Markdown to HTML converter

Introduction

The objective of the assignment is to implement a parser that converts a Markdown document to its equivalent HTML document.

The parser needs to support the markdown features such as headings, paragraphs, line break, bold, italic emphasis, ordered lists, unordered lists, images, URLs, and tables.

This report conveys a brief description of the features of my parser, the tools it uses, build and testing methods and the limitations in the implementation.

Design and Tools

I have used Flex (Fast Lexical Analyzer Generator) for my lexical analysis. The input Markdown file is broken down into different tokens using the Lexical analyser, as per the specified regex expressions. Later the generated tokens are used by the parser, which is implemented using Bison to produce an abstract syntax tree (AST). The AST is mapped to HTML syntax to produce the final output HTML file.

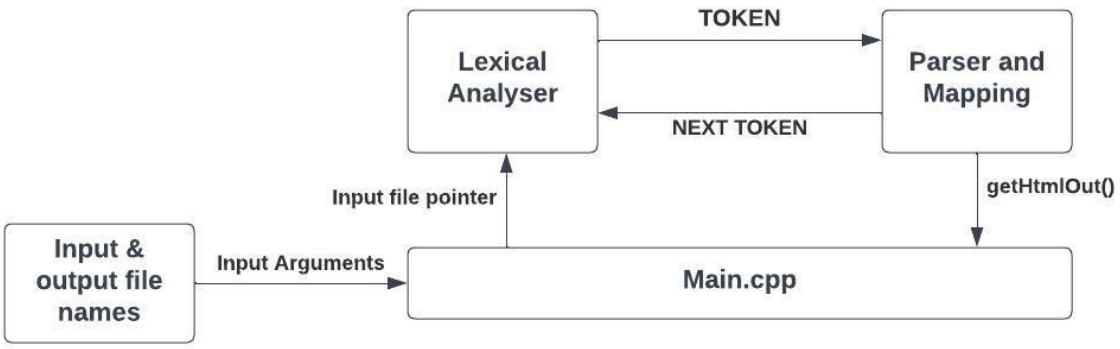
In bison, I have specified the context-free-grammar of Markdown using BNF expressions. This in turn produces a parser in C language.

An explicit implementation of tree using dynamic memory allocation of nodes are not done in this design. The mapping of Markdown AST to HTML AST is done directly in the bison code. I have used string datatype of C++, to perform this mapping. This design choice take as the output language is fixed.

The parser includes an interface function ***getHtmlOut()***, which returns the equivalent HTML tags of the input Markdown file in string format.

The Markdown file name is accepted using input arguments. Later the file is opened in read mode and the file pointer is mapped to the input file pointer of the lexer.

The output HTML file name is also accepted through input arguments and opened in write mode. The HTML body headers is written into the output file and the mapped HTML information is fetched using ***getHtmlOut()*** function and written into the output file.



For unit testing I have used GoogleTest framework. The framework is built for C++. I have generated predefined input markdown files and have validated the expected and actual results using the test framework. The framework is useful in generating a consolidated test report and in organizing test cases into various test suites. This helps in grouping similar features during test phase. The test report for my implementation is as follows:

```
swakath@LAPTOP-EJIR81GE:/mnt/d/IITD/Sem1/COL701-Software Systems Lab/markdown2html/gtest/build$ ./converterUnitTest
[==========] Running 16 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 16 tests from markdown2htmlcheck
[ RUN      ] markdown2htmlcheck.Heading1
[       OK ] markdown2htmlcheck.Heading1 (28 ms)
[ RUN      ] markdown2htmlcheck.Heading2
[       OK ] markdown2htmlcheck.Heading2 (11 ms)
[ RUN      ] markdown2htmlcheck.Heading3
[       OK ] markdown2htmlcheck.Heading3 (11 ms)
[ RUN      ] markdown2htmlcheck.Heading4
[       OK ] markdown2htmlcheck.Heading4 (10 ms)
[ RUN      ] markdown2htmlcheck.Heading5
[       OK ] markdown2htmlcheck.Heading5 (10 ms)
[ RUN      ] markdown2htmlcheck.Heading6
[       OK ] markdown2htmlcheck.Heading6 (13 ms)
[ RUN      ] markdown2htmlcheck.Paragraph
[       OK ] markdown2htmlcheck.Paragraph (11 ms)
[ RUN      ] markdown2htmlcheck.Line_Break
[       OK ] markdown2htmlcheck.Line_Break (9 ms)
[ RUN      ] markdown2htmlcheck.Soft_Break
[       OK ] markdown2htmlcheck.Soft_Break (10 ms)
[ RUN      ] markdown2htmlcheck.Italic
[       OK ] markdown2htmlcheck.Italic (6 ms)
[ RUN      ] markdown2htmlcheck.Strong
[       OK ] markdown2htmlcheck.Strong (10 ms)
[ RUN      ] markdown2htmlcheck.Strong_Italic
[       OK ] markdown2htmlcheck.Strong_Italic (7 ms)
[ RUN      ] markdown2htmlcheck.Ordered_List
[       OK ] markdown2htmlcheck.Ordered_List (12 ms)
[ RUN      ] markdown2htmlcheck.Unordered_List
[       OK ] markdown2htmlcheck.Unordered_List (12 ms)
[ RUN      ] markdown2htmlcheck.Link
[       OK ] markdown2htmlcheck.Link (5 ms)
[ RUN      ] markdown2htmlcheck.Image
[       OK ] markdown2htmlcheck.Image (10 ms)
[----------] 16 tests from markdown2htmlcheck (182 ms total)

[----------] Global test environment tear-down
[==========] 16 tests from 1 test suite ran. (183 ms total)
[  PASSED  ] 16 tests.
```

Build and Usage

The repository tree is as follows:

```
swakath@LAPTOP-EJIR81GE:/mnt/d/IITD/Sem1/COL701-Software Systems Lab/markdown2html$ tree -d
.
├── gtest
│   └── testFiles
├── learnings
│   ├── flex
│   │   ├── calculator
│   │   │   └── src
│   │   └── wc_mock
│   └── markdow_syntax
└── src
```

- **“src”** directory contains all the source code of the lexer and parser.
- **“gtest”** directory contains all the unit testing related code. It also contains the **“testFiles”** directory with contains all the input markdown files for unit testing.
- **“learnings”** directory contains all the flex and bison resources and example implementations to get familiar with the tools.

I have used CMake tool for build process. The build step for generating the final executable parser is as follows:

```
# Create a build directory inside markdown2html
mkdir build
cd build
cmake ..

# Parser Usage
./markdown2html input.md output.html
```

The build step for unit testing:

```
# Create a build directory inside gtest directory
mkdir build
cd build
cmake ..

# Testing usage
./converterUnitTest
```

Limitations

- The designed parser does not support tables and cascaded lists.
- Double newline after individual entries of the lists needs to be discarded as per the specifications, but the implementation treats it as two different lists.
- To distinguish between newline and soft break (\n treated as space) the implementation limits paragraphs to start with alphabets (small and caps). This does not match with the actual markdown specification.
- The implementation does not use an explicit tree data structure for AST. Due to this expanding the implementation for conversions to other languages in future is difficult.

Tool versions

- Flex 2.6 or above
- Bison 3.0 or above
- g++ compiler 7.5.0 or above
- CMake 3.1 or above
- GTest 1.8.0 or above