

**МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»**

**(РУТ (МИИТ)**

**ИНСТИТУТ ТРАНСПОРТНОЙ ТЕХНИКИ И СИСТЕМ УПРАВЛЕНИЯ**

**Курсовая работа**

**по дисциплине** «**Методы программирования**»

**Выполнил:** ст. гр. ТКИ-342

Речкалов Р.Ю.

Ярий А.А.

**Проверил:** к.т.н. Сафронов А.И.

**Москва – 2024 г**

Содержание

[1 Введение 3](#_Toc166350403)

[2 Цель работы 3](#_Toc166350404)

[3 Формулировка задачи 3](#_Toc166350405)

[4 Содержательная часть работы 4](#_Toc166350406)

[4.1 Анализ печатной формы 4](#_Toc166350407)

[4.2 Описание предназначения печатной формы 4](#_Toc166350408)

[4.3 Разнообразие печатных форм и выделение различий 4](#_Toc166350409)

[4.4 Унификация печатных форм для удобства обработки данных 4](#_Toc166350410)

[4.5 Разработка экранной формы для импорта сведений из печатной формы 5](#_Toc166350411)

[4.6 Разработка функции для конвертации файла из Excel в Word 7](#_Toc166350412)

[4.7 Разработка функции экспорта сведений из экранной формы в Excel 10](#_Toc166350414)

[4.8 Графический анализ данных, содержащихся в печатных формах 11](#_Toc166350415)

5 [Инструкция пользователя 17](#_Toc166350419)

[6 Заключение 19](#_Toc166350420)

[7 Выводы 19](#_Toc166350421)

# 1 Введение

# В рамках данной работы рассмотрена автоматизация чтения данных из документа Excel, и последующая конвертация в Word. Так же по документам строится статистика в виде столбчатых диаграмм, показывающих перевыполненную нагрузку преподавателей и отклонение от нормы.

# 2 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.

# 3 Формулировка задачи

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 / Excel - Word (в зависимости от исходно выданного документа).

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".

# Содержательная часть работы

## **4.1 Анализ печатных форм**

4.1.1 Цели анализа печатных форм

Анализ печатных форм ведомостей необходим для понимания структуры данных, которые используются для учета нагрузки преподавателей. Это важно для правильного импорта, обработки и анализа данных в приложении. Анализ позволяет выявить все необходимые поля, их форматы и возможные значения, а также обеспечить корректное отображение и обработку информации.

4.1.2 Основные разделы и поля печатной формы

Печатные формы ведомостей включают несколько ключевых разделов, каждый из которых содержит определенные поля. Ниже перечислены основные разделы и поля:

* Информация о преподавателе:
* Имя преподавателя: ФИО преподавателя.
* Ставка: Доля ставки, занимаемой преподавателем (например, 0.5, 1.0).

Виды деятельности:

* Лекции: Количество проведенных лекций.
* Экзамены: Количество проведенных экзаменов.
* Зачеты: Количество проведенных зачетов.
* ПК (Промежуточный контроль): Количество проведенных промежуточных контрольных мероприятий.
* Консультации: Количество проведенных консультаций.
* Практические занятия: Количество проведенных практических занятий.
* Рефераты, дом. задания, эссе: Количество проверенных рефератов, домашних заданий и эссе.
* Текущая аттестация: Количество текущих аттестаций.
* КСР (Контроль самостоятельной работы): Количество проверенных контрольных работ.
* Курсовые проекты, курсовые работы: Количество руководимых курсовых проектов и работ.
* Подготовка ВКР (Выпускная квалификационная работа): Количество руководимых ВКР.
* Учебная практика: Количество проведенных учебных практик.
* Преддипломная и производственная практика: Количество проведенных преддипломных и производственных практик.
* Государственные экзамены: Количество проведенных государственных экзаменов.
* Приёмная комиссия: Участие в приемной комиссии.
* Лабораторные работы: Количество проведенных лабораторных работ.
* Аспирантура: Работа с аспирантами.
* Посещение занятий: Посещение занятий других преподавателей.
* Другие виды занятий: Прочие виды деятельности.

Нагрузка:

* Планируемая нагрузка: Планируемое количество часов по каждому виду деятельности.
* Фактическая нагрузка: Фактическое количество часов по каждому виду деятельности.
* Разница (Y - X): Разница между планируемой и фактической нагрузкой (перевыполненная или невыполненная нагрузка).

Дополнительные сведения:

* Примечания: Примечания к ведомости, включающие дополнительные сведения или комментарии.

4.1.3 Важность каждого поля

* Имя преподавателя: Идентификация преподавателя.
* Ставка: Определение объема работы, возложенного на преподавателя.
* Виды деятельности: Подробное описание выполняемой работы.
* Нагрузка: Оценка выполненной и планируемой работы, определение перевыполненной или невыполненной нагрузки.
* Примечания: Дополнительные сведения, которые могут повлиять на анализ данных.

4.1.4 Формат данных и примеры данных

Формат данных включает строковые и числовые значения. Например:

* Имя преподавателя: "Иванов И.И."
* Ставка: 1.0
* Лекции: 30
* Экзамены: 5
* Планируемая нагрузка: 40
* Фактическая нагрузка: 50
* Примечания: "Дополнительные часы по договоренности"

4.1.5 Возможные проблемы и несоответствия

* Некорректные или пропущенные данные: Возможны ситуации, когда данные введены некорректно или отсутствуют.
* Несоответствие форматов: Разные ведомости могут иметь разные форматы, что затрудняет их обработку.
* Различия в терминах: Разные ведомости могут использовать различные термины для обозначения одного и того же вида деятельности.

## **4.2 Описание предназначения печатных форм**

## Печатная форма предназначена для учета и анализа рабочей нагрузки преподавателей. Она позволяет определить плановую и фактическую нагрузку, выявить отклонения от нормы и перевыполненную нагрузку. Эти данные могут использоваться для управления учебным процессом, распределения нагрузки и анализа эффективности работы преподавателей.

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

## Различия могут включать количество и типы полей, форматирование, используемые системы оценок и требования к дополнительным комментариям. Например, одна форма может включать поле "Аспирантура", а другая - нет. Также различия могут быть в формате данных, например, использование десятичных дробей или целых чисел для ставки.

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

## Унификация печатных форм необходима для автоматизации обработки данных и уменьшения вероятности ошибок. Это может включать стандартизацию форматов данных (например, все числа - десятичные дроби), использование одинаковых названий полей и унифицированных форматов представления данных. Унификация позволяет создать универсальный механизм импорта и экспорта данных, а также улучшить совместимость с различными системами управления учебным процессом.

## **4.5 Разработка экранной формы для импорта сведений из печатной формы**

Разработка экранной формы для импорта сведений из печатной формы включает создание удобного пользовательского интерфейса, который позволяет пользователю загружать и отображать данные из различных ведомостей. Основные задачи при разработке экранной формы включают:

* Создание интерфейса загрузки данных: Интерфейс должен включать элементы управления для выбора файла с ведомостью и запуска процесса загрузки данных. Это может быть реализовано с помощью комбинированного списка для выбора периода (весь период, по годам, по семестрам), а также выбора конкретного года и семестра.
* Отображение данных: Данные из загруженной ведомости должны быть отображены в удобном виде, позволяющем пользователю быстро оценить информацию. Это может включать таблицы, графики и другие визуальные элементы.
* Автоматическая загрузка данных: При запуске приложения данные из всех ведомостей, расположенных в папке "DataFiles", должны загружаться автоматически и группироваться по периодам (годам и семестрам). Пользователь должен иметь возможность выбирать различные периоды для анализа данных.
* Обработка и фильтрация данных: Экранная форма должна позволять пользователю фильтровать данные по выбранным критериям, таким как год и семестр. Это включает в себя правильное определение и обработку семестра из загруженных данных.
* Построение диаграмм: Важной функцией является построение различных типов диаграмм (перевыполненной нагрузки, невыполненной нагрузки, отклонения от нормы) на основе выбранных данных. Диаграммы должны быть наглядными и содержать легенду с фамилиями преподавателей, которые присутствуют на графике, и их соответствующими цветами.
* Экспорт данных: Пользователь должен иметь возможность экспортировать загруженные и обработанные данные в Excel и Word. Экспортируемые файлы должны сохранять структуру и формат данных, обеспечивая удобство дальнейшей работы с ними.

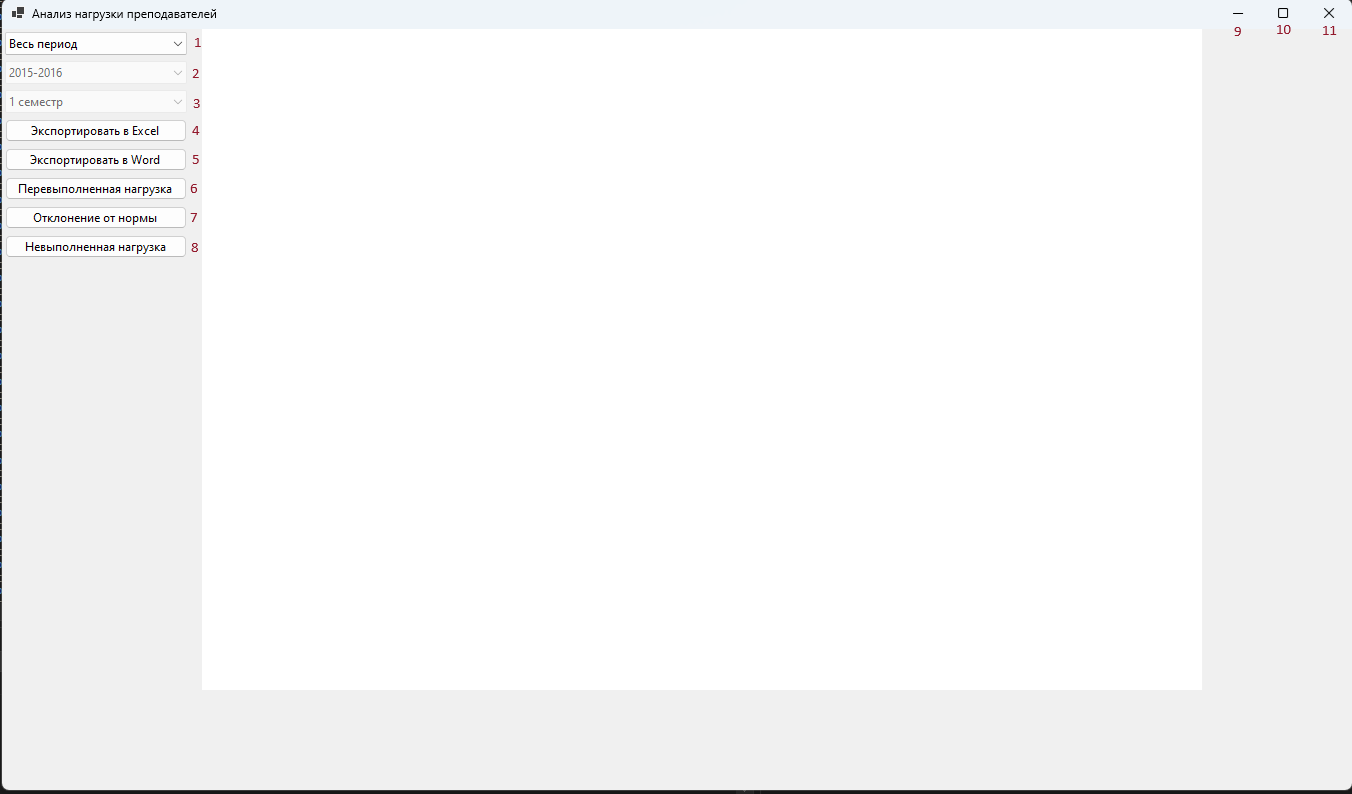
****

Рисунок 1 – Графический пользовательский интерфейс

1. Комбинированный список для выбора периода.
2. Комбинированный список для выбора временного промежутка.
3. Комбинированный список для выбора семестра.
4. Кнопка экспорта данных в Excel.
5. Кнопка экспорта данных в Word.
6. Кнопка построения диаграммы перевыполненной нагрузки.
7. Кнопка построения диаграммы отклонения от нормы.
8. Кнопка построения диаграммы невыполненной нагрузки.
9. Кнопка свернуть.
10. Кнопка на весь экран.
11. Кнопка закрытия.

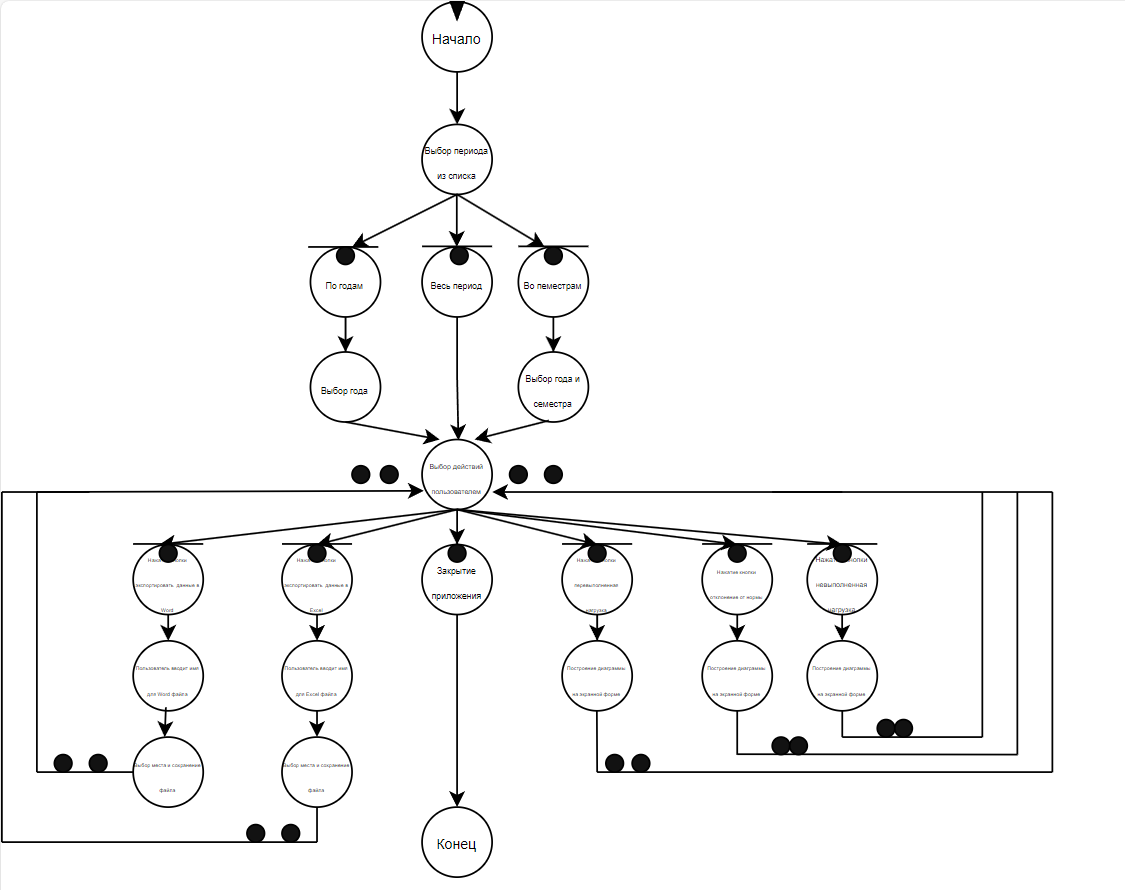


Рисунок 2 – Сеть петри

Листинг кода для реализации экранной формы:

private void InitializeComponent()

{

legendPanel = new FlowLayoutPanel

{

Dock = DockStyle.Bottom,

Height = 100,

FlowDirection = FlowDirection.LeftToRight,

AutoScroll = true

};

chart = new Chart();

periodComboBox = new ComboBox();

yearComboBox = new ComboBox();

semesterComboBox = new ComboBox();

exportToExcelButton = new Button();

exportToWordButton = new Button();

showTotalOverdoneLoadButton = new Button();

showDeviationFromNormButton = new Button();

showTotalUndoneLoadButton = new Button();

controlsPanel = new FlowLayoutPanel();

controlsPanel.SuspendLayout();

SuspendLayout();

//

// periodComboBox

//

periodComboBox.DropDownStyle = ComboBoxStyle.DropDownList;

periodComboBox.Items.AddRange(new string[] { "Весь период", "По годам", "По семестрам" });

periodComboBox.SelectedIndex = 0;

periodComboBox.Location = new Point(3, 3);

periodComboBox.Name = "periodComboBox";

periodComboBox.Size = new Size(182, 23);

periodComboBox.TabIndex = 0;

periodComboBox.SelectedIndexChanged += PeriodComboBox\_SelectedIndexChanged;

SetControlsState();

//

// yearComboBox

//

yearComboBox.DropDownStyle = ComboBoxStyle.DropDownList;

yearComboBox.Items.AddRange(new string[] { "2015-2016", "2016-2017", "2017-2018", "2018-2019" });

yearComboBox.SelectedIndex = 0;

yearComboBox.Location = new Point(3, 32);

yearComboBox.Name = "yearComboBox";

yearComboBox.Size = new Size(182, 23);

yearComboBox.TabIndex = 1;

//

// semesterComboBox

//

semesterComboBox.DropDownStyle = ComboBoxStyle.DropDownList;

semesterComboBox.Items.AddRange(new string[] { "1 семестр", "2 семестр" });

semesterComboBox.SelectedIndex = 0;

semesterComboBox.Location = new Point(3, 61);

semesterComboBox.Name = "semesterComboBox";

semesterComboBox.Size = new Size(182, 23);

semesterComboBox.TabIndex = 2;

//

// exportToExcelButton

//

exportToExcelButton.Location = new Point(3, 90);

exportToExcelButton.Name = "exportToExcelButton";

exportToExcelButton.Size = new Size(182, 23);

exportToExcelButton.TabIndex = 3;

exportToExcelButton.Text = "Экспортировать в Excel";

exportToExcelButton.Click += ExportToExcelButton\_Click;

//

// exportToWordButton

//

exportToWordButton.Location = new Point(3, 119);

exportToWordButton.Name = "exportToWordButton";

exportToWordButton.Size = new Size(182, 23);

exportToWordButton.TabIndex = 4;

exportToWordButton.Text = "Экспортировать в Word";

exportToWordButton.Click += ExportToWordButton\_Click;

//

// showTotalOverdoneLoadButton

//

showTotalOverdoneLoadButton.Location = new Point(3, 148);

showTotalOverdoneLoadButton.Name = "showTotalOverdoneLoadButton";

showTotalOverdoneLoadButton.Size = new Size(182, 23);

showTotalOverdoneLoadButton.TabIndex = 5;

showTotalOverdoneLoadButton.Text = "Перевыполненная нагрузка";

showTotalOverdoneLoadButton.Click += ShowTotalOverdoneLoadButton\_Click;

//

// showDeviationFromNormButton

//

showDeviationFromNormButton.Location = new Point(3, 177);

showDeviationFromNormButton.Name = "showDeviationFromNormButton";

showDeviationFromNormButton.Size = new Size(182, 23);

showDeviationFromNormButton.TabIndex = 6;

showDeviationFromNormButton.Text = "Отклонение от нормы";

showDeviationFromNormButton.Click += ShowDeviationFromNormButton\_Click;

//

// showTotalUndoneLoadButton

//

showTotalUndoneLoadButton.Location = new Point(3, 206);

showTotalUndoneLoadButton.Name = "showTotalUndoneLoadButton";

showTotalUndoneLoadButton.Size = new Size(182, 23);

showTotalUndoneLoadButton.TabIndex = 7;

showTotalUndoneLoadButton.Text = "Невыполненная нагрузка";

showTotalUndoneLoadButton.Click += ShowTotalUndoneLoadButton\_Click;

//

// controlsPanel

//

controlsPanel.Controls.Add(periodComboBox);

controlsPanel.Controls.Add(yearComboBox);

controlsPanel.Controls.Add(semesterComboBox);

controlsPanel.Controls.Add(exportToExcelButton);

controlsPanel.Controls.Add(exportToWordButton);

controlsPanel.Controls.Add(showTotalOverdoneLoadButton);

controlsPanel.Controls.Add(showDeviationFromNormButton);

controlsPanel.Controls.Add(showTotalUndoneLoadButton);

controlsPanel.Dock = DockStyle.Left;

controlsPanel.FlowDirection = FlowDirection.TopDown;

controlsPanel.Location = new Point(0, 0);

controlsPanel.Name = "controlsPanel";

controlsPanel.Size = new Size(200, 561);

controlsPanel.TabIndex = 0;

//

// MainForm

//

ClientSize = new Size(1350, 761);

Controls.Add(controlsPanel);

Controls.Add(chart);

Controls.Add(legendPanel); // Добавление панели легенды

Name = "MainForm";

Text = "Анализ нагрузки преподавателей";

controlsPanel.ResumeLayout(false);

ResumeLayout(false);

}

**4.6 Разработка функций экспорта сведений из печатной формы**

Функция экспорта данных из печатной формы включает преобразование данных в соответствующий формат и сохранение их в файл. Это может включать выбор данных, преобразование их в строковый формат и запись в файл с использованием стандартных разделителей (например, запятая или точка с запятой). Также необходимо обеспечить обработку возможных ошибок при записи файла.

Листинг кода для экспорта данных:

private void LoadFilesIntoComboBox()

{

string folderPath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "DataFiles");

if (!Directory.Exists(folderPath))

{

Directory.CreateDirectory(folderPath);

MessageBox.Show("Папка DataFiles создана. Пожалуйста, добавьте файлы ведомостей в эту папку.", "Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);

return;

}

data = new List<ProfessorLoad>();

var files = Directory.GetFiles(folderPath, "\*.xlsx");

foreach (var file in files)

{

var period = ExtractPeriodFromFileName(file);

var semester = ExtractSemesterFromFileNameOrSheet(file);

data.AddRange(LoadDataFromFile(file, period, semester));

}

if (data.Count == 0)

{

MessageBox.Show("Файлы ведомостей не найдены. Пожалуйста, добавьте файлы в папку DataFiles.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private string ExtractPeriodFromFileName(string fileName)

{

var fileNameWithoutExtension = Path.GetFileNameWithoutExtension(fileName);

var parts = fileNameWithoutExtension.Split('.');

if (parts.Length > 0)

{

return parts[0].Trim();

}

return string.Empty;

}

private string ExtractSemesterFromFileNameOrSheet(string fileName)

{

var fileNameWithoutExtension = Path.GetFileNameWithoutExtension(fileName);

if (fileNameWithoutExtension.Contains("1 семестр"))

{

return "1 семестр";

}

else if (fileNameWithoutExtension.Contains("2 семестр"))

{

return "2 семестр";

}

else

{

using (var workbook = new XLWorkbook(fileName))

{

var worksheet = workbook.Worksheet(1);

return worksheet.Cell("XY4").GetValue<string>();

}

}

}

private List<ProfessorLoad> LoadDataFromFile(string fileName, string period, string semester)

{

var dataList = new List<ProfessorLoad>();

using (var workbook = new XLWorkbook(fileName))

{

var worksheet = workbook.Worksheet(1);

var lastRow = worksheet.LastRowUsed().RowNumber();

for (int rowNum = 6; rowNum <= lastRow; rowNum++)

{

var row = worksheet.Row(rowNum);

if (rowNum == 34)

continue;

var professorLoad = new ProfessorLoad

{

ProfessorName = row.Cell(2).GetValue<string>() ?? string.Empty,

Notes = row.Cell(27).GetValue<string>() ?? string.Empty,

Period = period,

Semester = semester

};

try

{

professorLoad.Rate = row.Cell(3).GetValue<decimal>();

}

catch (InvalidCastException)

{

continue;

}

professorLoad.Lectures = GetValueOrDefault<int>(row.Cell(4));

professorLoad.Exams = GetValueOrDefault<int>(row.Cell(5));

professorLoad.Credits = GetValueOrDefault<int>(row.Cell(6));

professorLoad.IntermediateControls = GetValueOrDefault<int>(row.Cell(7));

professorLoad.Consultations = GetValueOrDefault<int>(row.Cell(8));

professorLoad.PracticalClasses = GetValueOrDefault<int>(row.Cell(9));

professorLoad.Essays = GetValueOrDefault<int>(row.Cell(10));

professorLoad.CurrentCertification = GetValueOrDefault<int>(row.Cell(11));

professorLoad.KSR = GetValueOrDefault<int>(row.Cell(12));

professorLoad.ControlWorks = GetValueOrDefault<int>(row.Cell(13));

professorLoad.CourseProjects = GetValueOrDefault<int>(row.Cell(14));

professorLoad.ThesisPreparation = GetValueOrDefault<int>(row.Cell(15));

professorLoad.AcademicPractice = GetValueOrDefault<int>(row.Cell(16));

professorLoad.PreDiplomaPractice = GetValueOrDefault<int>(row.Cell(17));

professorLoad.StateExam = GetValueOrDefault<int>(row.Cell(18));

professorLoad.AdmissionCommission = GetValueOrDefault<int>(row.Cell(19));

professorLoad.LaboratoryWork = GetValueOrDefault<int>(row.Cell(20));

professorLoad.Postgraduate = GetValueOrDefault<int>(row.Cell(21));

professorLoad.ClassAttendance = GetValueOrDefault<int>(row.Cell(22));

professorLoad.OtherActivities = GetValueOrDefault<int>(row.Cell(23));

professorLoad.PlannedLoad = GetValueOrDefault<int>(row.Cell(24));

professorLoad.ActualLoad = GetValueOrDefault<int>(row.Cell(25));

professorLoad.Difference = GetValueOrDefault<int>(row.Cell(26));

dataList.Add(professorLoad);

}

}

return dataList;

}

private T GetValueOrDefault<T>(IXLCell cell)

{

try

{

return cell.GetValue<T>();

}

catch (InvalidCastException)

{

return default;

}

}

**4.7 Разработка функций импорта сведений из оригинального и изменённого текстовых файлов с воссозданием Excel**

Функция импорта данных из текстовых файлов включает чтение данных из файла, преобразование их в соответствующие объекты и воссоздание исходного документа Excel. Это может включать чтение строк из файла, разбор их на отдельные поля, создание объектов данных и запись этих данных в новый файл Excel с использованием библиотеки ClosedXML.

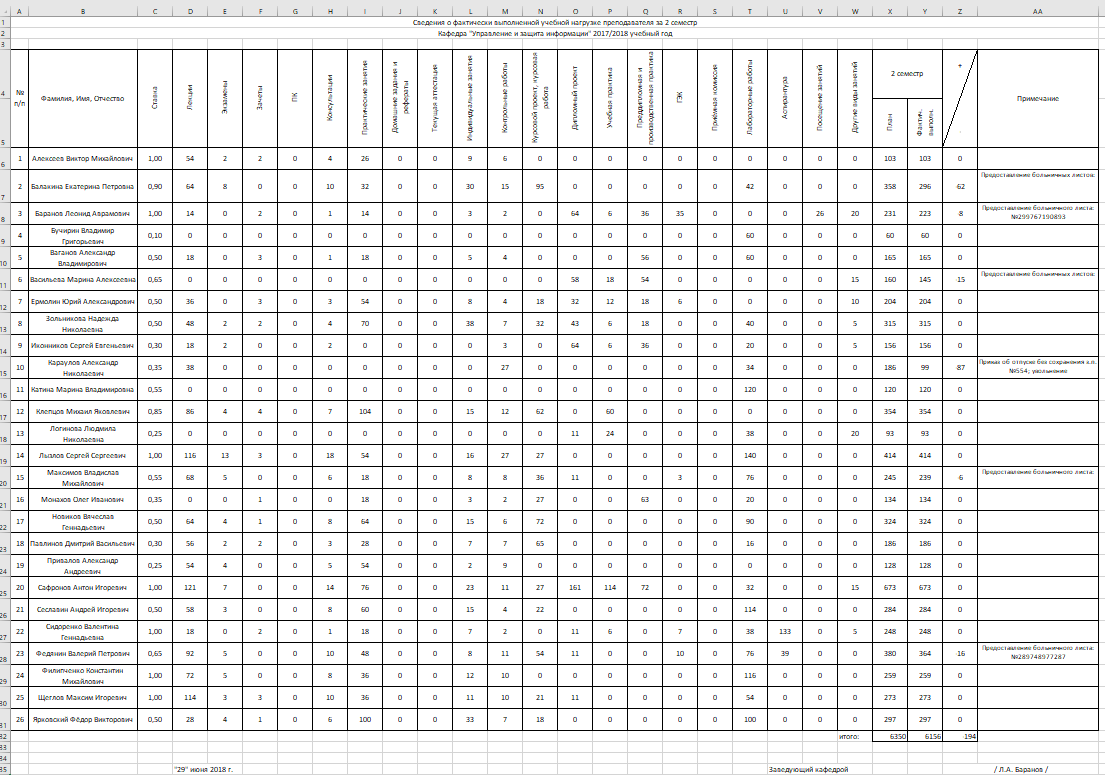


Рисунок 3 – Исходные данные

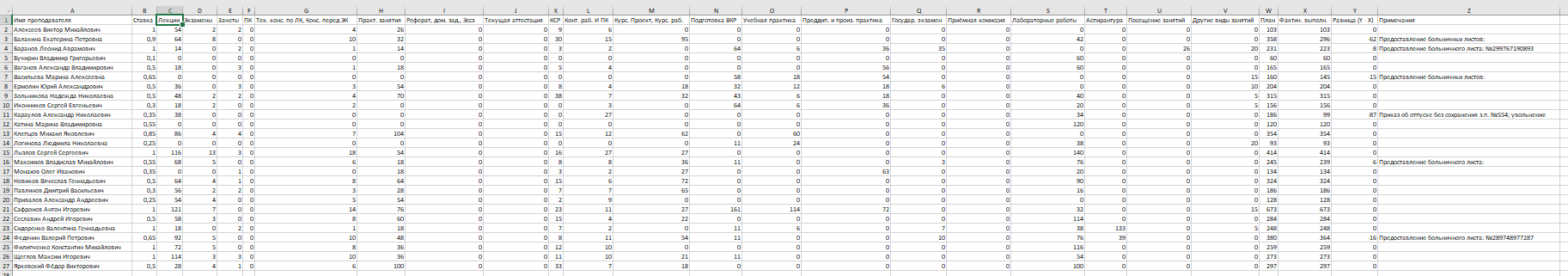


Рисунок 4 – Данные после экспорта в Excel

Листинг кода для функции импорта сведений:

private void ExportDataToExcel(List<ProfessorLoad> data, string filePath)

{

using (var workbook = new XLWorkbook())

{

var worksheet = workbook.AddWorksheet("Data");

worksheet.Cell(1, 1).Value = "Имя преподавателя";

worksheet.Cell(1, 2).Value = "Ставка";

worksheet.Cell(1, 3).Value = "Лекции";

worksheet.Cell(1, 4).Value = "Экзамены";

worksheet.Cell(1, 5).Value = "Зачеты";

worksheet.Cell(1, 6).Value = "ПК";

worksheet.Cell(1, 7).Value = "Тек. конс. по ЛК, Конс. перед ЭК";

worksheet.Cell(1, 8).Value = "Практ. занятия";

worksheet.Cell(1, 9).Value = "Реферат, дом. зад., Эссэ";

worksheet.Cell(1, 10).Value = "Текущая аттестация";

worksheet.Cell(1, 11).Value = "КСР";

worksheet.Cell(1, 12).Value = "Конт. раб. И ПК";

worksheet.Cell(1, 13).Value = "Курс. Проект, Курс. раб.";

worksheet.Cell(1, 14).Value = "Подготовка ВКР";

worksheet.Cell(1, 15).Value = "Учебная практика";

worksheet.Cell(1, 16).Value = "Преддип. и произ. практика";

worksheet.Cell(1, 17).Value = "Государ. экзамен";

worksheet.Cell(1, 18).Value = "Приёмная комиссия";

worksheet.Cell(1, 19).Value = "Лабораторные работы";

worksheet.Cell(1, 20).Value = "Аспирантура";

worksheet.Cell(1, 21).Value = "Посещение занятий";

worksheet.Cell(1, 22).Value = "Другие виды занятий";

worksheet.Cell(1, 23).Value = "План";

worksheet.Cell(1, 24).Value = "Фактич. выполн.";

worksheet.Cell(1, 25).Value = "Разница (Y - X)";

worksheet.Cell(1, 26).Value = "Примечания";

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

{

worksheet.Cell(i + 2, 1).Value = data[i].ProfessorName;

worksheet.Cell(i + 2, 2).Value = data[i].Rate;

worksheet.Cell(i + 2, 3).Value = data[i].Lectures;

worksheet.Cell(i + 2, 4).Value = data[i].Exams;

worksheet.Cell(i + 2, 5).Value = data[i].Credits;

worksheet.Cell(i + 2, 6).Value = data[i].IntermediateControls;

worksheet.Cell(i + 2, 7).Value = data[i].Consultations;

worksheet.Cell(i + 2, 8).Value = data[i].PracticalClasses;

worksheet.Cell(i + 2, 9).Value = data[i].Essays;

worksheet.Cell(i + 2, 10).Value = data[i].CurrentCertification;

worksheet.Cell(i + 2, 11).Value = data[i].KSR;

worksheet.Cell(i + 2, 12).Value = data[i].ControlWorks;

worksheet.Cell(i + 2, 13).Value = data[i].CourseProjects;

worksheet.Cell(i + 2, 14).Value = data[i].ThesisPreparation;

worksheet.Cell(i + 2, 15).Value = data[i].AcademicPractice;

worksheet.Cell(i + 2, 16).Value = data[i].PreDiplomaPractice;

worksheet.Cell(i + 2, 17).Value = data[i].StateExam;

worksheet.Cell(i + 2, 18).Value = data[i].AdmissionCommission;

worksheet.Cell(i + 2, 19).Value = data[i].LaboratoryWork;

worksheet.Cell(i + 2, 20).Value = data[i].Postgraduate;

worksheet.Cell(i + 2, 21).Value = data[i].ClassAttendance;

worksheet.Cell(i + 2, 22).Value = data[i].OtherActivities;

worksheet.Cell(i + 2, 23).Value = data[i].PlannedLoad;

worksheet.Cell(i + 2, 24).Value = data[i].ActualLoad;

worksheet.Cell(i + 2, 25).Value = data[i].Difference;

worksheet.Cell(i + 2, 26).Value = data[i].Notes;

}

workbook.SaveAs(filePath);

}

}

**4.8 Разработка функций экспорта сведений из экранной формы в Word**

Функция экспорта данных из экранной формы в Word включает преобразование данных, отображаемых в интерфейсе, в формат Word и запись их в файл. Это требует сбора данных из элементов управления, таких как DataGridView, и использования библиотеки OpenXML для создания и форматирования документа Word. Особое внимание уделяется настройке ориентации страницы (альбомная) и размера шрифта (7.5).

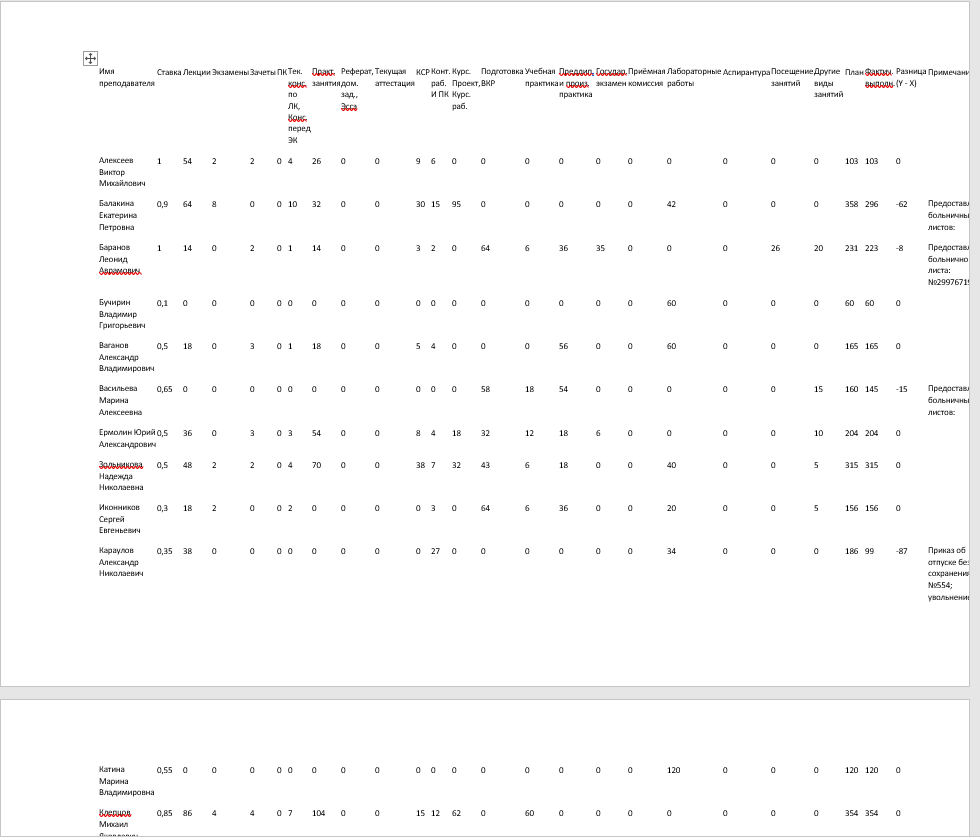


Рисунок 5 – Данные после экспорта в Word

Листинг кода для функции экспорта сведений:

private void ExportDataToWord(List<ProfessorLoad> data, string filePath)

{

using (var wordDocument = WordprocessingDocument.Create(filePath, WordprocessingDocumentType.Document))

{

var mainPart = wordDocument.AddMainDocumentPart();

var doc = new Document();

var body = new Body();

// Установка альбомной ориентации

var sectionProps = new SectionProperties(

new PageSize { Width = 16838, Height = 11906, Orient = PageOrientationValues.Landscape });

body.AppendChild(sectionProps);

var table = new Table();

var tableProperties = new TableProperties();

var tableWidth = new TableWidth { Width = "5000", Type = TableWidthUnitValues.Pct };

tableProperties.Append(tableWidth);

table.AppendChild(tableProperties);

var headerRow = new TableRow();

headerRow.Append(CreateTableCell("Имя преподавателя"));

headerRow.Append(CreateTableCell("Ставка"));

headerRow.Append(CreateTableCell("Лекции"));

headerRow.Append(CreateTableCell("Экзамены"));

headerRow.Append(CreateTableCell("Зачеты"));

headerRow.Append(CreateTableCell("ПК"));

headerRow.Append(CreateTableCell("Тек. конс. по ЛК, Конс. перед ЭК"));

headerRow.Append(CreateTableCell("Практ. занятия"));

headerRow.Append(CreateTableCell("Реферат, дом. зад., Эссэ"));

headerRow.Append(CreateTableCell("Текущая аттестация"));

headerRow.Append(CreateTableCell("КСР"));

headerRow.Append(CreateTableCell("Конт. раб. И ПК"));

headerRow.Append(CreateTableCell("Курс. Проект, Курс. раб."));

headerRow.Append(CreateTableCell("Подготовка ВКР"));

headerRow.Append(CreateTableCell("Учебная практика"));

headerRow.Append(CreateTableCell("Преддип. и произ. практика"));

headerRow.Append(CreateTableCell("Государ. экзамен"));

headerRow.Append(CreateTableCell("Приёмная комиссия"));

headerRow.Append(CreateTableCell("Лабораторные работы"));

headerRow.Append(CreateTableCell("Аспирантура"));

headerRow.Append(CreateTableCell("Посещение занятий"));

headerRow.Append(CreateTableCell("Другие виды занятий"));

headerRow.Append(CreateTableCell("План"));

headerRow.Append(CreateTableCell("Фактич. выполн."));

headerRow.Append(CreateTableCell("Разница (Y - X)"));

headerRow.Append(CreateTableCell("Примечания"));

table.AppendChild(headerRow);

foreach (var item in data)

{

var dataRow = new TableRow();

dataRow.Append(CreateTableCell(item.ProfessorName));

dataRow.Append(CreateTableCell(item.Rate.ToString()));

dataRow.Append(CreateTableCell(item.Lectures.ToString()));

dataRow.Append(CreateTableCell(item.Exams.ToString()));

dataRow.Append(CreateTableCell(item.Credits.ToString()));

dataRow.Append(CreateTableCell(item.IntermediateControls.ToString()));

dataRow.Append(CreateTableCell(item.Consultations.ToString()));

dataRow.Append(CreateTableCell(item.PracticalClasses.ToString()));

dataRow.Append(CreateTableCell(item.Essays.ToString()));

dataRow.Append(CreateTableCell(item.CurrentCertification.ToString()));

dataRow.Append(CreateTableCell(item.KSR.ToString()));

dataRow.Append(CreateTableCell(item.ControlWorks.ToString()));

dataRow.Append(CreateTableCell(item.CourseProjects.ToString()));

dataRow.Append(CreateTableCell(item.ThesisPreparation.ToString()));

dataRow.Append(CreateTableCell(item.AcademicPractice.ToString()));

dataRow.Append(CreateTableCell(item.PreDiplomaPractice.ToString()));

dataRow.Append(CreateTableCell(item.StateExam.ToString()));

dataRow.Append(CreateTableCell(item.AdmissionCommission.ToString()));

dataRow.Append(CreateTableCell(item.LaboratoryWork.ToString()));

dataRow.Append(CreateTableCell(item.Postgraduate.ToString()));

dataRow.Append(CreateTableCell(item.ClassAttendance.ToString()));

dataRow.Append(CreateTableCell(item.OtherActivities.ToString()));

dataRow.Append(CreateTableCell(item.PlannedLoad.ToString()));

dataRow.Append(CreateTableCell(item.ActualLoad.ToString()));

dataRow.Append(CreateTableCell(item.Difference.ToString()));

dataRow.Append(CreateTableCell(item.Notes));

table.AppendChild(dataRow);

}

body.Append(table);

doc.Append(body);

mainPart.Document = doc;

}

}

private TableCell CreateTableCell(string text)

{

var cell = new TableCell();

var paragraph = new Paragraph();

var run = new Run();

var runProperties = new RunProperties();

var fontSize = new FontSize { Val = "15" };

runProperties.Append(fontSize);

run.Append(runProperties);

run.Append(new Text(text));

paragraph.Append(run);

cell.Append(paragraph);

return cell;

}

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

Графический анализ данных включает использование Chart-элемента для визуализации данных о нагрузке преподавателей. Это требует настройки Chart-элемента для отображения гистограмм и графиков, включая настройку осей, меток, цвета столбцов и линий. Также необходимо обеспечить, чтобы все данные помещались на диаграмме, что достигается уменьшением размера столбцов и меток.

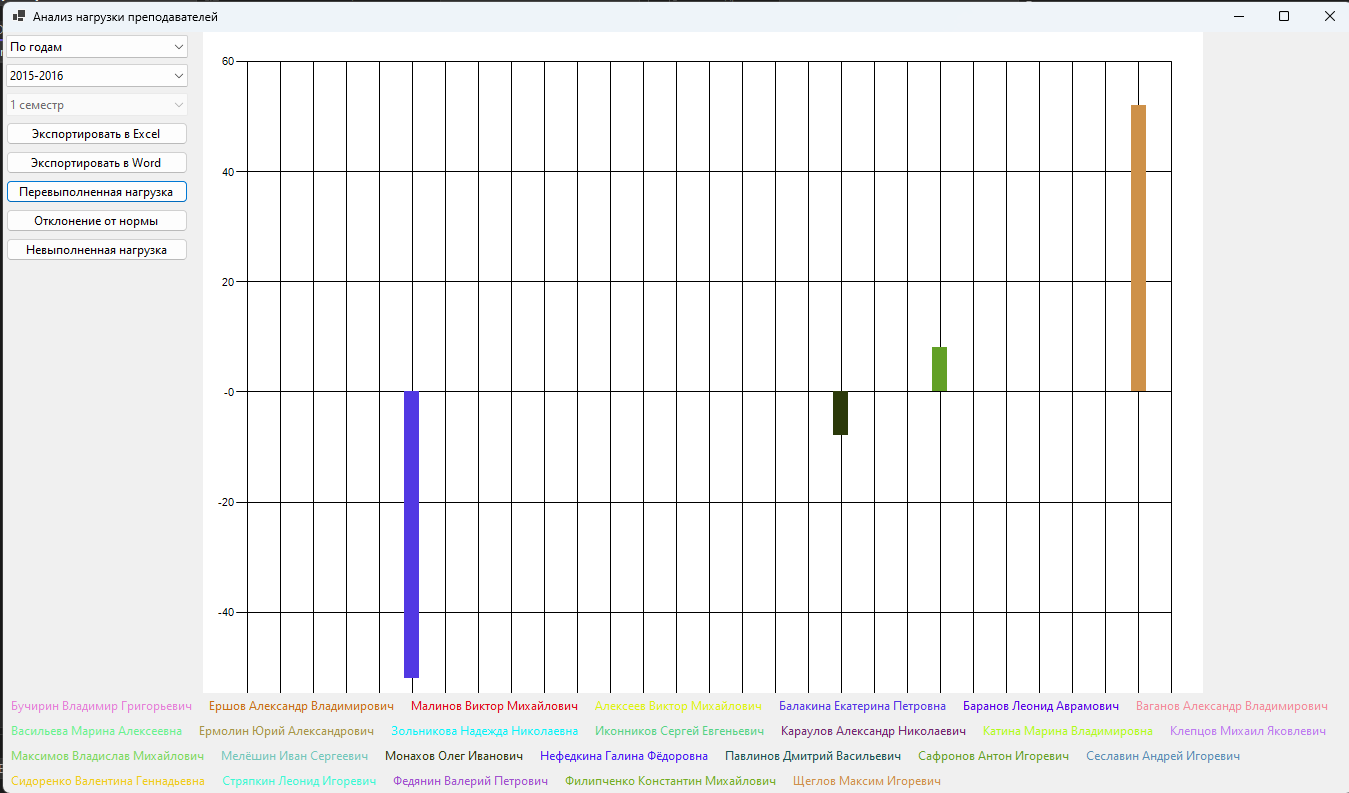


Рисунок 6 – График перевыполненной нагрузки

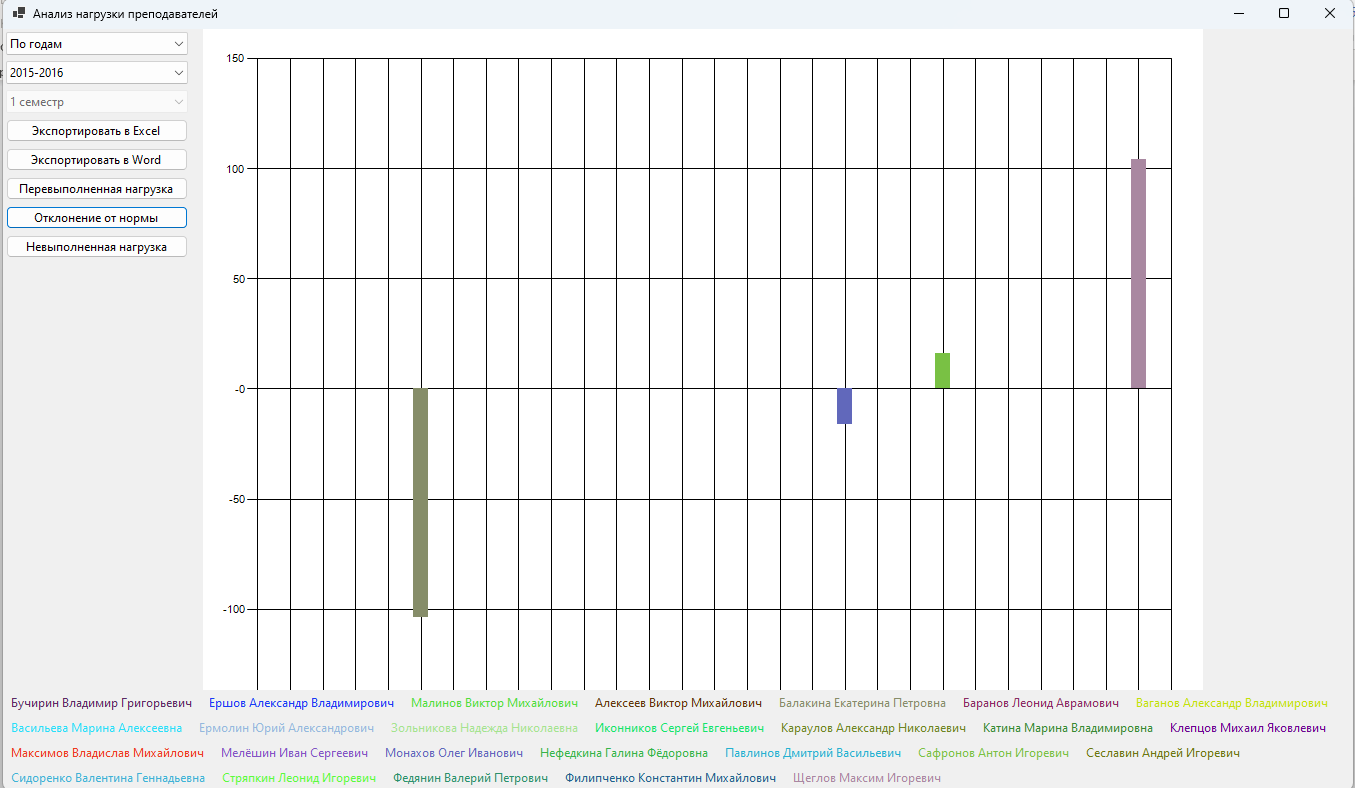


Рисунок 7 – График отклонения от нормы

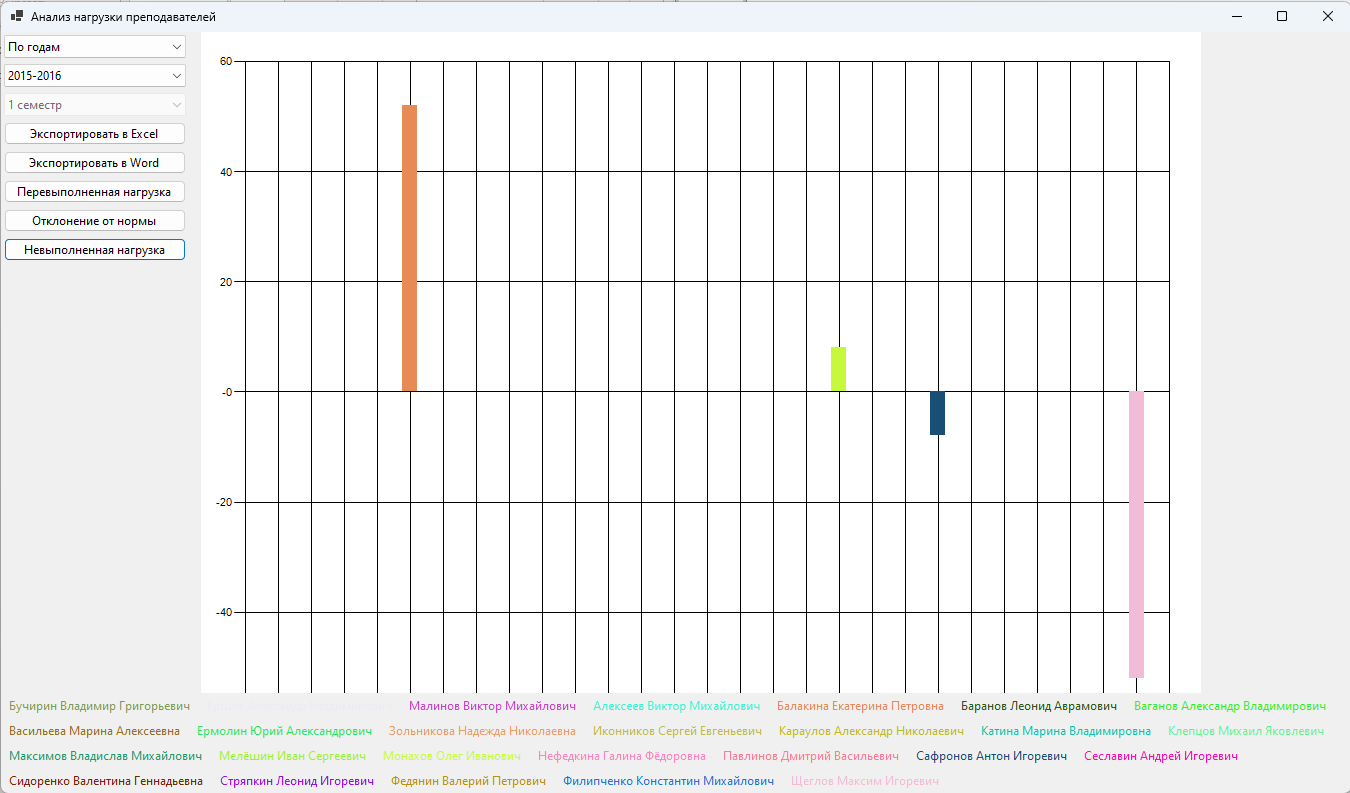


Рисунок 8 – График невыполненной нагрузки

Листинг кода для графического анализа данных:

private List<ProfessorLoad> FilterDataByPeriod(List<ProfessorLoad> data, string period, string year, string semester)

{

if (period == "Весь период")

{

return data;

}

else if (period == "По годам")

{

return data.Where(d => d.Period.Contains(year)).ToList();

}

else if (period == "По семестрам")

{

return data.Where(d => d.Period.Contains(year) && d.Semester == semester).ToList();

}

return new List<ProfessorLoad>();

}

private void ShowTotalOverdoneLoadChart(List<ProfessorLoad> data)

{

chart.Series.Clear();

chart.ChartAreas.Clear();

legendPanel.Controls.Clear(); // Очистить панель легенды

ChartArea chartArea = new ChartArea();

chart.ChartAreas.Add(chartArea);

var series = new Series("Перевыполненная нагрузка")

{

ChartType = SeriesChartType.Column,

["PointWidth"] = "0.4",

["PixelPointWidth"] = "15",

};

Random random = new Random();

Dictionary<string, System.Drawing.Color> professorColors = new Dictionary<string, System.Drawing.Color>();

foreach (var load in data)

{

int pointIndex = series.Points.AddY(load.OverdoneLoad);

System.Drawing.Color color = System.Drawing.Color.FromArgb(random.Next(256), random.Next(256), random.Next(256));

series.Points[pointIndex].Color = color;

professorColors[load.ProfessorName] = color;

}

chartArea.AxisX.Interval = 1;

chartArea.AxisX.LabelStyle.Enabled = false;

chartArea.AxisY.LabelStyle.Font = new System.Drawing.Font("Arial", 8);

chart.Series.Add(series);

chart.Invalidate();

foreach (var kvp in professorColors)

{

Label legendLabel = new Label

{

Text = kvp.Key,

ForeColor = kvp.Value,

AutoSize = true,

Margin = new Padding(5)

};

legendPanel.Controls.Add(legendLabel);

}

}

private void ShowDeviationFromNormChart(List<ProfessorLoad> data)

{

chart.Series.Clear();

chart.ChartAreas.Clear();

legendPanel.Controls.Clear(); // Очистить панель легенды

ChartArea chartArea = new ChartArea();

chart.ChartAreas.Add(chartArea);

var series = new Series("Отклонение от нормы")

{

ChartType = SeriesChartType.Column,

["PointWidth"] = "0.4",

["PixelPointWidth"] = "15",

};

Random random = new Random();

Dictionary<string, System.Drawing.Color> professorColors = new Dictionary<string, System.Drawing.Color>();

foreach (var load in data)

{

int pointIndex = series.Points.AddY(load.OverdoneLoad - load.UndoneLoad);

System.Drawing.Color color = System.Drawing.Color.FromArgb(random.Next(256), random.Next(256), random.Next(256));

series.Points[pointIndex].Color = color;

professorColors[load.ProfessorName] = color;

}

chartArea.AxisX.Interval = 1;

chartArea.AxisX.LabelStyle.Enabled = false;

chartArea.AxisY.LabelStyle.Font = new System.Drawing.Font("Arial", 8);

chart.Series.Add(series);

chart.Invalidate();

foreach (var kvp in professorColors)

{

Label legendLabel = new Label

{

Text = kvp.Key,

ForeColor = kvp.Value,

AutoSize = true,

Margin = new Padding(5)

};

legendPanel.Controls.Add(legendLabel);

}

}

private void ShowTotalUndoneLoadChart(List<ProfessorLoad> data)

{

chart.Series.Clear();

chart.ChartAreas.Clear();

legendPanel.Controls.Clear(); // Очистить панель легенды

ChartArea chartArea = new ChartArea();

chart.ChartAreas.Add(chartArea);

var series = new Series("Невыполненная нагрузка")

{

ChartType = SeriesChartType.Column,

["PointWidth"] = "0.4",

["PixelPointWidth"] = "15",

};

Random random = new Random();

Dictionary<string, System.Drawing.Color> professorColors = new Dictionary<string, System.Drawing.Color>();

foreach (var load in data)

{

int pointIndex = series.Points.AddY(load.UndoneLoad);

System.Drawing.Color color = System.Drawing.Color.FromArgb(random.Next(256), random.Next(256), random.Next(256));

series.Points[pointIndex].Color = color;

professorColors[load.ProfessorName] = color;

}

chartArea.AxisX.Interval = 1;

chartArea.AxisX.LabelStyle.Enabled = false;

chartArea.AxisY.LabelStyle.Font = new System.Drawing.Font("Arial", 8);

chart.Series.Add(series);

chart.Invalidate();

foreach (var kvp in professorColors)

{

Label legendLabel = new Label

{

Text = kvp.Key,

ForeColor = kvp.Value,

AutoSize = true,

Margin = new Padding(5)

};

legendPanel.Controls.Add(legendLabel);

}

}

private void CalculateOverdoneLoad(List<ProfessorLoad> data)

{

foreach (var load in data)

{

load.OverdoneLoad = load.ActualLoad - load.PlannedLoad;

}

}

private void CalculateDeviationFromNorm(List<ProfessorLoad> data)

{

foreach (var load in data)

{

load.UndoneLoad = load.PlannedLoad - load.ActualLoad;

}

}

private void CalculateUndoneLoad(List<ProfessorLoad> data)

{

foreach (var load in data)

{

load.UndoneLoad = load.PlannedLoad - load.ActualLoad;

}

}

**5 Инструкция пользователя**

1. Запуск программы

Запустите программу, дважды щелкнув по значку или через меню "Пуск".

При запуске программа автоматически загружает данные из всех ведомостей, находящихся в папке DataFiles.

2. Основной интерфейс программы

Интерфейс программы состоит из нескольких основных элементов:

Комбинированный список для выбора периода анализа (periodComboBox).

Комбинированный список для выбора года (yearComboBox).

Комбинированный список для выбора семестра (semesterComboBox).

Кнопки для построения различных диаграмм: "Перевыполненная нагрузка", "Отклонение от нормы", "Невыполненная нагрузка".

Кнопки для экспорта данных в Excel и Word.

Область для отображения диаграмм (chart).

Панель легенды для отображения цветов и имен преподавателей (legendPanel).

3. Выбор периода анализа

В комбинированном списке periodComboBox выберите интересующий вас период:

Весь период - для анализа данных за все годы.

По годам - для анализа данных по годам.

По семестрам - для анализа данных по семестрам.

При выборе По годам, комбинированный список semesterComboBox становится недоступным для выбора.

При выборе По семестрам, комбинированный список yearComboBox и semesterComboBox становятся доступными для выбора года и семестра соответственно.

4. Построение диаграмм

Выберите интересующий вас период, год и/или семестр.

Нажмите на одну из кнопок для построения диаграмм:

Перевыполненная нагрузка - для отображения диаграммы перевыполненной нагрузки преподавателей.

Отклонение от нормы - для отображения диаграммы отклонения от нормы нагрузки преподавателей.

Невыполненная нагрузка - для отображения диаграммы невыполненной нагрузки преподавателей.

5. Легенда диаграмм

После построения диаграммы, ниже графика будет отображена панель легенды с именами преподавателей и соответствующими им цветами, которые присутствуют на графике. Это позволяет легко сопоставить цвет столбцов на диаграмме с именами преподавателей.

6. Экспорт данных

Для экспорта данных в Excel нажмите кнопку Экспортировать в Excel.

Для экспорта данных в Word нажмите кнопку Экспортировать в Word.

В открывшемся диалоговом окне выберите место для сохранения файла и укажите его имя.

Нажмите кнопку Сохранить, чтобы завершить экспорт данных.

7. Общие рекомендации

Убедитесь, что все файлы ведомостей находятся в папке DataFiles, и они соответствуют формату, описанному в инструкции.

Если данные не загружаются или диаграммы не строятся, проверьте формат файлов и правильность указанного периода, года и семестра.

**6 Заключение**

В ходе работы над программой "Анализ нагрузки преподавателей" была разработана система для автоматизированного анализа и визуализации данных о нагрузке преподавателей. Программа позволяет пользователям легко загружать данные из ведомостей, отображать различные виды диаграмм для анализа перевыполненной, невыполненной нагрузки и отклонения от нормы. Кроме того, предусмотрены функции экспорта данных в форматы Excel и Word, что обеспечивает гибкость в использовании результатов анализа.

Основные достижения включают:

* Автоматическую загрузку данных из ведомостей, хранящихся в папке DataFiles.
* Поддержку анализа данных по различным периодам, включая весь период, по годам и по семестрам.
* Визуализацию данных с использованием цветовых обозначений для каждого преподавателя.
* Удобную панель легенды, которая отображает только тех преподавателей, чьи данные присутствуют на диаграмме.
* Функции экспорта данных в Excel и Word, что упрощает последующую обработку и представление данных.

**7 Выводы**

Разработка программы "Анализ нагрузки преподавателей" продемонстрировала важность автоматизации процесса анализа и представления данных. Основные выводы из проведенной работы включают:

* Эффективность автоматизации: Автоматическая загрузка и обработка данных из ведомостей существенно снижает время и усилия, необходимые для анализа нагрузки преподавателей.
* Гибкость анализа: Поддержка различных периодов анализа (весь период, по годам, по семестрам) позволяет пользователям получать детализированные и обобщенные данные в зависимости от их потребностей.
* Визуализация данных: Графическое представление данных с использованием цветовых обозначений улучшает восприятие информации и облегчает идентификацию ключевых моментов.
* Удобство экспорта: Возможность экспорта данных в Excel и Word обеспечивает гибкость в использовании результатов анализа для отчетности и дальнейшей работы.
* Интерактивность интерфейса: Интуитивно понятный интерфейс и удобные элементы управления (комбинированные списки, кнопки) делают программу доступной для пользователей с разным уровнем технической подготовки.

Программа "Анализ нагрузки преподавателей" представляет собой полезный инструмент для учебных заведений, позволяя легко и эффективно анализировать нагрузку преподавателей, выявлять перевыполненную и невыполненную нагрузку, а также отклонения от нормы. В дальнейшем возможно расширение функционала программы, например, добавление новых типов анализа или интеграция с другими системами управления учебным процессом.