РЕФЕРАТ

Выпускная работа бакалавра 48 с., 5 рис., 1 табл., 10 источн., 2 прил.

XML, WORD, DJVU, FB2, PDF, ПАРСИНГ, ЭЛЕКТРОННЫЙ ДОКУМЕНТ, ФОРМАТ, C#

Объектом исследования являются форматы электронных документов.

Цель работы – разработка модели хранения данных и их представления, а так же программы для их обработки.

В результате проделанной работы был реализован XML парсер, объединяющий информацию из двух файлов разного назначения.

Разработанную модель разделения данных можно использовать для создания электронных документов различного формата.

Рекомендуется использовать ее как средство промежуточного сохранения данных в программах-конструкторах различного назначения.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Информация – неотъемлемая часть нашей жизни. Мы потребляем и обрабатываем ее из различных источников в различных видах и различными способами. Один из видов информации – текст. Он может быть получен из документов, книг, статей, журналов, интернет-сайтов. В цифровую эпоху важным элементом работы с текстом является не только его хранение и передача, но и представление.

В разных структурах и организациях есть свои нормативы для составления документов – стандарты. Они затрагивают любой элемент документа: от заголовков и содержания до таблиц и рисунков. Стандарты позволяют создавать документы в единообразном стиле, избегать коллизий при оформлении того или иного участка (если такое предусмотрено стандартом), а также позволяют описывать структуру документа.

Однако, следовать всем требованиям, накладываемыми нормативами, для людей, создающих документы впервые – задача трудоемкая. Можно запомнить основные правила составления заголовков, абзацев, рисунков, нумерации – этого будет достаточно для создания простого документа. Также не простым является процесс редактирования документа для соответствия другому стандарту. В связи с этим становится актуальным вопрос упрощения создания электронных документов, на которые наложены требования оформления, использовав определенный формат представления данных, а также формализация требований отображений этих данных.

Целью выпускной квалификационной работы является разработка XML парсера, обрабатывающего данные определенного вида. Для ее выполнения необходимо разработать алгоритм обработки файлов, написанных на языке разметки XML. Поставленная цель позволит решить проблему изменения стиля содержимого, вследствие непреднамеренного упущения какого-либо правила при составлении документа. Для достижения поставленной цели целесообразно использовать язык разметки XML,

необходимый для хранения текстовых данных, и файл стилей по аналогии с файлом CSS для HTML.

Новизна данной работы обусловлена использованием подхода разделения данных от представления. Исходные данные и правила их отображения должны находиться в отдельных файлах и заключены в соответствующие теги с использованием XML разметки.

1 Описание существующих форматов хранения электронных документов

В контексте цифровых данных целесообразно называть любой цифровой источник текстовой информации, будь то статья, документ, электронная книга, одним словом – электронный документ. Такое обобщенное название условно, ведь для определённых типов документов целесообразней использовать предназначенные для них форматы: для электронных книг – ePub, fb2, djvu, для статей предпочтительней использовать pdf, для документооборота – docx, а для простого текста, не содержащего в себе мультимедиа контента, достаточно txt. Это обусловлено способом взаимодействия с этими данными: создание, редактирование или просто чтение; а также устройством, через которое с ними взаимодействуют создатели или конечные потребители.

Рассмотрим наиболее популярные на сегодняшний день форматы хранения и отображения электронных документов, проанализируем преимущества и недостатки этих форматов.

Большинство из используемых форматов являются контейнерами – представляют собой архив, хранящий основной текст, как правило, в виде XML файла, таблицу стилей, медиаданные и т. п.

1.1 EPub. Характеристики и особенности

Electronic Publication (ePub) – открытый формат электронных версий книг. Разработчик – Международный форум цифровых публикаций (IDPF). Вышел в свет в 2007 году и развивается до сих пор – последняя версия 3.3 вышла 25 мая 2023 года.

Данный формат является архивом, содержащим в себе сайт, т. к. накладывает требования на средства его чтения: они должны поддерживать

HTML5, JavaScript, CSS, SVG, что, по сути, является требованиям к web-браузерам [1].

Пример содержимого XML файла для формата ePub:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:epub="http://www.idpf.org/2007/ops"
xmlns:ev="http://www.w3.org/2001/xml-events" epub:prefix="media:
http://idpf.org/epub/vocab/media/#">
  <head>
    <meta charset="utf-8" />
    <link rel="stylesheet" type="text/css" href="../css/shared-
culture.css" />
  </head>
  <body>
    <section class="base">
      <h1>Introduction</h1>
      <p>This is an introduction to the topic of the ebook.</p>
      <img class="left" src="../intro.jpg" alt="Introduction" />
    </section>
  </body>
</html>
```

Приведем особенности формата ePub.

Гибкость отображения. Динамическая разметка позволяет комфортно просматривать документ на любом устройстве: от смартфонов до компьютеров, в зависимости от размеров дисплея.

Конвертируемость. Формат легко преобразовать для чтения в интернете, встроив в сайт или просто открыть через браузер.

Поиск по содержимому. Поиск можно осуществлять из содержащей книгу библиотеки.

А также поддерживаются некоторые другие функции: использование закладок, выделение отрывков текста и примечания, поддержка *DRM* (Digital rights management англ. – технические средства защиты авторских прав).

Главный недостаток формата ePub связан с информационной безопасностью. Системам чтения, реализующим данный стандарт полностью присуще те же уязвимости, что и web-браузерам. В статье Emma Woollacott от 1 июня 2021 года на сайте portswigger.net говорится: «Используя полуавтоматический испытательный стенд, доступный на GitHub , исследователи обнаружили, что 16 из 97 исследованных систем позволяли ePub сливать информацию о файловой системе пользователя и в восьми случаях извлекать содержимое файла. Злоумышленники, предупреждают они, могут добиться полной компрометации системы пользователя, используя определенные аспекты реализации систем чтения» [2].

Решением проблемы уязвимости является согласие или запрет пользователей на выполнение кода JavaScript в средствах чтения, а также ужесточения требований безопасности.

Формат ePub широко распространён как в российском сегменте Интернета, так и в иностранном. Книги данного формата доступны в Google Play Books, Apple Books, а также в Amazon Kindle.

1.2 FB2 и FB3 как альтернативный поход к литературным произведениям

FictionBook (англ. – художественная книга) или FeedBook – это формат хранения электронных книг от команды российских разработчиков, во главе с Д. Грибовым, являющегося тех. директором «ЛитРес». FB2 был разработан в 2004 году, последняя версия 2.21 вышла 16 января 2008 года.

По своей структуре это всего лишь XML файл, содержащий только текстовую информацию. Однако с использованием кодировки Base64 можно хранить и изображения.

Фрагмент книги М. Булгакова «Собачье сердце» в формате FB2:

```
<?xml version="1.0" encoding="UTF-8"?>
<FictionBook xmlns="http://www.gribuser.ru/xml/fictionbook/2.0"
xmlns:l="http://www.w3.org/1999/xlink">
  <description>
    <title-info>
      <genre>prose_classic</genre>
      <author>
        <first-name>Михаил</first-name>
        <middle-name>Афанасьевич</middle-name>
        <last-name>Булгаков</last-name>
        <id>2bd2e67f-2a82-102a-9ae1-2dfe723fe7c7</id>
      </author>
      <book-title>Собачье сердце</book-title>
      <date>1925</date>
      <coverpage>
        <image l:href="#cover.jpg"/>
      </coverpage>
      <lang>ru</lang>
    </title-info>
    <publish-info>
      <book-name>Собачье сердце</book-name>
      <publisher>Детская литература</publisher>
      <year>2007</year>
      <isbn>978-5-08-004244-7</isbn>
      <sequence number="0" name="Школьная библиотека"/>
    </publish-info>
  </description>
```

Перечислим достоинства формата FB2.

Модифицируемость. Достаточно открыть файл в простом текстовом редакторе, однако для его модификации все же нужно знать xml разметку.

Конвертируемость. Формат свободно преобразуется в html либо в pdf, имеет обратную совместимость с его приемником FB3.

Так же присутствует возможность автоматической обработки за счет использования тегов.

Недостатки формата FB2:

– отсутствует поддержка нумерованных и маркированных списков,

– нет сложной верстки текста,

– нет поддержки векторной графики,

– невозможно реализовать защиту авторских прав.

Эти недостатки не существенны, поскольку редко когда художественная литература действительно нуждается в перечисленных элементах. Эти недочеты были исправлены в следующей версии формата FB3, который уже представляет собой zip архив, хранящий текстовый и медиа контент и метаданные по аналогии с ePub. Требования к формату zip-файла и соглашения о его организации прописаны в стандарте ECMA-376 [3]. В отличие от ePub в нем нет поддержки JavaScript, потому FB3 лишен недостатков, связанных с запуском вредоносных скриптов на устройствах пользователей.

Этот формат является открытым и имеет собственный репозиторий на GitHub [4]. Так же на этом ресурсе в открытом доступе есть читатель [5] и редактор [6].

На данный момент можно говорить о большом распространении формата FB2, но только в русскоязычном сегменте Интернета. За рубежом про данный формат практически не известно. Мало популярен, даже в Рунете, и его приемник FB3.

1.3 DjVu. Книги в качестве изображений

Формат DjVu (фр. déjà vu – «уже виденное») предназначен для электронных книг и журналов. Начало разработки было положено в 1996

году AT&T Labs, Inc. – научно-исследовательским подразделением американской компании AT&T, специализирующейся на телекоммуникациях. Его разработка была мотивированна распространением отсканированных документов в сети.

Временем начала активного развития интернета можно назвать конец 90-х прошлого века. На тот момент активно поднимался вопрос цифровизации существующей на тот момент литературы. Это было реализовано технически: сканы можно было сохранить в виде картинки в формате jpeg, png. Но эти сканы дорого обходились: «…многие из важных документов имели цветные изображения и фотографии. Чтобы сохранить читаемость текста и качество картинок, приходилось делать сканы в высоком разрешении» [7], их вес мог достигать нескольких мегабайт за страницу. А в то время хранить, и тем более распространять такие тяжелые документы было затратно. Решить данную задачу распознаванием символов, и их в виде текста на тот момент было невозможно – точность была не высокая. Тогда решением могло быть разработка алгоритмов сжатия отсканированных снимков.

К 2001 году закончено создание технологии DjVu. Это позволило сжимать цветные изображения до 40-70 КБ, при оригинальном размере в 25 МБ, текстовые блоки до 15-40 КБ, а сканы древних документов до 100 КБ [8].

Документы DjVu основаны на формате файла контейнера Interchange, который представляет набор чанков – это структура определенного вида. Этот контейнер позволяет хранить данные любого типа: звук, текст, графика. На его основе формат DjVu хранит в цифровом виде информацию в виде изображений.

Сжатие происходит следующим образом: исходное изображение делится на три – передний план, задний план (фон), маска. Маска имеет то же разрешение, что и изображение, но является черно-белой и кодируется определенным образом с использованием алгоритма JB2. Он идентифицирует похожие фигуры на снимке и сохраняет только

информацию о расположении данных фигур на нем. Т. о. чем больше в снимке повторяющихся элементов, тем сильнее снимок сожмется. Передний план и фон хранятся в уменьшенном, в сравнении с оригиналом, разрешении и кодируются алгоритмом IW44, при котором изображение теряет в качестве, как и при использовании JPEG, однако, по словам разработчиков, алгоритм IW44 эффективнее его до двух раз.

Достоинства формата DjVu:

– меньший размер документов, за счет лучшего сжатия, чем у основного конкурента PDF;

– возможность производить поиск по тексту, за счет поддержки текстового слоя;

– открытый исходный код.

Однако его преимущество является и основным недостатком, которое затормозило его распространение и, впоследствии, уступило в конкуренции формату PDF. За счет использования алгоритма сжатия JB2 документы имели существенные искажения – похожие символы при низком разрешении сканирования могли интерпретироваться одинаково и заменены при сохранении. Описаны случаи таких замен символов принтерами Xerox Workcentre [9]. Правительственные организации европейских стран запретили использование алгоритма JBIG2 (JB2 является его разновидностью), в целях сохранения отсканированных документов.

В 2016 году проект Internet Archive, курирующий многочисленные проекты по оцифровке книг, заявил о прекращении использования формата DjVu. Основная причина – снижение популярности формата и сложностью поддержки просмотра на Java [10].

К малой популярности можно отнести еще некоторые недостатки:

– ограниченная возможность редактирования,

– ограниченный функциональные возможности,

– отсутствие механизма защиты авторских прав,

– отсутствие защиты от редактирования.

Так как формат хранит лишь отсканированные снимки, то его редактирование может быть осложнено и требовать дополнительных инструментов, особенно при отсутствии текстового слоя. В отличие от главного конкурента PDF, DjVu может хранить только статичную информацию: текст и изображения. Формат не реализует DRM: его нельзя защитить паролем, файл не может быть зашифрован, нет встроенной поддержки цифровой подписи, отсутствуют ограничения на печать, копирование, изменение файлов.

Несмотря на недостатки формат DjVu все еще пользуется популярностью. Однако его развитие остается под вопросом: последняя версия – 26 вышла в апреле 2005 года.

1.4 PDF и его портативность

PDF (Portable Document Format) был создан Adobe в начале 90-х годов. Этот формат предназначен для того, чтобы документы сохраняли своё оформление при обмене, могли просматриваться на разных устройствах и операционных системах, а также чтобы их можно было печатать на разнообразном оборудовании и с использованием различного программного обеспечения.

В первую очередь этот документ решал проблему, когда у пользователей не было необходимых шрифтов для отображения символов, из которых был составлен документ. Их отсутствие не влияло на просмотр документа.

С начала разработки его внедрение в электронный документооборот было медленным. Это было связано с несколькими факторами: первые версии не поддерживали гиперссылки, большой, на момент развития средств передачи документов, размер файлов, но основная причина – проприетарный режим работы с файлами. На тот момент единственным средством просмотра был Adobe Reader, разработанный той же Adobe. Стоимость лицензии

составляла 50 долларов, по этой причине только компании, и некоторые частые лица могли позволить себе работать в этой программе.

В 1994 году компания выпустила версию 2.0, которая распространялась бесплатно для любого пользователя. Это главным образом повлияло на популярность формата в Интернете. В 2008 году Adobe опубликовало публичную патентную лицензию по стандарту ISO 32000-1, предполагающей безвозмездное использование PDF. Однако некоторые его спецификации курируются только Adobe, и реализуются только в ее программных продуктах.

Файл PDF является бинарным, хранящий информацию о содержащихся в нем объектах, которыми могут быть текст, изображения, векторная графика, аудио или видео и даже 3D-объекты и их расположением внутри документа. Так же документ позволяет заменять или встраивать шрифты и распространять их вместе с документом, без необходимости установки на устройство пользователя.

Отличительной особенностью PDF от DjVu является тот факт, что PDF возлагает процесс рендеринга документа на программу - читателя, а в DjVu этот процесс выполняет программа - создатель. Это связанно с тем, что PDF кодирует графику и текст в векторизованные данные, и в таком случае программам - читателям необходимо выполнять работу над векторной графикой. В случае же с DjVu он кодирует эти данные как растровые изображения.

PDF-документы могут иметь две структуры: нелинейную (неоптимизированную) и линейную (оптимизированную). При использовании нелинейной структуры файл занимает меньше места на накопителе, но доступ к данным замедляется, так как элементы, необходимые для отображения страниц, разбросаны по всему файлу. Оптимизированные PDF-документы, иногда называемые веб-оптимизированными, предназначены для просмотра в браузерах без полной загрузки файла, так как ключевые элементы для отображения первой

страницы находятся в начале файла. Поэтому иногда меньший по размеру PDF-файл может загружаться дольше, чем более объемный файл.

Перечислим достоинства документа PDF:

– возможность просмотра на большом числе устройств,

– поддержка данных в различных формах (текст, графика, медиа файлы и т.п.),

– реализация различных средств защиты,

– поддержка JavaScript скриптов (опционально),

– стандартизация,

– конвертируемость.

Файлы PDF могут быть просмотрены на любом устройстве конечного пользователя, однако при просмотре на мобильных устройствах, актуальной остается проблема масштабируемости. Ранее было сказано, что документы данного формата могут содержать информацию разного вида. Однако не каждая программа имеет возможность ее корректно отобразить – вместо этого файл может просто не открыться, либо вместо проблемного объекта будет отображен другой, если это предусмотрел создатель. Скрипты языка JavaScript поддерживаются отдельной спецификацией PDF, которая не является свободно распространяемой и полноценную поддержку может осуществить ограниченный набор программ.

Встроенные средства защиты документа позволяют, например, защитить документ паролем или сделать доступным только для чтения. В нем так же поддерживается цифровая подпись. Другим немаловажным аспектом в цифровую эпоху является поддержка механизма защиты авторских прав. Приобретенную книгу или иной документ можно просмотреть в определенной программе, введя дополнительную информацию, подтверждающую факт покупки.

Файл преобразуется в различные другие форматы, если это позволяют его настройки. Однако не вся информация может правильно преобразоваться. Проблемы могут возникнуть с текстом. В случае, если на устройстве

пользователя отсутствует шрифт, который применялся в PDF, документ может потерять текстовую информацию. То же касается и кодировки. Для успешной конвертации необходимо знать ту кодировку, которая использовалась при создании PDF.

Существует еще одна проблема при работе с документами, связанная с текстом. Он может быть отображен как растровый объект, проще говоря – изображением. Такая ситуация возникает, когда реальный объект сканируется в PDF, без распознавания текста с помощью оптического распознавания символов, а также при намеренном преобразовании в такой вид. Это усложняет работу с данными которые необходимо извлечь из документа.

Несмотря на многочисленные преимущества, PDF также имеет ряд существенных недостатков, помимо тех, которые описаны ранее. Часть из них может негативно сказаться на пользовательском опыте и функциональности документов. Сложности возникают при редактировании PDF, особенно с работой с текстом. Некоторые веб-браузеры позволяют вносить изменения, но только дополняя информацию в файле, например выделив существующие элементы, но не удаляя их.

Еще одна не очевидная проблема связана с информационной безопасностью. Простейшим примером того, как может использоваться документ PDF является фишинг. Злоумышленникам достаточно разместить ссылку на вредоносный ресурс в документе. Но это проблема не связана напрямую с форматом – ссылки могут быть расположены в любом текстовом документе.

PDF-файлы могут быть заражены вирусами, троянами и другими вредоносными программами. В некоторых случаях даже просмотр документа может быть фатальным для жертвы хакеров. Злоумышленники пользуются уязвимостями не только формата, но и программ-читателей. В случае если в читателе включена функция запуска JavaScript скриптов, пользователь может стать жертвой хакеров. Специалист по кибербезопасности Стив Гибсон

рекомендует отключать эту функцию, не смотря на пользу, которую она могла бы дать при добросовестном использовании скриптов в документе. Так же описаны случаи, когда сразу при открытии файла вызывалось контекстное меню программы-читателя с предложением сохранить файл PDF. В системе на самом деле сохранялся не PDF а EXE файл, являющийся вирусом.

Большинство известных уязвимостей активно устраняется Adobe, выпуская обновления для софта и обновляя стандарты безопасности. Но те устройства и программы, которые не обновлены до последних версий все еще находятся в группе риска.

Несмотря на описанные недостатки PDF остается основным форматом, который используется в разных видах деятельности человека: от создания статей до электронного документооборота.

1.5 docx и его универсальность

Формат .docx был впервые представлен вместе с выпуском Microsoft Office 2007. Он стал частью стандарта Office Open XML (OOXML), который был разработан компанией Microsoft как открытый формат для хранения документов. Разработка стандарта Office Open XML была начата в 2003 году и была принята как международный стандарт ISO/IEC 29500 в 2008 году.

Технически документ этого формата – это zip архив, содержащий 2 типа файлов: xml и медиафайлы. Первый предназначается может быть представлен двумя форматами .xml и .rels. Они содержат структурированную информацию о форматировании, стилях, текстового содержимого. Медиафайлы чаще представляют собой статичные изображения. Технически документ может содержать файлы любого типа. Их можно просмотреть с помощью сторонней программы, но только изображения видны в самом документе. Это связано с тем, что изображения хранятся в архиве в запакованном виде, а другие файлы в бинарном.

Создадим пустой файл с расширением .docx используя Microsoft Office Word и распакуем его программой 7zip. Содержимое архива отображено на рисунке 1.

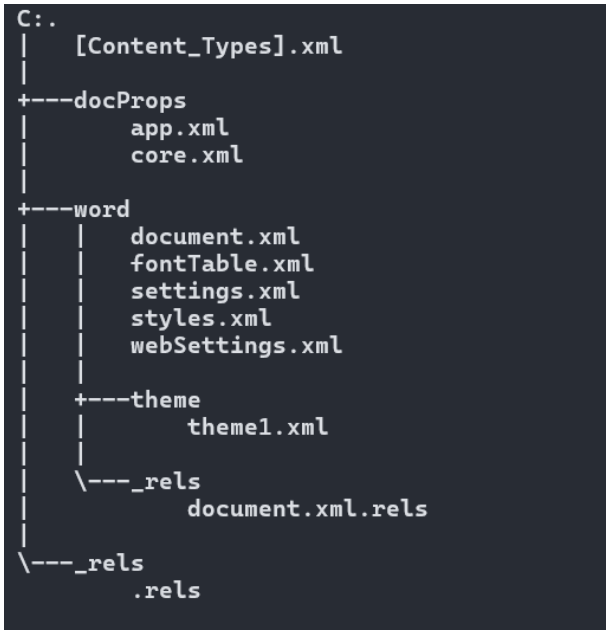

Рисунок 1 – Структура файла .docx

Разберем, как соглашение об открытой упаковке, разработанное Microsoft, описывает это содержимое.

Content_Types.xml. Этот файл используется для определения типов содержимого и связанных с ними расширений файлов внутри документа. Эта информация позволяет программам, обрабатывающим файлы .docx, правильно интерпретировать и обрабатывать содержимое документа, обеспечивая его корректное отображение и функциональность при работе с ним.

docProps/app.xml. Общая информация о документе: количество страниц, слов, символов, название приложения в котором был создан документ и т. п.

docProps/core.xml. Содержит метаданные, такие как автор документа, описание, дата создания, дата изменения и т.п.

word/document.xml. В файле хранится вся информация, которую мы видим, открыв документ в редакторе или в читателе. Он содержит не только текст, но и его форматирование.

word/fontTable.xml. В этом файле перечислены все используемые шрифты в документе.

word/settings.xml. Содержит настройки, относящиеся к текущему документу.

word/styles.xml. Оределяет стили.

word/webSettings.xml. Содержит настройки, связанные с веб-публикацией или предназначенные для отображения документа в веб-браузерах.

word/theme/theme1.xml. Сосредотачивается исключительно на определении набора цветов, шрифтов и эффектов, который является частью данной темы оформления. Его отличия от других xml файлов заключаются в том, что он специализируется на определении темы оформления, включая цвета, шрифты и эффекты, которые не всегда включены в общие стили. Он упрощает процесс управления оформлением документа, поскольку он позволяет применять сразу целый набор стилей и эффектов, в то время как стили (в styles.xml) могут быть более детализированными и применяться к отдельным элементам. Так же с его помощью к стилям можно подключить элементы, не содержащиеся в других файлах.

word/_rels/document.xml.rels. Обеспечивает связь с внешними ресурсами, например с изображениями или шрифтами.

_rels/.rels. Обеспечивает связь между частями документа.

Документ не ограничивается этими файлами, при добавлении информации его содержимое, как архива, будет меняться. Продемонстрируем это на примере: заполним созданный ранее документ некоторым содержимым. Добавим текст, формулу, изображение. Содержимое файла покажем на рисунке 2, а структуру документа на рисунке 3.

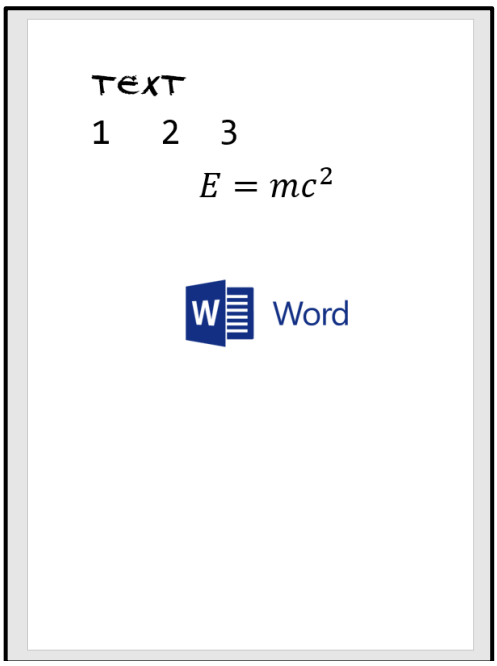

Рисунок 2 – Модифицированный файл .docx

```
C:.
|   [Content_Types].xml
|
+---docProps
|       app.xml
|       core.xml
|
+---word
|   |   document.xml
|   |   fontTable.xml
|   |   settings.xml
|   |   styles.xml
|   |   webSettings.xml
|   |
|   +---media
|   |       image1.png
|   |
|   +---theme
|   |       theme1.xml
|   |
|   \---_rels
|           document.xml.rels
|
\---_rels
        .rels
```

Рисунок 3 – Изменённая структура файла .docx

В папке word появилась папка media, и в ней файл image1.png. Рассмотрим содержимое файла document.xml. Часть документа, содержащая слово «Text», выглядит следующим образом:

```
<w:p w14:paraId="4F9C2915" w14:textId="1D79AD1B" w:rsidR="003F4152"
w:rsidRPr="00DA5F0A" w:rsidRDefault="00DA5F0A">
```

```
    <w:pPr>
        <w:rPr>
            <w:rFonts w:ascii="DJ Gross" w:hAnsi="DJ Gross"
w:cs="Calibri" />
            <w:sz w:val="72" />
            <w:szCs w:val="72" />
            <w:lang w:val="en-US" />
        </w:rPr>
    </w:pPr>
    <w:r w:rsidRPr="00DA5F0A">
        <w:rPr>
            <w:rFonts w:ascii="DJ Gross" w:hAnsi="DJ Gross"
w:cs="Calibri" />
            <w:sz w:val="72" />
            <w:szCs w:val="72" />
            <w:lang w:val="en-US" />
        </w:rPr>
        <w:t>Text</w:t>
    </w:r>
</w:p>
```

Далее идет часть, содержащая набор цифр, написанных через знак табуляции:

```
<w:p w14:paraId="5FF2B2BB" w14:textId="69D6CEF1" w:rsidR="00DA5F0A"
w:rsidRPr="003C655E"
    w:rsidRDefault="00DA5F0A" w:rsidP="003C655E">
    <w:pPr>
        <w:tabs>
            <w:tab w:val="left" w:pos="1843" />
            <w:tab w:val="left" w:pos="3402" />
        </w:tabs>
        <w:rPr>
```

```
            <w:rFonts w:ascii="Consolas" w:hAnsi="Consolas"
w:cs="Calibri" />
            <w:sz w:val="96" />
            <w:szCs w:val="96" />
            <w:lang w:val="en-US" />
        </w:rPr>
    </w:pPr>
    <w:r w:rsidRPr="003C655E">
        <w:rPr>
            <w:rFonts w:ascii="Consolas" w:hAnsi="Consolas"
w:cs="Calibri" />
            <w:sz w:val="96" />
            <w:szCs w:val="96" />
            <w:lang w:val="en-US" />
        </w:rPr>
        <w:t>1</w:t>
    </w:r>
```

Можем заметить, что сначала указываются позиции табуляции, и только потом размещаются цифры на позиции. Абзац, содержащий формулу, в xml имеет больше 50 строк. Отобразим лишь начало данного фрагмента:

```
<m:oMathPara>
  <m:oMath>
    <m:r>
      <w:rPr>
        <w:rFonts w:ascii="Cambria Math" w:hAnsi="Cambria Math"
w:cs="Courier New" />
        <w:sz w:val="96" />
        <w:szCs w:val="96" />
        <w:lang w:val="en-US" />
      </w:rPr>
      <m:t>E=m</m:t>
    </m:r>
```

Изображение представлено в документе в виде ссылки `<a:blip r:embed="rId4">`, которая в файле `word/_rels/document.xml.rels` указывает на расположение изображения в архиве:

```
…
<Relationship Id="rId4"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationsh
ips/image" Target="media/image1.png"/></Relationships>
```

Документы в формате .docx поддерживают макросы – это набор инструкций или команд, написанных на языке программирования VBA. Эти команды выполняются программами, в которых открыт документ. Макросы позволяют автоматизировать некоторые действия, упростить форматирование документа (например, автоматическое присваивание номеров рисункам, таблицам), обработать данные из таблиц. Полный функционал реализуются программами-редакторами.

Однако функция поддержки макросов нуждается в особом внимании. Она может быть полезна для автоматизации задач и улучшения производительности, но также представляет потенциальную угрозу безопасности для устройств. Макросы могут содержать вредоносный код, который может быть запущен при открытии документа, и в результате причинить вред компьютеру или сети. Это может привести к утечке конфиденциальной информации, повреждению файлов или даже заражению вирусами. По умолчанию редактор Microsoft Office Word открывает документы с отключенной функцией запуска макросов и предупреждает пользователя о потенциальном вреде для устройства.

Файлы docx по-разному интерпретируются программами, в которых были открыты. Вследствие этого у пользователей могут возникать проблемы с отображением контента. Например, один и тот же текст может занимать разный объём на странице. Или при открытии в программе, отличной от той,

в который был сделан документ, могут пропасть символы-разделители. Но эти проблемы связаны не с самой структурой файла, а с их интерпретаторами.

Просмотр документов формата docx не вызывает затруднений на мобильных устройствах – в отличии от PDF, при просмотре присутствует эффект масштабируемости. Поэтому у пользователей обычно нет необходимости скролить страницу влево-вправо при чтении.

Файлы docx можно защитить различными способами. На них можно установить пароль, защитить от редактирования и установить цифровую подпись.

2 Описание модели

Для создания документов нам необходимо придерживаться правила разделения данных от представления. Контент будет содержаться в XML файле. Правила его отображения – в другом файле, либо в специальной секции того же файла.

Документ может содержать различные типы данных: текст, рисунки, таблицы, формулы. Необходимо описать, как эти данные должны быть отображены в «сыром» виде на языке разметки XML. Далее потребуется задокументировать правила, которые могут быть наложены на контент, например шрифт или выравнивание. Данный раздел по своей сути является документацией к оформлению содержимого и его представления.

2.1 Правила написания текста

Любой текст можно представить как набор заголовков абзацев, и других структур специального вида. К таким структурам относятся:

– списки или перечисления,

– текст на языке программирования,

– формулы.

К каждому из этих видов применяются различные стили оформления, поэтому целесообразно размещать эти структуры в соответствующих им тегах. Правила их написания на языке разметки XML будут отражены в соответствующих пунктах.

2.1.1 Простой текст. Абзацы

Абзацем назовем такой текст, который состоит только из предложений, не содержащий остальных, перечисленных ранее структур. Всё содержимое абзаца размещается внутри тега <p>.

Однако внутри абзацев могут содержаться слова, различающиеся написанием от других слов. Например, слова могут быть выделены

курсивом, иметь полужирное начертание и т.д. Для этого их нужно заключить в соответствующие теги:

– <b> для выделения полужирным,
– <i> для написания курсивом,
– <u> для подчёркивания,
– <s> для зачёркивания,
– <sup> для надстрочного написания,
– <sub> надстрочного написания.

2.1.2 Заголовки

Заголовки необходимо включать в тег <h>. Они могут различаются уровнем, поэтому тег <h> содержит необязательный атрибут lvl, принимающий целые значения, начиная с 1.

Пример, содержащий заголовки разных уровней:

```
<h lvl="1">
    Заголовок уровня 1
</h>
<h lvl="2">
    заголовок уровня 2
</h>
```

2.1.3 Перечисления

Разделяют два типа перечислений: нумерованные (упорядоченные) и ненумерованные. В некоторых случаях используется комбинация этих типов, когда элементом одного списка является другой список.

Нумерованный список следует размещать в теге <ol> (ordered list), ненумерованный размещать внутри тега <ul> (unordered list). Тег <ol> содержит атрибут type для обозначения вида перечисления: значение abc для буквенного или значение num для числового; а также атрибут sep отвечающий за разделитель между цифрой или буквой и элементом списка.

Разделителем может быть точка, скобка или он может отсутствовать вовсе. Сами же объекты списка следует размещать внутри тега <li> (list item).

Пример нумерованного списка на языке разметки XML:

```
<ol type="num" sep="">
    <li>[первый объект списка]</li>
    <li>[второй объект списка]</li>
[…]
</ol>
```

Пример ненумерованного списка на языке разметки XML:

```
<ul>
    <li>яблоки</li>
    <li>бананы</li>
    <li>груши</li>
</ul>
```

2.1.4 Код на языке программирования

Текст, написанный на каком-либо языке программирования или на языке текстовой разметки, можно разместить в документах в единообразном с обычным текстом виде. Но в этом случае теряется одна отличительная черта такого рода текста – его читаемость. Она предназначена для человека, читающего код, а не для компьютеров, которые этот код исполняют. Стиль оформления кода отличается от обычного текста следующими правилами:

– используются моноширинные шрифты, например Consolas или Courier New;

– текст выровнен по левому краю, но рекомендуется соблюдение табуляции для различных структур кода;

– междустрочный интервал обычно равен одной строке;

– если код содержит много символов, разумно использовать меньший размер шрифта.

Поэтому рациональным решением является размещение кода в отдельный тег – <code>. Этот тег так же целесообразно наделить атрибутом lang, который будет нести дополнительную информацию о языке, на котором написан текст, будь то язык программирования или язык разметки. Она может пригодиться для подсветки синтаксиса языка в различных редакторах.

Пример оформленного в тег кода:

```
<code lang="js">
    console.log(“Hello world”)
</code>
```

2.1.5 Формулы и различные математические объекты

Формулы и объекты из сферы математики так же характеризуются отличительным написанием от остального текста. Такого рода объекты размещаются внутри тега <math> в формате LaTeX. Поэтому для их размещения необходимо предварительно перевести формулу с помощью соответствующих программ или веб-сервисов, например latexeditor.lagrida.com.

Рассмотрим формулу нормального распределения:

$$\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \quad (1)$$

Формула (1) будет записана в теге <math> следующим образом:

```
<math>
  \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)
</math>
```

2.2 Вставка изображений

Так как XML файл — это текстовый файл и в него нельзя вставить изображения, то для их отображения в конечном документе необходимо использовать ссылки. Эти ссылки обозначают расположение изображений на устройстве.

Для вставки ссылки изображения используется конструкция вида:

```
<img src="" [desc=""] />
```

Атрибут src содержит ссылку на изображение, а атрибут desc, не являющийся обязательным, – описание изображения.

2.3 Правила оформления таблиц

Таблицы, написанные на языке разметки XML, отличаются тем, что данные организованы в виде дерева, а не в виде классических строк и столбцов. В XML каждая запись таблицы, которая в табличной форме соответствует строке, является отдельным элементом, содержащим вложенные элементы для каждой ячейки данных. В отличие от традиционных таблиц, где столбцы имеют фиксированное положение и идентификацию, в XML данные структурированы иерархически, что позволяет гибко добавлять и изменять элементы.

В общем виде оформить таблицу с использованием языка разметки XML выглядит так:

```
<table [desc=""]>
[  <th>
    <td>[столбик1]<td>
    <td>[столбик2]<td>
  </th>]
```

```
  <tr>
    <td>[данные]<td>
    <td>[данные]<td>
  </tr>
…
</table>
```

Тег <table> содержит необязательный атрибут desc, служащий для описания таблицы. Внутри тега может сначала располагаться тег <th> (table header) необходимый для отображения заголовков таблицы. Но он может отсутствовать, в таком случае таблица целиком будет состоять из наборов тегов <tr> (table row). Внутри тегов <th> или <tr> располагаются теги <td>, внутри которых располагается информация.

Рассмотрим таблицу 1, содержащую некоторую информацию о студентах.

Таблица 1 – Информация о студентах

Студент	Возраст	Группа	Оценка
Иванов И.	20	М01	4
Петров П.	19	М02	5

Покажем, как таблица 1 должна быть описана на языке разметки XML:

```
<table desc=" Информация о студентах">
  <th>
    <td>Студент</td>
    <td>Возраст</td>
    <td>Группа</td>
    <td>Оценка</td>
  </th>
  <tr>
    <td>Иванов И.</td>
```

```
    <td>20</td>
    <td>M01</td>
    <td>4</td>
  </tr>
  <tr>
    <td>Петров П.</td>
    <td>19</td>
    <td>M02</td>
    <td>5</td>
  </tr>
</table>
```

2.4 Формализация требований

Формализации подлежат правила, относящиеся к внешнему виду документа, но в меньшей степени к его структурному содержанию. Это объясняется тем фактом, что внутреннее содержание документа, определяет его автор, а правила внешнего вида диктуются другими субъектами.

Из описанной в разделах 2.1, 2.2, 2.3 модели требуется сформировать правила отображения структурных элементов. Данные правила будут описаны в виде xml дерева. Каждый элемент характеризуется своими свойствами. Например, текст имеет кегль (параметр), который может быть задан в разных единицах измерения: pt, px, em, cm и т.п. В общем случае правила описаны следующим образом:

```
<тег (элемент)>
    <свойство1 параметр1="" [параметр1="" ]* />
…
</тег>
```

XML файл, с формальным описанием правил содержимого документа, соответствующий нашей модели задокументирован в приложении А.

3 Алгоритм преобразования XML документа

Для создания документа требуемого вида необходимо сначала проанализировать 2 файла: содержимого и представления. Прочитав программой XML файл содержимого, необходимо далее соотнести настройки его отображения, хранящиеся в файле представления. Сформированные таким образом данные позволят создавать документы любого удобного формата соответствующими инструментами.

Покажем на примере использования средств .Net Framework и Windows Forms программу, реализующую данный алгоритм.

Требуется сначала указать пути расположения XML документов. После программа проанализирует и сопоставит их содержимое. Первый файл имеет следующее содержимое:

```
<?xml version="1.0" encoding="UTF-8"?>

<document>
  <h lvl="1">ВВЕДЕНИЕ</h>
  <p>Информация – неотъемлемая часть нашей <b>жизни</b>...</p>
  <ol type="num" sep=" ">
    <li>[первый объект списка]</li>
    <li>[второй объект списка]</li>
  </ol>
  <ul>
    <li>яблоки</li>
    <li>бананы</li>
    <li>груши</li>
  </ul>
  <code lang="js">
    console.log(“Hello world”)
  </code>
  <math>
```

```
    \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(x-
\mu)^2}{2\sigma^2}\right)
  </math>
  <img src="cat.jpg" desc="котик"/>
  <table desc="">
    <th>
      <td>студент</td>
      <td>номер</td>
    </th>
    <tr>
      <td>иванов</td>
      <td>4</td>
   </tr>
  </table>
</document>
```

Файл настроек является модификацией файла из приложения А, добавлением в него параметров и их значений.

Внешний вид программы показан на рисунке 4.

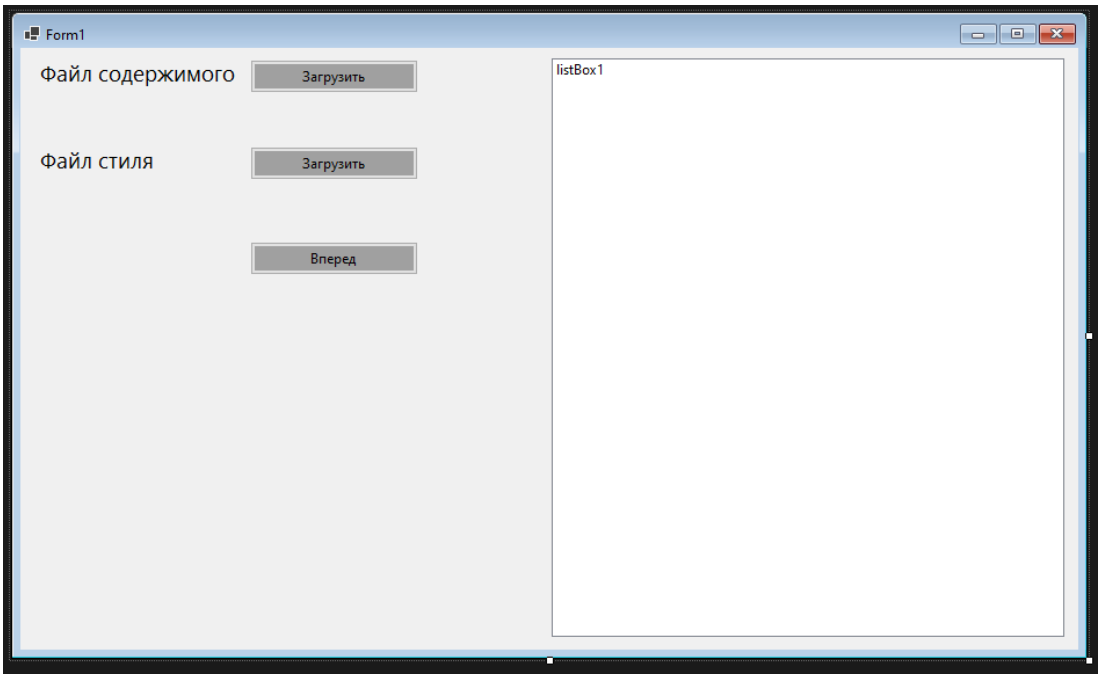

Рисунок 4 – Стартовое окно программы

После указания необходимых путей файлов результат работы программы будет отображен в правой части её окна. Продемонстрируем этот результат на рисунке 5.

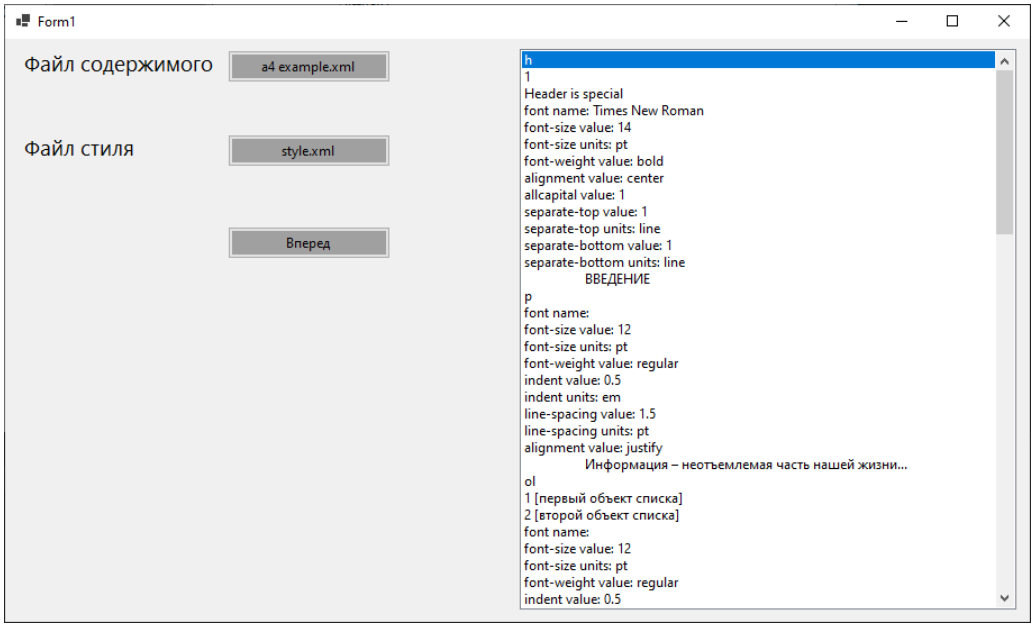

Рисунок 5 – Результат работы программы

Можем заметить порядок обработки файла: сначала указывается тег, который соответствует типу объекта, далее значение атрибута (при наличии), далее перечисляются параметры и их значения, в конце указывается содержимое. Этот процесс повторяется для каждого тега. Используя полученный результат, мы можем создать документы, используя, например, средства C#, или созданием HTML файла с настройкой стиля.

Исходный код программы размещен в приложении Б.

ЗАКЛЮЧЕНИЕ

В рамках проделанной работы были исследованы популярные форматы хранения электронных документов, выделены их преимущества и недостатки. Была описана модель раздельного хранения данных и их представления с использованием языка разметки XML. Разработан алгоритм, обработки файлов, реализующие данную модель. Демонстрирует работу алгоритма созданная программа, являющаяся XML-парсером.

Используя различные фреймворки для создания документов, можно расширить функционал данной программы. Это позволит изменить внешний вид документа нажатием одной кнопки, если будут описаны требуемые правила его отображения.

Так же модель хранения данных позволит с легкостью внедрить их в HTML разметку Интернет страниц.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 W3.org : цифровой ресурс : сайт. – URL: https://www.w3.org/publishing/epub3/ (дата обращения 23.04.2024) – Режим доступа: сеть Интернет. – Текст : электронный.

2 portswigger.net : цифровой ресурс : сайт. – URL: https://portswigger.net/daily-swig/epub-vulnerabilities-electronic-reading-systems-riddled-with-browser-like-flaws (дата обращения 23.04.2024) – Режим доступа: сеть Интернет. – Текст : электронный.

3 ecma-international.org: цифровой ресурс : сайт. – URL: https://ecma-international.org/publications-and-standards/standards/ecma-376/ (дата обращения 23.04.2024) – Режим доступа: сеть Интернет. – Текст : электронный.

4 github.com : цифровой ресурс : сайт. – URL: https://github.com/gribuser/FB3 (дата обращения 23.04.2024) – Режим доступа: сеть Интернет. – Текст : электронный.

5 github.com : цифровой ресурс : сайт. – URL: https://github.com/Litres/FB3Reader (дата обращения 23.04.2024) – Режим доступа: сеть Интернет. – Текст : электронный.

6 github.com : цифровой ресурс : сайт. – URL: https://github.com/Litres/FB3Editor (дата обращения 23.04.2024) – Режим доступа: сеть Интернет. – Текст : электронный.

7 Habr.com : цифровой ресурс : сайт. – URL: https://habr.com/ru/companies/maccentre/articles/411545/ (дата обращения 23.04.2024) – Режим доступа: сеть Интернет. – Текст : электронный.

8 Ботту Л. Высококачественное сжатие изображений документов с помощью DjVu / Л. Ботту, П. Хаффнер, П. Ховард, Патрис Симар, Й Бенджио, Я. Ле Кун. – Текст : непосредственный – Линкрофт, Нью Джерси, 1988. – С. 7-24

9 Xerox scanners/photocopiers randomly alter numbers in scanned documents: цифровой ресурс: сайт – URL: https://www.dkriesel.com/en/blog/2013/0802_xerox-workcentres_are_switching_written_numbers_when_scanning (дата обращения 23.04.2024) – Режим доступа: сеть Интернет. – Текст : электронный.

10 Internet Archive Forums: djvu files for new uploads: цифровой ресурс: сайт – URL: https://archive.org/post/1053214/djvu-files-for-new-uploads (дата обращения 23.04.2024) – Режим доступа: сеть Интернет. – Текст : электронный.

ПРИЛОЖЕНИЕ А

(обязательное)

Шаблон настроек представления содержимого, записанный на языке разметки XML

```
<?xml version="1.0" encoding="UTF-8"?>

<style>

  <p>
    <!-- Используемый шрифт-->
    <font name="" />
    <!-- Размер шрифта -->
    <font-size value="12" units="pt" />
    <!-- Ширина шрифта-->
    <font-weight value="1" units="pt" />
    <!-- Абзацный отступ -->
    <indent value="0.5" units="em" />
    <!-- Расстояние между строками -->
    <line-spacing value="1.5" units="pt" />
    <!-- Выравнивание текста на странице-->
    <alignment value="justify" />
  </p>

  <h>
    <font name="" />
    <font-size value="" units="" />
    <indent value="" units="" />
    <line-spacing value="" units="" />
    <alignment value="" />
    <!-- Отделение пустой строкой сверху-->
    <separate-top value="1" units="line" />
    <!-- Отделение пустой строкой снизу-->
    <separate-bottom value="1" units="line" />
    <!-- Специальные настройки -->
    <special>
      <!-- Список отдельных разделов-->
      <names></names>
      <font name="" />
      <font-size value="" units="" />
      <font-weight value="" units="" />
      <alignment value="" />
    </special>
  </h>

  <ol>
    <indent value="" units="" />
    <alignment value="" />
```

```
</ol>
<ul>
  <indent value="" units="" />
  <alignment value="" />
</ul>

<li>
  <font name="" />
  <font-size value="" units="" />
</li>

<code>
  <font name="" />
  <font-size value="" units="" />
  <indent value="" units="" />
  <line-spacing value="" units="" />
  <alignment value="" />
  <separate-top value="" units="" />
  <separate-bottom value="" units="" />
</code>

<math>
  <font name="" />
  <font-size value="" units="" />
  <indent value="" units="" />
  <line-spacing value="" units="" />
  <alignment value="" />
  <separate-top value="" units="" />
  <separate-bottom value="" units="" />
</math>

<img>
  <alignment value="" />
  <width value="" units="" />
  <height value="" units="" />
  <separate-top value="" units="" />
  <separate-bottom value="" units="" />
</img>

<table>
  <indent value="" units="" />
  <separate-top value="" units="" />
  <separate-bottom value="" units="" />
  <alignment value="" />
</table>

<th>
  <font name="" />
  <font-size value="" units="" />
  <alignment value="" />
```

```
  </th>
  <td>
    <font name="" />
    <font-size value="" units="" />
    <alignment value="" />
  </td>
</style>
```

ПРИЛОЖЕНИЕ Б

(обязательное)

Исходный код XML парсера

```
namespace XMLParce;

using System.Xml;

public partial class Form1 : Form
{
    string rowTextPath = string.Empty;
    string styleFilePath = string.Empty;

    public Form1()
    {
        InitializeComponent();
        listBox1.IntegralHeight = true;
    }

    private void button1_Click(object sender, EventArgs e)
    {
        OpenFileDialog openFileDialog = new()
        {
            Filter = "Файлы XML (*.xml)|*.xml"
        };

        if (openFileDialog.ShowDialog() == DialogResult.OK)
        {
            rowTextPath = openFileDialog.FileName;
            button1.Text = Path.GetFileName(rowTextPath);

        }
    }

    private void ProcessXmlFile()
    {
        XmlDocument xmlDocument = new();
        xmlDocument.Load(rowTextPath);

        XmlDocument style = new();
        style.Load(styleFilePath);

        using XmlNodeList? nodes =
xmlDocument.SelectNodes("/document/*");
        if (nodes is null)
        {
            return;
        }
        foreach (XmlNode node in nodes)
```

```
        {
            if (node is null) { continue; }
            string nodeName = node.Name;
            string value = string.Empty;
            listBox1.Items.Add(nodeName);

            switch (nodeName)
            {
                case "h":
                    if (node.Attributes is null)
                    {
                        break;
                    }
                    string level = node.Attributes["lvl"]?.Value ??
"1";
                    listBox1.Items.Add(level);
                    value = node.InnerText.Trim();

                    string[]? specialHeaders =
style.SelectSingleNode("/style/h/special/names")?.InnerText?.ToLower()
.Split(',');

                    XmlNodeList? styleNodes = null;
                    if (specialHeaders is not null &&
specialHeaders.Contains(value.ToLower()))
                    {
                        listBox1.Items.Add("Header is special");
                        styleNodes =
style.SelectNodes("/style/h/special/*");
                    }
                    else
                    {
                        styleNodes = style.SelectNodes("/style/h/*");
                    }

                    ShowTagNodesWithAttributes("h", styleNodes);
                    break;

                case "p":
                    value = node.InnerText.Trim();

                    styleNodes = style.SelectNodes("/style/p/*");

                    ShowTagNodesWithAttributes("p", styleNodes);
                    break;

                case "ol":
                    if (node.Attributes is null)
                    {
                        break;
```

```
                    }
                    string type = node.Attributes["type"]?.Value ??
string.Empty;
                    string sep = node.Attributes["sep"]?.Value ??
string.Empty;
                    string alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
                    int index = 1;
                    XmlNodeList liNodes = node.SelectNodes("li");
                    foreach (XmlNode liNode in liNodes)
                    {
                        string begin = string.Empty;
                        if (type == "num")
                        {
                            begin = index.ToString() + sep;
                        }
                        if (type == "abc")
                        {
                            begin = alphabet.ToLower()[index - 1] +
sep;
                        }

                        string listItem = liNode.InnerText;
                        listBox1.Items.Add(begin + listItem);
                        index++;
                    }

                    styleNodes = style.SelectNodes("/style/p/*");
                    ShowTagNodesWithAttributes("p", styleNodes);
                    styleNodes = style.SelectNodes("/style/ol/*");
                    break;

                case "ul":
                    liNodes = node.SelectNodes("li");
                    foreach (XmlNode liNode in liNodes)
                    {
                        string listItem = liNode.InnerText;
                        listBox1.Items.Add(listItem);
                    }

                    styleNodes = style.SelectNodes("/style/p/*");
                    ShowTagNodesWithAttributes("p", styleNodes);
                    styleNodes = style.SelectNodes("/style/ul/*");
                    ShowTagNodesWithAttributes("ul", styleNodes);
                    break;

                case "code":
                    if (node.Attributes is null)
                    {
                        break;
                    }
```

```
                    string lang = node.Attributes["lang"]?.Value ??
string.Empty;
                    listBox1.Items.Add(lang);
                    value = node.InnerText.Trim();

                    styleNodes = style.SelectNodes("/style/code/*");
                    ShowTagNodesWithAttributes("code", styleNodes);
                    break;

                case "math":
                    value = node.InnerText.Trim();

                    styleNodes = style.SelectNodes("/style/math/*");
                    ShowTagNodesWithAttributes("math", styleNodes);
                    break;

                case "img":
                    if (node.Attributes is null) { break; }
                    string src = node.Attributes["src"]?.Value ??
string.Empty;
                    string description =
node.Attributes["desc"]?.Value ?? string.Empty;
                    listBox1.Items.Add(description);
                    listBox1.Items.Add(src);

                    styleNodes = style.SelectNodes("/style/img/*");
                    ShowTagNodesWithAttributes("img", styleNodes);
                    break;

                case "table":
                    if (node is null || node.Attributes is null)
                    {
                        break;
                    }
                    description = node.Attributes["desc"]?.Value ??
string.Empty;
                    listBox1.Items.Add(description);

                    XmlNodeList thNodes = node.SelectNodes("th");
                    XmlNodeList trNodes = node.SelectNodes("tr");
                    foreach (XmlNode thNode in thNodes)
                    {
                        XmlNodeList? tdNodes =
thNode.SelectNodes("td");

                        if (tdNodes != null)
                        {
                            string[] thValues =
tdNodes.Cast<XmlNode>()
                                .Select(tdNode => tdNode.InnerText)
```

```
                                .ToArray();
                            listBox1.Items.AddRange(thValues);
                        }
                    }
                    foreach (XmlNode trNode in trNodes)
                    {
                        string[] rowValues =
trNode.SelectNodes("td").Cast<XmlNode>().Select(tdNode =>
tdNode.InnerText).ToArray();
                        listBox1.Items.AddRange(rowValues);
                    }
                    break;

                default:
                    break;
            }

            listBox1.Items.Add("\t" + value);
        }
    }

    private void button2_Click(object sender, EventArgs e)
    {
        OpenFileDialog openFileDialog = new()
        {
            Filter = "Файлы XML (*.xml)|*.xml"
        };

        if (openFileDialog.ShowDialog() == DialogResult.OK)
        {
            styleFilePath = openFileDialog.FileName;
            button2.Text = Path.GetFileName(styleFilePath);
        }
    }

    private void button3_Click(object sender, EventArgs e)
    {
        if (rowTextPath == string.Empty && styleFilePath ==
string.Empty)
        {
            label3.Text = "файлы не выбраны"; return;
        }

        ProcessXmlFile();
    }

    private void ShowTagNodesWithAttributes(string parentTag,
XmlNodeList? styleNodes)
    {
        if (styleNodes is null)
```

```
        {
            listBox1.Items.Add($"style is not defined for tag
{parentTag}");
            return;
        }

        foreach (XmlNode tag in styleNodes)
        {
            string tagName = tag.Name;
            if (tag.Attributes is not null)
            {
                foreach (XmlAttribute xmlAttribute in tag.Attributes)
                {
                    listBox1.Items.Add(tagName + " " +
xmlAttribute.Name + ": " + xmlAttribute.Value);
                }
            }
        }
    }
}
```