

Processamento de Linguagens e Compiladores
Pré-processador para L^AT_EX
Relatório de Desenvolvimento

André Sá (76361)

João Bastos (47419)

Pedro Sá (78164)

14 de Outubro de 2018

Resumo

O trabalho apresentado consiste na criação de um facilitador à criação e edição de documentos em \LaTeX recorrendo a uma linguagem mais simplificada e um pré-processador que faz a correspondente "tradução" para \LaTeX .

Conteúdo

1	Introdução	3
1.1	Descrição informal do problema	3
1.2	Estrutura do Relatório	3
2	Linguagem Simplificada	4
2.1	Acentos e Cedilhas	4
2.2	Cabeçalhos e Preâmbulos	4
2.2.1	Tipo de Documento	4
2.2.2	Título e Autor	5
2.2.3	Índices	5
2.2.4	Secções e Sub-secções	5
2.3	Texto formatado	5
2.3.1	Itálico	5
2.3.2	Bold	5
2.4	Texto não formatado	5
2.5	Listas	6
2.5.1	Itens	6
2.5.2	Descritivas	6
2.5.3	Numeradas	6
2.6	Macros	6
3	Concepção/desenho da Resolução	7
3.1	Início e Fim do documento	7
3.2	Conteúdo do Documento	7
3.3	Bold e Italic	8
3.4	Verbatim	8
3.5	Listas	9
3.6	Secções e Sub-secções	9
4	Concepção/desenho da Resolução	10
4.1	Início e Fim do documento	10
4.2	Conteúdo do Documento	10
4.3	Bold e Italic	11
4.4	Verbatim	11
4.5	Listas	12
4.6	Secções e Sub-secções	12
4.7	Estruturas de Dados	13

4.8	Algoritmos	13
5	Codificação e Testes	14
5.1	Testes realizados e Resultados	14
5.1.1	Doctype, Titl e Autor	14
5.1.2	Caracteres Especiais	14
5.1.3	Lista Numeradas	15
5.1.4	Lista de Itens	15
5.1.5	Lista Descritivas	15
5.1.6	Verbatim	15
5.1.7	Níveis de secção	15
6	Conclusão	17
A	Código do Programa	18

Capítulo 1

Introdução

1.1 Descrição informal do problema

O \LaTeX é um sistema tipográfico, bastante útil e completo, para a produção de todo o tipo de documentos. Porém, a edição do documento pode-se tornar morosa devido ao peso das marcas (comandos do \LaTeX) necessárias para anotar o texto.

O trabalho realizado teve como premissa precisamente facilitar a escrita de documentos em \LaTeX simplificando certas anotações de forma, conteúdo ou formato. Assim, foi criada uma linguagem mais simples baseada na ferramenta PPP¹. De seguida recorrendo à ferramenta Flex, criamos um processador que transforme a notação referida em \LaTeX .

1.2 Estrutura do Relatório

Neste relatório começamos com a Introdução (Capítulo 1) onde, de uma forma breve tentamos descrever o enquadramento do documento e a estrutura do mesmo.

No capítulo 2 explicamos em que consiste a Linguagem Simplificada e no capítulo seguinte (Capítulo 3) a forma como desenvolvemos a ferramenta que utilizará as regras dessa linguagem para criar o ficheiro \LaTeX .

O Capítulo 4 será composto pelos testes executados e no Capítulo 5 a Conclusão onde fazemos uma análise ao trabalho executado.

Por último o documento tem o apêndice onde foi introduzido o código fonte do ficheiro Lex.

¹PPP (Ptext PreProcessor) - <http://www4.di.uminho.pt/~jcr/AULAS/plc2008/tp1/ppp.html>

Capítulo 2

Linguagem Simplificada

Neste capítulo indicaremos a sintaxe definida nesta **Linguagem Simplificada**, doravante representada por **LS**, para os diferentes aspectos. Em cada caso iremos apresentar na forma de tabelas a sintaxe na **LS** e a correspondência em L^AT_EX.

2.1 Acentos e Cedilhas

Esta será a maior mais-valia desta ferramenta devido à acentuação e cedilhas presentes na língua portuguesa que obrigavam em L^AT_EX à constante notação sobre palavras travando a fluidez na escrita.

LS	L ^A T _E X
á	\'a
à	\'a
ã	\~a
â	\^a
é	\'e
ê	\^e
í	\{i}
Í	\'I
ó	\'o
õ	\~o
ô	\^o
ú	\'u
ü	\"u
ç	\c{c}
Ç	\c{C}

2.2 Cabeçalhos e Preâmbulos

2.2.1 Tipo de Documento

Ao iniciar o documento, o L^AT_EX necessita saber qual o layout mediante o tipo de documento que pretendemos. Na **LS** usaremos a seguinte sintaxe sendo o comando introduzido no início da linha:

LS	L ^A T _E X
=doctype=[tipo_de_documento]	\documentclass{tipo_de_documento}

Foi definido que o corpo do documento só será iniciado quando for encontrada a primeira linha em branco pelo que antes disso cabe ao utilizador começar o ficheiro com introdução do tipo de documento, título e autores.

2.2.2 Título e Autor

O título e autores do documento são escritos no início de linha com a seguinte sintaxe:

<i>LS</i>	\LaTeX
=title=[título]	\title{título}
=author=[autor1 \and autor2 \and ...]	\author{autor1 \and autor2 \and ...}

2.2.3 Índices

O índice é gerado no local onde o respectivo comando é introduzido sendo possível definir o local no documento onde o pretendemos, no entanto de momento apenas é possível gerar um índice por documento. O comando deverá ser introduzido no início da linha. A sintaxe é a seguinte:

<i>LS</i>	\LaTeX
=indice=	\tableofcontents

2.2.4 Secções e Sub-secções

As secções e os diversos níveis de sub-secções são introduzidos iniciando uma linha com os "=" seguido do nome da secção ou subsecção, sendo a sua numeração gerada automaticamente. O nível da secção corresponde ao número de "="'s introduzidos.

Nível	<i>LS</i>	\LaTeX
1	=capítulo	\chapter{capítulo}
2	==secção	\section{secção}
3	===sub-secção	\subsection{sub-secção}
4	====sub-subsecção	\subsubsection{sub-subsecção}
5	=====parágrafo	\paragraph{parágrafo}

2.3 Texto formatado

Qualquer pedaço de texto pode ser formatado em Itálico ou Bold. Para tal colocamos o texto que pretendemos formatar dentro dos respectivos comandos tal como exemplificamos de seguida.

2.3.1 Itálico

<i>LS</i>	\LaTeX
\it{texto}	\textit{texto}

2.3.2 Bold

<i>LS</i>	\LaTeX
\bf{texto}	\textbf{texto}

2.4 Texto não formatado

Para quando é necessário ter como input exactamente o que foi escrito o \LaTeX tem o comando Verbatim que optamos por simplificar da seguinte forma:

<i>LS</i>	\LaTeX
\bverb texto \everb	\begin{verbatim} texto \end{verbatim}

2.5 Listas

Existem três tipos de listas suportadas pela **LS**. Cada um destes comandos são iniciados por ":" no final de uma linha. Sendo que nos últimos

2.5.1 Itens

Utilizaremos uma descrição por itens para descrever este comando:

- Iniciado por ":" no final de uma linha;
- Cada item será iniciado no início de cada linha por "." seguido de um caracter diferente de "[" ou dígito;
- Termina com o caracter "#".

2.5.2 Descritivas

Utilizaremos uma lista descritiva para descrever este comando:

Primeiro Iniciado por ":" no final de uma linha;

Segundo Cada item será iniciado no início de cada linha por "." seguido de uma ou mais palavras entre "[" e "];

Terceiro Termina com o caracter "#".

2.5.3 Numeradas

Utilizaremos uma lista numerada para descrever este comando:

1. Iniciado por ":" no final de uma linha;
2. Cada item será iniciado no início de cada linha por "." seguido de um dígito;
3. Termina com o caracter "#".

2.6 Macros

Foram definidas duas macros relacionadas com indicador de ordem em feminino e masculino. Bastará escrever o que se pretende como input que a tradução em L^AT_EX será feita.

LS	L ^A T _E X
8 ^o	8\textordmasculine
11 ^a	11\textordfeminine

Capítulo 3

Concepção/desenho da Resolução

Neste capítulo iremos descrever a concepção/desenho do analisador léxico iniciando cada secção com um pequeno resumo seguido do código referido.

3.1 Início e Fim do documento

No ficheiro flex elaborado optamos por manipulação directa da stack de estados utilizando as funções `yy_push_state` e `yy_pop_state`. Para tal introduzimos `%option stack` para além dos estados por nós definidos, como se pode verificar na Figura 4.9.

```
%}  
  
%option stack  
  
%x AUTHOR  
%x BODY  
%x BODY_HEADER  
%x DOCTYPE  
%x LIST_ENUMERATE  
%x LIST_ITEMIZE  
%x LIST_DESCRIPTION  
%x TITLE  
%x VERBATIM  
%x BOLD  
%x ITALIC  
  
%%
```

Figura 3.1: Estados

Como podemos verificar pela Figura 4.10, ao iniciar, o analisador léxico encontrará a indicação do tipo de documento, título e autores e só após encontrar a primeira linha em branco irá passar para o estado BODY que vai conter todo o conteúdo do documento. Encontrando o EOF é introduzido o comando para o fim do documento em `LATEX` (Figura 4.11).

3.2 Conteúdo do Documento

Já dentro do estado BODY, ou seja, dentro do conteúdo do documento, o analisador léxico irá verificar se encontra correspondência para as restantes anotações de forma, conteúdo ou formato que iremos apresentar de seguida. Caso tenha correspondência é encaminhado para o estado devido (Figura 4.12). Caso no início de uma linha encontre `"=índice="` será impresso o comando `\tableofcontents` que introduz o índice nesse mesmo local.

```

^\\s*$ {
    /*
     * An empty line starts the body of the document,
     * so all the headers must come before any empty line
     */
    printf("\\begin{document}");
    stpush(BODY);
}

^=author=\\[ stpush(AUTHOR);
^=doctype=\\[ stpush(DOCTYPE);
^=title=\\[ stpush(TITLE);

<AUTHOR>[^\\]\\n+ printf("\\author{%s}", yytext);
<DOCTYPE>[^\\]\\n+ printf("\\documentclass{%s}", yytext);
<TITLE>[^\\]\\n+ printf("\\title{%s}", yytext);

<TITLE,AUTHOR,DOCTYPE>\\]$ stpop();

```

Figura 3.2: Estados DOCTYPE, TITLE, AUTHOR e BODY

```

<<EOF>> {
    printf("\\end{document}");
    /* Se nao acabarmos a funcao, o programa fica em loop */
    return 0;
}

```

Figura 3.3: Fim do documento.

```

<BODY>^=indice= { printf("\\tableofcontents"); }
<BODY>\\bverb[ \\n\\t] { printf("\\begin{verbatim}"); stpush(VERBATIM); }
<BODY>^#+ { nh = yyleng; stpush(BODY_HEADER); }
<BODY>:/\\n\\. { printf("\\begin{itemize}"); stpush(LIST_ITEMIZE); }
<BODY>:/\\n\\. [0-9 { printf("\\begin{enumerate}"); stpush(LIST_ENUMERATE); }
<BODY>:/\\n\\. \\[ { printf("\\begin{description}"); stpush(LIST_DESCRIPTION); }
<BODY>\\bfbf{ { printf("\\textbf{"); stpush(BOLD); }
<BODY>\\itit{ { printf("\\textit{"); stpush(ITALIC); }

<BODY>\\. |\\n ECHO;

```

Figura 3.4: Estado BODY e encaminhamento para os diferentes estados.

3.3 Bold e Italic

Nestes 2 casos é feita uma simples substituição dos comandos e escrito tudo o que se encontra entre chavetas (Figura 4.5).

```

<ITALIC>\\bf\\ { printf("\\textbf{"); stpush(BOLD); }
<BOLD>\\it\\ { printf("\\textit{"); stpush(ITALIC); }
<BOLD,ITALIC>[^\\("\\bf"|"it")]*\\ { ECHO; stpop(); }

```

Figura 3.5: BOLD e ITALIC.

3.4 Verbatim

Depois de o analisador léxico encontrar o comando de inicialização substitui pelo correspondente em $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ e faz ECHO até encontrar o comando de finalização que também substitui pelo correspondente (Figura 4.6).

```

<VERBATIM>\\everb { printf("\\end{verbatim}");
                  stpop(); }
<VERBATIM>\\. |\\n ECHO;

```

Figura 3.6: VERBATIM.

3.5 Listas

No estado BODY é encontrada a condição de inicialização de cada um dos três tipos de lista e impresso o respectivo comando de início. Daí o analisador léxico entra no respectivo estado e adiciona os itens no formato pretendido. Ao encontrar o ”#” imprime o comando de fecho do comando. Dentro destes três estados também se procura correspondência com os comandos de formatação itálico e bold de forma a poder ser aplicado também em listas (Figura 4.7).

```
<LIST_ITEMIZE>^\s*      { printf("\\item "); }
<LIST_ENUMERATE>^\s*[0-9]+\s* { printf("\\item "); }
<LIST_DESCRIPTION>^\s*[{
{ printf("\\item {"); }

<LIST_ENUMERATE,LIST_ITEMIZE,LIST_DESCRIPTION>,\n      ECHO;
<LIST_ENUMERATE,LIST_ITEMIZE,LIST_DESCRIPTION>\\bf{    { printf("\\textbf{"); stpush(BOLD); }
<LIST_ENUMERATE,LIST_ITEMIZE,LIST_DESCRIPTION>\\it{    { printf("\\textit{"); stpush(ITALIC); }

<LIST_ENUMERATE>^\s*=\s*$      { printf("\\end{enumerate}"); stpop(); }
<LIST_ITEMIZE>^\s*=\s*$      { printf("\\end{itemize}"); stpop(); }
<LIST_DESCRIPTION>^\s*=\s*$  { printf("\\end{description}"); stpop(); }
```

Figura 3.7: Listas.

3.6 Secções e Sub-secções

Existe um limite para os níveis de secções que podem ser criadas. Para resolver esse problema criamos uma função que, com auxílio de um apontador para um array de Strings, permite saber que nível pretendemos para imprimir o comando correcto e quando atinge o seu limite (Figura 4.8).

```
/* Supported LaTeX header tags */
static const char * chaps[] = {
    "chapter",
    "section",
    "subsection",
    "subsubsection",
    "paragraph",
    "subparagraph",
};

/* Number of supported LaTeX header tags */
static const unsigned nc = sizeof(chaps) / sizeof(*chaps);

/* Header level, 0-nc */
int nh = 0;
%}
```

(a)

```
<BODY_HEADER>[^#\n]*$ {
    /* Make sure `nh` is in bounds */
    if (((unsigned) --nh) >= nc)
        nh = nc - 1;

    printf("\\%s{", chaps[nh], yytext);
    nh = 0;
    stpop();
}
```

(b)

Figura 3.8: BODY_HEADER

Capítulo 4

Concepção/desenho da Resolução

Neste capítulo iremos descrever a concepção/desenho do analisador léxico iniciando cada secção com um pequeno resumo seguido do código referido.

4.1 Início e Fim do documento

No ficheiro flex elaborado optamos por manipulação directa da stack de estados utilizando as funções `yy_push_state` e `yy_pop_state`. Para tal introduzimos `%option stack` para além dos estados por nós definidos, como se pode verificar na Figura 4.9.

```
%}  
  
%option stack  
  
%x AUTHOR  
%x BODY  
%x BODY_HEADER  
%x DOCTYPE  
%x LIST_ENUMERATE  
%x LIST_ITEMIZE  
%x LIST_DESCRIPTION  
%x TITLE  
%x VERBATIM  
%x BOLD  
%x ITALIC  
  
%%
```

Figura 4.1: Estados

Como podemos verificar pela Figura 4.10, ao iniciar, o analisador léxico encontrará a indicação do tipo de documento, título e autores e só após encontrar a primeira linha em branco irá passar para o estado BODY que vai conter todo o conteúdo do documento. Encontrando o EOF é introduzido o comando para o fim do documento em `LATEX` (Figura 4.11).

4.2 Conteúdo do Documento

Já dentro do estado BODY, ou seja, dentro do conteúdo do documento, o analisador léxico irá verificar se encontra correspondência para as restantes anotações de forma, conteúdo ou formato que iremos apresentar de seguida. Caso tenha correspondência é encaminhado para o estado devido (Figura 4.12). Caso no início de uma linha encontre `"=índice="` será impresso o comando `\tableofcontents` que introduz o índice nesse mesmo local.

```

^\\s*$ {
    /*
     * An empty line starts the body of the document,
     * so all the headers must come before any empty line
     */
    printf("\\begin{document}");
    stpush(BODY);
}

^=author=\\[ stpush(AUTHOR);
^=doctype=\\[ stpush(DOCTYPE);
^=title=\\[ stpush(TITLE);

<AUTHOR>[^\\]\\n+ printf("\\author{%s}", yytext);
<DOCTYPE>[^\\]\\n+ printf("\\documentclass{%s}", yytext);
<TITLE>[^\\]\\n+ printf("\\title{%s}", yytext);

<TITLE,AUTHOR,DOCTYPE>\\]$ stpop();

```

Figura 4.2: Estados DOCTYPE, TITLE, AUTHOR e BODY

```

<<EOF>> {
    printf("\\end{document}");
    /* Se nao acabarmos a funcao, o programa fica em loop */
    return 0;
}

```

Figura 4.3: Fim do documento.

```

<BODY>^=indice= { printf("\\tableofcontents"); }
<BODY>\\bverb[ \\n\\t] { printf("\\begin{verbatim}"); stpush(VERBATIM); }
<BODY>^#+ { nh = yyleng; stpush(BODY_HEADER); }
<BODY>:/\\n\\. { printf("\\begin{itemize}"); stpush(LIST_ITEMIZE); }
<BODY>:/\\n\\. [0-9 { printf("\\begin{enumerate}"); stpush(LIST_ENUMERATE); }
<BODY>:/\\n\\. \\[ { printf("\\begin{description}"); stpush(LIST_DESCRIPTION); }
<BODY>\\bfbf{ { printf("\\textbf{"); stpush(BOLD); }
<BODY>\\itit{ { printf("\\textit{"); stpush(ITALIC); }

<BODY>\\. |\\n ECHO;

```

Figura 4.4: Estado BODY e encaminhamento para os diferentes estados.

4.3 Bold e Italic

Nestes 2 casos é feita uma simples substituição dos comandos e escrito tudo o que se encontra entre chavetas (Figura 4.5).

```

<ITALIC>\\bfbf{ { printf("\\textbf{"); stpush(BOLD); }
<BOLD>\\itit{ { printf("\\textit{"); stpush(ITALIC); }
<BOLD,ITALIC>[^\\("\\bfbf"|"itit")]*\\} { ECHO; stpop(); }

```

Figura 4.5: BOLD e ITALIC.

4.4 Verbatim

Depois de o analisador léxico encontrar o comando de inicialização substitui pelo correspondente em $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ e faz ECHO até encontrar o comando de finalização que também substitui pelo correspondente (Figura 4.6).

```

<VERBATIM>\\everb { printf("\\end{verbatim}");
                  stpop(); }
<VERBATIM>\\. |\\n ECHO;

```

Figura 4.6: VERBATIM.

4.5 Listas

No estado BODY é encontrada a condição de inicialização de cada um dos três tipos de lista e impresso o respectivo comando de início. Daí o analisador léxico entra no respectivo estado e adiciona os itens no formato pretendido. Ao encontrar o “#” imprime o comando de fecho do comando. Dentro destes três estados também se procura correspondência com os comandos de formatação itálico e bold de forma a poder ser aplicado também em listas (Figura 4.7).

```
<LIST_ITEMIZE>^\s*      { printf("\\item "); }
<LIST_ENUMERATE>^\s*[0-9]+\s* { printf("\\item "); }
<LIST_DESCRIPTION>^\s*  { printf("\\item ["); }

<LIST_ENUMERATE,LIST_ITEMIZE,LIST_DESCRIPTION>.\n      ECHO;
<LIST_ENUMERATE,LIST_ITEMIZE,LIST_DESCRIPTION>\\bf\\{   { printf("\\textbf{"); stpush(BOLD); }
<LIST_ENUMERATE,LIST_ITEMIZE,LIST_DESCRIPTION>\\it\\{   { printf("\\textit{"); stpush(ITALIC); }

<LIST_ENUMERATE>^\s*=\s*$ { printf("\\end{enumerate}"); stpop(); }
<LIST_ITEMIZE>^\s*=\s*$  { printf("\\end{itemize}"); stpop(); }
<LIST_DESCRIPTION>^\s*=\s*$ { printf("\\end{description}"); stpop(); }
```

Figura 4.7: Listas.

4.6 Secções e Sub-secções

Existe um limite para os níveis de secções que podem ser criadas. Para resolver esse problema criamos uma função que, com auxílio de um apontador para um array de Strings, permite saber que nível pretendemos para imprimir o comando correcto e quando atinge o seu limite (Figura 4.8).

```
/* Supported LaTeX header tags */
static const char * chaps[] = {
    "chapter",
    "section",
    "subsection",
    "subsubsection",
    "paragraph",
    "subparagraph",
};

/* Number of supported LaTeX header tags */
static const unsigned nc = sizeof(chaps) / sizeof(*chaps);

/* Header level, 0-nc */
int nh = 0;

%}
```

(a)

```
<BODY_HEADER>[^#\n]*$ {
    /* Make sure `nh` is in bounds */
    if ((unsigned) --nh >= nc)
        nh = nc - 1;

    printf("\\s{ %s}", chaps[nh], yytext);
    nh = 0;
    stpop();
}
```

(b)

Figura 4.8: BODY_HEADER

Início e Fim do documento No ficheiro flex elaborado optamos por manipulação directa da stack de estados utilizando as funções `yy_push_state` e `yy_pop_state`. Para tal introduzimos “%option stack” para além dos estados por nós definidos, como se pode verificar na Figura 4.9.

Como podemos verificar pela Figura 4.10, ao iniciar, o analisador léxico encontrará a indicação do tipo de documento, título e autores e só após encontrar a primeira linha em branco irá passar para o estado BODY que vai conter todo o conteúdo do documento. Encontrando o EOF é introduzido o comando para o fim do documento em \LaTeX (Figura 4.11).

```
%}

%option stack

%x AUTHOR
%x BODY
%x BODY_HEADER
%x DOCTYPE
%x LIST_ENUMERATE
%x LIST_ITEMIZE
%x LIST_DESCRIPTION
%x TITLE
%x VERBATIM
%x BOLD
```

```

^s*$ {
    /*
     * An empty line starts the body of the document,
     * so all the headers must come before any empty line
     */
    printf("\\begin{document}");
    stpush(BODY);
}

^=author=[ stpush(AUTHOR);
^=doctype=[ stpush(DOCTYPE);
^=title=[ stpush(TITLE);

<AUTHOR>[^\\n]+ printf("\\author{%s}", yytext);
<DOCTYPE>[^\\n]+ printf("\\documentclass{%s}", yytext);
<TITLE>[^\\n]+ printf("\\title{%s}", yytext);

<TITLE,AUTHOR,DOCTYPE>\\]$ stpop();

```

Figura 4.10: Estados DOCTYPE, TITLE, AUTHOR e BODY

Conteúdo do Documento Já dentro do estado BODY, ou seja, dentro do conteúdo do documento, o analisador léxico irá verificar se encontra correspondência para as restantes anotações de forma, conteúdo ou formato que iremos apresentar de seguida. Caso tenha correspondência é encaminhado para o estado devido (Figura 4.12).

4.7 Estruturas de Dados

4.8 Algoritmos

```

<<EOF>> {
    printf("\\end{document}");
    /* Se nao acabarmos a funcao, o programa fica em loop */
    return 0;
}

```

Figura 4.11: Fim do documento.

```

<BODY>^=indice= { printf("\\tableofcontents"); }
<BODY>\\bverb[ \\n\\t] { printf("\\begin{verbatim}"); stpush(VERBATIM); }
<BODY>^=#+ { nh = yyleng; stpush(BODY_HEADER); }
<BODY>:/\\n\\. { printf("\\begin{itemize}"); stpush(LIST_ITEMIZE); }
<BODY>:/\\n\\. [0-9] { printf("\\begin{enumerate}"); stpush(LIST_ENUMERATE); }
<BODY>:/\\n\\. \\[ { printf("\\begin{description}"); stpush(LIST_DESCRIPTION); }
<BODY>\\bf\\{ { printf("\\textbf{"); stpush(BOLD); }
<BODY>\\it\\{ { printf("\\textit{"); stpush(ITALIC); }

<BODY>\\. |\\n ECHO;

```

Figura 4.12: Estado BODY e encaminhamento para os diferentes estados.

Capítulo 5

Codificação e Testes

5.1 Testes realizados e Resultados

Nesta secção encontra-se os resultados do nosso analisador léxico no seguinte formato:

- Subsecções que se referem a um tipo de conversão de *LS* para L^AT_EX;
- Cada subsecção é formado por duas imagens, onde a da esquerda é o input no do nosso analisador léxico em *LS* e a da direita é o output em L^AT_EX;

5.1.1 Doctype, Title e Autor

```
=doctype=[report]
=title=[Trabalho 1 (testes)]
=author=[André Sá \and João Bastos \and Pedro Sá]
```

(a)

```
\documentclass{report}
\title{Trabalho 1 (testes)}
\author{André Sá \and João Bastos \and Pedro Sá}
```

(b)

Figura 5.1: Conversão de cabeçalho e preâmbulo

5.1.2 Caracteres Especiais

```
área
àquele
vão
Lâmpada
ética
paciência
índice
Índice
Póvoa
verão
avô
saúde
acção
11º
89ª
```

(a)

```
\area
\aquele
\va-o
L\^a npada
\et\i ca
pac\i\^en cia
\i\^i ndice
\I ndice
P\o voa
ver\~a o
av\^o
sa\~u de
ac\c(c)\-ao
89\textordfem\line
```

(b)

Figura 5.2: Conversão de cabeçalho e preâmbulo

5.1.3 Lista Numeradas

```
Teste de lista enumerada:  
.1 Com \bf{bold} dentro  
.2 ponto 2  
.3 com \it{texto} em itálico  
.4 outros  
=
```

(a)

```
Teste de lista enumerada\begin{enumerate}  
\item Com \textbf{bold} dentro  
\item ponto 2  
\item com \textit{texto} em itálico  
\item outros  
\end{enumerate}
```

(b)

Figura 5.3: Conversão de lista numeradas

5.1.4 Lista de Itens

```
Teste de lista de itens:  
. pão  
. salada  
. frango  
=
```

(a)

```
Teste de lista de itens\begin{itemize}  
\item pão  
\item salada  
\item frango  
\end{itemize}
```

(b)

Figura 5.4: Conversão de lista de itens

5.1.5 Lista Descritivas

```
Teste de lista descritiva:  
.[primeiro] item1  
.[segundo] item2  
.[outros] item3  
=
```

(a)

```
Teste de lista descritiva\begin{description}  
\item [primeiro] item1  
\item [segundo] item2  
\item [outros] item3  
\end{description}
```

(b)

Figura 5.5: Conversão de lista descritivas

5.1.6 Verbatim

```
\bverb  
sdasdas  
tudo o que estiver cá dentro:  
.1 fica igual  
=  
\everb
```

(a)

```
\begin{verbatim}sdasdas  
tudo o que estiver cá dentro:  
.1 fica igual  
=  
\end{verbatim}
```

(b)

Figura 5.6: Conversão de sintaxe verbatim

5.1.7 Níveis de secção

```

#Primeiro Capitulo
##Secção

Teste de lista enumerada:
.1 Com \bf{bold} dentro
.2 ponto 2
.3 com \it{texto} em itálico
.4 outros
=

###Sub-secção

Teste de lista de itens:
. pão
. salada
. frango
=

####Sub-subsecção

Teste de lista descritiva:
.[primeiro] item1
.[segundo] item2
.[outros] item3
=

#####Parágrafo

#####Sub-parágrafo

```

(a)

```

\chapter{Primeiro Capitulo}
\section{Secção}

Teste de lista enumerada\begin{enumerate}
\item Com \textbf{bold} dentro
\item ponto 2
\item com \textit{texto} em itálico
\item outros
\end{enumerate}

\subsection{Sub-secção}

Teste de lista de itens\begin{itemize}
\item pão
\item salada
\item frango
\end{itemize}

\subsubsection{Sub-subsecção}

Teste de lista descritiva\begin{description}
\item [primeiro] item1
\item [segundo] item2
\item [outros] item3
\end{description}

\paragraph{Parágrafo}

\subparagraph{Sub-parágrafo}

```

(b)

Figura 5.7: Conversão de níveis de secção

Capítulo 6

Conclusão

Este projecto mostrou-nos o poder que um analisador léxico tem e que nos pode ser útil em várias situações do dia-a-dia, como neste caso do \LaTeX que é uma linguagem bastante pesada em termos de sintaxe.

A nossa *LS* aborda as sintaxes mais usuais de \LaTeX mas tendo em conta a abrangência de composição de vários tipos de documentos e suporte a várias formas e estruturas de escrita/formatação, o nosso projecto poderá continuar ser desenvolvido para abranger tais sintaxes, e está codificado de forma a ser fácil essa introdução de sintaxe, devido à utilização da *stack*.

Apêndice A

Código do Programa