DOCUMENTATION

LaTeX to Markdown Converter (Compiler)

Author – Gaurav Nirala [2024MCS2466]

INDIAN INSTITUTE OF TECHNOLOGY , DELHI

COP701 : Software System Lab

Table of Contents

INTRODUCTION

The LaTeX to Markdown Converter Compiler is a tool designed to convert LaTeX documents into Markdown format. Utilizing Flex for lexical analysis and Bison for syntax parsing, this compiler processes LaTeX commands and structures, converting them into an equivalent Markdown representation. The goal is to simplify the conversion process for complex LaTeX documents, making them more accessible for platforms that use Markdown.

Project Structure

Provide an overview of the project directory structure, explaining the role of each component:

- ast_tree/: Contains the AST implementation.
- src/: Holds the lexer and parser source files.
- Traversor/: Implements the traversal logic for AST to Markdown conversion.
- main.cpp: The entry point of the application.
- Makefile: Automates the build process.
- test.tex: A sample LaTeX file for testing.
- googletest/ and test.cpp: For unit testing with Google Test.

The features(tags) of LaTeX considered:-

\documentclass	**\usepackage**	**\title**	**\date**	**\par**
\begin{document}	**\end{document}**	**\section**	**\subsection**	**\hrule**
\subsubsection	**\includegraphics**	**\begin{verbatim}**	**\end{verbatim}**	**\textbf**
\begin{itemize}	**\end{itemize}**	**\begin{enumerate}**	**\end{enumerate}**	**\textit**
\begin{tabular}	**\end{tabular}**	**& , \ **	**\hline**	**\href**

File Structure

File structure -->

```
latex2mdConvertor ( main directory )
    |-> ast_tree/ [ABSTRACT_SYNTAX_TREE]
    |       |-> ast.h (ast header file)
    |       |-> ast.cpp
    |
    |->images/
    |     |-> images.jpg
    |
    |-> src/  [SOURCE_FILES]
    |    |-> lexer.l
    |    |-> parser.y
    |
    |->Traversor/ [SYNTAX_TO_SEMENTIC(markdown)]
    |      |-> traverse.h (Traverse header file)
    |      |-> traverse.cpp
    |
    |->   main.cpp [MAIN_FILE]
    |
    |->   latex2md.out [COMPILER]
    |
    |->   Makefile [ALL necessory sequence of command]
    |
    |->   test.tex [TEST_LATEX_FILE]
    |
    |->   googletest/ [gtest tools e.g. lgtest lgtest_main]
    |
    |->   test.cpp   [MIAN TESTING file]
    |
    |->   gtest_result     [output file of gtest]
```

Installation and Setup

Prerequisites

- C++ Compiler: Ensure that you have a C++ compiler installed (e.g., g++).
- Flex: Install Flex for lexical analysis.
- Bison: Install Bison for syntax parsing.
- Google Test: Google Test framework for unit testing.

Setup

- **Clone the repository to your local machine.**

```
$ git clone https://github.com/GauravNirala05/Cop701-assignment01.git
```

- Navigate to the project directory.

```
$ cd latex2mdConvertor
```

- Compile the project using the provided Makefile.

```
$ make compiler
```

Usage

Running the Compiler

1. **Ensure the project is compiled (latex2md.out).**
2. **Run the compiler with a LaTeX file as input.**

```
$ ./latex2md.out  input.tex  output.md
```

- **Replace input.tex with your LaTeX file.**
- **The Markdown output will be saved to output.md.**

How To run code-

1. cd into latex2mdConvertor
2. run- "make compiler"
3. run ./latex2md.out test.tex output.md
4.

How To run gtest -

1. cd into latex2mdConvertor
2. run- "make gtest"
3. run ./gtest_result

Testing

Running Unit Tests

1. Compile the test suite using the Makefile.

```
$ make gtest
```

2. Run the tests.

```
$ ./gtest_result
```

3. Check the output in the gtest_result file for test results.

Test Framework

Testing for this project was conducted using Google Test (gtest), a robust C++ testing framework that provides a simple and effective way to write unit tests. The choice of gtest allows for automated testing of the compiler's functionality, ensuring that each component of the LaTeX to Markdown conversion process works as expected.

Test Cases

- LaTeX Document Coverage: The test cases cover a variety of LaTeX commands, including sections, bold and italic text, lists, tables, and images.
- Test Implementation (test.cpp): The test cases are implemented in test.cpp, which includes assertions to validate the correctness of the Markdown output against expected results.
- Automated Testing: Using the gtest target in the Makefile, tests are automatically compiled and executed, with results output to gtest_result.

GTEST RESULT

```
gaurav@Gaurav:/mnt/c/Users/gaura/OneDrive/Desktop/cop701/latex2mdConvertor$ ./gtest_result
[==========] Running 13 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 13 tests from Latex_To_Markdown
[ RUN      ] Latex_To_Markdown.section
[       OK ] Latex_To_Markdown.section (18 ms)
[ RUN      ] Latex_To_Markdown.subsection
[       OK ] Latex_To_Markdown.subsection (15 ms)
[ RUN      ] Latex_To_Markdown.subsubsection
[       OK ] Latex_To_Markdown.subsubsection (16 ms)
[ RUN      ] Latex_To_Markdown.italictext
[       OK ] Latex_To_Markdown.italictext (16 ms)
[ RUN      ] Latex_To_Markdown.boldtext
[       OK ] Latex_To_Markdown.boldtext (16 ms)
[ RUN      ] Latex_To_Markdown.hrule
[       OK ] Latex_To_Markdown.hrule (15 ms)
[ RUN      ] Latex_To_Markdown.href
[       OK ] Latex_To_Markdown.href (14 ms)
[ RUN      ] Latex_To_Markdown.paragraph
[       OK ] Latex_To_Markdown.paragraph (16 ms)
[ RUN      ] Latex_To_Markdown.verbatim
[       OK ] Latex_To_Markdown.verbatim (14 ms)
[ RUN      ] Latex_To_Markdown.graphics
[       OK ] Latex_To_Markdown.graphics (13 ms)
[ RUN      ] Latex_To_Markdown.itemize
[       OK ] Latex_To_Markdown.itemize (13 ms)
[ RUN      ] Latex_To_Markdown.enumerate
[       OK ] Latex_To_Markdown.enumerate (16 ms)
[ RUN      ] Latex_To_Markdown.table
[       OK ] Latex_To_Markdown.table (17 ms)
[----------] 13 tests from Latex_To_Markdown (204 ms total)

[----------] Global test environment tear-down
[==========] 13 tests from 1 test suite ran. (204 ms total)
[  PASSED  ] 13 tests.
gaurav@Gaurav:/mnt/c/Users/gaura/OneDrive/Desktop/cop701/latex2mdConvertor$
```

NOTE : LaTeX comments (e.g. % .....) are not considered in this compiler

Development Environment And Technology used

- **Operating System:** Windows and WSL(Ubuntu)
- **Compiler:** GCC and G++
- **Flex & Bison Versions:** (flex 2.6.4)(bison (GNU Bison) 3.8.2).
- **Gtest : v1.15.2**
- **CMake** : version 3.22.1
- **Editor/IDE:** E.g., VS Code.