Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра экономической математики, информатики и статистики (ЭМИС)

Программная генерация и редактирование текстовых документов

Отчёт по лабораторной работе

по дисциплине «Специализированная подготовка разработчиков бизнес приложений

Студент гр. з-38

\_\_\_\_\_\_\_\_\_\_\_\_ М.Е. Савельев

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2023г.

Руководитель

ст.преподаватель кафедры ЭМИС

\_\_\_\_\_\_\_\_\_\_\_\_\_ Я.В. Костелей

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2023г.

Томск 2023

1. Цель работы

Целью лабораторной работы является получение навыков работы с программной генерацией и модификацией текстовых документов с использованием библиотек работы с форматом PDF.

1. Ход работы

Запустим IDE «Microsoft Visual Studio 2022» (далее – MS VS). Создадим новый проект – «Консольное приложение для .NET Framework».

Затем подключим библиотеку «iTextSharp» для работы с форматом «pdf». Для этого в менеджере NuGet найдем соответствующий модуль и нажмём «Установить».

После установки библиотеки переходим к написанию кода приложения.

Указываем путь до файла шаблона и результирующего PDF файла, а также задаём лист шаблонных фраз для разметки документа (рисунок 2.2).

Также укажем переменные для идентификации номера рисунка, таблицы и раздела. Далее, создадим метод «Make», в котором будет формироваться.

В данном методе считываем строки шаблонного документа, устанавливаем шрифт по умолчанию для основного текста и создаём документа с указанными параметрами (рисунок 2.3).

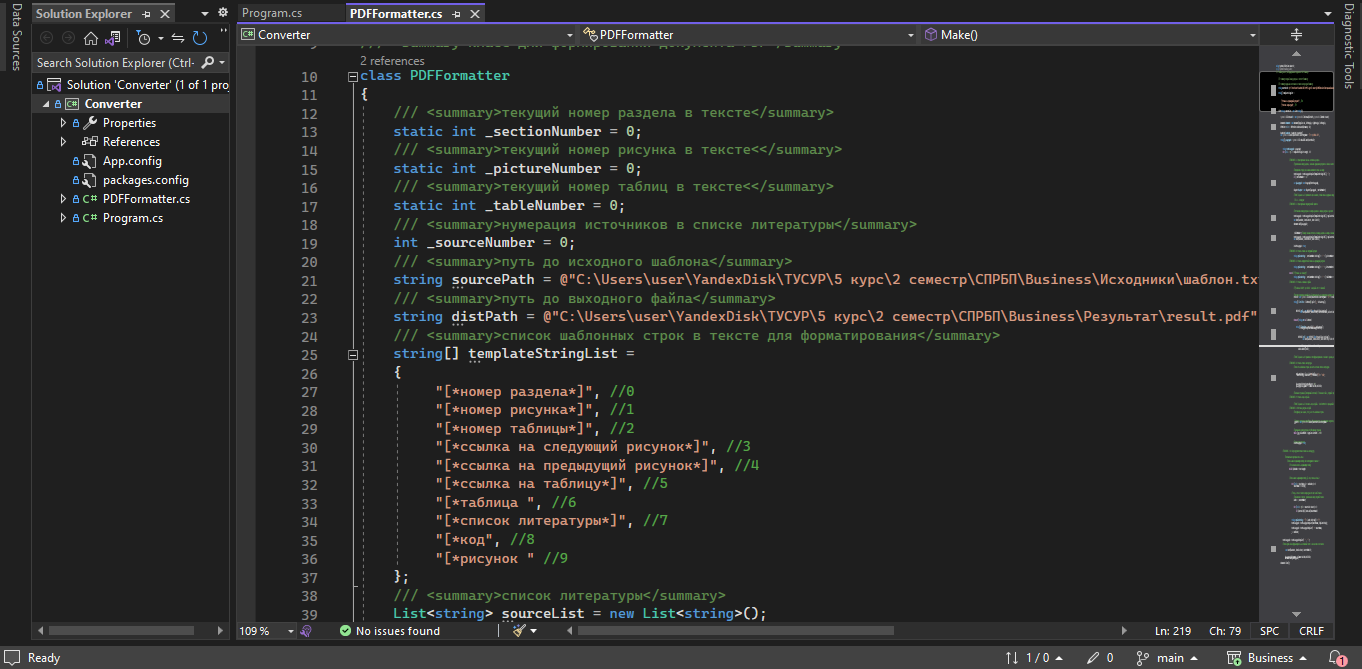


Рисунок 2.2 – скриншот основных переменных

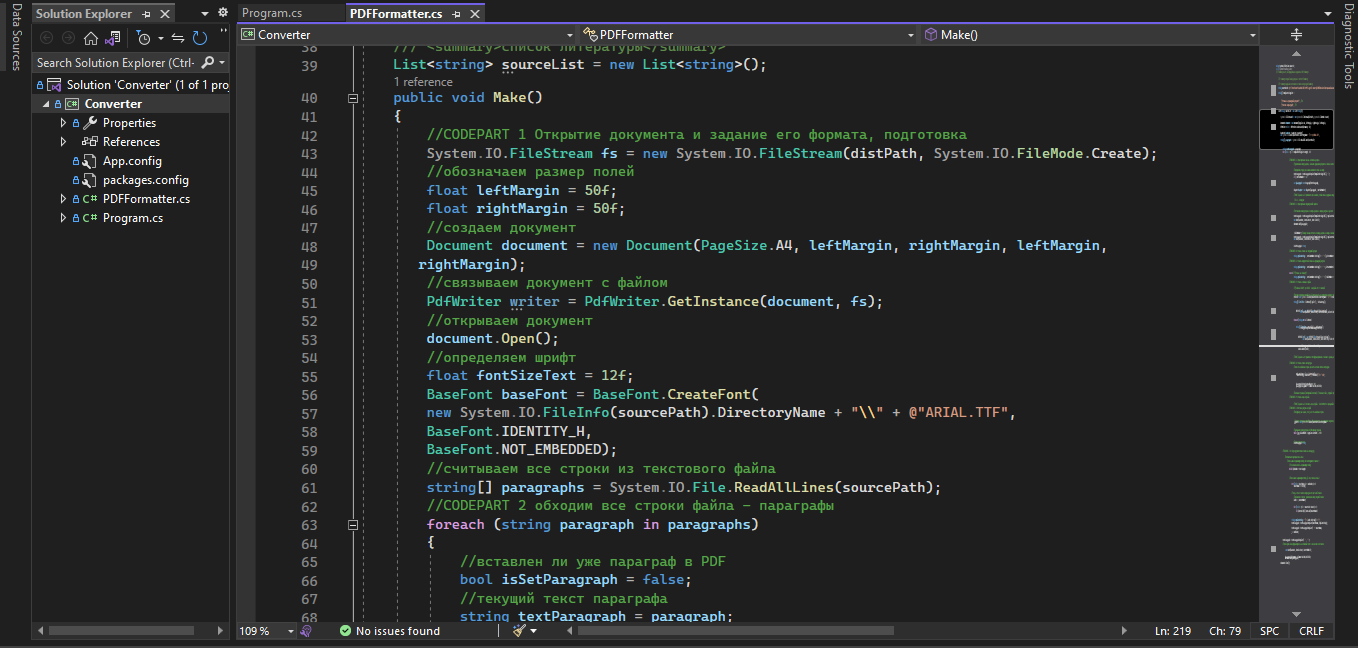


Рисунок 2.3 – скриншот метода «Make»

Все строки шаблонного документа были считаны в лист «paragraphs». В цикле по листу ищем совпадения фраз из созданного ранее (рисунок 2.2) листа «templateStringList». С помощью инструмента «Switch – case» определяем участки кода, в которых произведём редактирование текста согласно шаблонным фразам (рисунок 2.4).

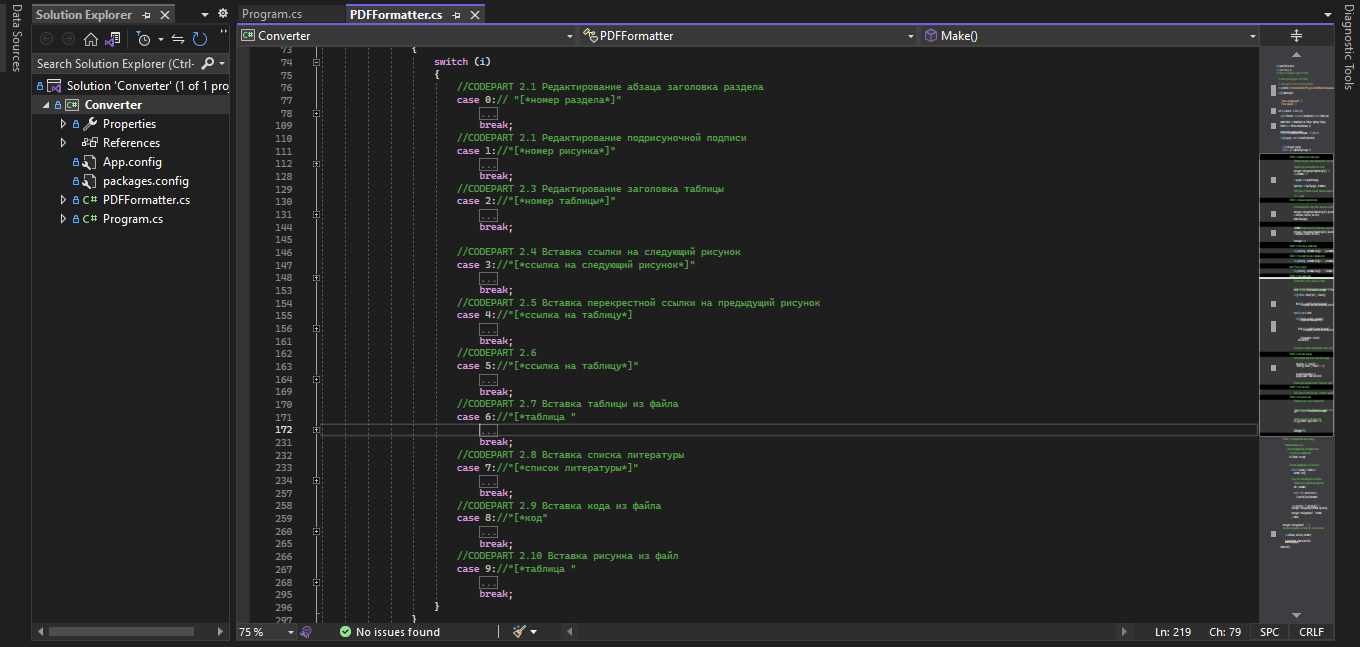


Рисунок 2.4 – скриншот условий инстурмента «Switch – case»

После блока «Switch – case» задаём блок условия «!isSetParagraph», который отвечает за стандартное редактирования абзаца, если он не задан. (рисунок 2.5). В блоке определяем параметры редактирования текста и отступы.

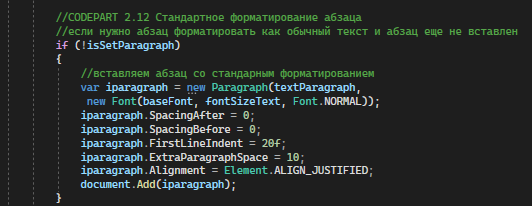


Рисунок 2.5 – скриншот блока условия «!isSetParagraph»

Затем в «Main» вносим следующий код и запускаем приложения для проверки срабатывания блока условия (рисунок 2.6).

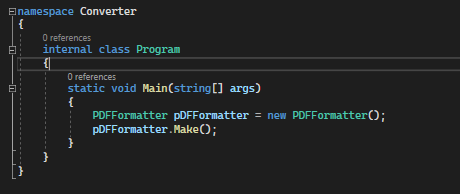


Рисунок 2.6 – скриншот функции «Main»

Результат запуска приложения изображён на рисунке 2.7.

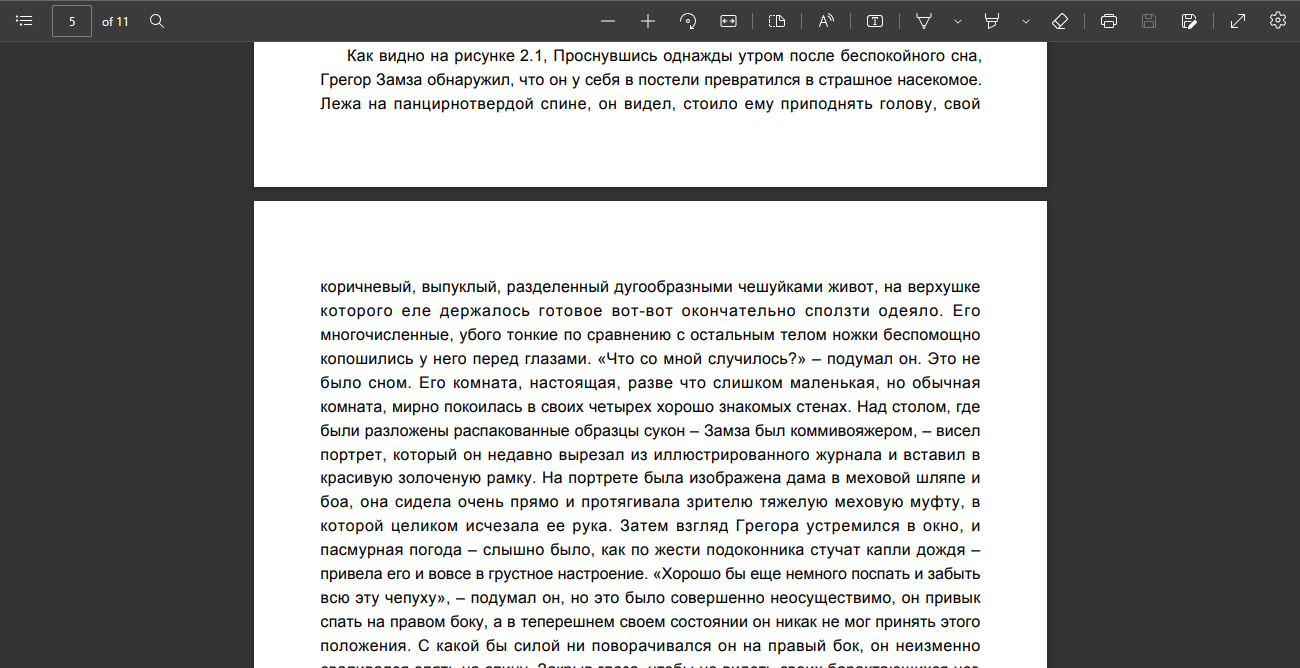
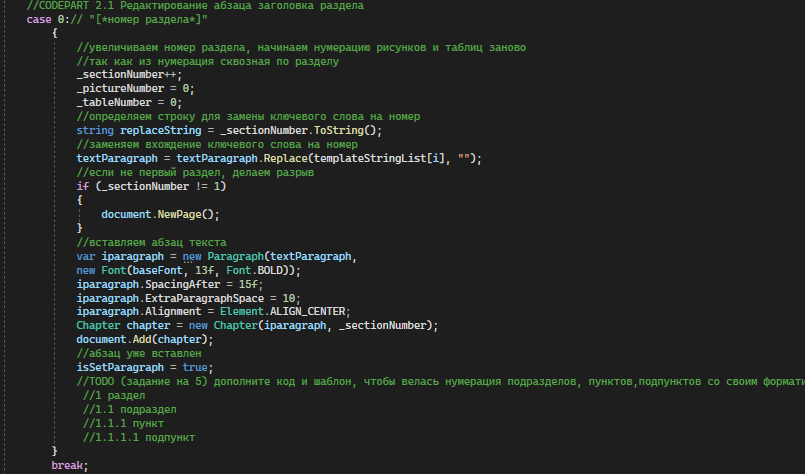


Рисунок 2.7 – скриншот сформированного PDF документа

Переходим к редактированию разделов (case 0). На рисунке 2.8 изображено создание новой страницы, установка типа и размера шрифта, добавление заголовка на страницу и в навигационную панель документа.

Рисунок 2.8 – скриншот редактирования блока «Номер раздела»

Сохраняем изменения файла. Запускаем приложение в режиме «Отладка». Результат выполнения приложения изображён на рисунке 2.9.

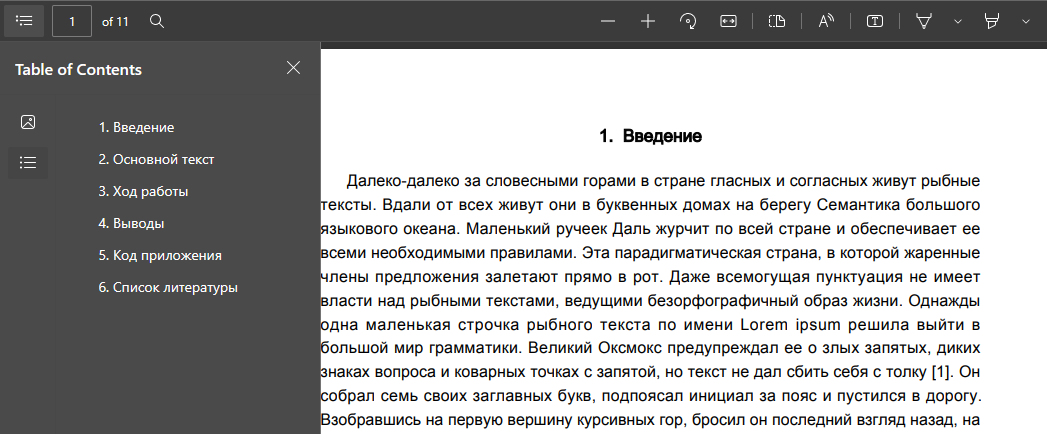


Рисунок 2.9 – Скриншот списка разделов PDF документа

Выполним коммит «Вставка абзацев разделов (2)» (рисунок 2.10).

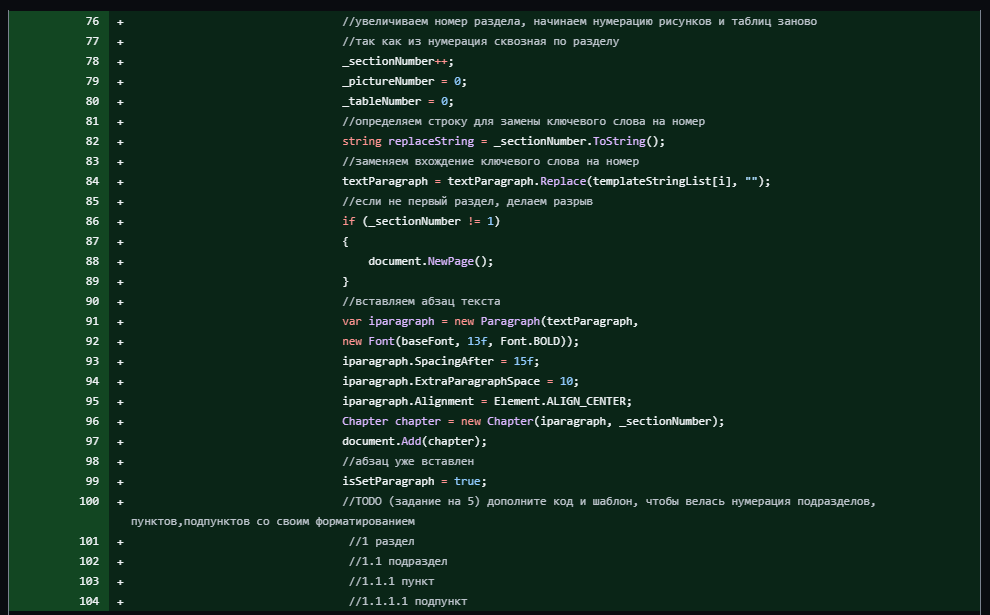


Рисунок 2.10 – Скриншот коммита в репозитории

Переходим к добавлению подрисуночных подписей и заголовков таблицы (рисунок 2.11).

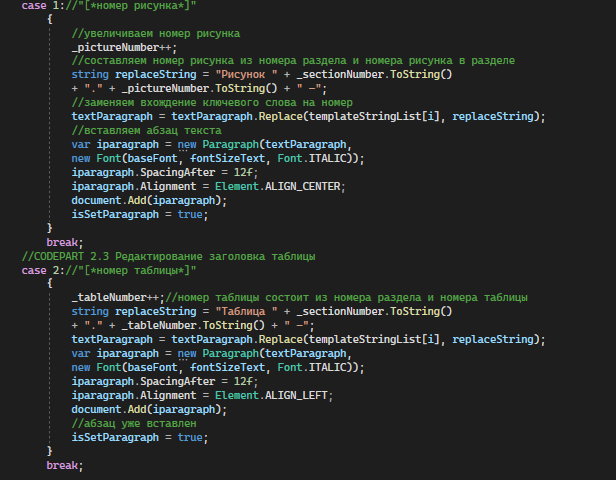


Рисунок 2.11 – Скриншот окна формы документа

Сохраняем изменения и запускаем приложение в режиме «Отладка». Результат выполнения изображён на рисунке 2.12. Выполняем коммит «Вставка картинок из файла (4)».

Отредактируем объявления и упоминания рисунков и таблиц в тексте.

На рисунке 2.13 изображен код для редактирования замены шаблонных фраз на номера рисунков и таблиц.

Результат выполнения кода видим на рисунке 2.12 на строке перед рисунком, аналогичное объявление у таблиц и ссылок на предыдущие рисунки. Выполняем коммит «Вставка ссылок (5)» и переходим к вставке таблицы из CSV файла.

Выбираем файл таблицы из директории и записываем все строки из него в массив «listRows». Отдельно, в переменную «listTitle», сохраняем названия колонок таблицы. Создаём таблицу с размером равным количеству элементов массива заголовков.

Далее вносим в таблицу сами заголовки и строки, применяя редактирование текса по умолчанию. Одновременно вносим правки для изменения стиля таблицы в соответствии с заданием под звездочкой. Код представлен на рисунке 2.14.

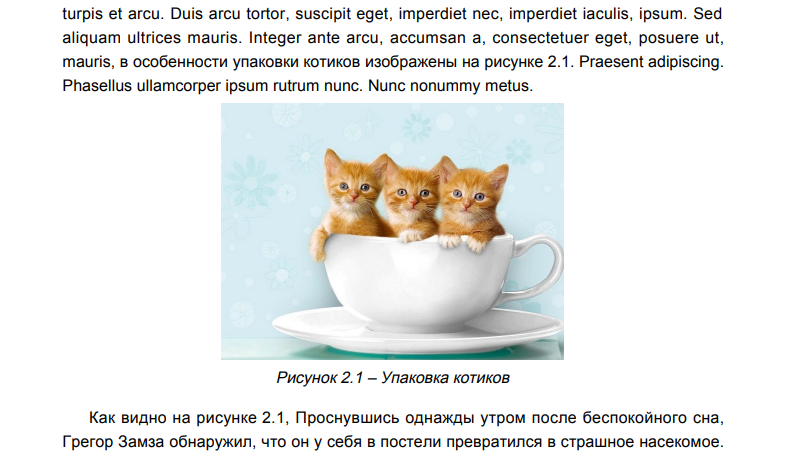


Рисунок 2.12 – Скриншот подрисуночной подписи и ссылки на рисунок

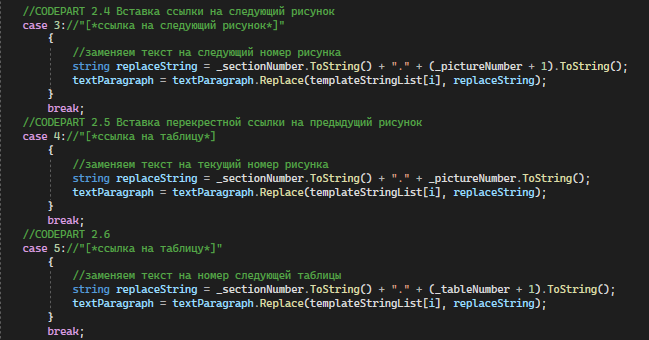


Рисунок 2.13 – Скриншот редактирования подписей рисунков и таблиц



Рисунок 2.14 – Скриншот вставки таблицы из файла

Выполним коммит «Вставка таблицы из файла (6)» и коммит «Форматирование таблицы (0)» и переходим к заполнению списка литературы.

На рисунке 2.15 изображён код для формирования списка литературы, установка таких параметров текста как: выравнивание, отступы…

Результат выполнения приложения в режиме «Отладка» изображён на рисунке 2.16. Выполняем коммит «Сборка и вставка списка литературы (7)»

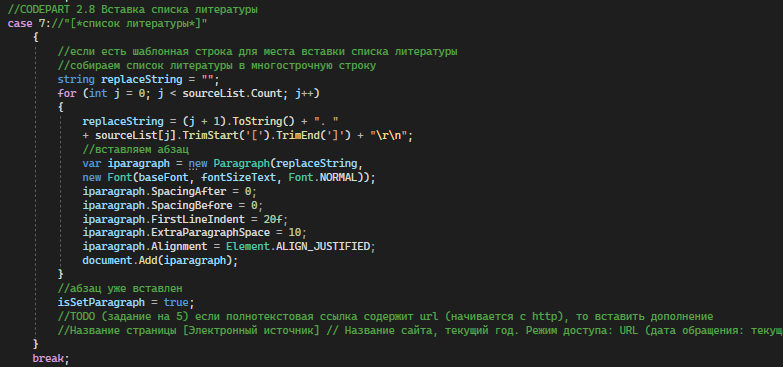


Рисунок 2.15 – Скриншот кода редактирования списка литературы

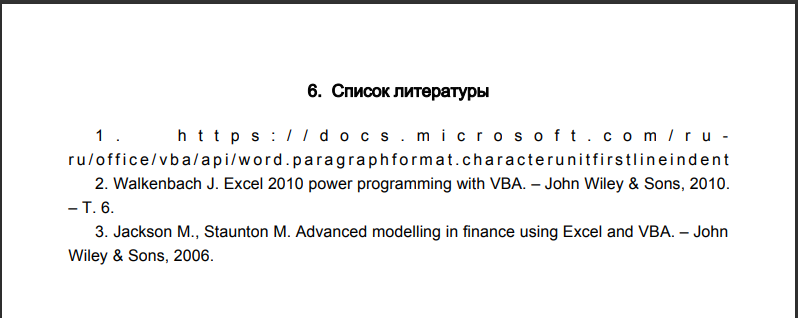


Рисунок 2.16 – Скриншот списка литературы

Выполним задание под звездочкой. Необходимо удалить из документа двойные пробелы (кроме кода), перед знаком «[» вставить неразрывный пробел (не просто пробел, а '\u00A0'). Для реализации этого функционала были внесены изменения в код, представленные на рисунке 2.17, и сделан коммит «Замена пробелов (0)».

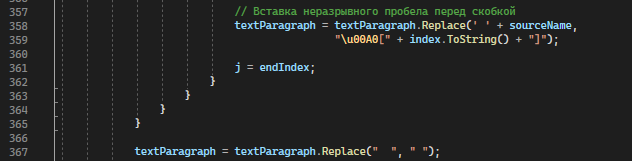


Рисунок 2.16 – Скриншот изменений в коде

Код приложения

using iTextSharp.text;

using iTextSharp.text.pdf;

using System;

using System.Collections.Generic;

using System.Reflection;

using System.IO;

using System.Threading.Tasks;

class PDFFormatter

{

static int \_sectionNumber = 0;

static int \_pictureNumber = 0;

static int \_tableNumber = 0;

int \_sourceNumber = 0;

string sourcePath = @"шаблон.txt";

string distPath = @"result.pdf";

string[] templateStringList =

{

"[\*номер раздела\*]",

"[\*номер рисунка\*]",

"[\*номер таблицы\*]",

"[\*ссылка на следующий рисунок\*]",

"[\*ссылка на предыдущий рисунок\*]",

"[\*ссылка на таблицу\*]",

"[\*таблица ",

"[\*cписок литературы\*]",

"[\*код",

"[\*рисунок "

};

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

public void Make()

{

System.IO.FileStream fs = new System.IO.FileStream(distPath, System.IO.FileMode.Create);

float leftMargin = 50f;

float rightMargin = 50f;

Document document = new Document(PageSize.A4, leftMargin, rightMargin, leftMargin,

rightMargin);

PdfWriter writer = PdfWriter.GetInstance(document, fs);

document.Open();

float fontSizeText = 12f;

BaseFont baseFont = BaseFont.CreateFont(

new System.IO.FileInfo(sourcePath).DirectoryName + "\\" + @"ARIAL.TTF",

BaseFont.IDENTITY\_H,

BaseFont.NOT\_EMBEDDED);

string[] paragraphs = System.IO.File.ReadAllLines(sourcePath);

foreach (string paragraph in paragraphs)

{

bool isSetParagraph = false;

string textParagraph = paragraph;

for (int i = 0; i < templateStringList.Length; i++)

{

if (paragraph.Contains(templateStringList[i]))

{

switch (i)

{

case 0:

{

\_sectionNumber++;

\_pictureNumber = 0;

\_tableNumber = 0;

string replaceString = \_sectionNumber.ToString();

textParagraph = textParagraph.Replace(templateStringList[i], "");

if (\_sectionNumber != 1)

{

document.NewPage();

}

var iparagraph = new Paragraph(textParagraph,

new Font(baseFont, 13f, Font.BOLD));

iparagraph.SpacingAfter = 15f;

iparagraph.ExtraParagraphSpace = 10;

iparagraph.Alignment = Element.ALIGN\_CENTER;

Chapter chapter = new Chapter(iparagraph, \_sectionNumber);

document.Add(chapter);

isSetParagraph = true;

}

break;

case 1:

{

\_pictureNumber++;

string replaceString = "Рисунок " + \_sectionNumber.ToString()

+ "." + \_pictureNumber.ToString() + " –";

textParagraph = textParagraph.Replace(templateStringList[i], replaceString);

var iparagraph = new Paragraph(textParagraph,

new Font(baseFont, fontSizeText, Font.ITALIC));

iparagraph.SpacingAfter = 12f;

iparagraph.Alignment = Element.ALIGN\_CENTER;

document.Add(iparagraph);

isSetParagraph = true;

}

break;

case 2:

{

\_tableNumber++;

string replaceString = "Таблица " + \_sectionNumber.ToString()

+ "." + \_tableNumber.ToString() + " –";

textParagraph = textParagraph.Replace(templateStringList[i], replaceString);

var iparagraph = new Paragraph(textParagraph,

new Font(baseFont, fontSizeText, Font.ITALIC));

iparagraph.SpacingAfter = 12f;

iparagraph.Alignment = Element.ALIGN\_LEFT;

document.Add(iparagraph);

isSetParagraph = true;

}

break;

case 3:

{

string replaceString = \_sectionNumber.ToString() + "." + (\_pictureNumber + 1).ToString();

textParagraph = textParagraph.Replace(templateStringList[i], replaceString);

}

break;

case 4:

{

string replaceString = \_sectionNumber.ToString() + "." + \_pictureNumber.ToString();

textParagraph = textParagraph.Replace(templateStringList[i], replaceString);

}

break;

case 5:

{

string replaceString = \_sectionNumber.ToString() + "." + (\_tableNumber + 1).ToString();

textParagraph = textParagraph.Replace(templateStringList[i], replaceString);

}

break;

case 6:

{

string csvPath = textParagraph.Replace(templateStringList[i], "")

.Replace("\*", "").Replace("\r", "").Replace("]", "");

csvPath = new System.IO.FileInfo(sourcePath).DirectoryName + "\\" + csvPath;

string[] listRows = System.IO.File.ReadAllLines(csvPath);

string[] listTitle = listRows[0].Split(";,".ToCharArray(),

StringSplitOptions.RemoveEmptyEntries);

PdfPTable table = new PdfPTable(listTitle.Length);

foreach (string title in listTitle)

{

PdfPCell cell = new PdfPCell(new Phrase(title.ToString(),

new Font(baseFont, fontSizeText, Font.BOLDITALIC, Color.YELLOW)));

cell.BackgroundColor = Color.GRAY;

cell.BorderWidthBottom = 1;

table.AddCell(cell);

}

int bgColorIterator = 0;

foreach (string row in listRows)

{

bgColorIterator++;

if (row == listRows[0]) continue;

string[] listValue = row.Split(";,".ToCharArray(),

StringSplitOptions.RemoveEmptyEntries);

foreach (string value in listValue)

{

PdfPCell style = new PdfPCell();

style.BorderColor = Color.RED;

style.HorizontalAlignment = 1;

PdfPCell cell = new PdfPCell(new Phrase(value.ToString(),

new Font(baseFont, fontSizeText, Font.HELVETICA, Color.CYAN)));

cell.Border = 0;

if (bgColorIterator % 2 == 0)

cell.BackgroundColor = Color.GRAY;

else

cell.BackgroundColor = Color.WHITE;

table.AddCell(cell);

}

}

document.Add(table);

isSetParagraph = true;

}

break;

case 7:

{

string replaceString = "";

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

{

replaceString = (j + 1).ToString() + ". "

+ sourceList[j].TrimStart('[').TrimEnd(']') + "\r\n";

var iparagraph = new Paragraph(replaceString,

new Font(baseFont, fontSizeText, Font.NORMAL));

iparagraph.SpacingAfter = 0;

iparagraph.SpacingBefore = 0;

iparagraph.FirstLineIndent = 20f;

iparagraph.ExtraParagraphSpace = 10;

iparagraph.Alignment = Element.ALIGN\_JUSTIFIED;

document.Add(iparagraph);

}

isSetParagraph = true;

}

break;

case 8:

{

//если есть шаблонная строка для места вставки кода

textParagraph = "тут будет ваш код";

//TODO (задание на 5) вставить код из файла - CourierNew 8 пт одинарный без отступа в рамке

}

break;

case 9:

{

string jpgPath = textParagraph.Replace(templateStringList[i],

"").Replace("\*", "").Replace("\r", "").Replace("]", "");

jpgPath = new System.IO.FileInfo(sourcePath).DirectoryName

+ "\\" + jpgPath;

Image jpg = Image.GetInstance(jpgPath);

jpg.Alignment = Element.ALIGN\_CENTER;

jpg.SpacingBefore = 12f;

float procent = 90;

while (jpg.ScaledWidth > PageSize.A4.Width / 2.0f)

{

jpg.ScalePercent(procent);

procent -= 10;

}

document.Add(jpg);

isSetParagraph = true;

}

break;

}

}

}

string text = textParagraph;

if (text.Contains("["))

{

for (int j = 0; j < text.Length - 1; j++)

{

if (text[j] == '[' && text[j + 1] != '\*')

{

int startIndex = j;

int endIndex = startIndex + 1;

while (endIndex < text.Length

&&

text[endIndex] != ']')

{

endIndex++;

}

string sourceName = "";

if (text[endIndex] == ']')

{

for (int k = startIndex; k <= endIndex; k++)

{

sourceName += text[k];

}

int index = 0;

if (!sourceList.Contains(sourceName))

{

sourceList.Add(sourceName);

\_sourceNumber++;

index = \_sourceNumber;

}

else

{

for (int k = 0; k < sourceList.Count; k++)

{

if (sourceList[k].Contains(sourceName))

{

index = k + 1;

}

}

}

string replaceString = "[" + index.ToString() + "]";

textParagraph = textParagraph.Replace(sourceName, replaceString);

textParagraph = textParagraph.Replace(' ' + sourceName,

"\u00A0[" + index.ToString() + "]");

j = endIndex;

}

}

}

}

textParagraph = textParagraph.Replace(" ", " ");

if (!isSetParagraph)

{

var iparagraph = new Paragraph(textParagraph,

new Font(baseFont, fontSizeText, Font.NORMAL));

iparagraph.SpacingAfter = 0;

iparagraph.SpacingBefore = 0;

iparagraph.FirstLineIndent = 20f;

iparagraph.ExtraParagraphSpace = 10;

iparagraph.Alignment = Element.ALIGN\_JUSTIFIED;

document.Add(iparagraph);

}

}

document.Close();

}

}

Выводы

В ходе выполнения лабораторной работы получены навыки работы с программной генерацией и модификацией текстовых документов с использованием библиотеки «iTextSharp» для работы с форматом PDF. Сгенерирован документа формата «\*.pdf» из исходного шаблона с разметкой разделов, подрисуночными подписями, внутритекстовыми ссылками на литературу, таблицей и списком литературы.