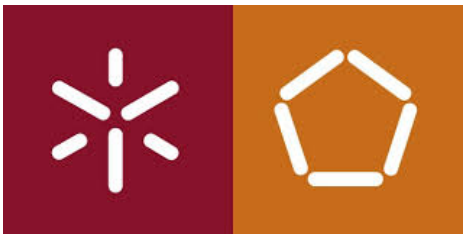

Processamento de Linguagens - TP2 - Processador para LaTex

Gonçalo Camaz A76861 | Jorge Oliveira A78660 | José Ferreira A78452

(7 de Maio de 2018)

Resumo

No presente relatório vamos apresentar uma linguegem mais leve do que o latex e o programa que trasformar a mesma em texto latex. Apresentamos a nossa linguagem e as suas caraterísticas, a forma de colocar o texto com ênfase(negrito, itálico e sublinhado) e também de listas, tabelas e vários níveis de "cabeçalhos". Depois apresentamos também como realizamos o processo de transformação da nossa linguagem para latex com o auxílio da ferramenta *Flex* e das suas caraterísticas bastante poderosas, nomeadamente as condições de contexto.

Conteúdo

Lista de Figuras

1 Introdução

O presente relatório serve para descrever todo o processo para a criação de uma linguagem, que tem como objetivo facilitar a escrita de texto com a sintaxe do LaTeX.

O nosso principal objetivo era conseguir realizar todos os desafios propostos no enunciado. Só após esta base estar pronta é que pensaríamos em realizar algo mais, que poderia passar, pela abstração de tabelas ou então o encadeamento de marcas.

Tal tarefa deve ser efetuada usando o *flex*. Este é também um outro objetivo, tentar usar ao máximo o *flex* e não cair no erro de tentar resolver o problema usando código C.

2 Estrutura do Relatório

No capítulo 3 será feita uma apresentação do problema que estamos a tratar e ainda a linguagem que nós construímos. Desta forma o leitor terá um melhor enquadramento sobre o porquê de realizarmos este trabalho e ainda a constituição da linguagem.

No capítulo 3.3 apresentaremos a forma como abordamos o problema em termos algorítmicos e no capítulo 4 serão apresentados os resultados obtidos pelo nosso programa. O capítulo 5 concluí o relatório com a apresentação da conclusão.

3 Análise e Especificação

3.1 Descrição informal do problema

O latex é bastante utilizado pela comunidade científica e matemática mundial devido à sua grande qualidade tipográfica. O LaTeX fornece um conjunto de macros alto-nível que tornam mais fácil e rápida a produção de documentos em TeX e é utilizado para produzir todo o tipo de documentos.

O principal objetivo do LaTeX é que o autor se possa distanciar da apresentação visual do documento e assim concentrar-se no seu conteúdo. Possui variadíssimas formas de lidar com bibliografias, citações, formatos de páginas, referências e tudo mais que não seja relacionado com conteúdo do documento em si.

Porém, é necessário, editar o próprio texto para colocar com a sintaxe do LaTeX, um ato que por vezes se torna maçador. O ideal seria termos um conjunto de anotações mais simples e mais intuitivas na sua escrita (o ideal seria até que estas anotações fossem o máximo independente da escrita em LaTeX), e que depois de executar um pré-processamento estas anotações mais simples se transformassem em anotações de acordo com o LaTeX. Um caso deste é o *MarkDown*.

O nosso objetivo, com a conceção deste trabalho, é criar umas anotações mais

simples para a escrita de documentos em LaTeX. De modo que, quem esteja a escrever o seu texto, se foque essencialmente no texto e se abstraia mais de qual comando tem de colocar para que o seu texto fique editado de uma forma correta.

3.2 Apresentação da Linguagem

Como já foi referido anteriormente, o LaTeX tem uma vasta biblioteca para a edição correta do texto, mas nós focámo-nos nos marcadores que possam ser mais utilizados. A escrita em negrito, itálico ou sublinhado e ainda vários níveis de títulos (*chapter, section, subsection, subsubsection*). Vários tipos de lista, numeradas, não-numeradas e dicionário. E ainda o uso de tabelas.

De seguida passamos a apresentar a linguagem utilizada:

O símbolo $ indica o nível da *section*. Um $ indica que é uma *section*, $$ indica *subsection* e $$$ *subsubsection*.

palavra - Caso uma sequência de carateres esteja entre dois asteriscos então é uma indicação de que deve ser escrita em negrito. Para esta anotação, inspirámo-nos no que uma aplicação bem conhecida já faz. O Facebook, caso coloquemos algo entre dois asteriscos, automaticamente aplica um efeito de negrito.

&palavra& - Caso uma sequência de carateres esteja entre dois & significa que essa sequência vai ser escrita em itálico. Para esta anotação, baseámo-nos na anterior, e para que fosse mais intuitivo colocamos o símbolo do E comercial.

palavra - Caso uma sequência de carateres esteja entre dois /underscores significa que essa sequência vai ser sublinhada. Para esta anotação, baseámo-nos mais uma vez nas anteriores, e utilizamos o _ já que da a ideia de que a palavra deve ser sublinhada ao utilizador.

Tratamento das Listas:

Os seguintes símbolos indicam o inicio de uma lista. Para tal, o utilizador deve colocar a seguinte sintaxe: #Opção# sendo que a Opção deve ser um carater dentro do seguinte conjunto: {O,N,D}.

#O# - Inicio de uma lista do tipo ordenada.

#N# - Inicio de uma lista do tipo não ordenada.

#D# - Inicio de uma lista do tipo dicionário.

Após a indicação do inicio da lista, seguem-se os vários elementos, que devem ser escritos também eles entre os símbolos cardinais.

Quando terminar a escrita dos vários elementos, deverá colocar o seguinte símbolo #FOpção# (Opção pertence {O,N,D}), que deverá corresponder ao inicio da lista que o utilizador iniciou.

Tratamento da Tabela: Para a escrita da tabela o utilizador deverá iniciar através do símbolo F. Porém as tabelas trazem muitas variantes para a sua formatação. O grupo decidiu focar-se em realizar uma tabela simples, onde permitimos que de uma forma intuitiva o utilizador decida se pretende centrar

a tabela, que coloque uma legenda para a mesma e ainda os vários elementos da tabela. De salientar que o utilizador deverá colocar a sintaxe do *begintabular*, já que traz variadíssimas opções e dessa forma, deixámos à escolha do utilizador quais as formatações que deseja.

#F# - Inicio de uma tabela.

>C - Inica que a tabela deve ser centrada (opcional)

>L legenda >FL - Adiciona uma legenda à tabela.

| - A barra serve para separar cada elemento por coluna. Para a mudança de linha deve indicar um #.

#FT# - Fim da tabela.

3.3 Concepção/desenho da Resolução

Passada a fase de especificação da linguagem, estávamos em condições de passar para a implementação da aplicação que transforma a mesma em linguagem *LaTex*. A primeira definição que realizamos foi quais seriam as condições de contexto que necessitamos para desenvolver a conversão. Como tínhamos diferentes parâmetros de conversões decidimos ter as seguintes condições de contexto:

- LISTA - esta condição de contexto corresponde a texto que se encontra num contexto de lista ordenada ou não-ordenada
- NEGRITO - corresponde a texto que se encontra num contexto de negrito
- ITALICO - corresponde a texto que se encontra num contexto de itálico
- SUBLINHADO - corresponde a texto que se encontra num contexto de sublinhado
- CABECALHO - corresponde a texto que se encontra num contexto de cabeçalho
- DICIONARIO - corresponde a texto que se encontra num contexto de lista do tipo dicionário
- TABELA - corresponde a texto que se encontra num contexto de tabela

Depois verificamos que necessitávamos de uma *stack* de condições de contexto pelo facto de que estas condições se poderiam aninhar umas nas outras (em certos casos), pelo que ativamos a opção *stack* que nos fornece essa funcionalidade. Começando pelo carater de escape que utilizamos sempre que encontrávamos texto com esse caractere (em qualquer contexto) seguido de um dos carateres definidos como especiais da linguagem imprimimos o texto, pois estava *escaped*.

```
<*>\\[*_&#$] {printf("%s",yytext);}
```

Sempre que em qualquer contexto encontrássemos também as expressões definidas como de início de alguma parte especial de texto então iniciamos essa parte e

introduzimos o contexto referente na *stack* através da operção **yy_push_state**. Antes de introduzir o contexto temos de imprimir o texto correspondente em *LaTex* para início da parte em questão. Exemplo do texto negrito:

```
<*>\*   {
        printf("\\textbf{");
        yy_push_state(NEGRITO);

        }
```

Exemplo do dicionário:

```
<*>#D# {
        printf("\\begin{description}");
        yy_push_state(DICIONARIO);
}
```

No entanto, em certos casos nem sempre se pode iniciar a parte correspondente, por exemplo as tabelas não podem ser iniciadas em nenhum contexto, apenas no inicial o que acontece também com os cabeçalhos. Daí nestes casos termos definido que apenas realizamos a operação quando este se encontra na condição inicial. Exemplo do cabeçalho:

```
$+  {
    int tam = strlen(yytext);
    switch(tam){
        case 1: printf("\\chapter{"); break;
        case 2: printf("\\section{"); break;
        case 3: printf("\\subsection{"); break;
        default: printf("\\subsubsection{");
    }
    yy_push_state(CABECALHO);
}
```

Exemplo da tabela:

```
#T# {
    printf("\\begin{table}");
    yy_push_state(TABELA);
}
```

Quando estamos num contexto e aparece a expressão que representa o fim do mesmo decidimos imprimir o caractere que corresponde ao fim do mesmo em *LaTex* e de seguida retiramos o contexto da *stack* através da operação **yy_pop_state**. Exemplo de negrito:

```
<NEGRITO>\* {
        printf("}");
```

```
        yy_pop_state();
    }
```

Exemplo de dicionário:

```
<DICIONARIO>#FD#    {
    printf("\\end{description}");
    yy_pop_state();
}
```

Exemplo da tabela:

```
<TABELA>#FT# { printf("\\end{table}"); yy_pop_state();}
```

Quando estamos num contexto de lista e identificamos a expressão que nos diz que estamos na presença de um novo item então temos de imprimir a expressão correspondente em *LaTex*:

```
 \item.

<LISTA>#    {

        printf("\\item");
}
```

Quando aparece a expressão do fim de lista ordenada ou não ordenada também temos de imprimir o texto correspondente. Neste caso temos também de sair do contexto atual realizando a operação **yy_pop_state**.

```
<LISTA>#FO# {
    printf("\\end{enumerate}");
    yy_pop_state();
}

<LISTA>#FN# {
    printf("\\end{itemize}");
    yy_pop_state();
}
```

O mesmo acontece para o caso do dicionário, sendo que neste temos de ter um caso especial. É que ao encontrar o início de um item este vem com a correspondente palavra entre parênteses retos e a definição a seguir pelo que temos de imprimir o texto:

```
 \item
```

e também a parte da palavra entre parêntese retos, que se encontra na variável **yytext**, mas inicia-se na posição 1 desse *array* de carateres visto que apanhamos também o # que corresponde ao início de um novo item.

```
<DICIONARIO>#\[[^#*_&]+\]   {

        printf("\\item%s",yytext+1);

}

<DICIONARIO>#FD#    {
    printf("\\end{description}");
    yy_pop_state();
}
```

Quando estamos num contexto de tabelas e nos aparece a expressão ">C" a mesma significa que é para centrar, pelo que imprimimos para o texto a expressão correspondente em *LaTex*: *begin centering*.

```
<TABELA>\>C {printf("\\centering");}
```

Da mesma forma para a legenda da tabela definimos como a expressão ">L" para imprimir a expressão correspondente em *LaTeX*.

```
<TABELA>\>L {printf("\\caption{");}
```

Precisamos no entanto de definir uma expressão para o fim da *caption* e para isso definimos a expressão ">FL" imprimindo o texto "}" visto que em *LaTex* a legenda aparece dentro de chavetas.

```
<TABELA>\>FL {printf("}");}
```

Depois como definimos que cada elemento da tabela é separado pelo caractere '|' temos de, ao identificar o mesmo num contexto de Tabela, imprimir o correspondente na linguagem desejada, o caractere '&'.

```
<TABELA>\| {printf("&");}
```

Definimos também o fim da linha de uma tabela como o caractere '#' e imprimimos o mesmo em *LaTex*, ou seja a expressão '
'.

```
<TABELA># {printf("\\\\");}
```

Identificamos ainda o fim da tabela como a expressão "#FT#" e decidimos imprimir a expressão que corresponde ao fim do mesmo em LaTex. De seguida retiramos o contexto da *stack* através da operação **yy_pop_state**.

```
<TABELA>#FT# { printf("\\end{table}"); yy_pop_state();}
```

De referir que apenas apresentamos expressões para "apanhar" os carateres especiais definidos para a linguagem, visto que nos restantes casos queremos deixar o texto como está e assim, pelo mecanismo do *Flex* isso é realizado. Também salientamos que qualquer texto em *LaTex* que seja necessário poderá ser acrescentado ao original visto que a nossa linguagem não identificará esse texto.

4 Codificação e Testes

4.1 Testes realizados e Resultados

De seguida vamos apresentar alguns exemplos que mostram a simplicidade da nossa linguagem relativa ao *LaTex* e do resultado das transformções do texto da nossa linguagem para a linguagem desejada.

4.1.1 Exemplo 1

Texto Original

```
\documentclass{report}
\usepackage[utf8]{inputenc}

\title{TestesPL}
\author{JoseJorgeGoncalo}
\date{May 2018}

\begin{document}

\maketitle

\section{Introduction}


#O#

# ola

#N#

# *cenas*
# cenos

#D#

#[FCP] &Futebol Clube& do Porto
#[FPF] Federação _Portuguesa de_ Futebol

#FD#

#FN#

# ole
# oli
```

```
# olo

#FO#


\end{document}
```

Texto Produzido

```
\documentclass{report}
\usepackage[utf8]{inputenc}

\title{TestesPL}
\author{JoseJorgeGoncalo}
\date{May 2018}

\begin{document}

\maketitle

\section{Introduction}


\begin{enumerate}

\item ola

\begin{itemize}

\item \textbf{cenas}
\item cenos

\begin{description}

\item[FCP] \textit{Futebol Clube} do Porto
\item[FPF] Federação \underline{Portuguesa de} Futebol

\end{description}

\end{itemize}

\item ole
\item oli
\item olo
```

```
\end{enumerate}


\end{document}
```

Resultado em formato PDF

De seguida apresentamos as imagens correspondentes ao texto acima descrito. A página de título é igual para todos os exemplos, pelo que apenas será apresentado aqui.

Figura 1: titulo

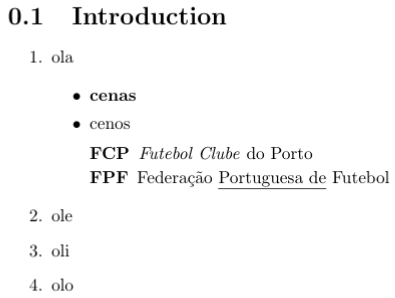

Figura 2: Exemplo1

4.1.2 Exemplo 2

Texto Original

```
\documentclass{report}
\usepackage[utf8]{inputenc}

\title{TestesPL}
\author{JoseJorgeGoncalo}
\date{May 2018}

\begin{document}

\maketitle

\section{Introduction}

*ola*
&ole&
_oli_

#O#

# ola
# ole
# oli
# olo

#N#
```

```
# cenas
# cenos

#FN#

#FO#

#D#

#[FCP] Futebol Clube do Porto
#[FPF] Federação Portuguesa de Futebol

#FD#

$ Teste $

Vamos lá testar!

$$ Devia dar $

Será que vai dar?

$$$ TODOS juntos $

Todos juntos conseguimos.

#T#

>C

>L OlaCenas >FL

\begin{tabular}{\|c\|c\|c\|c\|}

\hline

A | B | C | D #
\hline
A | B | C | D #

\hline

\end{tabular}

#FT#
```

```
\end{document}
```

Texto Produzido

```
\documentclass{report}
\usepackage[utf8]{inputenc}

\title{TestesPL}
\author{JoseJorgeGoncalo}
\date{May 2018}

\begin{document}

\maketitle

\section{Introduction}

\textbf{ola}
\textit{ole}
\underline{oli}

\begin{enumerate}

\item ola
\item ole
\item oli
\item olo

\begin{itemize}

\item cenas
\item cenos

\end{itemize}

\end{enumerate}

\begin{description}

\item[FCP] Futebol Clube do Porto
\item[FPF] Federação Portuguesa de Futebol

\end{description}
```

```
\chapter{ Teste }

Vamos lá testar!

\section{ Devia dar }

Será que vai dar?

\subsection{ TODOS juntos }

Todos juntos conseguimos.

\begin{table}

\centering

\caption{ OlaCenas }

\begin{tabular}{|c|c|c|c|}

\hline

A & B & C & D \\
\hline
A & B & C & D \\

\hline

\end{tabular}

\end{table}

\end{document}
```

Resultado em formato PDF

De seguida apresentamos as imagens correspondentes ao texto acima descrito.

0.1 Introduction

ola *ole* oli

1. ola
2. ole
3. oli
4. olo
    - cenas
    - cenos

FCP Futebol Clube do Porto

FPF Federação Portuguesa de Futebol

Figura 3: exemplo2_1

Chapter 1

Teste

Vamos lá testar!

1.1 Devia dar

Será que vai dar?

1.1.1 TODOS juntos

Todos juntos conseguimos.

Table 1.1: OlaCenas

A	B	C	D
A	B	C	D

Figura 4: exemplo2_2

4.1.3 Exemplo 3

Texto Original

```
\documentclass{report}
\usepackage[utf8]{inputenc}

\title{TestesPL}
\author{JoseJorgeGoncalo}
\date{May 2018}

\begin{document}

\maketitle

\section{Introduction}


#T#

>C

>L OlaCenas >FL

\begin{tabular}{\|c\|c\|c\|c\|}

\hline

*ola* | &ole& | C | D #
\hline
A |

\begin{minipage}{1.5in}
#O#

# ola
# ole
# oli
# olo

#N#

# cenas
# cenos
```

```
#FN#

#FO#
\end{minipage}
| _oli_ | D #

\hline

\end{tabular}

#FT#

\end{document}
```

Texto Produzido

```
\documentclass{report}
\usepackage[utf8]{inputenc}

\title{TestesPL}
\author{JoseJorgeGoncalo}
\date{May 2018}

\begin{document}

\maketitle

\section{Introduction}


\begin{table}

\centering

\caption{ OlaCenas }

\begin{tabular}{|c|c|c|c|}

\hline

\textbf{ola} & \textit{ole} & C & D \\
\hline
A &
```

```
\begin{minipage}{1.5in}
\begin{enumerate}

\item ola
\item ole
\item oli
\item olo

\begin{itemize}

\item cenas
\item cenos

\end{itemize}

\end{enumerate}
\end{minipage}
& \underline{oli} & D \\

\hline

\end{tabular}

\end{table}

\end{document}
```

Resultado em formato PDF

De seguida apresentamos a imagem correspondente ao texto acima descrito.

Table 1: OlaCenas

ola	*ole*	C	D
A	1. ola 2. ole 3. oli 4. olo • cenas • cenos	oli	D

0.1 Introduction

Figura 5: exemplo3

4.2 Código produzido

```
%{
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
%}

%x LISTA NEGRITO ITALICO SUBLINHADO CABECALHO DICIONARIO ELEMENTO TABELA

%option stack

%%


#T# {
  printf("\\begin{table}");
  yy_push_state(TABELA);
}


<*>\\[*_&#$] {printf("%s",yytext);}
```

```
<CABECALHO>\$ {
  printf("}");
  int top = yy_top_state();

  //printf("TOP: %d\n",top);
  //if(top!=0){
    yy_pop_state();
  //}
  //else{ BEGIN 0; }
}

<NEGRITO>\* {
    printf("}");
    int top = yy_top_state();

    //printf("TOP: %d\n",top);
  //  if(top!=0){
      yy_pop_state();
  //  }
  //  else{ BEGIN 0; }
  }

<ITALICO>& {

    printf("}");

    int top = yy_top_state();

    //printf("TOP: %d\n",top);
  //  if(yy_top_state()!=0){
      yy_pop_state();
  //  }
  //  else{ BEGIN 0; }
}

<SUBLINHADO>_ {

    printf("}");

    int top = yy_top_state();

    //printf("TOP: %d\n",top);
  //  if(yy_top_state()!=0){
      yy_pop_state();
  //  }
  //  else{ BEGIN 0; }
```

```
}

<LISTA>#  {

    printf("\\item");
}

<LISTA>#FO# {
  printf("\\end{enumerate}");
  yy_pop_state();
}

<LISTA>#FN# {
  printf("\\end{itemize}");
  yy_pop_state();
}

<DICIONARIO>#\[[^#*_&]+\] {

    printf("\\item%s",yytext+1);

}

<DICIONARIO>#FD#  {
  printf("\\end{description}");
  yy_pop_state();
}

<TABELA>\\\> {printf(">");}

<TABELA>\>C {printf("\\centering");}

<TABELA>\>L {printf("\\caption{");}

<TABELA>\>FL {printf("}");}

<TABELA>\| {printf("&");}

<TABELA>\\\| {printf("|");}

<TABELA># {printf("\\\\");}

<TABELA>#FT# { printf("\\end{table}"); yy_pop_state();}

$+  {
```

```
  int tam = strlen(yytext);
  switch(tam){
    case 1: printf("\\chapter{"); break;
    case 2: printf("\\section{"); break;
    case 3: printf("\\subsection{"); break;
    default: printf("\\subsubsection{");
  }
  yy_push_state(CABECALHO);
}

<*>\* {
    printf("\\textbf{");
    yy_push_state(NEGRITO);

    }

<*>&  {
    printf("\\textit{");
    yy_push_state(ITALICO);


}

<*>_ {

    printf("\\underline{");
    yy_push_state(SUBLINHADO);

}

<*>#O# {
    printf("\\begin{enumerate}");
    yy_push_state(LISTA);
}

<*>#N# {
    printf("\\begin{itemize}");
    yy_push_state(LISTA);
}

<*>#D# {
    printf("\\begin{description}");
    yy_push_state(DICIONARIO);
}
```

```
%%

int yywrap(){
  return 1;
}

int main(int argc, char** argv) {

  yylex();


  return 0;
}
```

5 Conclusão

Em suma, podemos afirmar que o trabalho realizado foi concluído com sucesso. O nosso principal objetivo, conseguir realizar tudo o que era proposto no enunciado, foi conseguido, já que a nossa linguagem consegue abstrair o uso de "código latex" para os casos de o utilizador querer utilizar as marcas que estavam referidas no enunciado. Para além deste aspeto, ainda conseguimos implementar uns extras com o encadeamento das marcas (por exemplo uma mesma palavra pode ser escrita a itálico e sublinhada), uma linguagem simplista para o uso de tabelas e ainda escrevemos uma "main" que permitia ao utilizador colocar como input o nome dos ficheiros, desta forma é mais intuitiva trabalhar, do que efetuar um redireccionamento a partir da bash.

Não só conseguimos resolver o que era pedido e ainda acrescentamos uns extras, como também o núcleo da resolução do problema passa pelo uso de Expressões Regulares e também do *[*flex. O uso da *stack* "nativa" do flex revelou-se uma ferramenta extremamente útil e eficaz, permitindo-nos, de uma forma mais fácil, aplicar então o encadeamento de instruções. Pelo facto mencionado anteriormente, podemos afirmar que conseguimos concluir o segundo objetivo delimitado conforme pretendido.

Mais uma vez, a realização deste projeto clarificou-nos o uso correto para a ferramenta *flex* e o quão poderosa esta poderá ser aliada ainda a outras. Ficamos agradados com os resultados obtidos, e verificamos que o uso de uma anotação do tipo *Markdown* poderá facilitar, e muito, a produção de textos em latex, por exemplo.