Программные системы и вычислительные методы
Правильная ссылка на статью:
Нуриев М.Г., Белашова Е.С., Барабаш К.А. — Конвертер Markdown-файлов в LaTeX-документ // Программные системы и вычислительные методы. – 2023. – № 1. DOI: 10.7256/2454-0714.2023.1.39547 EDN: SNAYLQ URL: https://nbpublish.com/library_read_article.php?id=39547

Конвертер Markdown-файлов в LaTeX-документ

Нуриев Марат Гумерович

кандидат технических наук

старший преподаватель, кафедра автоматизированных систем обработки информации и управления, Казанский национальный исследовательский технический университет им. А.Н. Туполева-КАИ

420015, Россия, республика Татарстан, г. Казань, ул. Большая Красная, 55

marat_nu1@mail.ru

Белашова Елена Семеновна

кандидат физико-математических наук

доцент, кафедра компьютерных систем, Казанский национальный исследовательский технический университет им. А.Н.Туполева-КАИ

420015, Россия, республика Татарстан, г. Казань, ул. Большая Красная, 55

bel_lena@mail.ru

Барабаш Константин Алексеевич

студент кафедры компьютерных систем Казанского национального исследовательского технического университета им. А.Н.Туполева-КАИ

420015, Россия, республика Татарстан, г. Казань, ул. Большая Красная, 55

kostyandriy@mail.ru

Статья из рубрики "Языки программирования"

DOI:

10.7256/2454-0714.2023.1.39547

EDN:

SNAYLQ

Дата направления статьи в редакцию:

25-12-2022

Дата публикации:

01-01-2023

Аннотация: Привычные пользователю текстовые редакторы такие как Microsoft Word, Notepad++ и другие являются «громоздкими». При своем огромном функционале они не

исключают риска неправильной конвертации документа, например, при открытии тех же Word-файлов на более старых или наоборот более новых версиях Microsoft Word. Выходом является применение языков разметок, которые позволяют маркировать блоки текста в целях их представления в нужной стилистике. В настоящее время большую популярность имеют LaTeX (набор макрорасширений системы компьютерной вёрстки TeX) и Markdown (облегчённый язык разметки, созданный с целью обозначения форматирования в простом тексте). Поэтому актуален вопрос преобразования Markdown-документа в LaTeX-документ. Существуют различные инструменты конвертации Markdown-файлов в LaTeX-документ, например, библиотека Pandoc, Markdown.lua, Lunamark и другие. Но большинство из них имеют избыточные шаги по формированию выходного документа. В данной статье освещается метод решения посредством интеграции Markdown-файла в LaTeX-документ, который потенциально позволит сократить время формирования выходного документа в отличие от существующих решений. Разработанный конвертер Markdown-файлов в LaTeX-документ позволит автоматически получить результирующий документ и снизить вероятность ошибок при ручном преобразовании текста из Markdown-формата в LaTeX-формат.

Ключевые слова:

Markdown, LaTeX, программирование, конвертер, Python, язык разметки, Overleaf, преобразование текста, Word, регулярные выражения

Введение

Привычные пользователю текстовые редакторы такие как Microsoft Word, Notepad++ и другие являются «громоздкими». При своем огромном функционале они не исключают риска неправильной конвертации документа, например, при открытии тех же Word-файлов на более старых или наоборот более новых версиях Microsoft Word. Текстовый редактор от компании Microsoft имеет два расширения: doc и docx. Хоть первый из них уже не актуален, однако старые версии Microsoft Word до сих пор используют его. Doc-файл является форматом документа используемым текстовым редактором Microsoft Word, а .docx это следующая версия, она более эффективная, создает файлы, в меньшей степени подверженные повреждению. Для большей надежности сохранения форматов и редакции в документе можно воспользоваться языком разметок [1]. Язык Markdown – один из представителей языков разметок. Он обладает следующими преимуществами:

- Универсальность: Документы с синтаксисом Markdown это простые текстовые файлы, которые можно открыть в любом текстовом редакторе.
- Простота: Язык Markdown очень прост для освоения, для этого не требуется никакие дополнительные знания.
- Большой выбор инструментов: Благодаря высокой универсальности с языком разметки Markdown можно работать в любом редакторе, выбор пользователя почти ничем не ограничивается.
- Конвертируемость: Документы Markdown легко экспортировать в любые форматы: PDF, DOC, ODT. При этом их форматирование остаётся неизменным [2].

Благодаря использованию языков разметки, в частности, Markdown можно достичь высокой сохранности редакции файла чем обусловлена актуальность разработки ПО для конвертации Markdown файлов в LaTeX документы. LaTeX документ – текстовый файл содержащий команды языка разметки.

Язык упрощённой разметки Markdown

Целью языка Markdown является легкая запись и чтение. Внешне Markdown напоминает язык HTML, однако не является им и не может его заменить так как имеет очень мало типов синтаксиса, идея Markdown состоит в том, чтобы упростить чтение, запись и изменение документов. В отличие от HTML, в связи с огромным количеством разнообразных тегов, является сложным для чтения и понимания того, как будет выглядеть результат, Markdown не обладает этой проблемой, потому что является настолько легким для чтения на сколько это возможно. Для использования Markdown достаточно просто применить простые теги к тексту. Язык разметки Markdown обладает блочными (рис. 1) и строчными элементами (рис. 2). Как видно из примера Markdown, в отличие от HTML, не требует больших каскадных выделений для создания отформатированного абзаца.

````
Here's an example:

```js
function test() {
  console.log("notice the blank line before this function?");
}
```

syntax highlighting:
```ruby
require 'redcarpet'
markdown = Redcarpet.new("Hello World!")
puts markdown.to_html
```
````

Рисунок 1. Пример блочного кода в Markdown

```
# This is an H1

## This is an H2

###### This is an H6
```

Рисунок 2. Пример строчного кода в Markdown

Структура документов LaTeX

LaTeX – набор макрорасширений системы компьютерной вёрстки TeX, который облегчает набор сложных документов. Основная идея LaTeX заключается в том, что авторам нужно думать только о содержании, не беспокоясь о конечном визуальном облике. Разрабатывая свой документ автор указывает логическую структуру текста (разбивая его на главы, разделы, таблицы, изображения), а LaTeX решает вопросы его отображения. Так содержание отделяется от оформления. Оформление при этом или определяется заранее (стандартное), или разрабатывается для конкретного документа.

Документ LaTeX гораздо менее читаемый чем документ Markdown и обладает жесткой структурой, которая сравнима с программным кодом. Этот документ содержит специальные команды языка разметки и делится на преамбулу и тело. Преамбула содержит информацию: о классе документа, об авторе, о дате создания и прочее.

Тело документа содержит текст документа и команды разметки, оно ограничивается командами begin{document} и end{document}.

Веб-редактор Overleaf

Веб-редактор Overleaf – редактор LaTeX файлов, однако помимо формата .tex, являющегося стандартным форматом LaTeX документом, может читать формат .sty, в котором можно описать стиль текста, что позволяет не тратить время на редакцию текста, отступы и прочее – достаточно описать текст, заключив его в блоки LaTeX документа. Помимо вышеуказанного преимущества имеет удобный интерфейс (рис. 3).

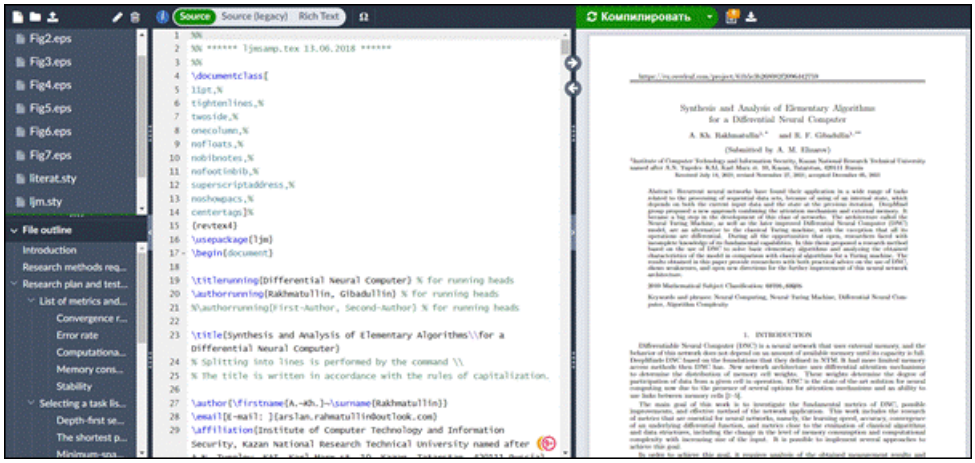

Рисунок 3. Интерфейс Overleaf

Также позволяет редактировать один и тот же файл нескольким пользователям, поддерживает практически все функции LaTeX и позволяет скомпилировать документа в формат .pdf. Overleaf написан на языке CoffeScript, использует Node.js и такие СУБД как MangoDB и Redis [3].

Предпосылки к разработке конвертера

Разработка конвертера позволит автоматизировать процесс преобразования Markdown-документов в LaTeX-документы, что устранит или сведет к минимуму процедуру ручного преобразования и позволит избежать ошибок, вызванных человеческим фактором.

Разрабатываемый конвертер Markdown-файлов в LaTeX-документ будет предназначен для:

- Подготовки выбранного документа к конвертации.
- Снабжения результирующего файла необходимыми библиотеками, требуемыми для корректного отображения всех функций исходного документа.
- Конвертации документа .md в формат .tex.
- Создания результирующих файлов, а именно отфильтрованного .md и финального .tex файла.

При этом конвертер должен выполнять обработку выбранного файла за короткий промежуток времени (не более 30 секунд на конвертацию документа размера 500 КБ), иметь интерактивный пользовательский интерфейс.

Структурная схема конвертера Markdown-файлов в LaTeX-документ

На основании поставленной задачи разработана структурная схема конвертера Markdown-файлов в LaTeX-документ (рис. 4).

На этапе ввода файла пользователь выбирает исходный файл, который прежде чем стать частью LaTeX-документа должен пройти предварительную обработку. Основным этапом создания конвертера Markdown-файлов в LaTeX-документ является этап обработки файлов, в котором исходный файл должен быть подготовлен к включению в шаблон LaTeX. Он представляет собой стандартный документ LaTeX, в котором включены все

необходимые компоненты. На этом этапе документ будет обрабатываться программой, написанной на Python, после чего файл будет подготовлен к выводу.

Вывод будет осуществлен в виде двух файлов. Первый файл – это Markdown-документ, в котором будет храниться текст. Вторым файлом является шаблон LaTeX, в котором хранятся данные о разметки текста.

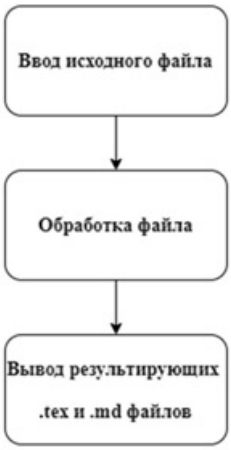

Рисунок 4. Структурная схема обработки документа

Алгоритм работы конвертера Markdown-файлов в LaTeX-документ

Схема алгоритма работы конвертера Markdown-файла в LaTeX-документ показан на рисунке 5.

Рисунок 5. Алгоритм работы конвертера Markdown-файла в LaTeX-документ

Пользователь выбирает Markdown-файл и одновременно с этим происходит создание LaTeX-документа, в котором хранится информация о стиле и библиотеках, используемых для корректного отображения файла в формате .tex. Tex-компоненты позволяют использовать русский язык, обозначают кодировку символов, включают корректное

отображение программного кода, а также делают его цветным.

С помощью следующих ключевых слов:

- usepackage{packages/sleek-title}
- usepackage{packages/sleek-theorems}
- usepackage{packages/sleek-listings}

задается стиль текста.

На этапе очистки из исходного файла удаляются объекты, автоматическое добавление которых в LaTeX-документ затруднено или не является возможным, это является узким местом выбранного метода решения задачи. На рисунке 6 показано как выглядит титул в Markdown-файле, этот текст не читается в LaTeX-документе, конвертер выдаст ошибку при обработке. На рисунке 7 показано как выглядит титул в LaTeX-документе.

```
---
layout: post
title: "Параллельная программа проверки орфографии"
date: 2021-05-01
---
```

Рисунок 6. Пример титульного листа в markdown файле

```
\title{Тест}
\author{тест}
\date{22.11.22}
\begin{document}
\maketitle
\markdownInput{result.md}
\end{document}
```

Рисунок 7. Пример титульного листа в LaTeX-документе

Внутри фигурных скобок ключевых слов title {Заголовок}, author{автор} и date{дата} вводятся соответствующий текст заголовка, ФИО автора и дата публикации, а ключевое слово maketitle в теле LaTeX-документа создает титульный лист. Однако в заголовке Markdown-файла лежит информация, которая считывается с помощью программы и записывается в начало файла .md. Далее исходный Markdown-файл также требуется очистить от списка источников, так как в LaTeX-документе ссылки представляются иной языковой конструкцией. Далее Markdown-файл добавляется в шаблон LaTeX. С помощью ключевого слово markdownInput{пример.md} файл markdown присоединяется к LaTeX-документу. Данное ключевое слово должно следовать после команды maketitle, в противном случае заголовок будет создан после основного текста [4].

Допустим и другой способ, когда содержимое Markdown-файла находится в теле LaTeX-документа вместо ключевого слова markdownInput{пример.md}.

Программная реализация конвертера Markdown-файлов в Latex-документ

Программная реализация состоит из нескольких этапов.

В рамках первого этапа с помощью блока очистки выполняется выделение полезной информации и запись в промежуточный файл. Второй этап заключается в повторной очистке, который полностью подготовит документ к конвертации, устранив из файла не

поддающие преобразованию фрагменты. После обработки модулем очистки результирующий для него файл будет прикреплен к шаблону LaTeX с помощью ключевого слова MarkdownInput{файл}, где файл шаблона создается в момент, когда будет готов очищенный файл. Последним этапом работы является вывод двух результирующих файлов: один – LaTeX-файл, который хранит в себе ссылку на второй файл формата Markdown, а также данные о разметки текста и необходимые LaTeX библиотеки для корректной работы и отображения результата. На рисунке 8 приведена общая структурная схема программного обеспечения.

Рисунок 8. Структурная схема программного обеспечения конвертера Markdown-файлов в LaTeX-документ

Таким образом, программное обеспечение можно разделить на несколько модулей, таких как:

1. Модуль очистки;
2. Модуль создания LaTeX шаблона;
3. Модуль вывода результата.

Модуль очистки открывает файл и построчно с помощью оператора while обрабатывает текст исходного файла. Первая строка обрабатываемого файла содержит значение “---”, в кодировке Markdown оно создает прямую горизонтальную линию внутри блока, огражденного парой таких значений. В данной строке находится непереводимый в LaTeX заголовок файла. С помощью Python-выражения *file = open ("имя_файла.md", "r", encoding='utf-8')* открываем файл "имя_файла.md", для чтения в кодировке UTF-8. Далее с помощью инструкции file.readline считываем первую строку файла и сдвигаем первую строку, предварительно записав ее значение в переменную. Это требуется для того, чтобы найти конец заголовка. На листинге 1 приведен код программы, выполняющей первый этап очистки файла.

Листинг 1. Первый этап очистки

```
    del_pattern_list = r"title:"
    first_line = in_file.readline()
    line_index=0
    skip = False
    while True:
    # считываем строку
        line = in_file.readline()
    # прерываем цикл, если строка пустаяif not line:
        print("конец", line_index)break
    # выделение заголовка
        if re.search(del_pattern_list, line):
            test = re.split(r'\"', line)
            header= "# "+ test[1] +"\n\n"
            filter_file.writelines(header)
    # поиск лишнего заголовка
        if line == first_line:
            skip = True
        if skip:
            filter_file.writelines(in_file.readlines())
        line_index+=1
    filter_file.close
```

В результате выполнения данного кода (см. листинг 1) файл очищается от заголовка, при этом в переменную *header* записывается значение заголовка, написанного в исходном файле, с пометкой *tittle* . Поиск происходит при помощи использования регулярного выражения, которое является удобным инструментом при работе с текстом. Текст исходного файла записывается в промежуточный файл начиная со следующей строки после нахождения второй границы заголовка. Данное решение позволяет удалять заголовок за 4 итерации, что увеличивает скорость обработки, так как не нужно исследовать каждую строку файла.

Второй этап обработки требуется для удаления списка источников. Здесь происходит обработка файла *filter_file* , созданного в первом этапе. Задачей данного этапа является определение и удаление списка источников, так как в Markdown-файле он формируется с помощью непереводимых в LaTeX команд, вызывающих ошибку при конвертации. На листинге 2 приведен код программы, решающий задачу второго этапа очистки файла.

Листинг 2. Второй этап очистки

```
    def sourse_list_check(line, line_index):
        sourse_pattern = r'\[\^\d\]'
        sourse_list =re.match(sourse_pattern, line)
        if sourse_list:
            return line_index


    line_index=0
    while True:
        line = filter_file.readline()
        if not line:
            print("конец", line_index)
            break
        if sourse_list_check(line, line_index) != None:
            sourse_index = line_index
            print(sourse_index)
            break
        line_index+=1
    filter_file.close
    filter_file = open("_posts/test_filter.md", "r",encod-
ing='utf-8')
    result = open("results/result.md", "w", encoding='utf-8')
    result.writelines(filter_file.readlines()[:sourse_index-1])
```

Здесь также применяются регулярные выражения. С помощью функции

sourse_list_check(line, line_index) , которая принимает текущую строку и её номер, происходит поиск по заданному шаблону с возвращением номера строки при совпадении. Номер строки запоминается и происходит выход из цикла, далее данные из промежуточного файла записываются в результирующий файл до строки с началом списка источников. Для этого из найденного функцией значения вычитаем 1, иначе первая строка со списком источников войдет в файл.

В результате выполнения двух этапов очистки создается результирующий Markdown-файл.

Модуль создания LaTeX шаблона

Для создания LaTeX шаблона использовался пустой шаблон LaTeX документа, в который были добавлены требуемые библиотеки. На листинге 3 показан готовый LaTeX шаблон.

Листинг 3. LaTeX шаблон

```
\documentclass{article}
\usepackage[utf8]{inputenc}
\usepackage[english,russian]{babel}
\usepackage[T1]{fontenc}
\usepackage[fencedCode,inlineFootnotes,citations,definitionL
ists,hashEnumerators,smartEllipses,hybrid]{markdown}
\usepackage{packages/sleek-title}
\usepackage{packages/sleek-theorems}
\usepackage{packages/sleek-listings}
\usepackage[noheader]{packages/sleek}
\usepackage{minted}
\title{Тест}
\author{тест}
\date{22.11.22}
\begin{document}
\maketitle
\markdownInput{result.md}
\end{document}
```

documentclass[12pt, letterpaper]{article} - определяет тип документа. В команду можно передать некоторые дополнительные параметры, заключенные в скобки и разделенные запятыми. В примере дополнительные параметры задают размер шрифта (12pt), размер по умолчанию – 10pt и размер бумаги (letterpaper).

usepackage[utf8]{inputenc} – это кодировка документа, позволяющая использовать в тексте символы за пределами ASCII (например, à, ü, č...). Его можно опустить или изменить на другую кодировку, но рекомендуется использовать utf-8.

usepackage[english,russian]{babel} определяет используемые языки по умолчанию, русский язык не доступен.

usepackage[T1]{fontenc} – это кодировка шрифта (определяет, какой шрифт используется.

usepackage[fencedCode,inlineFootnotes,citations,definitionLists,hashEnumerators,smartEllipses,hybrid]{markdown} – множество библиотек для Markdown, требуемых для корректного отображения его функций.

fencedCode – используется для корректного отображения кода.

hashEnumerators – используется для создания упорядоченных списков.

inlineFootnotes – позволяет создать ссылки на внешние сайты.

hybrid – позволяет использовать LaTeX код внутри Markdown документа.

Строки:

usepackage{packages/sleek-title}

usepackage{packages/sleek-theorems}

usepackage{packages/sleek-listings}

usepackage[noheader]{packages/sleek}

подключают соответствующий шаблон, параметр noheader выключает стандартный заголовок стиля.

Внутри файлов шаблона описываются: разметка текста, стили вывода уравнений, кода, заголовков, обычного текста.

Результат выводится после выполнения работы блока очистки, однако он не является читаемым для пользователя, так как закодирован. С помощью сервиса overleaf происходит чтение результирующего файла и конвертация в формат pdf.

Это решение имеет ряд преимуществ, например, высокая производительность, возможность редактировать Markdown-файл после включения его в LaTeX документ, что сильно повышает гибкость решения.

Заключение

По результатам проделанной работы был разработан конвертер Markdown-файлов в LaTeX-документ, нацеленный на исключение рутинной ручной работы по преобразованию Markdown-файлов в LaTeX-документ.

Решены задачи:

1. Представлен программный проект конвертера;
2. Разработан и реализован алгоритм работы модулей очистки;
3. Разработан модуль формирования выходного документа с заданным LaTeX-шаблоном.

В будущем планируется развивать данный проект посредством расширения приложения новыми стилями выходного документа, оптимизацией кода, по критерию быстродействия [5,6], добавлением новых возможностей, таких как пакетная обработка файлов, интеграция программы в качестве расширения для браузера и обеспечения доступа к базе данных с результирующими документами с соблюдением принципов информационной безопасности [7,8].

Библиография

1. Мехмонов И. Н. Инструментарии автоматизированного формирования динамических документов //Прикладная математика и информатика: Современные исследования в области естественных и технических наук. – 2020. – С. 883-886.
2. Павлов Д. А. Автоматическая вёрстка и оформление научной и программной документации //Компьютерные инструменты в образовании. – 2018. – №. 6. – С. 39-46.

3. B. Luo, W. Zhu, P. Li and Z. Han, "Distributed Dynamic Cuckoo Filter System Based on Redis Cluster," 2018 IEEE 4th International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing, (HPSC) and IEEE International Conference on Intelligent Data and Security (IDS), 2018, pp. 244-247, doi: 10.1109/BDS/HPSC/IDS18.2018.00059.
4. J. Tippayachai and S. Kiattisin, "Academic Publishing Solution Based on LATEX Class Package Implementation for ITMSOC Journal," 2018 3rd Technology Innovation Management and Engineering Science International Conference (TIMES-iCON), 2018, pp. 1-5, doi: 10.1109/TIMES-iCON.2018.8621689.
5. Gibadullin, R.F., Lekomtsev, D.V. & Perukhin, M.Y. Analysis of Industrial Network Parameters Using Neural Network Processing. Sci. Tech. Inf. Proc. 48, 446–451 (2021). https://doi.org/10.3103/S0147688221060046.
6. Гибадуллин Р.Ф. Потокобезопасные вызовы элементов управления в обогащенных клиентских приложениях // Программные системы и вычислительные методы. – 2022. – № 4. – С. 1-19. DOI: 10.7256/2454-0714.2022.4.39029 EDN: IAXOMA URL: https://nbpublish.com/library_read_article.php?id=39029.
7. Гибадуллин Р.Ф. Организация защищенной передачи данных в сенсорной сети на базе микроконтроллеров AVR // Кибернетика и программирование. – 2018. – № 6. – С. 80-86. DOI: 10.25136/2306-4196.2018.6.24048 URL: https://nbpublish.com/library_read_article.php?id=24048.
8. Гибадуллин Р. Ф. Развитие единообразного формализма защиты точечных, линейных и площадных объектов картографии // Вестник Казанского государственного технического университета им. А.Н. Туполева. – 2010. – №. 2. – С. 101-105.

Результаты процедуры рецензирования статьи

В связи с политикой двойного слепого рецензирования личность рецензента не раскрывается.
Со списком рецензентов издательства можно ознакомиться здесь.

Предмет исследования – разработка конвертера Markdown-файлов в LaTeX-документ.

Методология исследования основана на сочетании теоретического и эмпирического подходов с применением методов анализа, обобщения, сравнения, синтеза, программирования.

Актуальность исследования определяется широким распространением информационно-коммуникационных технологий, важностью проектирования и реализации соответствующих программных продуктов.

Научная новизна связана с разработкой автором программного продукта (конвертера), который включая алгоритм работы модулей очистки, а также модуля формирования выходного документа с заданным LaTeX-шаблоном.

Статья написана русским литературным языком. Стиль изложения научный.

Структура рукописи включает следующие разделы: Введение ("громоздкие" текстовые редакторы Microsoft Word, Notepad++ и др., риск неправильной конвертации документа, язык разметок, преимущества языка Markdown), Язык упрощённой разметки Markdown

(цель языка, отличия от языка HTML), Структура документов LaTeX (основная идея LaTeX, документ LaTeX), Веб-редактор Overleaf (формат .sty, удобный интерфейс, редактирование файла нескольким пользователям), Предпосылки к разработке конвертера (структурная схема конвертера Markdown-файлов в LaTeX-документ, структурная схема обработки документа, алгоритм работы конвертера Markdown-файлов в LaTeX-документ, пример титульного листа в Markdown-файле, пример титульного листа в LaTeX-документе),
Программная реализация конвертера Markdown-файлов в Latex-документ (структурная схема программного обеспечения конвертера Markdown-файлов в LaTeX-документ, модули очистки, создания LaTeX шаблона, вывода результата), Модуль создания LaTeX шаблона (разметка текста, стили вывода уравнений, кода, заголовков, обычного текста), Заключение (выводы), Библиография.

Текст включает восемь рисунков, три листинга. Листинги также можно обозначить рисунками.

Содержание в целом соответствует названию. Представлено описание конвертера Markdown-файлов в LaTeX-документ, нацеленный на исключение рутинной ручной работы, а также определены перспективы развития проекта (расширение новыми стилями, оптимизация кода, увеличение быстродействия, добавление пакетной обработки файлов и пр.).

Библиография включает восемь источника зарубежных авторов (научные статьи). Библиографические описания некоторых источников требуют корректировки в соответствии с ГОСТ и требованиями редакции, например:
3. Luo B., Zhu W., Li P., Han Z. Distributed Dynamic Cuckoo Filter System Based on Redis Cluster // IEEE 4th International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing, (HPSC) and IEEE International Conference on Intelligent Data and Security (IDS). Место издания ???, 2018. P. 244–247.
Возможно излишнее самоцитирование (Гибадуллин Р. Ф. с соавторами).

Апелляция к оппонентам (Мехмонов И. Н., Павлов Д. А., Luo B., Zhu W., Li P., Han Z., Tippayachai J., Kiattisin S. и др.) имеет место.

В целом материал представляет интерес для читательской аудитории и после доработки может быть опубликован в журнале «Программные системы и вычислительные методы».