого файла «Текущий».

Для просмотра диаграмм следует выбрать год и интересующую вас статистику.

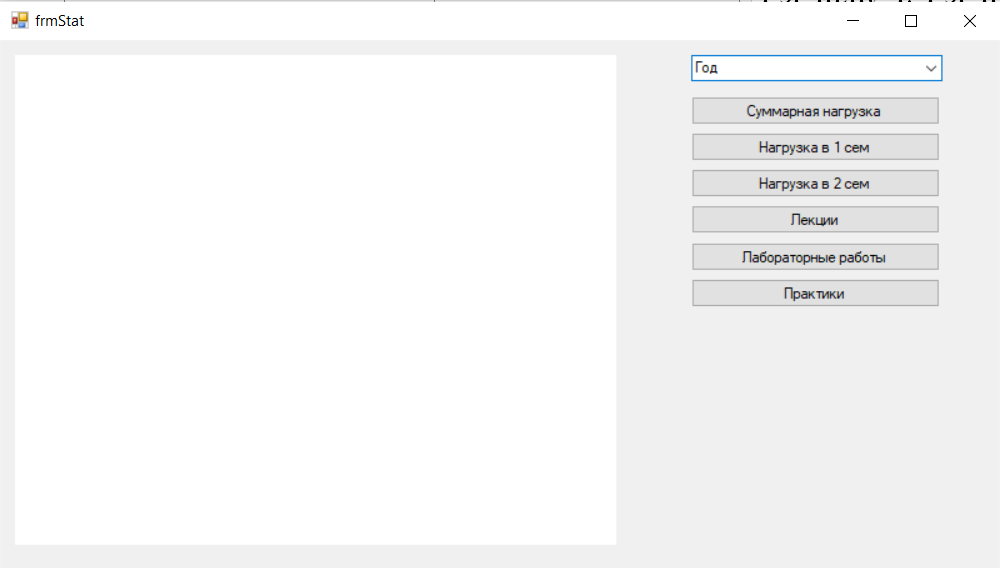


Рисунок 25 – Форма для статистики

## **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**

* **frmMain**

using System;

using System.Linq;

using System.Windows.Forms;

using System.IO;

using Word = Microsoft.Office.Interop.Word;

using Excel = Microsoft.Office.Interop.Excel;

namespace Coursework

{

public partial class frmMain : Form

{

private string csvFilePath;

//Функция для прямого доступа к ячейкам Exel ("A1" вместо (1, 1))

private string Excel\_cell(int row, int cl)

{

string name = "";

switch (cl)

{

case 1: { name = "A" + row.ToString(); break; }

case 2: { name = "B" + row.ToString(); break; }

case 3: { name = "C" + row.ToString(); break; }

case 4: { name = "D" + row.ToString(); break; }

case 5: { name = "E" + row.ToString(); break; }

case 6: { name = "F" + row.ToString(); break; }

case 7: { name = "G" + row.ToString(); break; }

case 8: { name = "H" + row.ToString(); break; }

case 9: { name = "I" + row.ToString(); break; }

case 10: { name = "J" + row.ToString(); break; }

case 11: { name = "K" + row.ToString(); break; }

case 12: { name = "L" + row.ToString(); break; }

case 13: { name = "M" + row.ToString(); break; }

case 14: { name = "N" + row.ToString(); break; }

case 15: { name = "O" + row.ToString(); break; }

case 16: { name = "P" + row.ToString(); break; }

case 17: { name = "Q" + row.ToString(); break; }

case 18: { name = "R" + row.ToString(); break; }

case 19: { name = "S" + row.ToString(); break; }

case 20: { name = "T" + row.ToString(); break; }

}

return name;

}

private Word.Application wordapp;

private Word.Document wd;

private Excel.Application exapp;

private Excel.Workbook ex;

string[] files;

public frmMain()

{

InitializeComponent();

}

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

{

this.MinimumSize = this.Size;

files = Directory.GetFiles(System.Windows.Forms.Application.StartupPath + "\\docs");

string[] temp;

foreach (string file in files)

{

temp = file.Split('\\');

this.cmbFiles.Items.Add(temp[temp.Length - 1]);

}

this.cmbFiles.SelectedIndex = 0;

}

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

{

wordapp = new Word.Application();

wd = wordapp.Documents.Open(this.files[this.cmbFiles.SelectedIndex], ReadOnly: true);

wordapp.Visible = true;

}

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

{

wordapp = new Word.Application();

exapp = new Excel.Application();

if (rbtnWordExcelOne.Checked)

{

wd = wordapp.Documents.Open(this.files[this.cmbFiles.SelectedIndex], ReadOnly: true);

ex = exapp.Workbooks.Add();

Excel.Worksheet pg = (Excel.Worksheet)ex.ActiveSheet;

string[] filename\_parts = this.cmbFiles.SelectedItem.ToString().Split('.');

//В зависимости от семестра номер таблицы-источника данных может меняться

string[] semeter\_parse\_parts = filename\_parts[filename\_parts.Length - 2].Split('\_');

Word.Table wt;

if (semeter\_parse\_parts[semeter\_parse\_parts.Length - 1] == "II")

{

wt = wd.Tables[wd.Tables.Count];

}

else

{

wt = wd.Tables[wd.Tables.Count - 1];

}

string filename = "";

for (int i = 0; i < filename\_parts.Count() - 1; i++) { filename = filename + filename\_parts[i] + "."; }

filename += "xlsx";

string temp;

foreach (Word.Cell cl in wt.Range.Cells)

{

temp = wt.Cell(cl.RowIndex, cl.ColumnIndex).Range.Text;

temp = temp.Remove(temp.Length - 2);

pg.Cells[cl.RowIndex, cl.ColumnIndex] = temp;

}

exapp.DisplayAlerts = false;

ex.SaveAs(System.Windows.Forms.Application.StartupPath + "\\excel\\" + filename);

exapp.DisplayAlerts = true;

ex.Close();

wd.Close();

}

else

{

for (int file\_i = 0; file\_i < this.cmbFiles.Items.Count - 1; file\_i++)

{

wd = wordapp.Documents.Open(this.files[file\_i], ReadOnly: true);

ex = exapp.Workbooks.Add();

Excel.Worksheet pg = (Excel.Worksheet)ex.ActiveSheet;

string[] filename\_parts = this.cmbFiles.Items[file\_i].ToString().Split('.');

//В зависимости от семестра номер таблицы может меняться

string[] semeter\_parse\_parts = filename\_parts[filename\_parts.Length - 2].Split('\_');

Word.Table wt;

if (semeter\_parse\_parts[semeter\_parse\_parts.Length - 1] == "II")

{

wt = wd.Tables[wd.Tables.Count];

}

else

{

wt = wd.Tables[wd.Tables.Count - 1];

}

string filename = "";

for (int i = 0; i < filename\_parts.Count() - 1; i++) { filename = filename + filename\_parts[i] + "."; }

filename += "xlsx";

string temp;

foreach (Word.Cell cl in wt.Range.Cells)

{

temp = wt.Cell(cl.RowIndex, cl.ColumnIndex).Range.Text;

temp = temp.Remove(temp.Length - 2);

pg.Cells[cl.RowIndex, cl.ColumnIndex] = temp;

}

exapp.DisplayAlerts = false;

ex.SaveAs(System.Windows.Forms.Application.StartupPath + "\\excel\\" + filename);

exapp.DisplayAlerts = true;

ex.Close();

wd.Close();

}

}

wordapp.Quit();

exapp.Quit();

}

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

{

string[] filename\_parts = this.cmbFiles.SelectedItem.ToString().Split('.');

string filename = "";

for (int i = 0; i < filename\_parts.Count() - 1; i++) { filename = filename + filename\_parts[i] + "."; }

filename += "xlsx";

exapp = new Excel.Application();

try

{

ex = exapp.Workbooks.Open(System.Windows.Forms.Application.StartupPath + "\\excel\\" + filename, ReadOnly: false);

exapp.Visible = true;

}

catch { MessageBox.Show("Запрашиваемый файл Excel не найден!"); }

}

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

{

wordapp = new Word.Application();

wd = wordapp.Documents.Open(this.files[this.cmbFiles.SelectedIndex], ReadOnly: false);

string[] filename\_parts = this.cmbFiles.SelectedItem.ToString().Split('.');

string filename = "";

for (int i = 0; i < filename\_parts.Count() - 1; i++) { filename = filename + filename\_parts[i] + "."; }

exapp = new Excel.Application();

ex = exapp.Workbooks.Open(System.Windows.Forms.Application.StartupPath + "\\excel\\" + filename + "xlsx", ReadOnly: false);

Excel.Worksheet pg = ex.Sheets[1];

//В зависимости от семестра номер таблицы может меняться

string[] semeter\_parse\_parts = filename\_parts[filename\_parts.Length - 2].Split('\_');

Word.Table wt;

if (semeter\_parse\_parts[semeter\_parse\_parts.Length - 1] == "II")

{

wt = wd.Tables[wd.Tables.Count];

}

else

{

wt = wd.Tables[wd.Tables.Count - 1];

}

string temp;

foreach (Word.Cell cl in wt.Range.Cells)

{

try

{

temp = wt.Cell(cl.RowIndex, cl.ColumnIndex).Range.Text;

temp = pg.get\_Range(Excel\_cell(cl.RowIndex, cl.ColumnIndex)).Text;

wt.Cell(cl.RowIndex, cl.ColumnIndex).Range.Text = temp;

}

finally { }

}

wordapp.DisplayAlerts = Word.WdAlertLevel.wdAlertsNone;

wd.SaveAs(System.Windows.Forms.Application.StartupPath + "\\docs\\" + filename + "docx");

wordapp.DisplayAlerts = Word.WdAlertLevel.wdAlertsAll;

wd.Close();

wordapp.Quit();

ex.Close();

exapp.Quit();

}

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

{

frmStat fs = new frmStat();

fs.Show();

}

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

{

wordapp = new Word.Application();

wd = wordapp.Documents.Open(this.files[this.cmbFiles.SelectedIndex], ReadOnly: true);

StreamWriter sw = new StreamWriter(Application.StartupPath + "\\csv\\" + ChngExtension(this.cmbFiles.SelectedItem.ToString(), ".txt"));

// Перебираем все таблицы в документе

foreach (Word.Table table in wd.Tables)

{

// Перебираем все строки в таблице

foreach (Word.Row row in table.Rows)

{

// Перебираем все ячейки в строке

foreach (Word.Cell cl in row.Cells)

{

string temp = cl.Range.Text;

temp = temp.Replace("\a", string.Empty).Replace("\n", string.Empty).Remove(temp.Length - 2);

Console.WriteLine(temp);

sw.Write(temp);

sw.Write(";");

}

Console.WriteLine("N");

sw.Write("\n");

}

sw.Write("\n\n"); // Добавляем отступ после каждой таблицы

}

sw.Close();

wordapp.Quit();

}

public string ChngExtension(string filename, string ext)

{

string[] temp = filename.Split('.');

return temp[0] + ext;

}

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

{

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

Word.Document wd = wordapp.Documents.Add();

Word.Paragraph wp = wd.Paragraphs[1];

Word.Table wt = wd.Tables.Add(wp.Range, 1, 6);

wt.Borders.Enable = 1; // Включаем отображение границ

wt.Borders.OutsideLineStyle = Word.WdLineStyle.wdLineStyleSingle; // Стиль линии для внешних границ

wt.Borders.InsideLineStyle = Word.WdLineStyle.wdLineStyleSingle; // Стиль линии для внутренних границ

StreamReader sr = new StreamReader(Application.StartupPath + "\\csv\\" + ChngExtension(this.cmbFiles.SelectedItem.ToString(), ".txt"));

int counter1 = 1;

int counter2 = 1;

string csv;

string[] atemp;

string[] top = { "Дисциплина", "Группа", "Вид занятий", "Кол-во часов", "Стоимость часа(руб.)", "Общая стоимость оплаты" };

bool isNewTable = true; // Флаг для определения начала новой таблицы

while ((csv = sr.ReadLine()) != null)

{

atemp = csv.Split(';');

if (isNewTable)

{

isNewTable = false;

}

counter2 = 1;

if (counter1 > 1) // Добавляем новые строки только в текущую таблицу

{

wt.Rows.Add();

}

foreach (string line in atemp)

{

if (counter2 > 6) // Если достигли конца текущей строки таблицы

{

isNewTable = true; // Начинаем новую таблицу

break;

}

wt.Cell(wt.Rows.Count, counter2).Range.Text = line; // Добавляем данные в последнюю строку таблицы

wt.Cell(wt.Rows.Count, counter2).Borders.OutsideLineStyle = Word.WdLineStyle.wdLineStyleSingle; // Границы для каждой ячейки

counter2++;

}

counter1++;

}

sr.Close();

string outputFileName = Application.StartupPath + "\\output\\" + this.cmbFiles.SelectedItem.ToString();

wd.SaveAs2(outputFileName);

wordapp.Quit();

MessageBox.Show("Границы в таблицах успешно добавлены. Файл сохранен как " + outputFileName, "Успех");

}

}

}

* **frmStat**

using System;

using System.Collections.Generic;

using System.Windows.Forms;

using System.IO;

using System.Windows.Forms.DataVisualization.Charting;

using Excel = Microsoft.Office.Interop.Excel;

namespace Coursework

{

public partial class frmStat : Form

{

//Функция для прямого доступа к ячейкам Exel ("A1" вместо (1, 1))

private string Excel\_cell(int row, int cl)

{

string name = "";

switch (cl)

{

case 1: { name = "A" + row.ToString(); break; }

case 2: { name = "B" + row.ToString(); break; }

case 3: { name = "C" + row.ToString(); break; }

case 4: { name = "D" + row.ToString(); break; }

case 5: { name = "E" + row.ToString(); break; }

case 6: { name = "F" + row.ToString(); break; }

case 7: { name = "G" + row.ToString(); break; }

case 8: { name = "H" + row.ToString(); break; }

case 9: { name = "I" + row.ToString(); break; }

case 10: { name = "J" + row.ToString(); break; }

case 11: { name = "K" + row.ToString(); break; }

case 12: { name = "L" + row.ToString(); break; }

case 13: { name = "M" + row.ToString(); break; }

case 14: { name = "N" + row.ToString(); break; }

case 15: { name = "O" + row.ToString(); break; }

case 16: { name = "P" + row.ToString(); break; }

case 17: { name = "Q" + row.ToString(); break; }

case 18: { name = "R" + row.ToString(); break; }

case 19: { name = "S" + row.ToString(); break; }

case 20: { name = "T" + row.ToString(); break; }

}

return name;

}

private Excel.Application exapp;

private Excel.Workbook ex;

public frmStat()

{

InitializeComponent();

}

string[] files;

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

{

this.MinimumSize = this.Size;

files = Directory.GetFiles(System.Windows.Forms.Application.StartupPath + "\\excel");

Console.WriteLine(files.Length);

string[] temp;

cmbYear.Items.Add("Все");

foreach (string file in files)

{

temp = file.Split('\\');

string filename = temp[temp.Length - 1];

string another\_year = filename.Split('\_')[0];

//Заполняем комбо-бокс с годами

if(!cmbYear.Items.Contains(another\_year))

{

cmbYear.Items.Add(filename.Split('\_')[0]);

}

}

}

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

{

if (cmbYear.SelectedIndex == -1)

{

MessageBox.Show("Рассматриваемый период не выбран!");

return;

}

string year = this.cmbYear.SelectedItem.ToString();

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

List<int> dats = new List<int>();

//Добыча данных из Exel при помощи его инструментов

exapp = new Excel.Application();

string[] temp;

foreach (string file in files)

{

temp = file.Split('\\');

string filename = temp[temp.Length - 1];

if (filename.Split('\_')[0] == year | year == "Все")

{

ex = exapp.Workbooks.Open(System.Windows.Forms.Application.StartupPath + "\\excel\\" + filename, ReadOnly: false);

string[] name\_parts = filename.Split('\_');

string name = "";

for (int i = 1; i < name\_parts.Length - 2; i++) { name += name\_parts[i] + " "; }

name += name\_parts[name\_parts.Length - 2];

Excel.Worksheet pg = ex.Sheets[1];

//Поиск столбца с названием "Количество часов"

Excel.Range cl = pg.get\_Range("1:1").Find("Кол-во часов");

//Поиск строки с названием "Всего за семестр"

Excel.Range row = pg.get\_Range("A:A").Find("Всего за семестр");

//Данные на их пересечении

int data;

bool parced = int.TryParse(pg.get\_Range(Excel\_cell(row.Row, cl.Column)).Text, out data);

Console.WriteLine(parced);

if (names.Contains(name) & parced)

{

dats[names.IndexOf(name)] += data;

}

else

{

if (parced)

{

names.Add(name);

dats.Add(data);

}

}

ex.Close();

}

}

exapp.Quit();

chrtStat.Series.Clear();

chrtStat.Titles.Clear();

chrtStat.Titles.Add("Суммарная нагрузка за " + year + " год");

chrtStat.ChartAreas[0].AxisX.Maximum = 1 + 0.4;

chrtStat.ChartAreas[0].AxisX.Minimum = 1 - 0.4;

chrtStat.ChartAreas[0].AxisX.LabelStyle.Enabled = false;

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

{

Series series = this.chrtStat.Series.Add(names[i]);

series.Points.Add(dats[i]);

series.ChartType = SeriesChartType.Column;

}

}

private void btn1semTime\_Click(object sender, EventArgs e)

{

if (cmbYear.SelectedIndex == -1)

{

MessageBox.Show("Рассматриваемый период не выбран!");

return;

}

string year = this.cmbYear.SelectedItem.ToString();

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

List<int> dats = new List<int>();

//Добыча данных из Exel при помощи его инструментов

exapp = new Excel.Application();

string[] temp;

foreach (string file in files)

{

temp = file.Split('\\');

string filename = temp[temp.Length - 1];

string[] name\_parts = filename.Split('\_');

string sem = name\_parts[name\_parts.Length - 1].Split('.')[0];

if ((filename.Split('\_')[0] == year | year == "Все") & sem == "II")

{

Console.WriteLine(filename);

ex = exapp.Workbooks.Open(System.Windows.Forms.Application.StartupPath + "\\excel\\" + filename, ReadOnly: false);

string name = "";

for (int i = 1; i < name\_parts.Length - 2; i++) { name += name\_parts[i] + " "; }

name += name\_parts[name\_parts.Length - 2];

Excel.Worksheet pg = ex.Sheets[1];

//Поиск столбца с названием "Количество часов"

Excel.Range cl = pg.get\_Range("1:1").Find("Кол-во часов");

//Поиск строки с названием "Всего за семестр"

Excel.Range row = pg.get\_Range("A:A").Find("Всего за семестр");

//Данные на их пересечении

int data = 0;

bool parced = int.TryParse(pg.get\_Range(Excel\_cell(row.Row, cl.Column)).Text, out data);

if (names.Contains(name) & parced)

{

dats[names.IndexOf(name)] += data;

}

else

{

if (parced)

{

names.Add(name);

dats.Add(data);

}

}

ex.Close();

}

}

exapp.Quit();

chrtStat.Series.Clear();

chrtStat.Titles.Clear();

chrtStat.Titles.Add("Суммарная нагрузка за 1 семестр " + year + " года");

chrtStat.ChartAreas[0].AxisX.Maximum = 1 + 0.4;

chrtStat.ChartAreas[0].AxisX.Minimum = 1 - 0.4;

chrtStat.ChartAreas[0].AxisX.LabelStyle.Enabled = false;

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

{

Series series = this.chrtStat.Series.Add(names[i]);

series.Points.Add(dats[i]);

series.ChartType = SeriesChartType.Column;

}

}

private void btn2semTime\_Click(object sender, EventArgs e)

{

if (cmbYear.SelectedIndex == -1)

{

MessageBox.Show("Рассматриваемый период не выбран!");

return;

}

string year = this.cmbYear.SelectedItem.ToString();

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

List<int> dats = new List<int>();

//Добыча данных из Exel при помощи его инструментов

exapp = new Excel.Application();

string[] temp;

foreach (string file in files)

{

temp = file.Split('\\');

string filename = temp[temp.Length - 1];

string[] name\_parts = filename.Split('\_');

string sem = name\_parts[name\_parts.Length - 1].Split('.')[0];

if ((filename.Split('\_')[0] == year | year == "Все") & sem == "I")

{

Console.WriteLine(filename);

ex = exapp.Workbooks.Open(System.Windows.Forms.Application.StartupPath + "\\excel\\" + filename, ReadOnly: false);

string name = "";

for (int i = 1; i < name\_parts.Length - 2; i++) { name += name\_parts[i] + " "; }

name += name\_parts[name\_parts.Length - 2];

Excel.Worksheet pg = ex.Sheets[1];

//Поиск столбца с названием "Количество часов"

Excel.Range cl = pg.get\_Range("1:1").Find("Кол-во часов");

//Поиск строки с названием "Всего за семестр"

Excel.Range row = pg.get\_Range("A:A").Find("Всего за семестр");

//Данные на их пересечении

int data = 0;

bool parced = int.TryParse(pg.get\_Range(Excel\_cell(row.Row, cl.Column)).Text, out data);

if (names.Contains(name) & parced)

{

dats[names.IndexOf(name)] += data;

}

else

{

if (parced)

{

names.Add(name);

dats.Add(data);

}

}

ex.Close();

}

}

exapp.Quit();

chrtStat.Series.Clear();

chrtStat.Titles.Clear();

chrtStat.Titles.Add("Суммарная нагрузка за 2 семестр " + year + " года");

chrtStat.ChartAreas[0].AxisX.Maximum = 1 + 0.4;

chrtStat.ChartAreas[0].AxisX.Minimum = 1 - 0.4;

chrtStat.ChartAreas[0].AxisX.LabelStyle.Enabled = false;

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

{

Series series = this.chrtStat.Series.Add(names[i]);

series.Points.Add(dats[i]);

series.ChartType = SeriesChartType.Column;

}

}

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

{

if (cmbYear.SelectedIndex == -1)

{

MessageBox.Show("Рассматриваемый период не выбран!");

return;

}

string year = this.cmbYear.SelectedItem.ToString();

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

List<int> dats = new List<int>();

//Добыча данных из Exel при помощи его инструментов

exapp = new Excel.Application();

string[] temp;

foreach (string file in files)

{

temp = file.Split('\\');

string filename = temp[temp.Length - 1];

string[] name\_parts = filename.Split('\_');

if (filename.Split('\_')[0] == year | year == "Все")

{

ex = exapp.Workbooks.Open(System.Windows.Forms.Application.StartupPath + "\\excel\\" + filename, ReadOnly: false);

string name = "";

for (int i = 1; i < name\_parts.Length - 2; i++) { name += name\_parts[i] + " "; }

name += name\_parts[name\_parts.Length - 2];

Excel.Worksheet pg = ex.Sheets[1];

//Поиск строки с названием "Лекции"

Excel.Range cl ;

cl = pg.get\_Range("C1:C65536").Find("Лекции");

while(cl != null)

{

//Смещение выделения в клетку с количеством часов

Excel.Range data\_cl = cl.Offset[0,1];

int data = 0;

bool parced = int.TryParse(data\_cl.Text, out data);

if (names.Contains(name) & parced)

{

dats[names.IndexOf(name)] += data;

}

else

{

if (parced)

{

names.Add(name);

dats.Add(data);

}

}

cl = pg.get\_Range("C" + (cl.Row + 1).ToString() + ":C65536").Find("Лекции");

}

ex.Close();

}

}

exapp.Quit();

chrtStat.Series.Clear();

chrtStat.Titles.Clear();

chrtStat.Titles.Add("Суммарная лекционная нагрузка за " + year + " год");

chrtStat.ChartAreas[0].AxisX.Maximum = 1 + 0.4;

chrtStat.ChartAreas[0].AxisX.Minimum = 1 - 0.4;

chrtStat.ChartAreas[0].AxisX.LabelStyle.Enabled = false;

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

{

Series series = this.chrtStat.Series.Add(names[i]);

series.Points.Add(dats[i]);

series.ChartType = SeriesChartType.Column;

}

}

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

{

if (cmbYear.SelectedIndex == -1)

{

MessageBox.Show("Рассматриваемый период не выбран!");

return;

}

string year = this.cmbYear.SelectedItem.ToString();

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

List<int> dats = new List<int>();

//Добыча данных из Exel при помощи его инструментов

exapp = new Excel.Application();

string[] temp;

foreach (string file in files)

{

temp = file.Split('\\');

string filename = temp[temp.Length - 1];

string[] name\_parts = filename.Split('\_');

if (filename.Split('\_')[0] == year | year == "Все")

{

ex = exapp.Workbooks.Open(System.Windows.Forms.Application.StartupPath + "\\excel\\" + filename, ReadOnly: false);

string name = "";

for (int i = 1; i < name\_parts.Length - 2; i++) { name += name\_parts[i] + " "; }

name += name\_parts[name\_parts.Length - 2];

Excel.Worksheet pg = ex.Sheets[1];

//Поиск строки с названием "Лабораторные работы"

Excel.Range cl;

cl = pg.get\_Range("C1:C65536").Find("Лабораторные работы");

while (cl != null)

{

//Смещение выделения в клетку с количеством часов

Excel.Range data\_cl = cl.Offset[0, 1];

int data = 0;

bool parced = int.TryParse(data\_cl.Text, out data);

if (names.Contains(name) & parced)

{

dats[names.IndexOf(name)] += data;

}

else

{

if (parced)

{

names.Add(name);

dats.Add(data);

}

}

cl = pg.get\_Range("C" + (cl.Row + 1).ToString() + ":C65536").Find("Лабораторные работы");

}

ex.Close();

}

}

exapp.Quit();

chrtStat.Series.Clear();

chrtStat.Titles.Clear();

chrtStat.Titles.Add("Суммарная лабораторная нагрузка за " + year + " год");

chrtStat.ChartAreas[0].AxisX.Maximum = 1 + 0.4;

chrtStat.ChartAreas[0].AxisX.Minimum = 1 - 0.4;

chrtStat.ChartAreas[0].AxisX.LabelStyle.Enabled = false;

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

{

Series series = this.chrtStat.Series.Add(names[i]);

series.Points.Add(dats[i]);

series.ChartType = SeriesChartType.Column;

}

}

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

{

if (cmbYear.SelectedIndex == -1)

{

MessageBox.Show("Рассматриваемый период не выбран!");

return;

}

string year = this.cmbYear.SelectedItem.ToString();

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

List<int> dats = new List<int>();

//Добыча данных из Exel при помощи его инструментов

exapp = new Excel.Application();

string[] temp;

foreach (string file in files)

{

temp = file.Split('\\');

string filename = temp[temp.Length - 1];

string[] name\_parts = filename.Split('\_');

if (filename.Split('\_')[0] == year | year == "Все")

{

ex = exapp.Workbooks.Open(System.Windows.Forms.Application.StartupPath + "\\excel\\" + filename, ReadOnly: false);

string name = "";

for (int i = 1; i < name\_parts.Length - 2; i++) { name += name\_parts[i] + " "; }

name += name\_parts[name\_parts.Length - 2];

Excel.Worksheet pg = ex.Sheets[1];

//Поиск строки с названием "Практические занятия"

Excel.Range cl;

cl = pg.get\_Range("C1:C65536").Find("Практические занятия");

while (cl != null)

{

//Смещение выделения в клетку с количеством часов

Excel.Range data\_cl = cl.Offset[0, 1];

int data = 0;

bool parced = int.TryParse(data\_cl.Text, out data);

if (names.Contains(name) & parced)

{

dats[names.IndexOf(name)] += data;

}

else

{

if (parced)

{

names.Add(name);

dats.Add(data);

}

}

cl = pg.get\_Range("C" + (cl.Row + 1).ToString() + ":C65536").Find("Практические занятия");

}

ex.Close();

}

}

exapp.Quit();

chrtStat.Series.Clear();

chrtStat.Titles.Clear();

chrtStat.Titles.Add("Суммарная практическая нагрузка за " + year + " год");

chrtStat.ChartAreas[0].AxisX.Maximum = 1 + 0.4;

chrtStat.ChartAreas[0].AxisX.Minimum = 1 - 0.4;

chrtStat.ChartAreas[0].AxisX.LabelStyle.Enabled = false;

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

{

Series series = this.chrtStat.Series.Add(names[i]);

series.Points.Add(dats[i]);

series.ChartType = SeriesChartType.Column;

}

}

}

}

## **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 также стал приобретённый навык построения схем Петри процессов и 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.

Код представляет собой набор методов, которые позволяют пользователю взаимодействовать с документами Microsoft Word и Excel. Он включает в себя функции для открытия и сохранения файлов, а также для импорта данных из CSV-файла в Word-документ. Основным результатом работы стала программа, способная обрабатывать печатные формы с индивидуальными планами преподавателей, конвертировать их в другие форматы. Программа способна строить диаграммы для анализа данных, собранных в результате её эксплуатации.

## **Библиографический список**

1. Лекционный материал по дисциплине «Методы программирования» Сафронов А. И., 2024.