

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

RAFAEL FELIPE TASAKA DE MELO

**RELATÓRIO: TRADUTOR DE LINGUAGEM LATEX PARA
MARKDOWN**

RELATÓRIO

PONTA GROSSA
2019

RAFAEL FELIPE TASAKA DE MELO

**RELATÓRIO: TRADUTOR DE LINGUAGEM LATEX PARA
MARKDOWN**

Relatório apresentado como requisito para o primeiro trabalho da matéria de Compiladores.

Orientador: Gleifer Vaz Alves

PONTA GROSSA
2019

SUMÁRIO

1	INTRODUÇÃO	3
2	BREVE EXPLICAÇÃO SOBRE O LATEX E O MARKDOWN	4
2.1	A LINGUAGEM LATEX	4
2.1.1	A estrutura do Latex	4
2.2	A LINGUAGEM MARKDOWN	5
2.2.1	A estrutura Markdown	5
3	DESENVOLVIMENTO	6
3.1	A GRAMÁTICA	6
3.2	O CÓDIGO	7
3.2.1	Analisador léxico	7
3.2.2	Analisador sintático	8
4	TESTES E RESULTADOS	10
4.1	COMPILANDO O TRADUTOR	10
4.2	ARQUIVOS BEM-SUCEDIDOS	10
4.3	ALGUNS RESULTADOS	12
4.4	CASOS EM QUE A TRADUÇÃO FOI MAL-SUCEDIDA	13
5	CONCLUSÃO	15

1 INTRODUÇÃO

Na maioria das vezes traduzir arquivos manualmente pode ser um trabalho tedioso. Para isso algoritmos de automatização são feitos para que, dado um arquivo de entrada, outro seja criado já com a solução proposta.

Paralelamente a isso diversas linguagens de texto são utilizadas para se escrever na web como o HTML e o Markdown. Em outras áreas, linguagens como o Latex são amplamente usadas para elaboração de documentos.

Este trabalho tem como objetivo criar um tradutor de arquivos .lex (Latex) para .md (Markdown) utilizando as ferramentas Flex e Bison.

2 BREVE EXPLICAÇÃO SOBRE O LATEX E O MARKDOWN

Nesta seção serão apresentados alguns conceitos básicos sobre as linguagens Markdown e Latex - bem como uma parte de suas gramáticas.

2.1 A LINGUAGEM LATEX

A linguagem Latex é uma linguagem de alta qualidade com recursos específicos para documentos, principalmente, científicos (LATEX..., 2019). Para muitos, o Latex é como "programar" um documento.

A estrutura do Latex é, basicamente, a declaração do tipo de documento e seus pacotes; e o documento em si. Dentro do documento podemos ir definindo várias coisas como estilos de texto (itálico, negrito, etc.), capítulos, seções, referências, entre outros. Para este trabalho, a gramática do Latex será apresentada mais adiante.

Diferentemente de outros métodos de processamento de textos como o Word ou o Pages, o Latex traz um jeito mais automático de formatação, logo a maior preocupação do usuário será o seu texto e como ele será organizado no documento.

2.1.1 A estrutura do Latex

A estrutura do Latex é bem grande e a medida que inserimos novos pacotes, novos comandos também são adicionados. Portanto, para este trabalho, será adotada uma linguagem mais simplificada, onde:

- "classtype" define o tipo de documento;
- "usepackage" insere novos pacotes no documento;
- 'begin{document}' e 'end{document}' são os delimitadores do documento;
- "paragraph" para um parágrafo
- "bf", "underline" e "it" são os personalizadores de texto (negrito, sublinhado e itálico, respectivamente);
- "begin/end itemize" e "begin/end numerate" são os delimitadores de uma lista não numerada e numerada, respectivamente e;
- "item" é um item da lista.

2.2 A LINGUAGEM MARKDOWN

A linguagem Markdown é umas das linguagens para escrita de textos em web (MASTERING...,). Um exemplo de seu uso são nos arquivos Read me de projetos de código aberto - geralmente encontrados em plataformas como o GitHub.

Sua estrutura é mais simples do que a do Latex. A linguagem é composta, basicamente, de alguns caracteres não alfabéticos - como o "#" e o "*", por exemplo - e textos simples.

2.2.1 A estrutura Markdown

Para este trabalho, a seguinte sintaxe será utilizada:

- "#" (ou mais) será(ão) utilizada(s) para mostrar o título, nome do autor, capítulo, seção e subseção por ordem hierárquica. Ou seja: "#" será para o título, "##" para o nome do autor e assim por diante;
- "*" para itens não numerados de uma lista;
- "1. " para itens numerados de uma lista ¹
- "*texto*" para itálico;
- "***texto***" para negrito;

¹ Apesar de todos os itens possuírem o índice 1, somente o primeiro é obrigatório. Os demais podem admitir qualquer outro número. A linguagem subentende que uma lista numerada deve ser feita e a organiza para o usuário.

3 DESENVOLVIMENTO

Esta seção apresentará partes do desenvolvimento do projeto com trechos do código e breves explicações sobre o funcionamento do analisadores.

3.1 A GRAMÁTICA

Precisaremos de dois itens para fazer o tradutor: um analisador léxico e um analisador sintático. Uma vez com a gramática em mãos podemos definir os tokens.

A gramática está definida a seguir em notação BNF¹:

```

<documentoLatex> ::= <configuracao> <identificacao> <principal>
  <configuracao> ::= CLASSE PACOTE
  | CLASSE
  <identificacao> ::= TITULO AUTOR
  | TITULO
  <principal> ::= <inicio> <corpoLista> <fim>
  <inicio> ::= begin '{' document '}'
  <fim> ::= end '{' document '}'
  <corpoLista> ::= <capitulo> <secao> <subsecao> <corpoLista>
  | <corpo>
  <capitulo> ::= chapter '{' CONTEUDO '}' <corpo> <capitulo>
  | chapter '{' CONTEUDO '}'
  <secao> ::= section '{' CONTEUDO '}' <corpo> <secao>
  | <corpo>
  <subsecao> ::= subsection '{' CONTEUDO '}' <corpo> <subsecao>
  | <corpo>
  <corpo> ::= <texto>
  | <texto> <corpo>
  | <textoEstilo> corpo>
  | <listas> corpo>
  <texto> ::= paragraph '{' CONTEUDO '}'
  | ∅
  <textoEstilo> ::= bf '{' CONTEUDO '}'
  | underline '{' CONTEUDO '}'

```

¹ A gramática foi dada pelo enunciado do projeto, contudo ela sofreu alterações pois, originalmente, ela estava ambígua em um ponto crítico, gerando problemas de erro de sintaxe

```

| it '{' CONTEUDO '}'
  <listas> ::= <listaNumerada>
| <listaItens>
  <listaNumerada> ::= begin '{' enumerate '}' <itensLNumerada> end '{' enumerate '}'
  <itensLNumerada> ::= item '{' CONTEUDO '}'
| item '{' CONTEUDO '}'
| <listas>
  <listaItens> ::= begin '{' itemize '}' <itensLItens> end '{' itemize '}'
  <itensLItens> ::= item '{' CONTEUDO '}'
| item '{' CONTEUDO '}'
| <listas>

```

Onde os tokens '{', '}', chapter, begin, end, section, subsection, paragraph, bf, underline, it, document, enumerate, itemize, item, CLASSE, PACOTE, TITULO, CONTEUDO e AUTOR são tokens terminais.

3.2 O CÓDIGO

Nesta seção será apresentado trechos do código - tanto do analisador léxico quanto do analisador sintático.

3.2.1 Analisador léxico

O analisador léxico pegará toda a entrada e analisará letras de maneira individual em tokens (MOGENSEN, 2010). Sua principal função é facilitar a análise para o analisador sintático. A ferramenta que fará isso será o Flex.

Uma vez esse tokens separados, o analisador sintático os recombinará - estruturando assim o corpo do texto.


```

TEXT [A-Za-z ]*[0-9]*[\\:;\\-\\.]*

%%

"\\chapter" {return chapter;}
"\\section" {return section;}
"\\subsection" {return subsection;}
"\\paragraph" {return paragraph;}
"\\bf" {return bf;}
"\\underline" {return underline;}
"\\it" {return it;}
"\\begin" {return begin;}
"\\end" {return end;}
"enumerate" {return enumerate;}
"\\item" {return item;}
"itemize" {return itemize;}
"document" {return document;}
{" |
}" {return yytext[0];}
"\\usepackage{"{TEXT}"}" {yyval.sval = strdup(yytext); return PACOTE;}
"\\author{"{TEXT}"}" {yyval.sval = strdup(yytext); return AUTOR;}
"\\title{"{TEXT}"}" {yyval.sval = strdup(yytext); return TITULO;}
"\\documentclass{"{TEXT}"}" {yyval.sval = strdup(yytext); return CLASSE;}
{TEXT} {yyval.sval = strdup(yytext); return CONTEUDO;}
. ;

```

Figura 1 – O analisador léxico

Não há grandes complicações nas expressões. Temos palavras reservadas para identificar autor, título, quando começamos uma lista, os delimitadores do documento, entre outros. Qualquer coisa diferente dessas palavras chaves será um texto - que é definido como CONTEUDO.

Uma vez que uma das regras é satisfeita o analisador lexico retorna ao analisador sintático qual token foi encontrado. Com o token em mãos o analisador sintático realizará o código C que será apresentado mais adiante.

3.2.2 Analisador sintático

O analisador sintático é mais complexo - uma vez que ele trabalha com a gramática, a tradução em si e a função main do programa.

Primeiramente o analisador sintático é dividido em três partes (assim como o analisador léxico)(LEVINE, 2009): declarações prévias, a gramática em si e as funções posteriores

Nas declarações prévias são declaradas algumas variáveis que serão utilizadas e a inclusão de bibliotecas como a `stdio.h` e a `stdlib.h`. Além disso também são declarados todos os tokens que serão utilizados.

A segunda parte do programa é a mais essencial para o funcionamento do tradutor. A medida que o analisador léxico retorna os tokens encontrados no arquivo de entrada, o analisador sintático os trata e faz alguns comandos em C para realizar a tradução em si.

Como o código é relativamente grande não serão tratadas todas as partes da gramática, mas alguns pontos principais serão explicados.

Como explicado anteriormente título, autor, capítulo, seção e subseção terão níveis de tamanhos diferentes no Markdown (indo de "#" até "#####"). A linguagem Latex já numera automaticamente capítulo, seção e subseção, logo, para o mesmo ocorrer em Markdown, foi adicionado contadores para cada um dos casos - assim como mostrado na figura a seguir:

```
capitulo: chapter '{' CONTEUDO '}' {countCapter++; fprintf(f, "####d. %s\n\n", countCapter, $3); free($3);} corpo capitulo
| chapter '{' CONTEUDO '}' {countCapter++; fprintf(f, "####d. %s\n\n", countCapter, $3); free($3);}
;
```

Figura 2 – Capítulos do analisador sintático

Para os estilos de texto a ideia adotada também já foi tratada. É importante ressaltar que a ausência do sublinhado em Markdown fará com que o tradutor somente transcreva o texto.

Para as listas um contador de espaços foi adicionado para ir indentando a lista, ou seja, uma vez que ele encontrar dentro de uma lista um "begin", uma tabulação é adicionada e, quando ele encontrar um "end", uma tabulação é retirada.

```
listaItens: begin '{' itemize '}' {countSpace++;} itensLIens end '{' itemize '}' {countSpace--; fprintf(f, "\n");} ;

itensLIens: item '{' CONTEUDO '}' {for(i = 0; i < countSpace; i++){fprintf(f, "\t");} fprintf(f, "* %s\n", $3);}
| item '{' CONTEUDO '}' {for(i = 0; i < countSpace; i++){fprintf(f, "\t");} fprintf(f, "* %s\n", $3);} itensLIens
| listas
;
```

Figura 3 – Trecho do analisador sintático sobre itens

A última parte do código engloba as funções main e yyerror. Na main é aberto o arquivo .md onde será escrita a tradução e evocamos yyparse() - que dá início à análise. A função yyerror será evocada quando um erro qualquer ocorrer na análise. Um exemplo é quando houver um erro de sintaxe.

4 TESTES E RESULTADOS

Com o código pronto foram realizados alguns testes para analisar tanto a gramática quanto os resultados gerados pela tradução.

4.1 COMPILANDO O TRADUTOR

Para compilar os seguintes comando devem ser digitados:

```
flex translator.l
```

```
bison -d translator.y
```

```
gcc -o $@ translator.tab.c lex.yy.c -ll
```

Para facilitar e agilizar o processo de testes, um makefile foi feito:

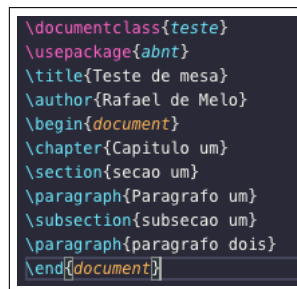
```
make translator
```

Para rodar o tradutor devemos digitar:

```
.translator < nome - do - arquivo.tex
```

4.2 ARQUIVOS BEM-SUCEDIDOS

Segue abaixo exemplos de arquivos bem-sucedidos na tradução, ou seja, que não obtiveram erros na sintaxe.



```
\documentclass{teste}
\usepackage{abnt}
\title{Teste de mesa}
\author{Rafael de Melo}
\begin{document}
\chapter{Capítulo um}
\section{secao um}
\paragraph{Parágrafo um}
\subsection{subsecao um}
\paragraph{parágrafo dois}
\end{document}
```

Figura 4 – Primeiro teste

```

\documentclass{teste}
\usepackage{abnt}
\title{Teste de mesa}
\author{Rafael de Melo}
\begin{document}
\chapter{Capitulo um}
\section{secao um}
\paragraph{Paragrafo um}
\subsection{subsecao um}
\paragraph{paragrafo dois}
\subsection{subsecao 2}
\bf{Exemplo de texto}
\paragraph{Este e um paragrafo da subsecao}
\begin{itemize}
\item{exemplo de item}
\end{itemize}
\begin{enumerate}
\item{exemplo de item}
\end{enumerate}
\end{document}

```

Figura 5 – Segundo teste

```

\documentclass{teste}
\usepackage{abnt}
\title{Teste de mesa}
\author{Rafael de Melo}
\begin{document}
\chapter{Capitulo um}
\paragraph{Lore ipsum da bla bla bla}
\chapter{Capitulo da largatixa}
\section{teste de secao}
\bf{BF}
\paragraph{ }
\underline{Underline}
\paragraph{ }
\it{It a coisa}
\end{document}

```

Figura 6 – Terceiro teste

```

\documentclass{teste}
\usepackage{abnt}
\title{Teste de mesa}
\author{Rafael de Melo}
\begin{document}
\chapter{Capitulo um}
\paragraph{Lore ipsum da bla bla bla}
\chapter{Capitulo da largatixa}
\section{teste de secao}
\paragraph{Um paragrafo}
\subsection{subsecao}
\begin{itemize}
\item{Item 1}
\item{Item 2}
\end{itemize}
\begin{enumerate}
\item{Na minha terra tem palmeiras}
\item{onde canta o sabia}
\end{enumerate}
\end{document}

```

Figura 7 – Quarto teste

4.3 ALGUNS RESULTADOS

Para melhor visualizar o resultado gerado, o arquivo .md foi utilizado no site downliverpreview.com/, dando uma prévia de que seria o arquivo em seu formato web.



Figura 8 – Primeiro resultado

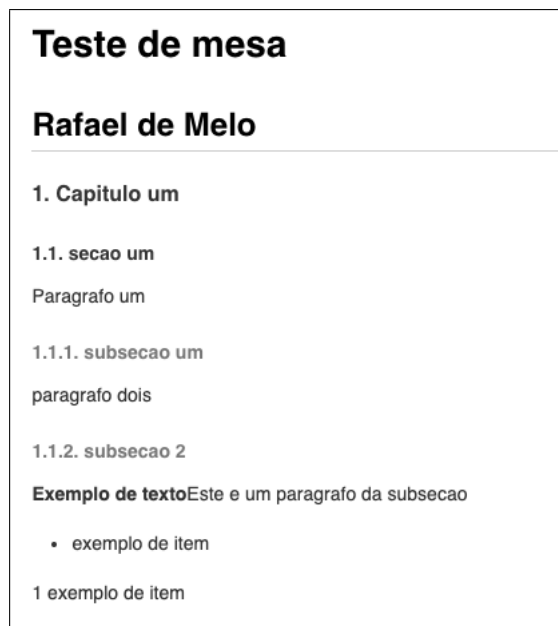


Figura 9 – Segundo resultado

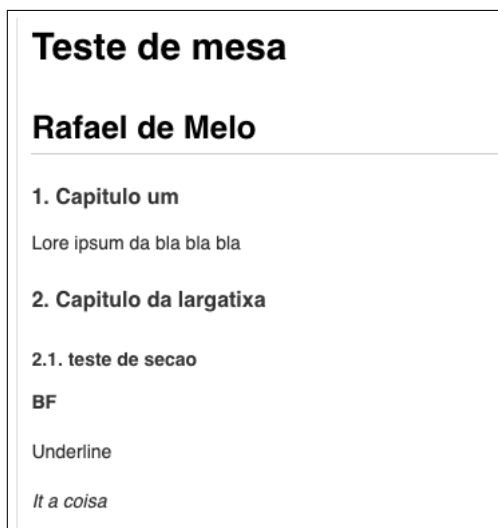


Figura 10 – Terceiro resultado

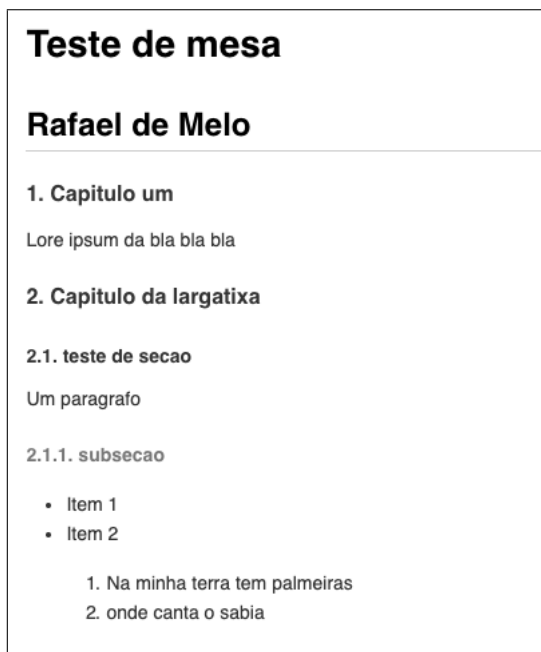


Figura 11 – Quarto resultado

4.4 CASOS EM QUE A TRADUÇÃO FOI MAL-SUCEDIDA

Por conta da gramática, nem todas as entradas serão aceitas. Alguns exemplos de quando isso acontece são:

- Estilos de texto dentro de parágrafos;
- Quando mais de um pacote é inserido;
- Falta de delimitadores;

- Arquivo escrito com tabulação;
- Capítulo contendo um parágrafo e, em seguida, encerrar o documento;

5 CONCLUSÃO

Algumas tarefas podem ser tediosas e maçantes. Entre elas está o trabalho de traduzir linguagens, para isso um tradutor de .tex para .md foi realizado como objetivo deste trabalho.

Trabalhar com o Flex e o Bison não é simples - uma vez que são bem sensíveis e algo fora da gramática não é aceito pelo programa. Muitas dificuldades foram encontradas para a realização (o próprio funcionamento do Bison foi complicado de entender), contudo referências foram o suficiente para sanar todas as dúvidas.

REFERÊNCIAS

LATEX – A document preparation system. 2019. Disponível em: <<https://www.latex-project.org/>>.

LEVINE, J. R. **flex bison**. [S.l.]: O'Reilly, 2009.

MASTERING Markdown. Disponível em: <<https://guides.github.com/features/mastering-markdown/>>.

MOGENSEN, T. Ægidius. **Basics of Compiler Design**. [S.l.]: lulu.com., 2010. ISBN 978-87-993154-0-6.