ECOTE - Final Documentation
Translator of a LaTeX subset to HTML

Krzysztof Rudnicki, 307585
Semester: 2023L

June 7, 2023

1 General overview and assumptions

My task was to create a translator of LaTeX subset to selected text format with focus on LaTeX tables
I decided to change to translator of LaTeX subset to HTML since I know LaTeX very well and HTML relatively well, I decide to translate LaTeXinto HTML since HTML is easy, a little bit different than LaTeXand popular which makes this translator a practical tool.

1.1 Assumptions

- "Tables" will be represented using LaTeX *table* environment
- All changes influenced by different LaTeX document class are ignored in html
- Everything between documentclass and begin document is ignored when generating html
- Program tries to generate final html even if encountering errors which would make final html not complete
- Only commands defined in LaTeX subset are translated, rest is translated literally as raw text to html

2 Functional requirements

The goal of the project is to transform .tex file to (working) .html file if the subset of .tex file is within project scope or output error message explaining why the html could not be outputed

2.1 LaTeX subset

This project will focus almost exclusively on *tabular* environment

- \documentclass{class}: Defines what layout standard LaTeXwill use
- \begin{document}: Ends (in our case empty) preamble
- \end{document}: Ends LaTeX document
- \begin{tabular}[pos]{table spec}: Opens environment used to typeset tables
- \end{tabular}: Closes environment used to typeset tables

Supported tabular arguments:

- l, c, r - respectively left-justified, centered and right-justified column
- | - single vertical line
- || - double vertical line
- p{'width'} - paragraph column (that wraps), aligned at the top
- m{'width'} - paragraph column (that wraps), aligned at the middle
- b{'width'} - paragraph column (that wraps), aligned at the bottom

Supported commands inside tabular environment

- & - separate columns
- \\ - start new row
- \hline - horizontal line

- \cline{i-j} horizontal line beginning in column i and ending in column j
- \newline - new line *inside* the cell

3 Implementation

I decided to use Python as a language in which I will implement my solution The reasons for using python are as follow:

1. It is the easiest language among those that I know
2. I know it enough to be confident in my ability to implement this solution in python
3. I want to learn python more through this project

Negative aspects of python which is that it is very slow language do not bother me as I believe the project scope will not be big enough for this to become an issue

3.1 General architecture

Module	Description
Main	Handles texfile, puts it to a function translating to html and saves output html file
FileHandler	Handles reading latex file and transforming it to string in python
ReadStart	Checks documentclass and begin document functions of latex file and if they are correct outputs starting html tags
HandleInside	Goes through inside of LaTeX document and transforms each of its part to HTML
HandleTable	Reads table, including table parameters and actual contents and transforms it to html
TabInsideHandler	Handles actual content of table and translates them to html

3.2 Data structures

- File entered by user is represented by python File class
- Parsed LaTeX code is represented by data structure based on node classes
- Tabular environment parameters are stored in an array, if the parameter contains additional optional parameters they are stored in a pair of the parent argument and array of all optional arguments
- Generated HTML code is stored in node classes
- Final HTML code is stored in plain text and then written to File

3.3 Module descriptions

3.3.1 Main

Puts latex filename to filehandler, sends retrieved string to function translating tex to html and then saves final html file

3.3.2 FileHandler

Handles texfile and transforms it to python string
Input:
tex_filename - LaTeX file filename
Functions:

- Handle arguments - Checks if arguments are correct, reads file and saves it to python string
- Invoke Latex Handler - Sends file from file handler to LatexHandler

3.3.3 LatexHandler

Transforms tex string to html Input: tex string Functions:

- ReadStart - Checks begininig of tex file and returns html code if the begining is correct
- HandleInside - Handles insides of tex file, both tables and loose text and translates it to html

3.3.4 ReadStart

Checks begininig of tex file and returns html code if the begining is correct
Input: tex string
Output: html_string

- read_document_class - Checks if document class exists and is written according to tex rules, returns <!DOCTYPE html> if yes
- read_begin_document - (optional) Checks if begin{document} function was written in tex file if yes adds <html> to html_string
- HandleInside - Passes parsed tex data and html_string to function handling inside of tex document

3.3.5 HandleInside

Translates tex string starting at begin document and ending at end document, returns final html_string
Parameters:

- tex_data - String read from tex file
- html_string - Final output string to which table data and loose text will be added

Functions:

- HandleTable - Converts tex tables to html tables
- HandleLoose - Goes through text that is not in table and converts it to html

3.3.6 HandleTable

Converts tex tables to html tables
Parameters:

- tex_data - String read from tex file

- html_string - Final output string to which table data and loose text will be added

Functions:

- TabArgHandler - Reads tex table parameters and translates them to html style
- TabInsideHandler - Reads inside of tex table and together with parse parameters outputs correct table layout and look

3.3.7 TabInsideHandler

Together with html table parameters translates inside of tex table to html
Parameters:

- TableContents - Latex string containing only inside of table

Functions:

- ReadTableContent - Parses table content and checks for errors
- ConvertToHTML - Converts table inside to html

3.3.8 TabArgHandler

Reads tex table parameters and translates them to html style
Parameters:

- TableArguments - Arguments provided with \begin{tabular} environment

Functions:

- ReadTableARguments - Parses table arguments and checks for errors
- ConvertToHTML - Converts table arguments to html style

3.4 Input/output description

Input is a .tex file (LaTeX file)
Outpus is an .html file
In case of errors error message will be outputed on the terminal
Input File path is entered as an argument to terminal with "-i" or "–input" flag for example:

```
python main.py -i texFile.tex
```

Output file path can be named by user by using "-o" or "–output" flag:

```
python main.py -i texFile.tex -o htmlFile.html
```

If no "-o" flag is issued the output file will have the same name as input file with changed extension to html (so in this example texFile.tex will become texFile.html)
If the path to file name consists of spaces, path name needs to be but in ""

```
python main.py -i "My␣Folder/input.tex"
```

3.5 Others

4 Unit tests

There are over 12 unit tests for program functions, all of them can be ran by using python test environment:

```
python -m pytest unit_tests

collected 12 items

unit_tests/test_code/test_begin_document.py .

    [  8%]
unit_tests/test_code/test_begin_tabular.py .

    [ 16%]
unit_tests/test_code/test_document_class.py .

    [ 25%]
```

```
unit_tests/test_code/test_length_conversions.py .

    [ 33%]
unit_tests/test_code/test_only_pipes_and_space.py .

    [ 41%]
unit_tests/test_code/test_split_columns.py .

    [ 50%]
unit_tests/test_code/test_split_rows.py .

    [ 58%]
unit_tests/test_code/test_tabular_columns_parameters.py .

    [ 66%]
unit_tests/test_code/test_tabular_parameters.py .

    [ 75%]
unit_tests/test_code/test_tabular_required_parameters.py .

    [ 83%]
unit_tests/test_code/test_translate_column.py .

    [ 91%]
unit_tests/test_code/test_translate_inside_to_html.py .

    [100%]

============================================================= 12
    passed in 0.02s
    ============================================================
```

5 Functional test cases

Title	Input (LaTeX)	Output
empty file		`Error! expected \documentclass at the begining of LaTeX file`
Document class	`\documentclass[options]{class}`	`Error! expected \begin{document} after document class`
`Extra text between document class and begin document`	`\documentclass[options]{class}` `"extra text"` `\begin{document}`	`Error! unexpected text between document class and begin document`
`Just document class and begin document`	`\documentclass[options]{class}` `\begin{document}`	`Error! no \end{document} at the end of LaTeX code`

Title	Input (LaTeX)	Output

Just document class and begin/ end document	`\documentclass[options]{class}`
	`\begin{document}`
	`\end{document}`
		`<html>`
		`</html>`

Plain text inside	`\documentclass[options]{class}`
	`\begin{document}`
	`Lorem ipsum dolor sit amet.`
	`\end{document}`
		`<html>`
		`Lorem ipsum dolor sit amet.`
		`</html>`

Reduntant \end{document} (ignored)	`\documentclass[options]{class}`
	`\begin{document}`
	`Lorem ipsum dolor sit amet.`
	`\end{document}`
	`\end{document}`
		`<html>`
		`Lorem ipsum dolor sit amet.`
		`</html>`

LaTeX comments	`\documentclass[options]{class}`
	`\begin{document}`
	`Lorem ipsum dolor sit amet.`
	`% some comment`
	`\end{document}`
	`\end{document}`
		`Error! LaTeX comment detected at line 3`

Title	Input (LaTeX)	Output
Table with vertical lines	\documentclass[options]{class} \begin{document} \begin{tabular}{ l | c | r } test & 2 & test \\\\ 4 & 5 & 6 \\\\ \end{tabular} \end{document}	<html> <table> <tr> <td align='left'>test</td> <td align='center' style="border-left: 1px solid black;">2</td> <td align='right'>test</td> </tr> <tr> <td align='left'>4</td> <td align='center' style="border-left: 1px solid black;">5</td> <td align='right'>6</td> </tr> </table> </html>
Missing &	\documentclass[options]{class} \begin{document} \begin{tabular}{ l c r } 1 & 2 & 3 \\\\ 4 & 5 6 \\\\ \end{tabular} \end{document}	Error! Missing third column in second row

Title	Input (LaTeX)	Output
Too much columns	`\documentclass[options]{class}` `\begin{document}` `\begin{tabular}{ l c r }` `1 & 2 & 3 & 4 & 5 \\` `4 & 5 6 \\` `\end{tabular}` `\end{document}`	`Error! Too much columns in row 1, expected 3, got 5`