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»

Выполнил: ст. гр. ТКИ-341

Пряхин И.Д.

Прокопович Д.Е.

Вариант №6

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

Москва – 2024 г.

Содержание

[1 Введение 3](#_Toc169840466)

[2 Цель работы 3](#_Toc169840467)

[3 Формулировка задачи 3](#_Toc169840468)

[4 Содержательная часть работы 4](#_Toc169840469)

[4.1 Анализ печатной формы 4](#_Toc169840470)

[4.2 Описание предназначения печатной формы 4](#_Toc169840471)

[4.3 Разнообразие печатных форм и выделение различий 4](#_Toc169840472)

[4.4 Унификация печатных форм для удобства обработки данных 4](#_Toc169840473)

[4.5 Считывание из Word на экранную форму 5](#_Toc169840474)

[4.6 Считывание из Excel на экранную форму 9](#_Toc169840475)

[4.7 Разработка функции конвертации документа Word в Excel 13](#_Toc169840476)

[4.8 Разработка функции конвертации документа Excel в Word 17](#_Toc169840477)

[4.9 Графический анализ данных, содержащихся в печатных формах 22](#_Toc169840478)

[4.9.1 Показывать столбчатую диаграмму по количеству изданий, запланированных под различные учебные дисциплины за всё время. 22](#_Toc169840479)

[4.9.2 Показывать столбчатую диаграмму по количеству изданий, запланированных под различные учебные дисциплины за выбранный год. 26](#_Toc169840480)

[4.9.3 Показывать столбчатую диаграмму по видам изданий за всё время. 30](#_Toc169840481)

[4.9.4 Показывать столбчатую диаграмму по видам изданий за выбранный год. 34](#_Toc169840482)

[4.9.5 Показывать столбчатую диаграмму по объёмам в авторских листах за всё время. 38](#_Toc169840483)

[4.9.6 Показывать столбчатую диаграмму по кварталам сдачи изданий за всё время. 42](#_Toc169840484)

[4.9.7 Показывать столбчатую диаграмму по кварталам сдачи изданий за выбранный год. 46](#_Toc169840485)

[4.10 Инструкция пользователя 50](#_Toc169840486)

[4.10.1 Главная форма 50](#_Toc169840487)

[4.10.2 Форма для работы с файлами Word 50](#_Toc169840488)

[4.10.3 Форма для работы с файлами Excel 51](#_Toc169840489)

[4.10.4 Форма для отображения диаграмм 52](#_Toc169840490)

[5 Заключение 53](#_Toc169840491)

[6 Выводы 53](#_Toc169840492)

# Введение

В рамках данной работы рассмотрена автоматизация чтения данных из документа Word и её последующая конвертация в Excel. 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 ходе курса «Методы программирования», в том числе: подключение библиотек, классов и их использования в программах, программная настройка электронных таблиц Excel и документов Word, конвертация файлов данных типов между собой (из Word в Excel, из Excel в Word). Освоить навыки работы с элементом Chart и построением графиков.

# 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

## Анализ печатной формы

Печатная форма представляет из себя список выпущенных работ преподавателями РУТ МИИТ на различные темы, а также различные виды работ. Также указано количество экземпляров, авторских листов, для каких специализаций была выпущена данная работа и квартал сдачи.

## Описание предназначения печатной формы

Форма предназначена для хранения данных о каждой работе, выпущенной преподавателями.

## Разнообразие печатных форм и выделение различий

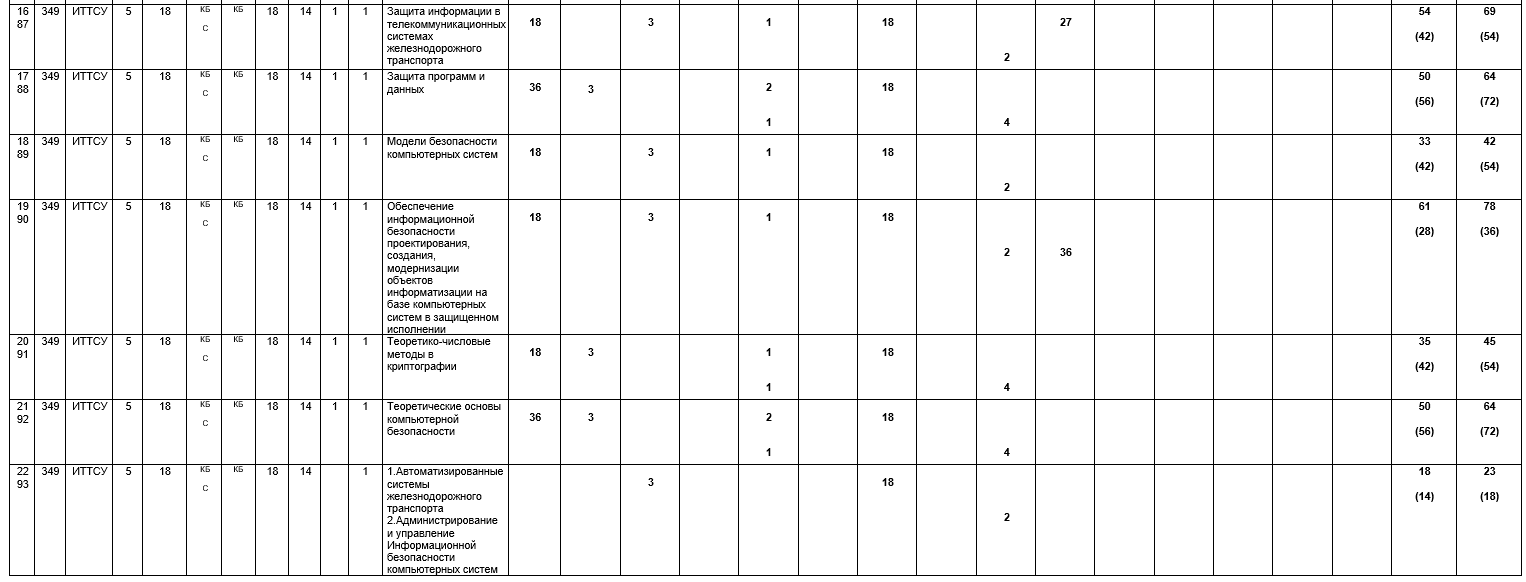
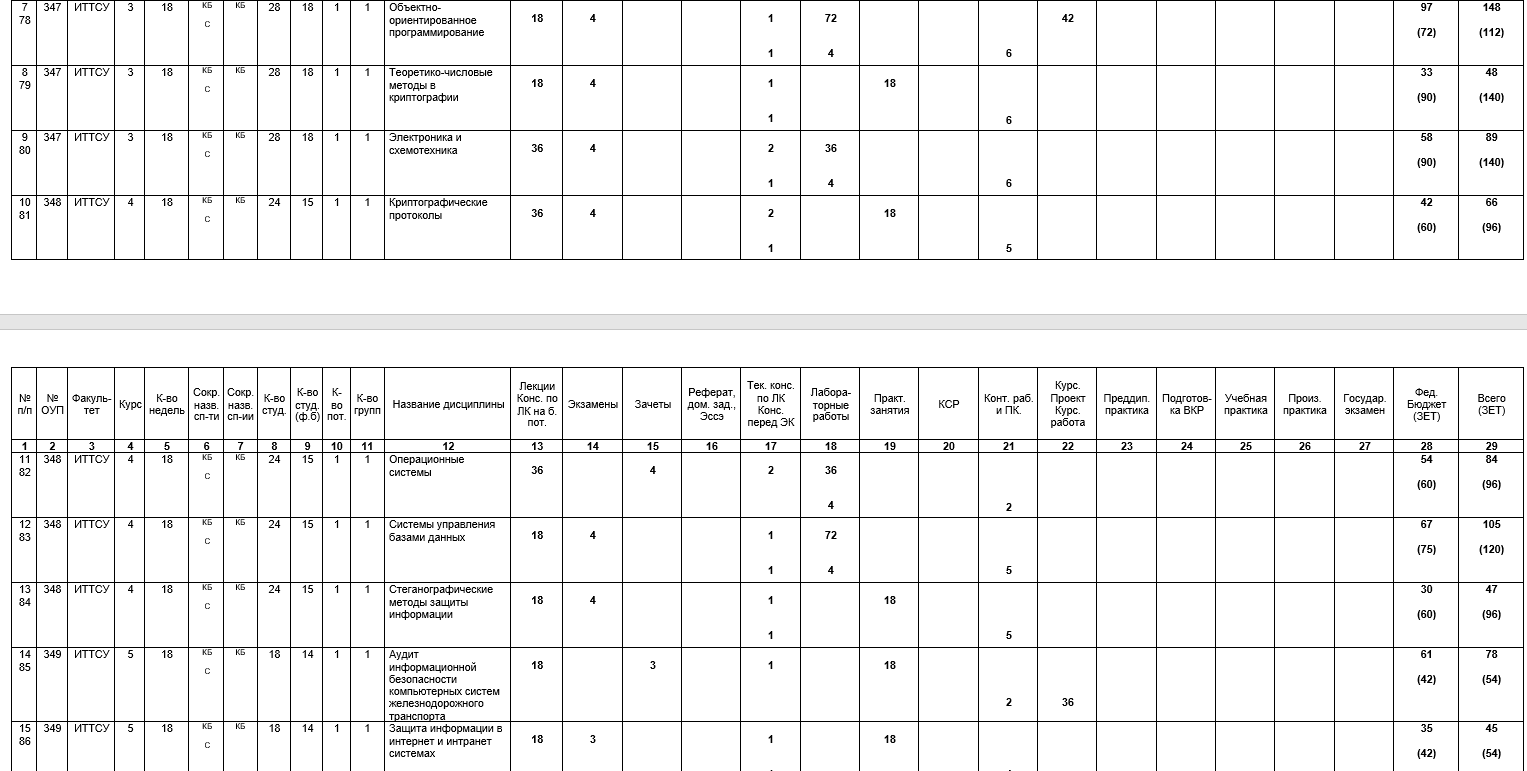
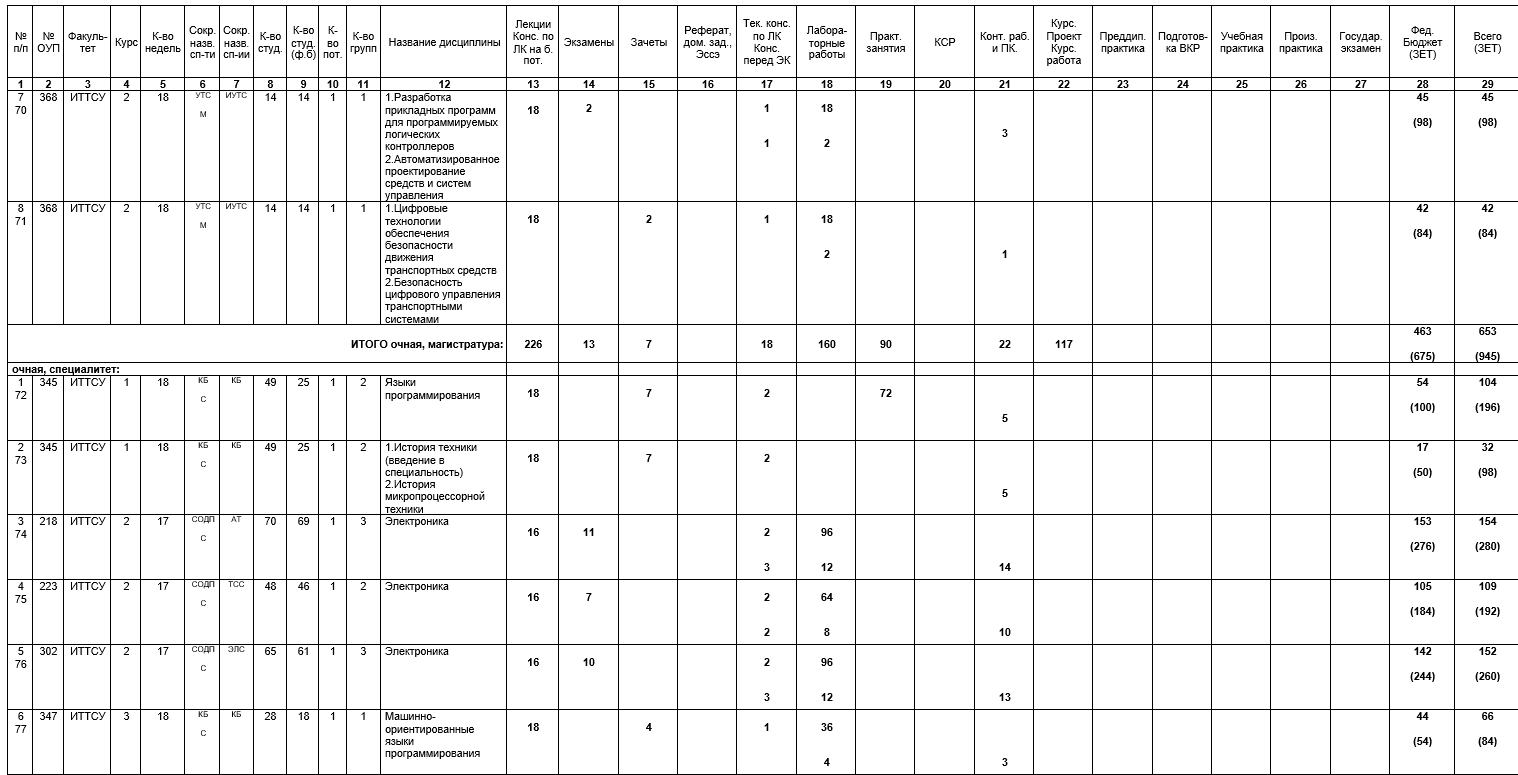
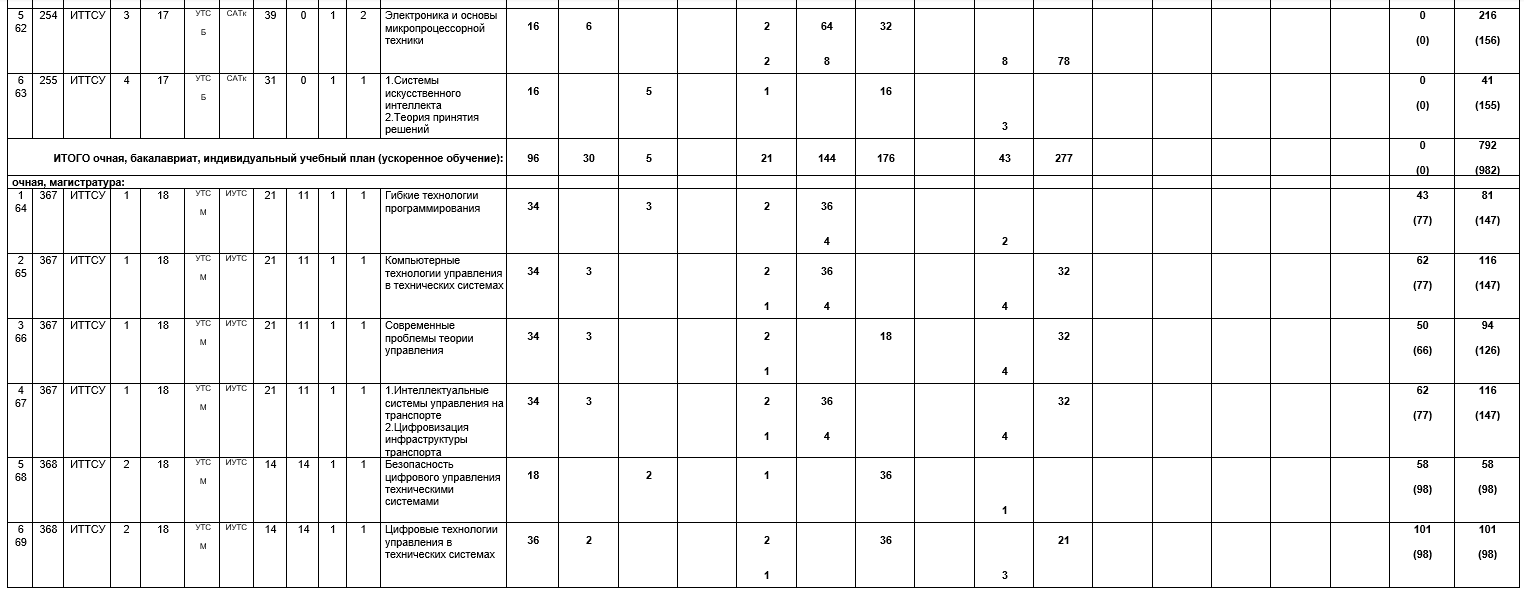
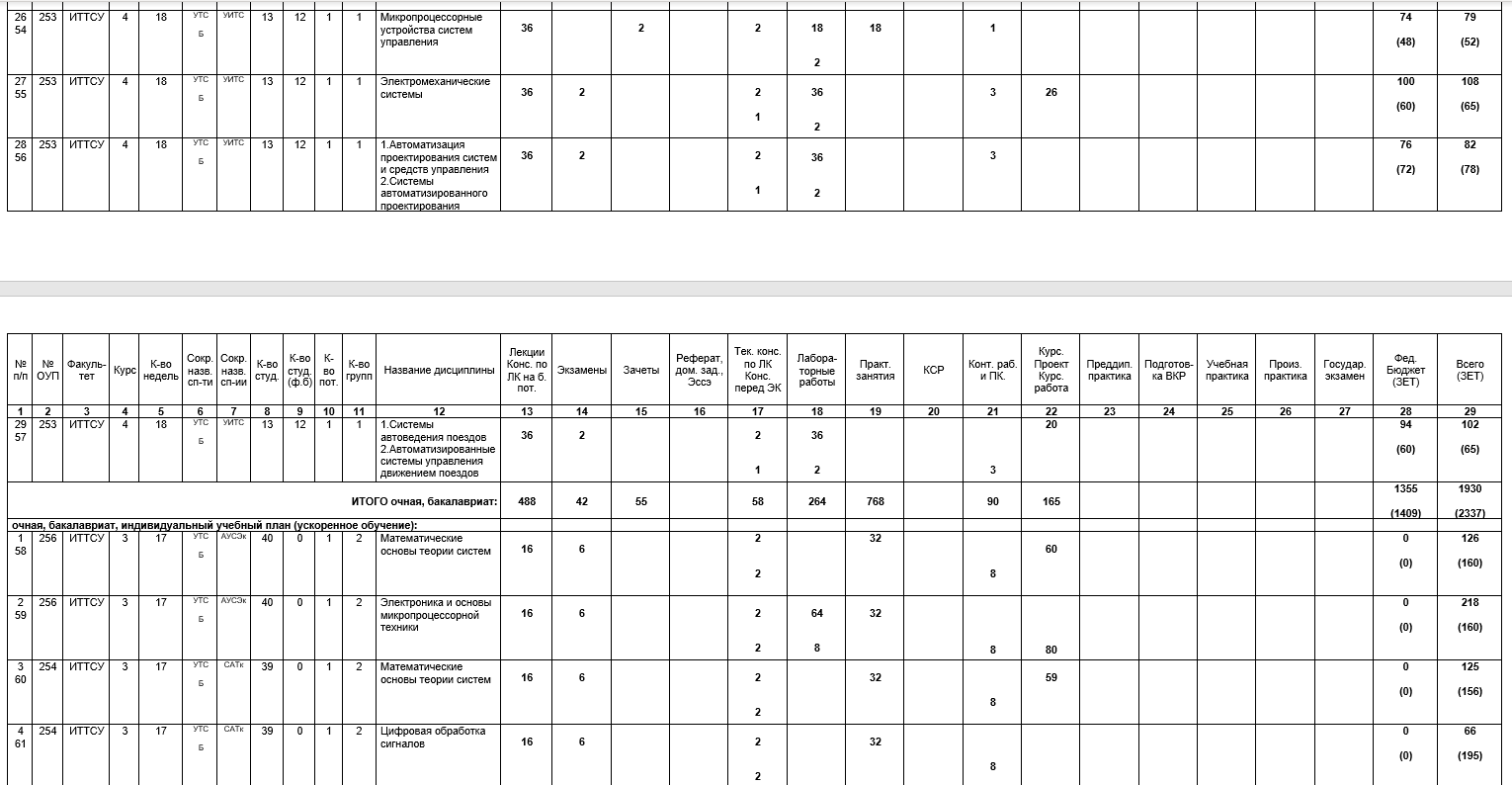
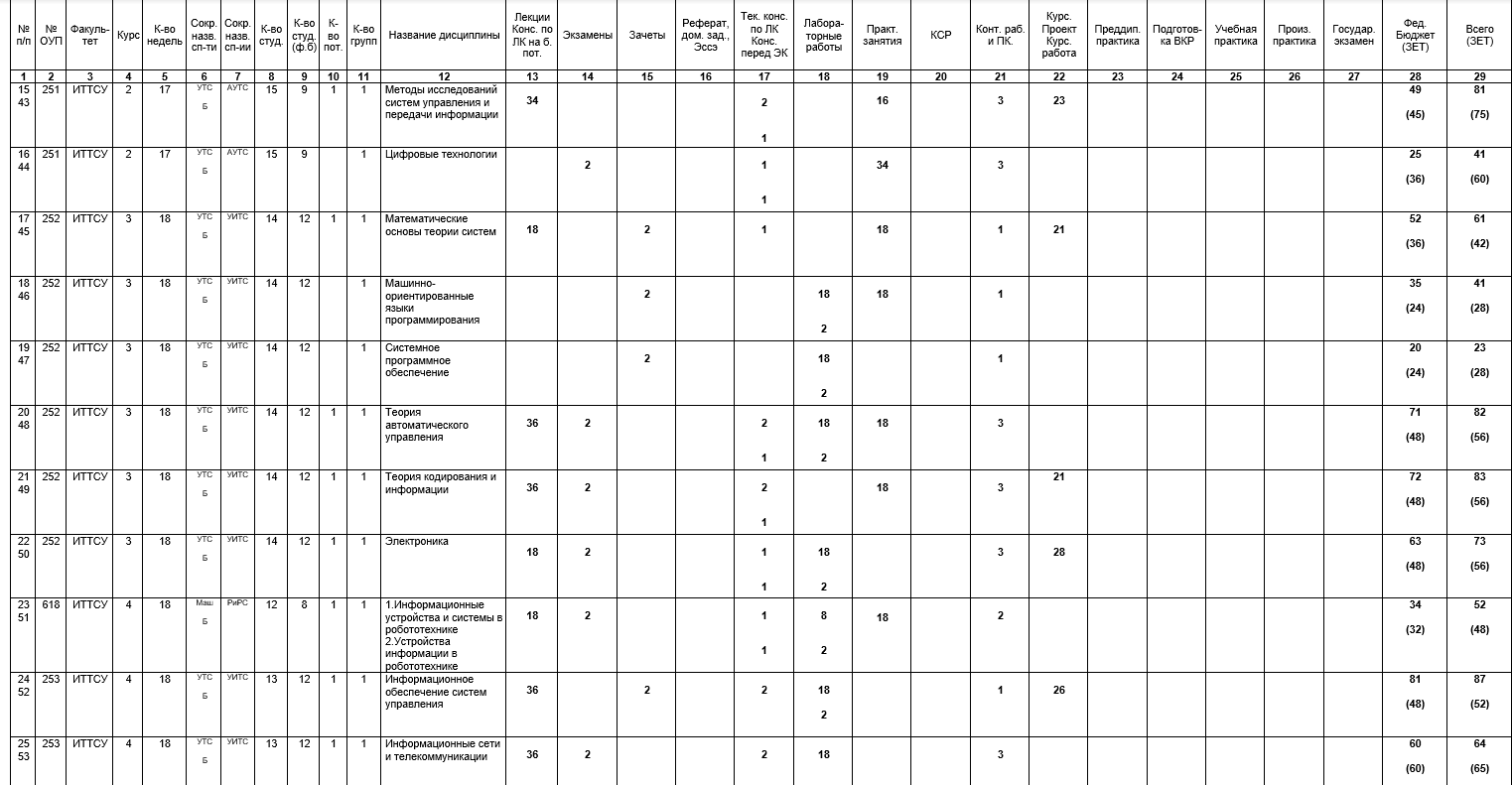
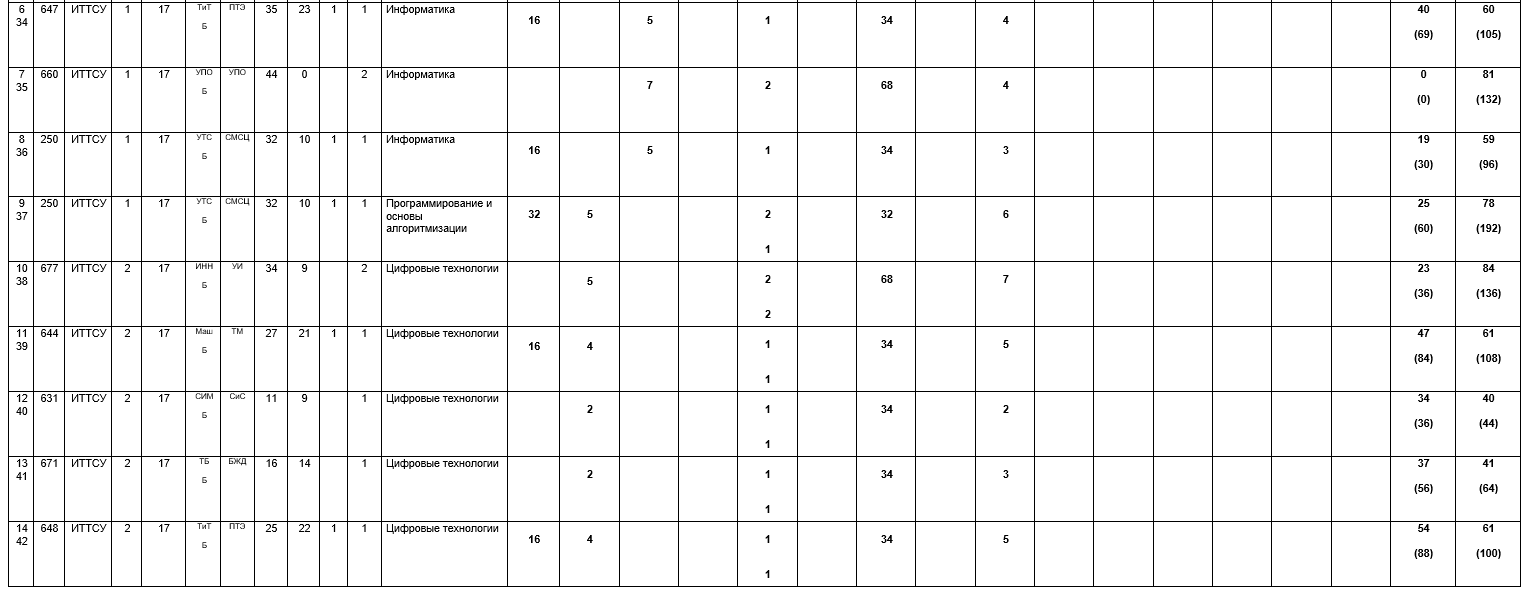
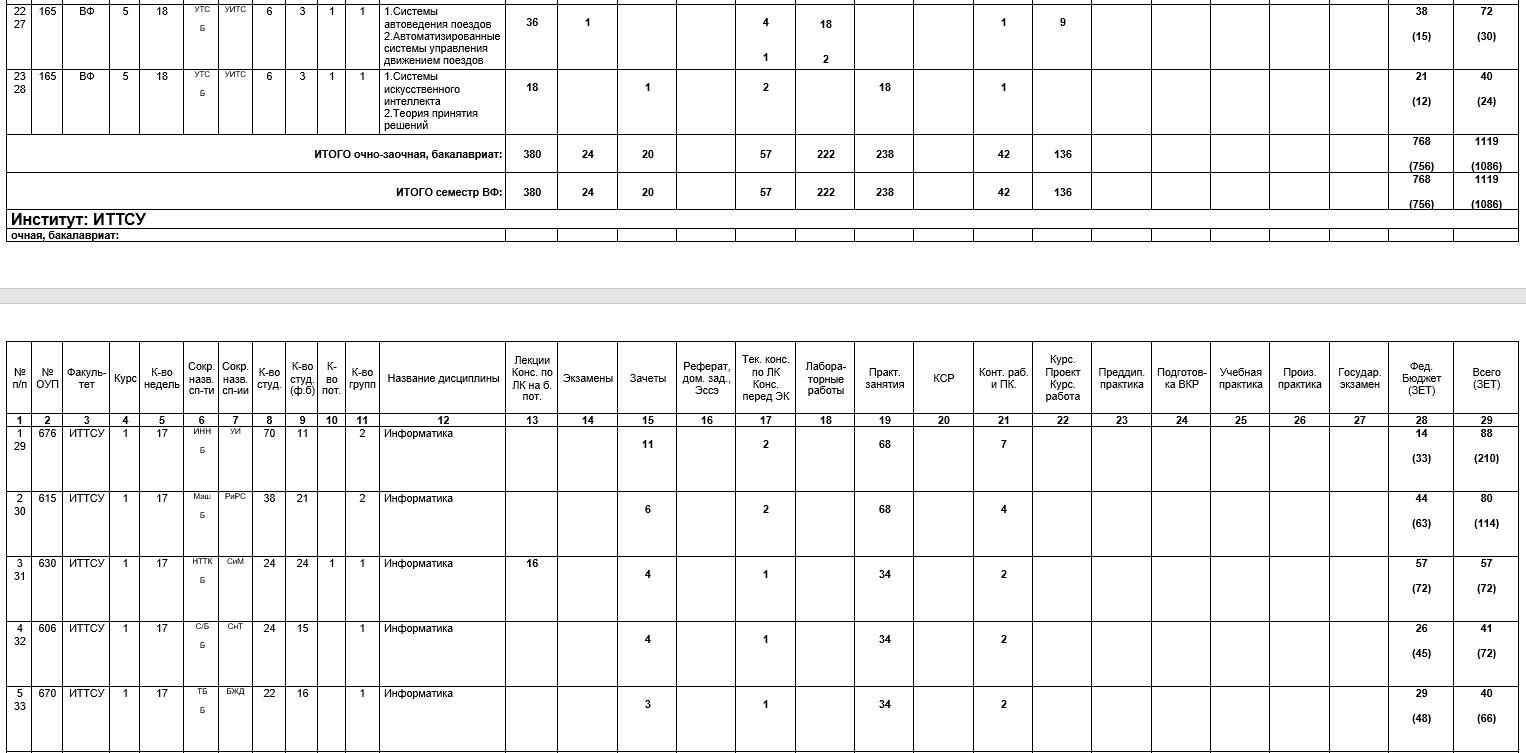
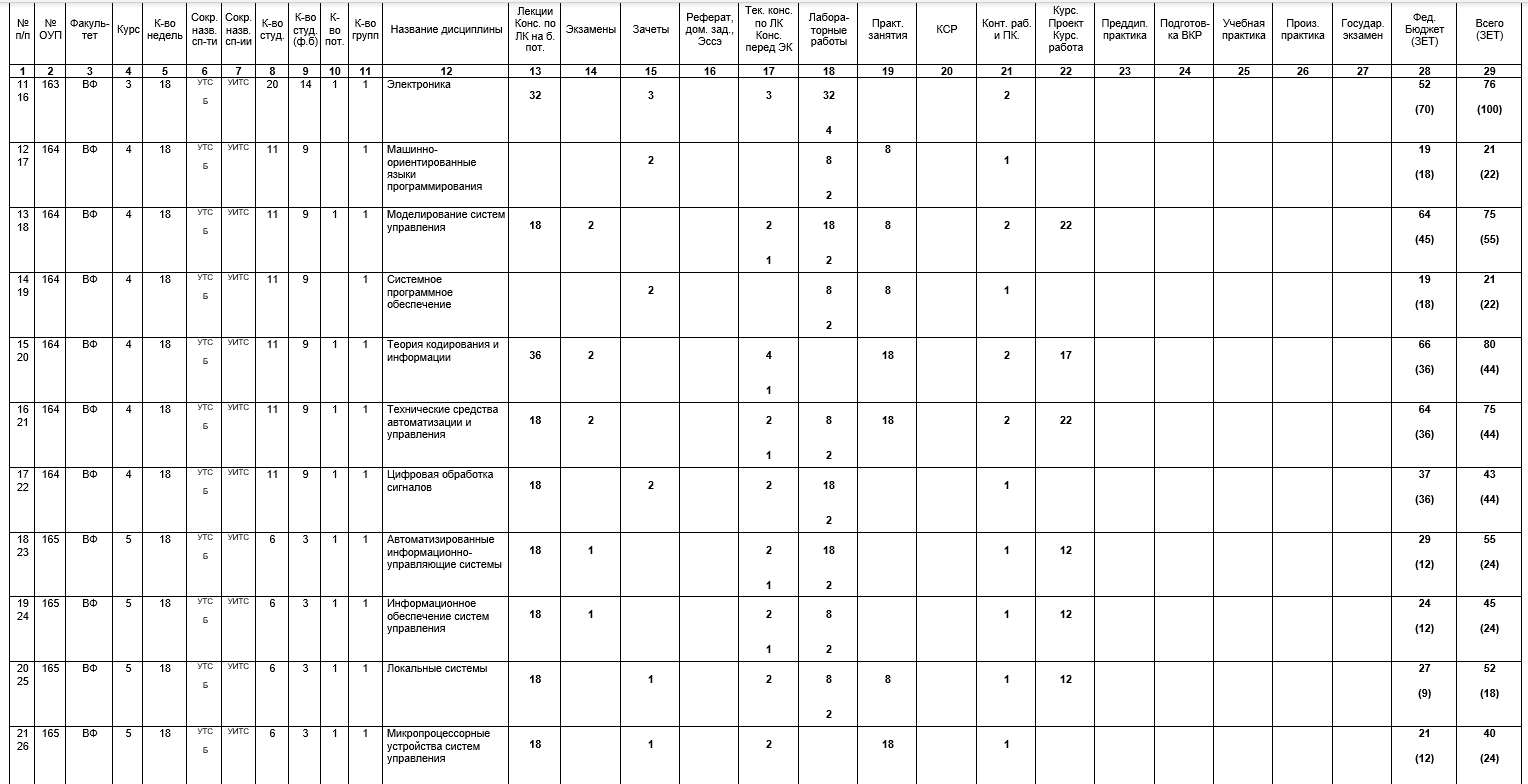
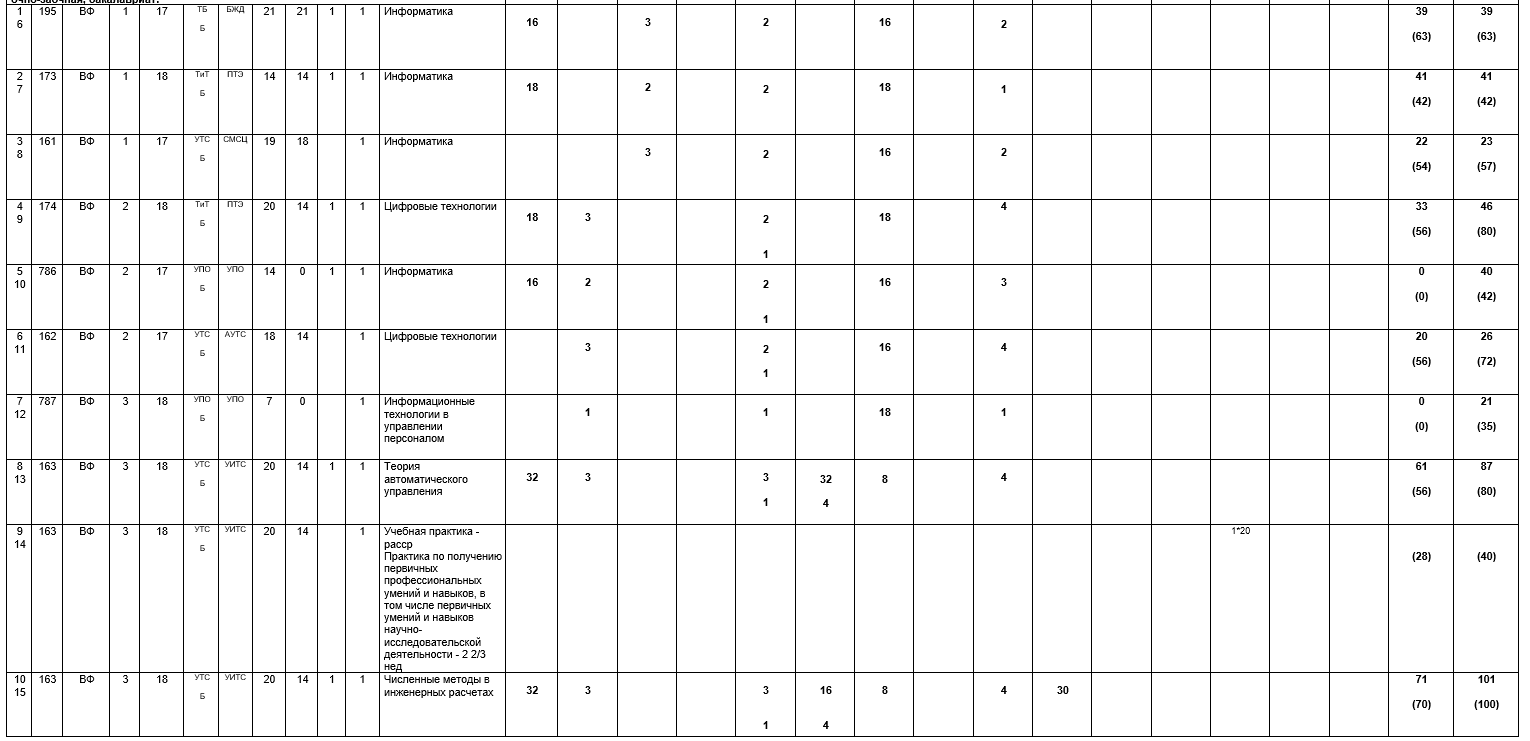
Формы в большинстве своём были одинакового размера, а также содержали одинаковые данные, однако форма 2021 года отличается наличием дополнительного столбца в таблице.

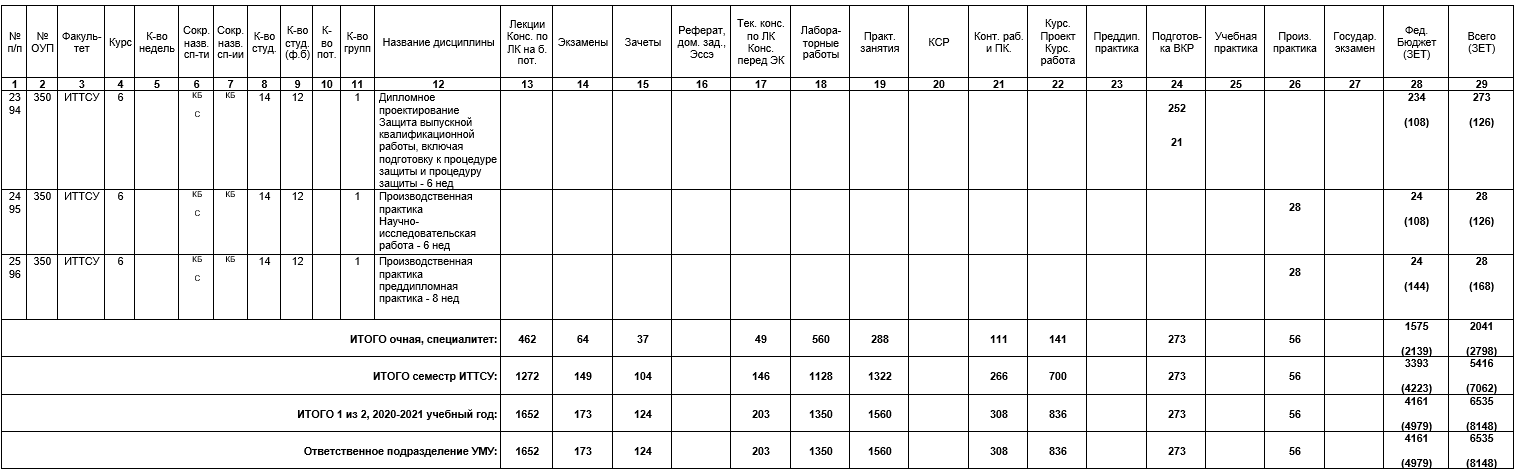
## Унификация печатных форм для удобства обработки данных

Чтобы было легче анализировать данные, было принято решение привести все таблицы в печатных формах к одному размеру, поэтому из форм 2021 года был удален 1 столбец в таблице.

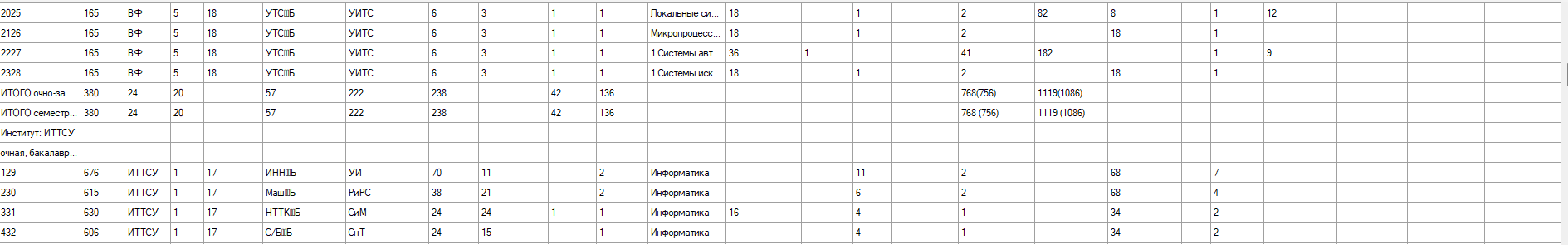
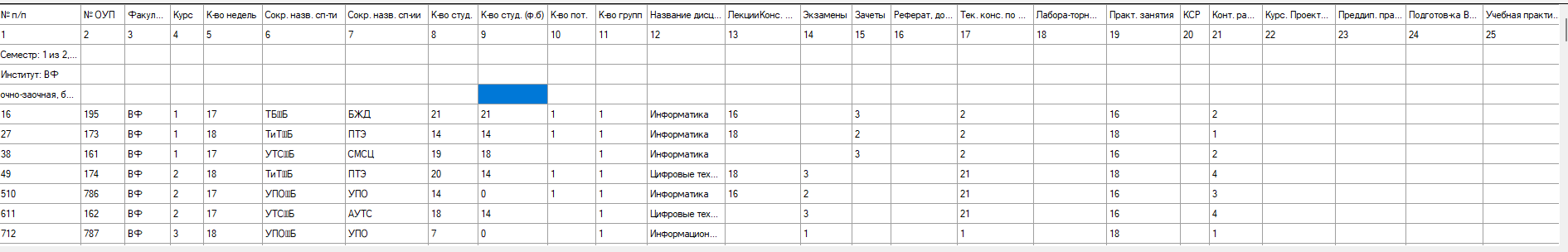
## Считывание из Word на экранную форму

Так как основную информацию в себе несёт именно таблица, в рамках данной работы было решено представить информацию в элементе DataGridView. Данные считывались напрямую из документа Word. Пример работы данной программы представлен на Рисунках 1 и 2.



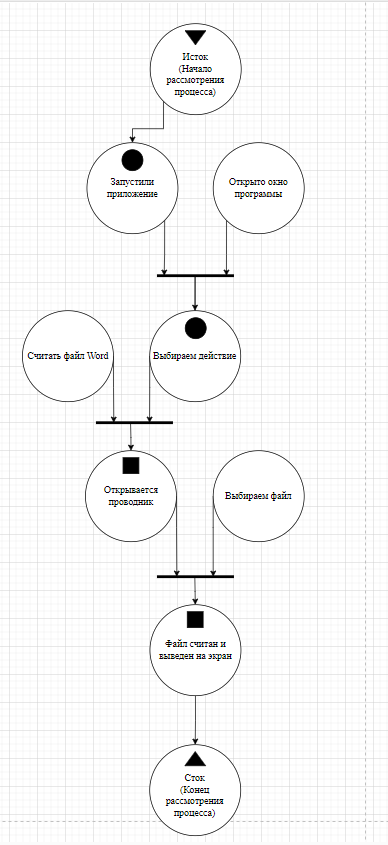


1. Исходная таблица в Word

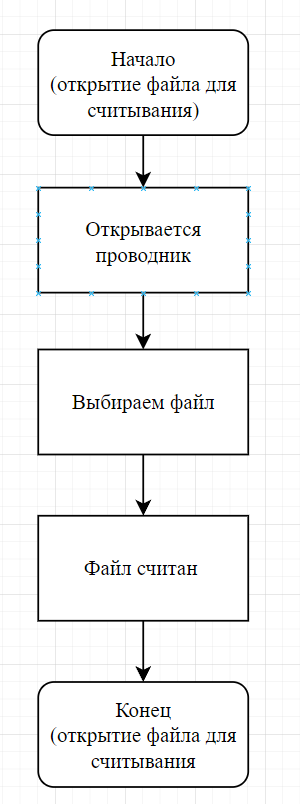


1. Таблица, считанная в элемент DataGridView

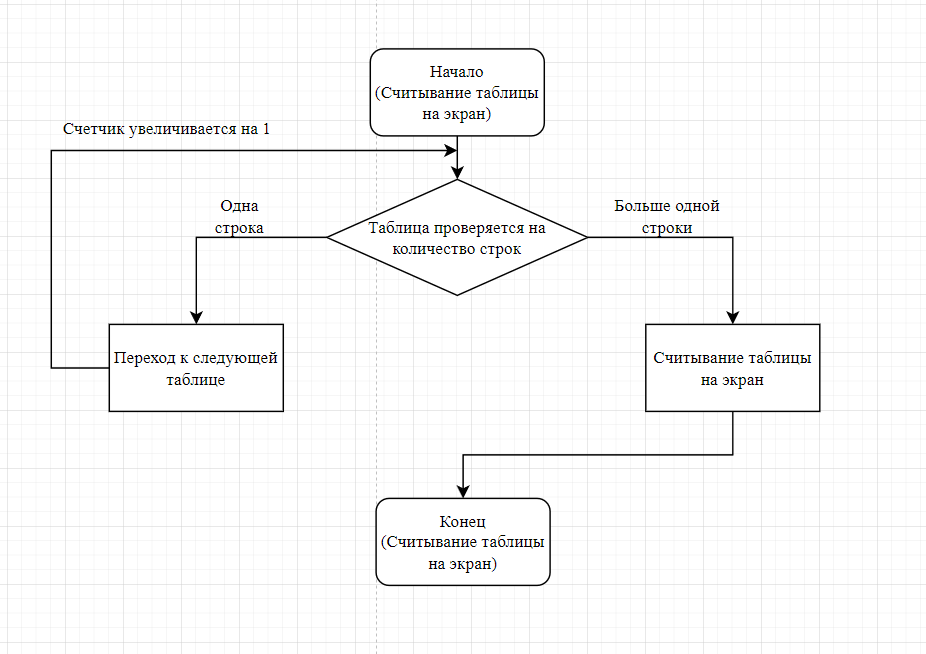
Сеть Петри для алгоритма считывания из Word на экранную форму:



1. Сеть Петри для алгоритма считывания из Word на экранную форму



1. Схема алгоритма “Выбор файла для считывания”



1. Схема алгоритма “Файл считывается и выводится на экран”

Листинг кода экранной формы:

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

{

Word.Application oWord;

Word.Document oDoc;

OpenFileDialog fileDialog = new OpenFileDialog();

fileDialog.ShowDialog();

string filePath = fileDialog.FileName.ToString();

int counter = 0;

oWord = new Word.Application();

oDoc = oWord.Documents.Open(filePath);

foreach (Word.Table oTab in oDoc.Tables)

{

if (oTab.Rows.Count > 1)

{

if (counter == 0)

{

DGVForWord.Rows.Clear();

DGVForWord.Columns.Clear();

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

{

try

{

DGVForWord.Columns.Add("col" + (i + 1), oTab.Cell(1, i + 1).Range.Text.TrimEnd(new char[] { '\r', '\a' }));

}

catch { }

}

DGVForWord.Rows.Add(oTab.Rows.Count);

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

{

for (int j = 0; j < oTab.Columns.Count; j++)

{

try

{

DGVForWord[j, i].Value = oTab.Cell(i + 1, j + 1).Range.Text.TrimEnd(new char[] { '\r', '\a' });

}

catch { }

}

}

}

counter++;

}

}

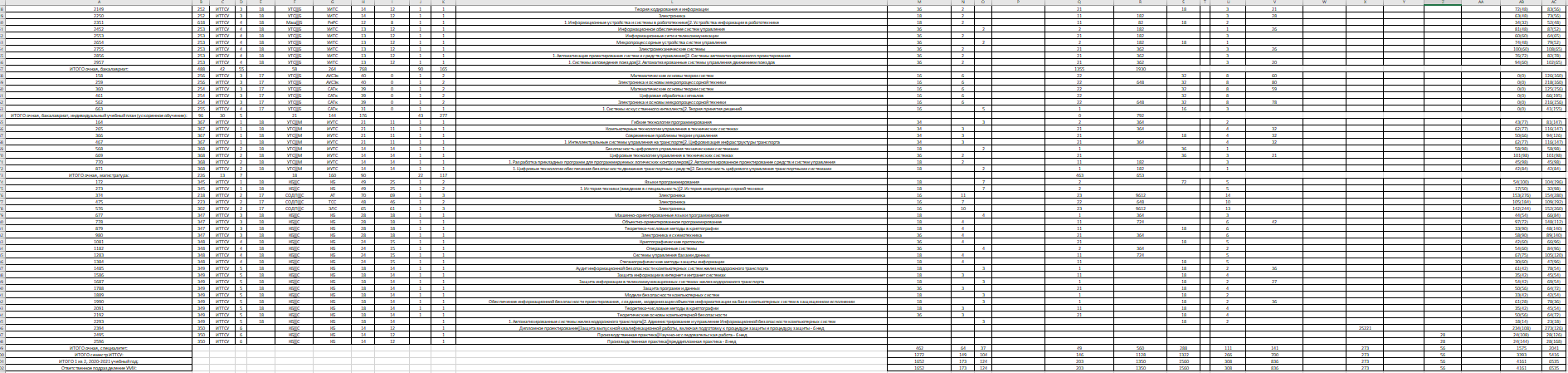
oDoc.Close();

oWord.Quit();

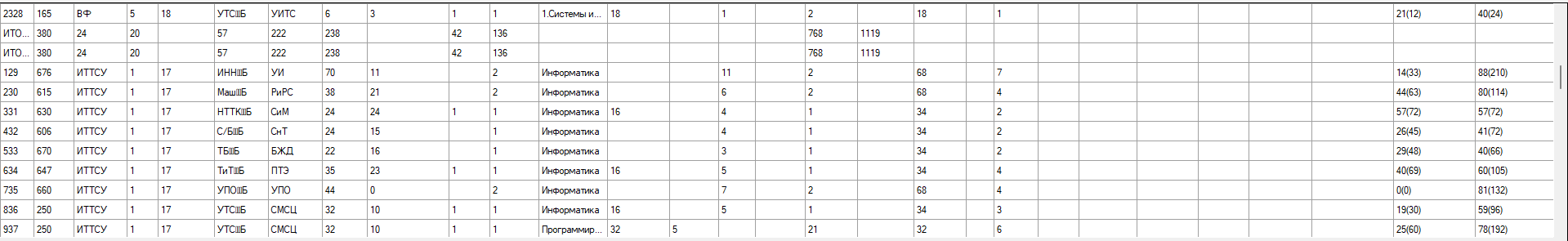
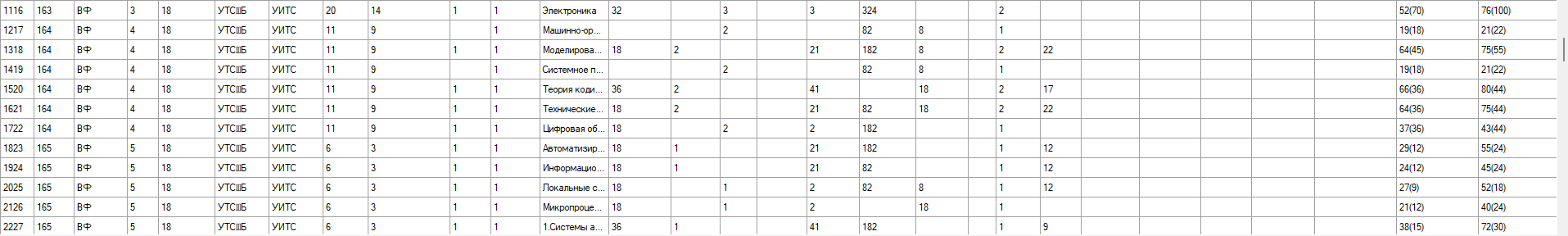
}

## Считывание из Excel на экранную форму

Данные считывались напрямую из документа Excel. Пример работы данной программы представлен на Рисунках 6 и 7.

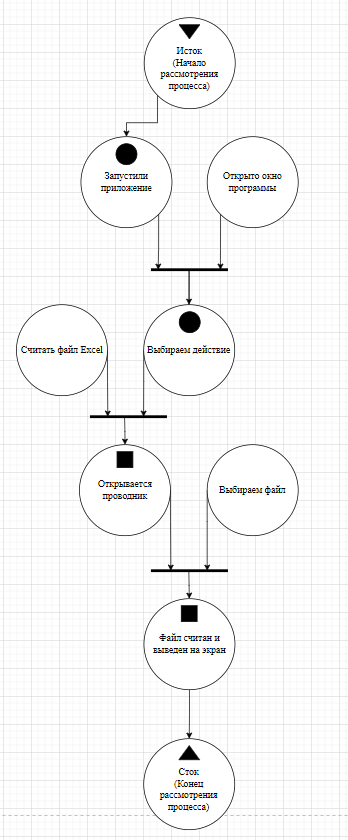


1. Исходная таблица в Excel



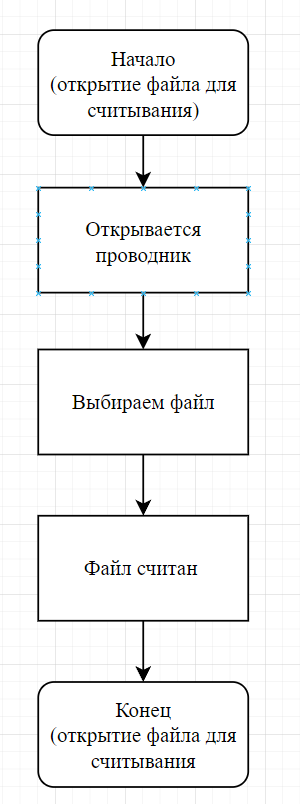
1. Таблица, считанная в элемент DataGridView

Сеть петри для алгоритма считывания файла Excel и вывода его на экран:



1. Сеть Петри для алгоритма считывания из Excel на экранную форму

Схема алгоритма “Выбор файла для считывания и выведение на экран” представлена на Рисунке 9.



1. Схема алгоритма “Открывается диалоговое окно для выбора файла”

Листинг кода экранной формы:

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

{

Word.Application oWord;

Word.Document oDoc;

OpenFileDialog fileDialog = new OpenFileDialog();

fileDialog.ShowDialog();

string filePath = fileDialog.FileName.ToString();

int counter = 0;

oWord = new Word.Application();

oDoc = oWord.Documents.Open(filePath);

foreach (Word.Table oTab in oDoc.Tables)

{

if (oTab.Rows.Count > 1)

{

if (counter == 0)

{

DGVForWord.Rows.Clear();

DGVForWord.Columns.Clear();

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

{

try

{

DGVForWord.Columns.Add("col" + (i + 1), oTab.Cell(1, i + 1).Range.Text.TrimEnd(new char[] { '\r', '\a' }));

}

catch { }

}

DGVForWord.Rows.Add(oTab.Rows.Count);

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

{

for (int j = 0; j < oTab.Columns.Count; j++)

{

try

{

DGVForWord[j, i].Value = oTab.Cell(i + 1, j + 1).Range.Text.TrimEnd(new char[] { '\r', '\a' });

}

catch { }

}

}

}

counter++;

}

}

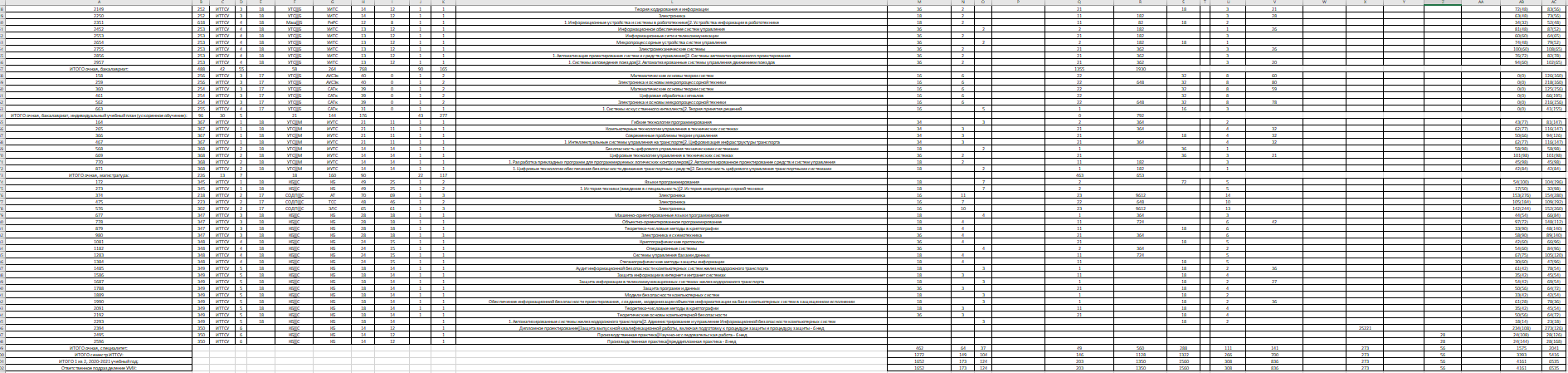
oDoc.Close();

oWord.Quit();

}

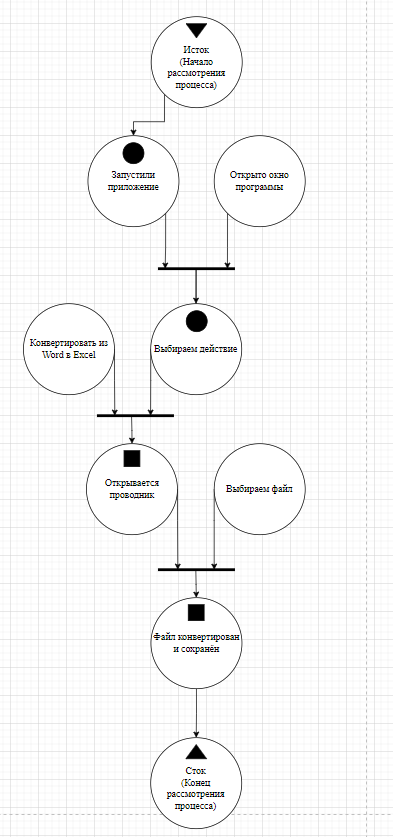
## Разработка функции конвертации документа Word в Excel

Пример работы представлен на Рисунке 13.

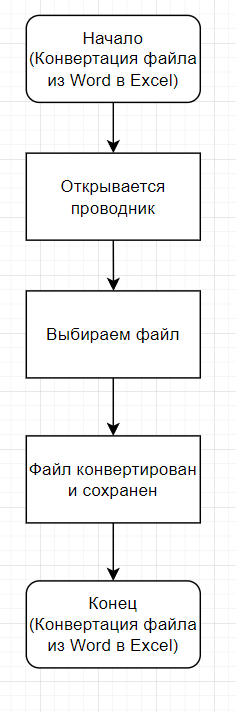


1. Генерируемый программой файл Excel

Сеть Петри для алгоритма конвертации из Word в Excel:



1. Сеть Петри для алгоритма для конвертации из Word в Excel



1. Схема алгоритма “Выбор файла для конвертации из Word в Excel”

Листинг кода функции конвертации в Excel:

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

{

Word.Application oWord;

Word.Document oDoc;

DateTime dt = DateTime.Now;

OpenFileDialog fileDialog = new OpenFileDialog();

fileDialog.ShowDialog();

string filePath = fileDialog.FileName.ToString();

oWord = new Word.Application();

oDoc = oWord.Documents.Open(filePath);

Excel.Application excelApp = new Excel.Application();

Excel.Workbook excelWorkbook = excelApp.Workbooks.Add();

Excel.Worksheet excelWorksheet = excelWorkbook.Sheets[1];

foreach (Word.Table table in oDoc.Tables)

{

if (table.Rows.Count > 1)

{

int rowIndex = 1;

int colIndex = 1;

foreach (Word.Row row in table.Rows)

{

foreach (Word.Cell cell in row.Cells)

{

excelWorksheet.Cells[rowIndex, colIndex] = cell.Range.Text.TrimEnd(new char[] { '\r', '\a' });

excelWorksheet.Cells[rowIndex, colIndex].BorderAround(Excel.XlLineStyle.xlContinuous, Excel.XlBorderWeight.xlThin);

excelWorksheet.Cells[rowIndex, colIndex].HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter;

colIndex++;

}

colIndex = 1;

rowIndex++;

}

}

}

string originalFileName = Path.GetFileNameWithoutExtension(filePath);

excelWorksheet.Columns.AutoFit();

excelWorkbook.SaveAs $@"D:\Изображения\Сафронов\Excel\{originalFileName} .xlsx");

excelWorkbook.Close();

excelApp.Quit();

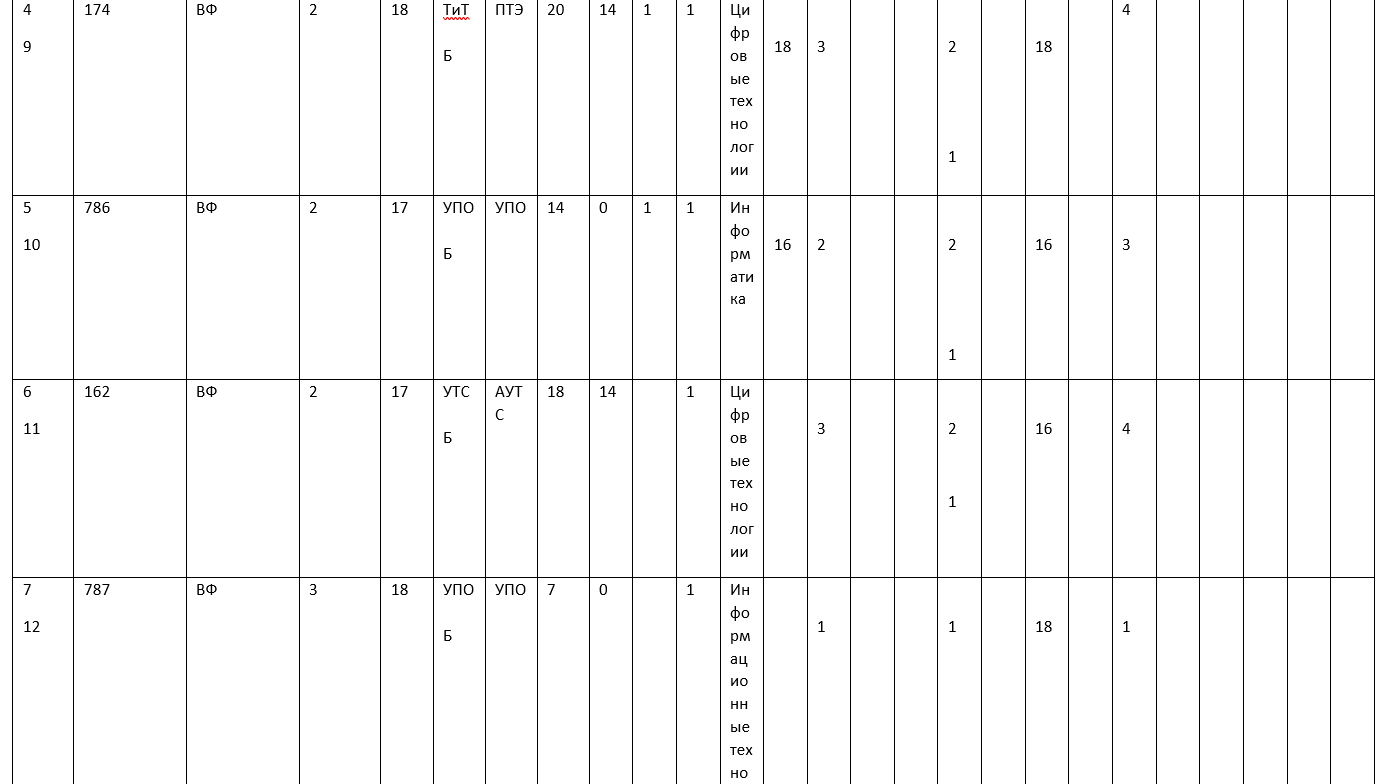
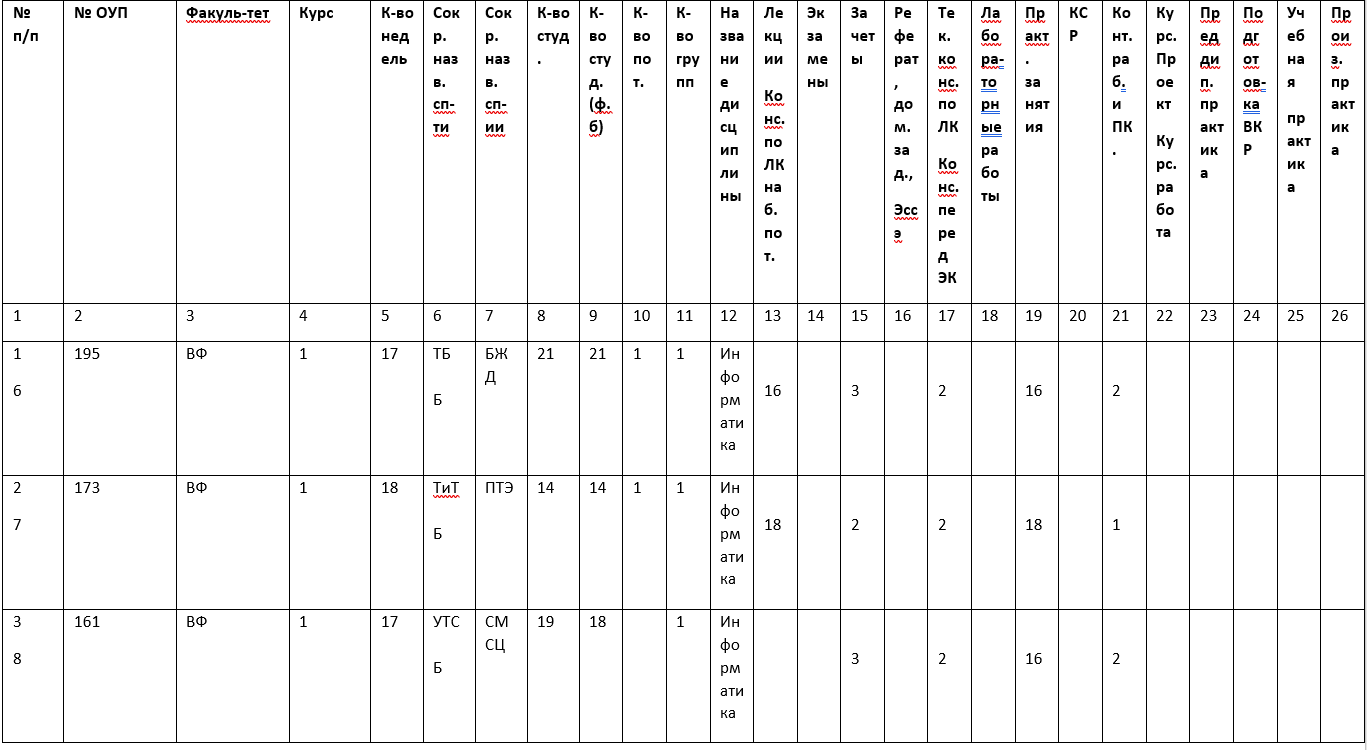
oWord.Quit();

MessageBox.Show("Файл создан");

}

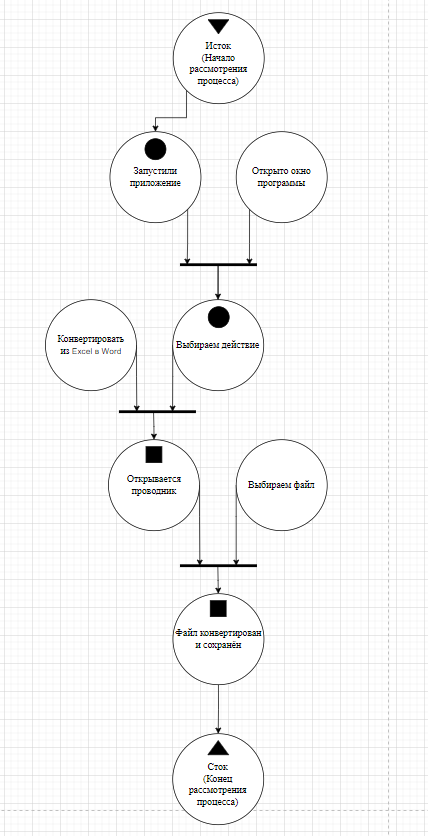
## Разработка функции конвертации документа Excel в Word

Пример работы представлен на Рисунке 16.

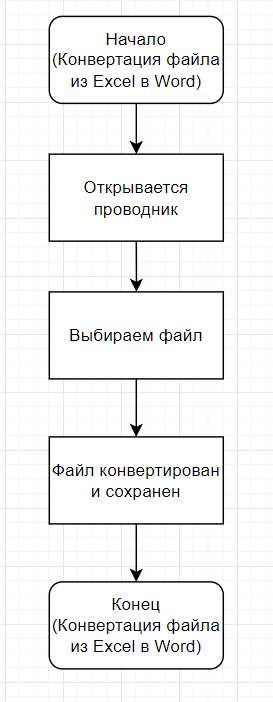


1. Генерируемый программой файл Word

Сеть Петри для алгоритма конвертации из Excel в Word:



1. Сеть Петри для алгоритма для конвертации из Excel в Word



1. Схема алгоритма “Выбор файла для конвертации из Excel в Word”

Листинг кода функции конвертации в Word:

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

{

DateTime dt = DateTime.Now;

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Filter = "Excel Files|\*.xls;\*.xlsx";

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

{

string excelFilePath = openFileDialog.FileName;

Excel.Application excelApp = new Excel.Application();

Excel.Workbook excelWorkbook = excelApp.Workbooks.Open(excelFilePath);

Excel.Worksheet excelWorksheet = excelWorkbook.Sheets[1];

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

Word.Document wordDoc = wordApp.Documents.Add();

wordDoc.PageSetup.Orientation = Word.WdOrientation.wdOrientLandscape;

Word.Table table = wordDoc.Tables.Add(wordDoc.Range(0, 0), excelWorksheet.UsedRange.Rows.Count, excelWorksheet.UsedRange.Columns.Count);

table.Range.Font.Size = 10;

table.Borders.Enable = 1;

for (int j = 1; j <= excelWorksheet.UsedRange.Columns.Count; j++)

{

object cellValue = excelWorksheet.Cells[1, j].Value;

table.Cell(1, j).Range.Text = cellValue != null ? cellValue.ToString() : "";

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

table.Rows[1].Height = 150;

if (j >= 5 && j <= 20)

{

table.Cell(1, j).Range.Orientation = Word.WdTextOrientation.wdTextOrientationUpward;

}

if (j == 1)

{

table.Columns[j].Width = 35;

}

if (j == 2)

{

table.Columns[j].Width = 65;

}

if (j == 3)

{

table.Columns[j].Width = 65;

}

if (j == 4)

{

table.Columns[j].Width = 47;

}

if (j >= 5 && j <= 8)

{

table.Columns[j].Width = 20;

}

}

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

{

for (int j = 1; j <= excelWorksheet.UsedRange.Columns.Count; j++)

{

object cellValue = excelWorksheet.Cells[i, j].Value;

table.Cell(i, j).Range.Text = cellValue != null ? cellValue.ToString() : "";

if (j == 1)

{

table.Columns[j].Width = 35;

}

if (j == 2)

{

table.Columns[j].Width = 65;

}

if (j == 3)

{

table.Columns[j].Width = 65;

}

if (j == 4)

{

table.Columns[j].Width = 47;

}

if (j >= 5 && j <= 8)

{

table.Columns[j].Width = 30;

}

}

}

string saveFilePath = $@"D:\Изображения\Сафронов\Word {dt.ToString("yyyy-MM-dd-hh-mm-ss")}.docx";

wordDoc.SaveAs2(saveFilePath);

excelWorkbook.Close();

excelApp.Quit();

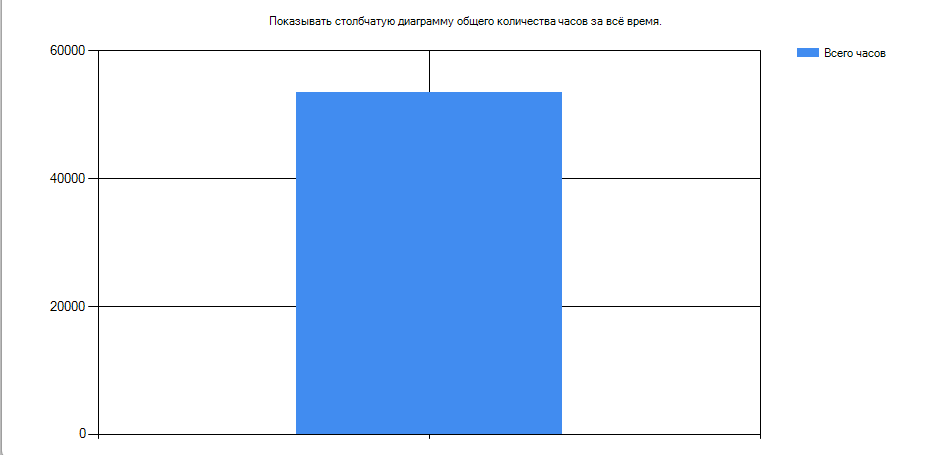
wordApp.Visible = true;

}

}

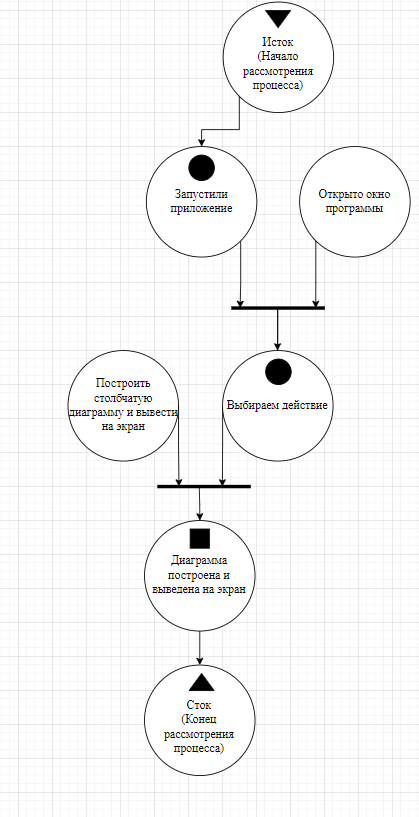
## Графический анализ данных, содержащихся в печатных формах

## 4.9.1 Показывать столбчатую диаграмму общего количества часов за всё время. Результат работы программы на Рисунке 16.

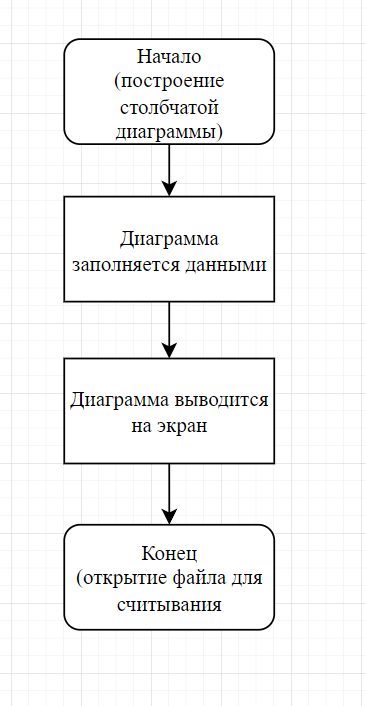


1. Диаграмма количества часов за все время

Сеть Петри для алгоритма вывода столбчатой диаграммы.



1. Сеть Петри для алгоритма для столбчатой диаграммы общего количества часов за все время.



1. Схема алгоритма “Заполнение диаграммы и вывод на экран”

Листинг кода для построения данной диаграммы:

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

{

string folderPath = @"D:\Изображения\Сафронов\excels\";

string[] files = Directory.GetFiles(folderPath);

Dictionary<string, double> processedNames = new Dictionary<string, double>();

foreach (string file in files)

{

Excel.Application excelApp = new Excel.Application();

Excel.Workbook excelWorkbook = excelApp.Workbooks.Open(file);

Excel.Worksheet excelWorksheet = excelWorkbook.Sheets[1];

string name = "Всего часов";

double value = Convert.ToDouble(excelWorksheet.Cells[101, 29].Value2); // Row 2, Column 7

if (!string.IsNullOrEmpty(name))

{

if (processedNames.ContainsKey(name))

{

processedNames[name] += value;

}

else

{

processedNames.Add(name, value);

}

}

excelWorkbook.Close(false);

excelApp.Quit();

}

foreach (var entry in processedNames)

{

string name = entry.Key;

double value = entry.Value;

bool seriesExists = false;

foreach (var series in chart1.Series)

{

if (series.Name == name)

{

series.Points.Clear();

series.Points.AddXY(name, value);

seriesExists = true;

break;

}

}

if (!seriesExists)

{

Series newSeries = chart1.Series.Add(name);

newSeries.ChartType = SeriesChartType.Column;

newSeries.Points.AddXY(name, value);

}

}

foreach (var series in chart1.Series)

{

series.BorderWidth = 1;

}

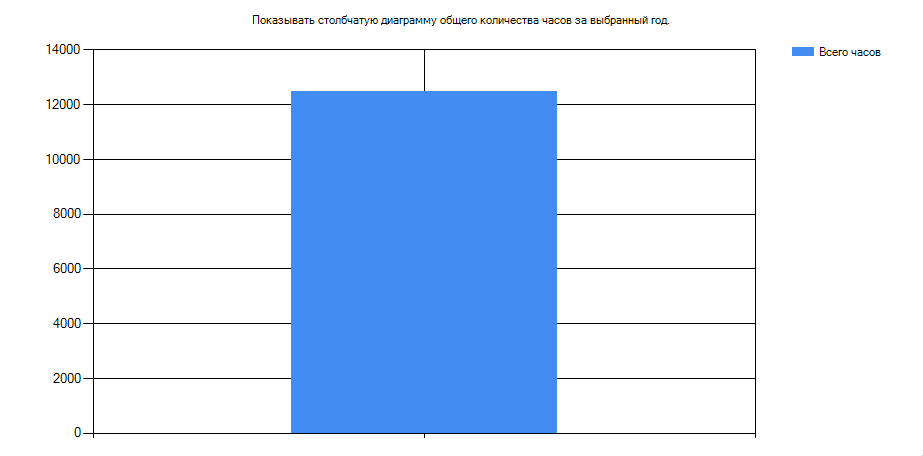
chart1.Titles.Clear();

chart1.Titles.Add("Показывать столбчатую диаграмму общего количества часов за всё время.");

chart1.ChartAreas[0].AxisX.LabelStyle.Enabled = false;

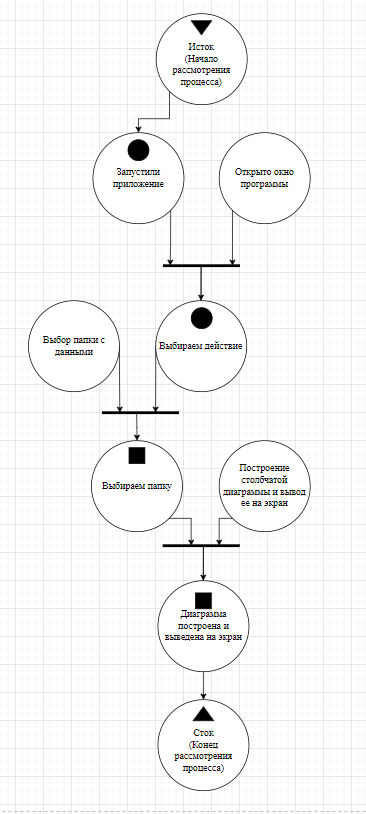
}

4.9.2 Показывать столбчатую диаграмму общего количества часов за выбранный учебный год. Результат работы программы на Рисунке 19.

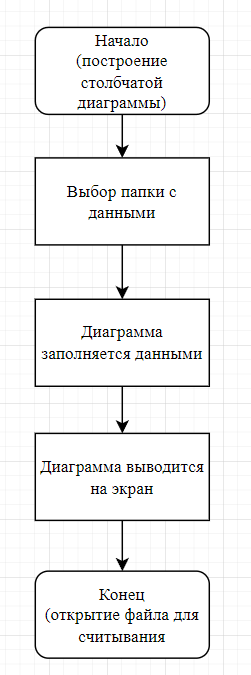


1. Диаграмма общего количества часов за выбранный учебный год

Сеть Петри для алгоритма вывода столбчатой диаграммы.



1. Сеть Петри для алгоритма для диаграммы общего количества часов за выбранный учебный год



1. Схема алгоритма “Диаграмма заполняется данными из папки и выводится на экран”

Листинг кода для построения данной диаграммы:

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

{

// Выбор папки с помощью диалогового окна

FolderBrowserDialog folderDialog = new FolderBrowserDialog();

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

{

string folderPath = folderDialog.SelectedPath;

string[] files = Directory.GetFiles(folderPath);

Dictionary<string, double> processedNames = new Dictionary<string, double>();

foreach (string file in files)

{

Excel.Application excelApp = new Excel.Application();

Excel.Workbook excelWorkbook = excelApp.Workbooks.Open(file);

Excel.Worksheet excelWorksheet = excelWorkbook.Sheets[1];

Excel.Range usedRange = excelWorksheet.UsedRange;

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

{

string name = usedRange.Cells[i, 3].Value2;

double value = Convert.ToDouble(usedRange.Cells[i, 7].Value2);

if (name != null)

{

if (processedNames.ContainsKey(name))

{

processedNames[name] += value;

}

else

{

processedNames.Add(name, value);

}

}

}

excelWorkbook.Close(false);

excelApp.Quit();

}

foreach (var entry in processedNames)

{

string name = entry.Key;

double value = entry.Value;

bool seriesExists = false;

foreach (var series in chart1.Series)

{

if (series.Name == name)

{

series.Points.Clear();

series.Points.AddXY(name, value);

seriesExists = true;

break;

}

}

if (!seriesExists)

{

Series newSeries = chart1.Series.Add(name);

newSeries.ChartType = SeriesChartType.Column;

newSeries.Points.AddXY(name, value);

}

}

foreach (var series in chart1.Series)

{

series.BorderWidth = 1;

}

chart1.Titles.Clear();

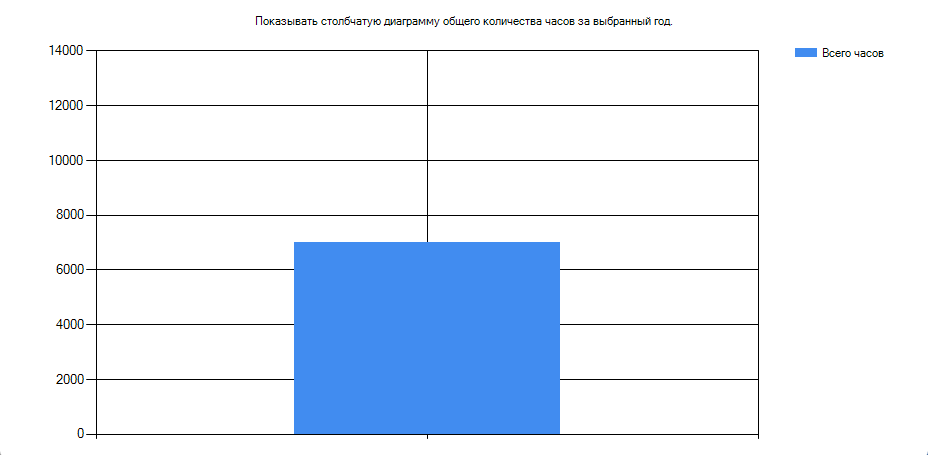
chart1.Titles.Add("Показывать столбчатую диаграмму общего количества часов за выбранный год.");

chart1.ChartAreas[0].AxisX.LabelStyle.Enabled = false;

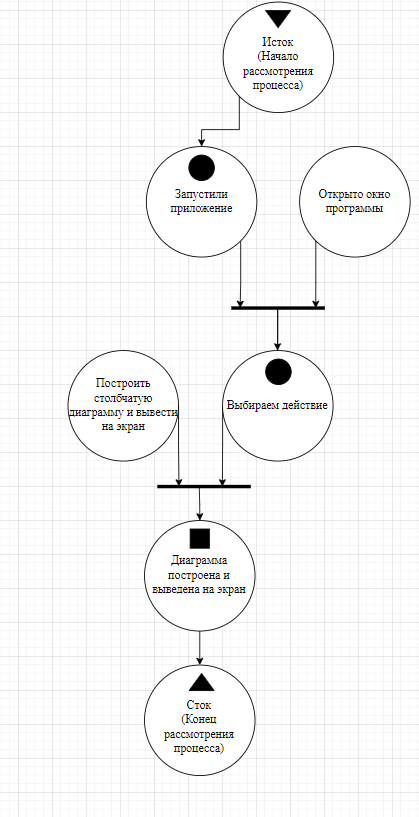
}

### }

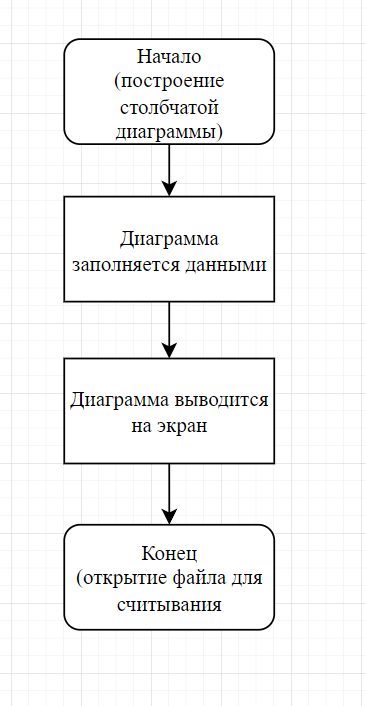
### Показывать столбчатую диаграмму общего количества часов за выбранный календарный год. Результат работы программы на Рисунке 22.



1. Диаграмма общего количества часов за выбранный календарный год



1. Сеть Петри для алгоритма для диаграммы общего количества часов за выбранный календарный год



1. Схема алгоритма “ Диаграмма заполняется данными из папки и выводится на экран”

Листинг кода для построения данной диаграммы:

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

{

// Выбор папки с помощью диалогового окна

FolderBrowserDialog folderDialog = new FolderBrowserDialog();

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

{

string folderPath = folderDialog.SelectedPath;

string[] files = Directory.GetFiles(folderPath);

Dictionary<string, double> processedNames = new Dictionary<string, double>();

foreach (string file in files)

{

Excel.Application excelApp = new Excel.Application();

Excel.Workbook excelWorkbook = excelApp.Workbooks.Open(file);

Excel.Worksheet excelWorksheet = excelWorkbook.Sheets[1];

Excel.Range usedRange = excelWorksheet.UsedRange;

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

{

string name = usedRange.Cells[i, 3].Value2;

double value = Convert.ToDouble(usedRange.Cells[i, 7].Value2);

if (name != null)

{

if (processedNames.ContainsKey(name))

{

processedNames[name] += value;

}

else

{

processedNames.Add(name, value);

}

}

}

excelWorkbook.Close(false);

excelApp.Quit();

}

foreach (var entry in processedNames)

{

string name = entry.Key;

double value = entry.Value;

bool seriesExists = false;

foreach (var series in chart1.Series)

{

if (series.Name == name)

{

series.Points.Clear();

series.Points.AddXY(name, value);

seriesExists = true;

break;

}

}

if (!seriesExists)

{

Series newSeries = chart1.Series.Add(name);

newSeries.ChartType = SeriesChartType.Column;

newSeries.Points.AddXY(name, value);

}

}

foreach (var series in chart1.Series)

{

series.BorderWidth = 1;

}

chart1.Titles.Clear();

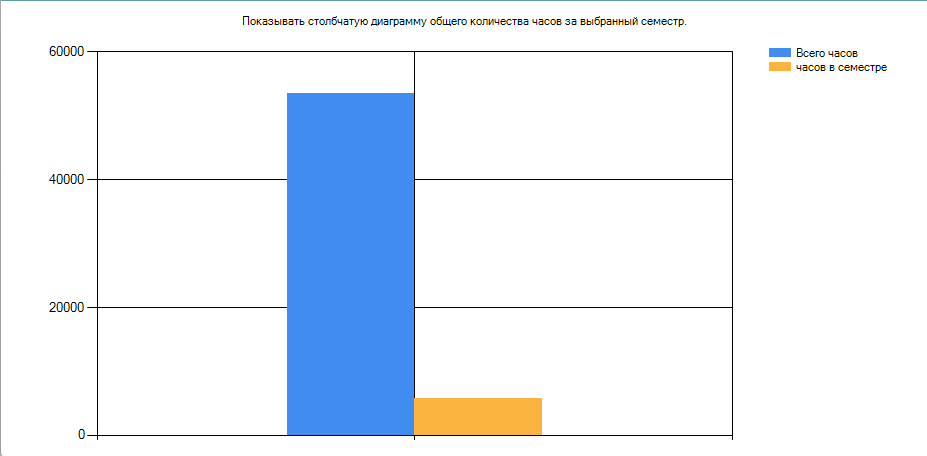
chart1.Titles.Add("Показывать столбчатую диаграмму общего количества часов за выбранный год.");

chart1.ChartAreas[0].AxisX.LabelStyle.Enabled = false;

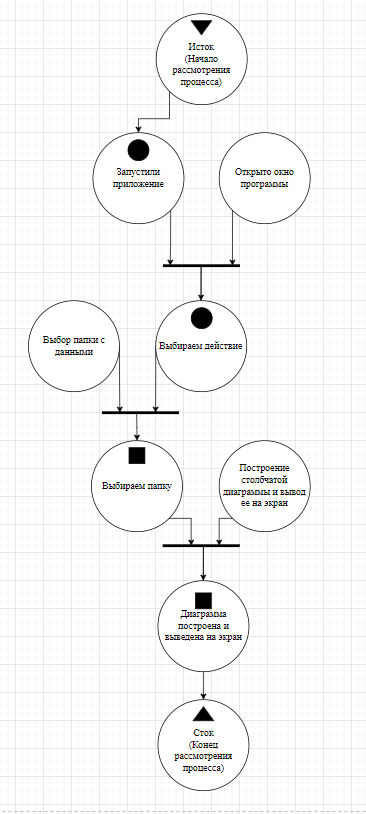
}

### }

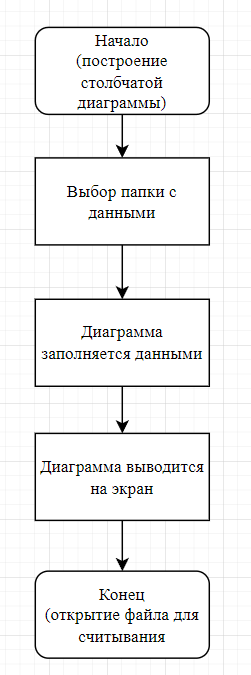
### Показывать столбчатую диаграмму общего количества часов за выбранный семестр. Результат работы программы на Рисунке 25.



1. Диаграмма общего количества часов за выбранный семестр



1. Сеть Петри для алгоритма для диаграммы общего количества часов за выбранный семестр



1. Схема алгоритма “Диаграмма заполняется и выводится на экран”

Листинг кода для построения данной диаграммы:

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

{

// Открытие диалогового окна выбора файла

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Filter = "Файлы Excel (\*.xls;\*.xlsx)|\*.xls;\*.xlsx";

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

{

string filePath = openFileDialog.FileName;

Dictionary<string, double> processedNames = new Dictionary<string, double>();

Excel.Application excelApp = new Excel.Application();

Excel.Workbook excelWorkbook = excelApp.Workbooks.Open(filePath);

Excel.Worksheet excelWorksheet = excelWorkbook.Sheets[1];

string name = "часов в семестре";

double value = Convert.ToDouble(excelWorksheet.Cells[101, 29].Value2); // Row 2, Column 7

if (!string.IsNullOrEmpty(name))

{

if (processedNames.ContainsKey(name))

{

processedNames[name] += value;

}

else

{

processedNames.Add(name, value);

}

}

excelWorkbook.Close(false);

excelApp.Quit();

foreach (var entry in processedNames)

{

string name1 = entry.Key;

double value1 = entry.Value;

bool seriesExists = false;

foreach (var series in chart1.Series)

{

if (series.Name == name)

{

series.Points.Clear();

series.Points.AddXY(name, value);

seriesExists = true;

break;

}

}

if (!seriesExists)

{

Series newSeries = chart1.Series.Add(name);

newSeries.ChartType = SeriesChartType.Column;

newSeries.Points.AddXY(name, value);

}

}

foreach (var series in chart1.Series)

{

series.BorderWidth = 1;

}

chart1.Titles.Clear();

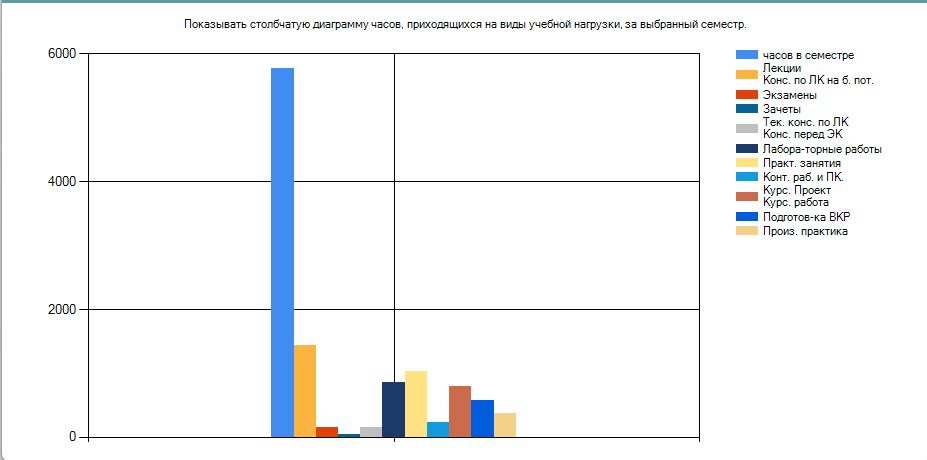
chart1.Titles.Add("Показывать столбчатую диаграмму общего количества часов за выбранный семестр.\r\n");

chart1.ChartAreas[0].AxisX.LabelStyle.Enabled = false;

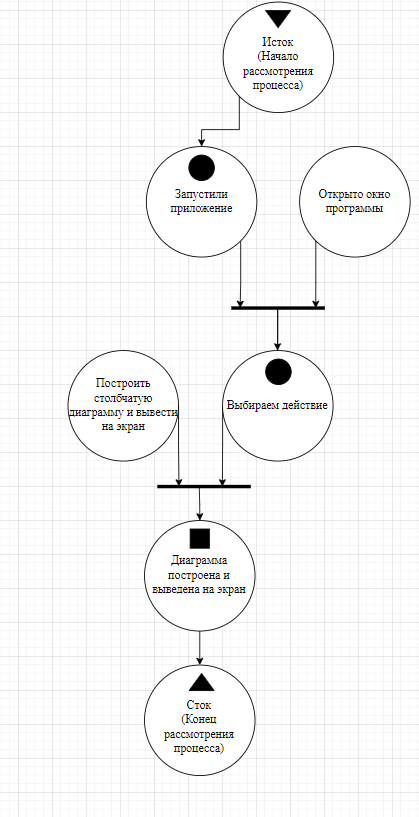
}

}

### Показывать столбчатую диаграмму количества часов, приходящихся на виды учебной нагрузки, за выбранный семестр. Результат работы программы на Рисунке 28.

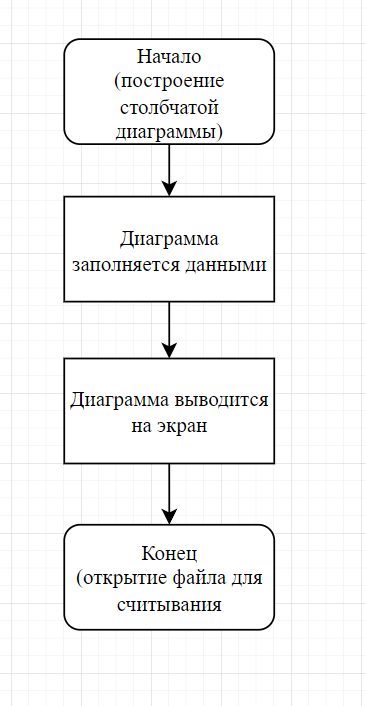


1. Диаграмма количества часов, приходящихся на виды учебной нагрузки, за выбранный семестр.



1. Сеть Петри для алгоритма для диаграммы количества часов, приходящихся на виды учебной нагрузки, за выбранный семестр.

.



1. Схема алгоритма “Диаграмма заполняется данными и выводится на экран”

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

{

// Открытие диалогового окна выбора файла

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Filter = "Файлы Excel (\*.xls;\*.xlsx)|\*.xls;\*.xlsx";

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

{

string filePath = openFileDialog.FileName;

// Определение наименований и соответствующих ячеек

Dictionary<string, string[]> nameToCellAddresses = new Dictionary<string, string[]>()

{

{ "Лекции\r\nКонс. по ЛК на б. пот. ", new string[] { "M101" } },

{ "Экзамены", new string[] { "N101" } },

{ "Зачеты ", new string[] { "O101" } },

{ "Тек. конс. по ЛК\r\nКонс. перед ЭК ", new string[] { "Q101" } },

{ "Лабора-торные работы ", new string[] { "R101" } },

{ "Практ. занятия ", new string[] { "S101" } },

{ "Конт. раб. и ПК. ", new string[] { "U101" } },

{ "Курс. Проект\r\nКурс. работа ", new string[] { "V101" } },

{ "Подготов-ка ВКР ", new string[] { "X101" } },

{ "Произ. практика ", new string[] { "Z101" } },

// Добавьте другие наименования и ячейки

};

Dictionary<string, double> processedNames = new Dictionary<string, double>();

Excel.Application excelApp = new Excel.Application();

Excel.Workbook excelWorkbook = excelApp.Workbooks.Open(filePath);

Excel.Worksheet excelWorksheet = excelWorkbook.Sheets[1];

foreach (var nameAndCells in nameToCellAddresses)

{

string name = nameAndCells.Key;

string[] cellAddresses = nameAndCells.Value;

foreach (string cellAddress in cellAddresses)

{

object cellValue = excelWorksheet.Range[cellAddress].Value2;

// Проверяем, не null ли значение и не является ли строкой " "

if (cellValue != null && !cellValue.ToString().Equals(" "))

{

// Преобразуем значение в число

if (double.TryParse(cellValue.ToString(), out double value))

{

if (processedNames.ContainsKey(name))

{

processedNames[name] += value; // Суммируем значение

}

else

{

processedNames.Add(name, value);

}

}

}

}

}

excelWorkbook.Close(false);

excelApp.Quit();

// Построение диаграммы

foreach (var entry in processedNames)

{

string name = entry.Key;

int value = (int)entry.Value;

bool seriesExists = false;

foreach (var series in chart1.Series)

{

if (series.Name == name)

{

series.Points.Clear();

series.Points.AddXY(name, value);

seriesExists = true;

break;

}

}

if (!seriesExists)

{

Series newSeries = chart1.Series.Add(name);

newSeries.ChartType = SeriesChartType.Column;

newSeries.Points.AddXY(name, value);

}

}

foreach (var series in chart1.Series)

{

series.BorderWidth = 1;

}

chart1.Titles.Clear();

chart1.Titles.Add("Показывать столбчатую диаграмму часов, приходящихся на виды учебной нагрузки, за выбранный семестр.");

chart1.ChartAreas[0].AxisX.LabelStyle.Enabled = false;

}

### Показывать столбчатую диаграмму часов, приходящихся на виды учебной нагрузки, за всё время. Результат работы программы на Рисунке 31.

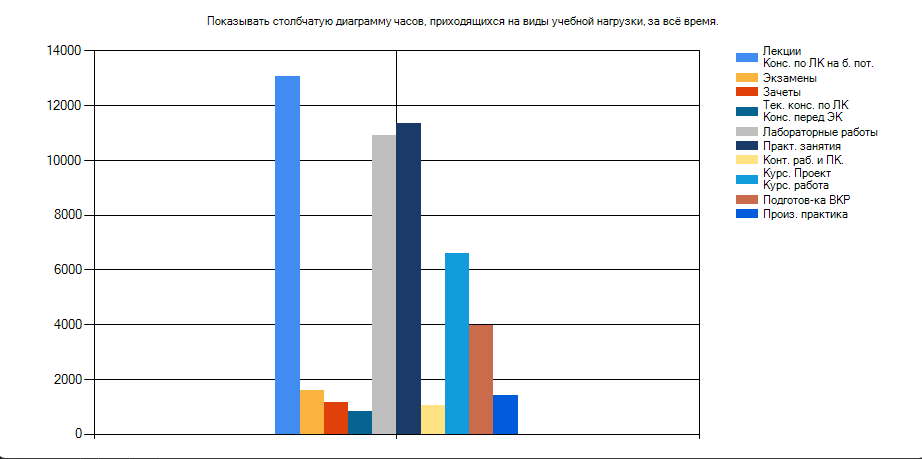


Рисунок 31 – Диаграмма часов, приходящихся на виды учебной нагрузки, за всё время

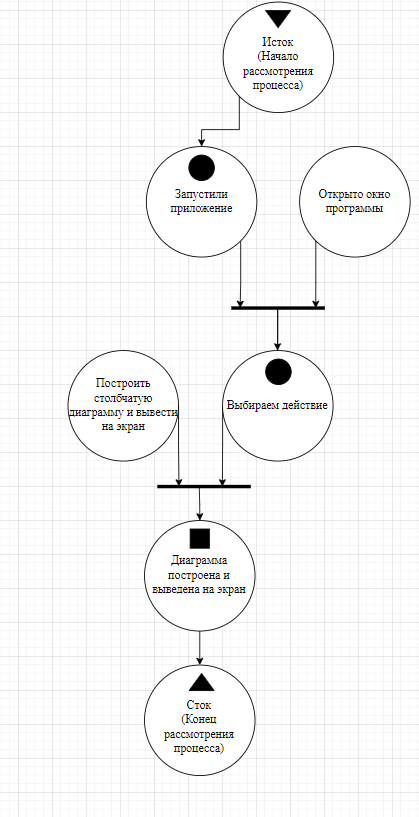


Рисунок 32 – Сеть Петри для алгоритма для диаграммы количества часов, приходящихся на виды учебной нагрузки, за всё время.

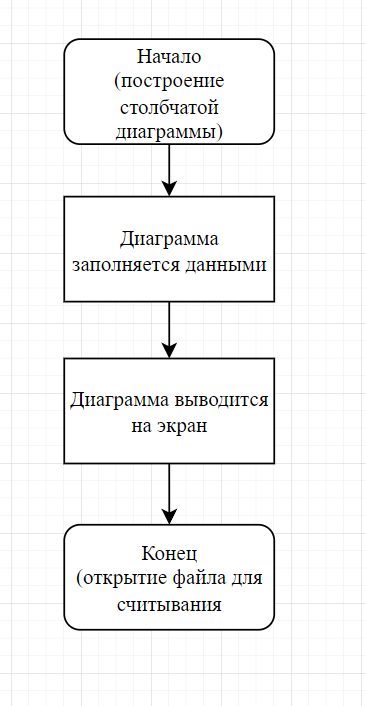


Рисунок 33 - Схема алгоритма “Диаграмма заполняется данными и выводится на экран”

Листинг программы:

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

{

string folderPath = @"D:\Изображения\Сафронов\excels\";

string[] files = Directory.GetFiles(folderPath);

// Определение наименований и соответствующих ячеек

Dictionary<string, string[]> nameToCellAddresses = new Dictionary<string, string[]>()

{

{ "Лекции\r\nКонс. по ЛК на б. пот. ", new string[] { "M101" } },

{ "Экзамены", new string[] { "N101" } },

{ "Зачеты ", new string[] { "O101" } },

{ "Тек. конс. по ЛК\r\nКонс. перед ЭК ", new string[] { "Q101" } },

{ "Лабораторные работы ", new string[] { "R101" } },

{ "Практ. занятия", new string[] { "S101" } },

{ "Конт. раб. и ПК.", new string[] { "U101" } },

{ "Курс. Проект\r\nКурс. работа", new string[] { "V101" } },

{ "Подготов-ка ВКР", new string[] { "X101" } },

{ "Произ. практика", new string[] { "Z101" } },

};

Dictionary<string, double> processedNames = new Dictionary<string, double>();

foreach (string file in files)

{

Excel.Application excelApp = new Excel.Application();

Excel.Workbook excelWorkbook = excelApp.Workbooks.Open(file);

Excel.Worksheet excelWorksheet = excelWorkbook.Sheets[1];

foreach (var nameAndCells in nameToCellAddresses)

{

string name = nameAndCells.Key;

string[] cellAddresses = nameAndCells.Value;

foreach (string cellAddress in cellAddresses)

{

object cellValue = excelWorksheet.Range[cellAddress].Value2;

// Проверяем, не null ли значение и не является ли строкой " "

if (cellValue != null && !cellValue.ToString().Equals(" "))

{

// Преобразуем значение в число

if (double.TryParse(cellValue.ToString(), out double value))

{

if (processedNames.ContainsKey(name))

{

processedNames[name] += value; // Суммируем значение

}

else

{

processedNames.Add(name, value);

}

}

}

}

}

excelWorkbook.Close(false);

excelApp.Quit();

}

// Построение диаграммы

foreach (var entry in processedNames)

{

string name = entry.Key;

int value = (int)entry.Value;

bool seriesExists = false;

foreach (var series in chart1.Series)

{

if (series.Name == name)

{

series.Points.Clear();

series.Points.AddXY(name, value);

seriesExists = true;

break;

}

}

if (!seriesExists)

{

Series newSeries = chart1.Series.Add(name);

newSeries.ChartType = SeriesChartType.Column;

newSeries.Points.AddXY(name, value);

}

}

foreach (var series in chart1.Series)

{

series.BorderWidth = 1;

}

chart1.Titles.Clear();

chart1.Titles.Add(" Показывать столбчатую диаграмму часов, приходящихся на виды учебной нагрузки, за всё время.");

chart1.ChartAreas[0].AxisX.LabelStyle.Enabled = false;

}

### Показывать столбчатую диаграмму количества часов, приходящихся на формы обучения, за всё время. Результат работы программы на Рисунке 34.

### 

Рисунок 34 – Диаграмма количества часов, приходящихся на формы обучения, за всё время

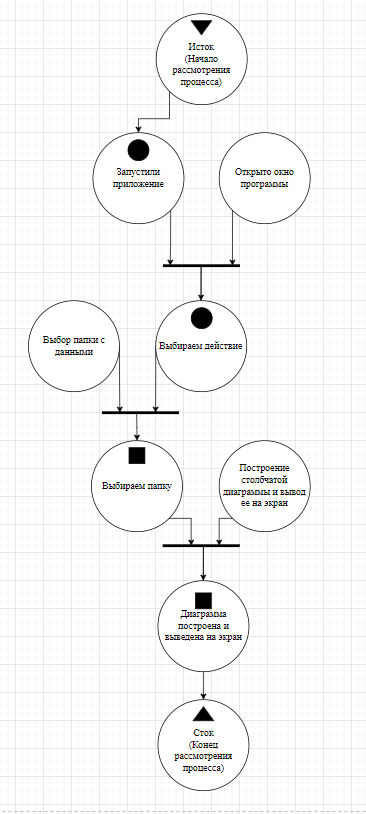


Рисунок 35 – Сеть Петри для алгоритма для диаграммы количества часов, приходящихся на формы обучения, за всё время

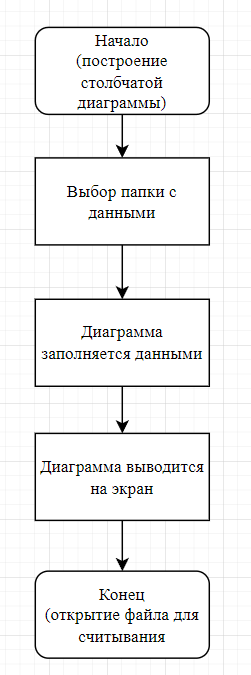


Рисунок 36 – Схема алгоритма “Диаграмма заполняется данными из папки и выводится на экран”

Листинг программы:

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

{

string folderPath = @"D:\Изображения\Сафронов\excels\";

string[] files = Directory.GetFiles(folderPath);

// Определение наименований и соответствующих ячеек

Dictionary<string, string[]> nameToCellAddresses = new Dictionary<string, string[]>()

{

{ "очно-заочная, бакалавриат ", new string[] { "R26" } },

{ "очная, бакалавриат", new string[] { "R57" } },

{ "очная, бакалавриат, индивидуальный учебный план (ускоренное обучение)", new string[] { "R64" } },

{ "очная, магистратура:", new string[] { "R73" } },

{ "очная, специалитет ", new string[] { "R99" } },

};

Dictionary<string, double> processedNames = new Dictionary<string, double>();

foreach (string file in files)

{

Excel.Application excelApp = new Excel.Application();

Excel.Workbook excelWorkbook = excelApp.Workbooks.Open(file);

Excel.Worksheet excelWorksheet = excelWorkbook.Sheets[1];

foreach (var nameAndCells in nameToCellAddresses)

{

string name = nameAndCells.Key;

string[] cellAddresses = nameAndCells.Value;

foreach (string cellAddress in cellAddresses)

{

object cellValue = excelWorksheet.Range[cellAddress].Value2;

// Проверяем, не null ли значение и не является ли строкой " "

if (cellValue != null && !cellValue.ToString().Equals(" "))

{

// Преобразуем значение в число

if (double.TryParse(cellValue.ToString(), out double value))

{

if (processedNames.ContainsKey(name))

{

processedNames[name] += value; // Суммируем значение

}

else

{

processedNames.Add(name, value);

}

}

}

}

}

excelWorkbook.Close(false);

excelApp.Quit();

}

// Построение диаграммы

foreach (var entry in processedNames)

{

string name = entry.Key;

int value = (int)entry.Value;

bool seriesExists = false;

foreach (var series in chart1.Series)

{

if (series.Name == name)

{

series.Points.Clear();

series.Points.AddXY(name, value);

seriesExists = true;

break;

}

}

if (!seriesExists)

{

Series newSeries = chart1.Series.Add(name);

newSeries.ChartType = SeriesChartType.Column;

newSeries.Points.AddXY(name, value);

}

}

foreach (var series in chart1.Series)

{

series.BorderWidth = 1;

}

chart1.Titles.Clear();

chart1.Titles.Add(" Показывать столбчатую диаграмму часов, приходящихся на формы обучения, за всё время.");

chart1.ChartAreas[0].AxisX.LabelStyle.Enabled = false;

}

}

## Инструкция пользователя

### Главная форма

При открытии программы появится интерфейс главной формы, состоящий из 3 элементов:

Word – Кнопка перехода на форму для работы с документами Word;

Excel – Кнопка переход на форму для работы с таблицами Excel;

Вывод Графиков – Кнопка переход на форму для построения столбчатых диаграмм;

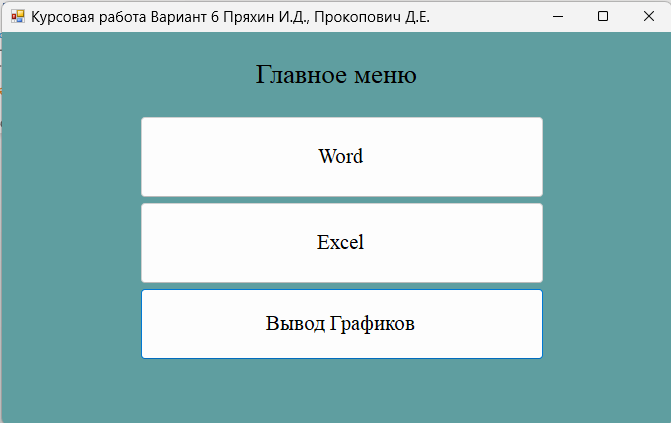


Рисунок 37 – Интерфейс главной формы

### Форма для работы с файлами Word

В данной форме присутствуют элемент DataGridView для отображения считываемой таблицы и 2 кнопки:

Считать таблицу из файла – считывает таблицу из выбранного документа Word

Конвертировать Excel – конвертирует выбранный документ Word в таблицу Excel

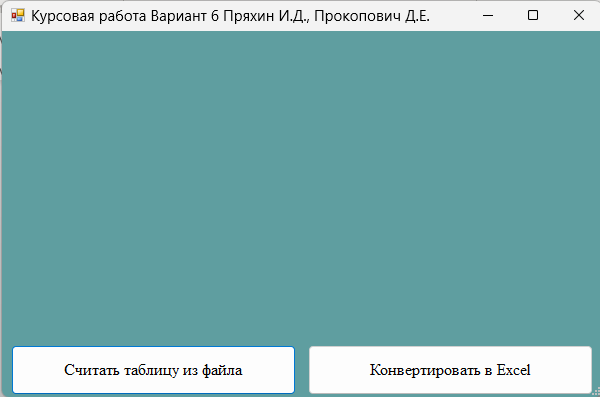


Рисунок 45 – Интерфейс формы для работы с файлами Word

### Форма для работы с файлами Excel

В данной форме присутствуют элемент DataGridView для отображения считываемой таблицы и 2 кнопки:

Считать файла – считывает выбранный файл Excel

Конвертировать в Word – конвертирует выбранную таблицу Excel в Документ Word

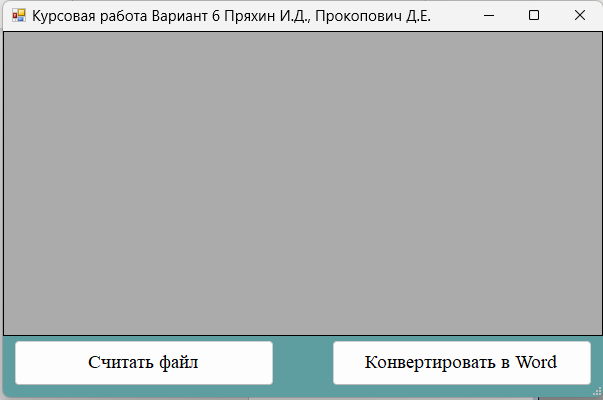


Рисунок 46 – Интерфейс формы для работы с файлами Excel

### Форма для отображения диаграмм

В данной форме присутствуют элемент Chart для отображения столбчатых диаграмм и 8 кнопок:

Очистить – кнопка очищает Chart-элемент

Количество изданий за всё время – кнопка выводит столбчатую диаграмму по количеству изданий за всё время

Объемы за всё время – кнопка выводит столбчатую диаграмму по объёмам авторских листов за всё время

Кварталы за всё время – кнопка выводит столбчатую диаграмму по кварталам сдачи за всё время

Количество изданий за год – кнопка выводит столбчатую диаграмму по количеству изданий за год

Кварталы за год – кнопка выводит столбчатую диаграмму по кварталам сдачи за год

Вид изданий за всё время – кнопка выводит столбчатую диаграмму по видам изданий за всё время

Вид изданий за год – кнопка выводит столбчатую диаграмму по видам изданий за год

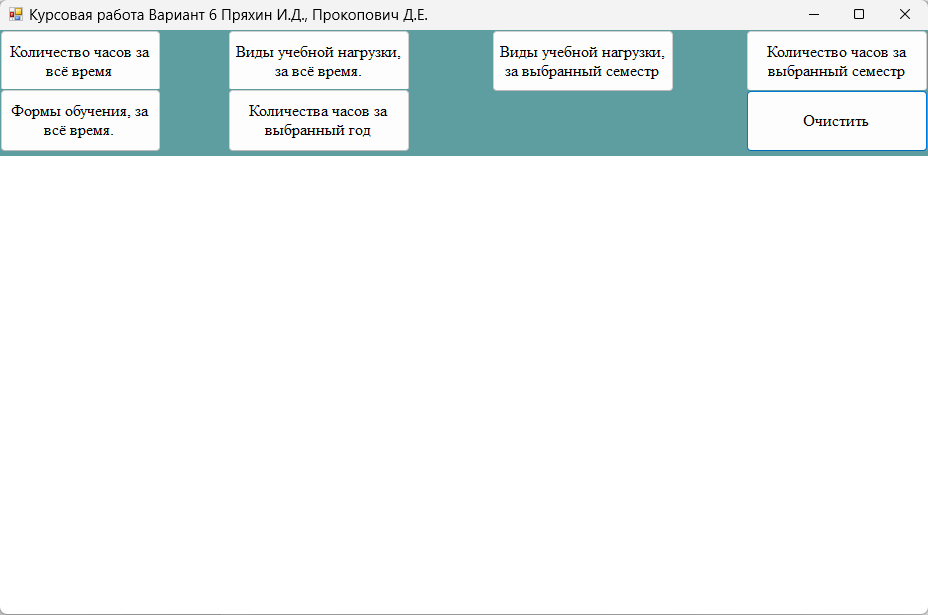


Рисунок 47 – Интерфейс формы для отображения диаграмм

# Заключение

В процессе выполнения данной работы были освоены следующие навыки:

* Анализ печатной формы
* Описание предназначения печатной формы
* Выявление различий в печатных формах
* Унификация печатных форм
* Разработка экранной формы для конвертации документов Word - Excel / Excel - Word
* Графический анализ данных, содержащихся в печатной форме
* Создание инструкции пользователя

# Выводы

Данная курсовая работа помогла закрепить навыки полученные в течение семестра, такие как:

* подключение библиотек и классов и их использование в программе;
* настройка электронных таблиц Excel с помощью кода;
* настройка электронных документов Word с помощью кода;
* конвертация файлов Excel и Word между собой.

Также познакомились с таким элементом как Chart, с его помощью научились анализировать документы и на основе анализа построили столбчатые диаграммы по следующим условиям:

1. Показывать столбчатую диаграмму общего количества часов за всё время.

2. Показывать столбчатую диаграмму общего количества часов за выбранный учебный год.

3. Показывать столбчатую диаграмму общего количества часов за выбранный календарный год.

4. Показывать столбчатую диаграмму общего количества часов за выбранный семестр.

5. Показывать столбчатую диаграмму часов, приходящихся на виды учебной нагрузки, за выбранный семестр.

6. Показывать столбчатую диаграмму часов, приходящихся на виды учебной нагрузки, за всё время.

7. Показывать столбчатую диаграмму часов, приходящихся на формы обучения, за всё время.