Documentación de MarkEngine

Realizado por: *Maldonado Ivan, Vera María*

Índice

1. Descripción

Este proyecto académico tiene como objetivo desarrollar una herramienta en C para la **generación automatizada de documentos en formato LaTeX**. La herramienta toma un conjunto de **reglas de producción definidas por el usuario** y las convierte en código LaTeX, lo que facilita la creación de documentos científicos o técnicos de manera estructurada y profesional.

El sistema permite a los usuarios describir la estructura y el contenido de su documento en un formato de entrada sencillo. Las reglas, que incluyen elementos como título, autor, capítulos, listas, tablas, imágenes, citas, etc., son analizadas y transformadas en código LaTeX automáticamente.

Este proyecto implementa un **analizador de sintaxis (parser)** utilizando herramientas como **Flex** y **Bison** para procesar el formato de entrada, reconocer los tokens definidos y generar el código LaTeX correspondiente, listo para ser compilado en un documento PDF.

2. Objetivo

El objetivo principal de **MarkEngine** es **automatizar la creación de documentos LaTeX** a partir de un conjunto estructurado de reglas definidas por el usuario. De este modo, los usuarios pueden concentrarse en el contenido y la estructura del documento sin preocuparse por la sintaxis de LaTeX, que se genera automáticamente.

El sistema **procesa** las entradas definidas por el usuario y **produce el código LaTeX necesario** para estructurar el documento según los parámetros establecidos.

3. Diseño

El diseño de la herramienta está basado en la **utilización de Flex y Bison**, dos herramientas ampliamente usadas para la creación de analizadores léxicos y sintácticos, respectivamente.

- **Flex**: se utiliza para definir y reconocer los **tokens** que se reconocerán en el formato de entrada.
- **Bison**: se emplea para analizar las reglas sintácticas y transformar los tokens en objetos C que luego se procesan para generar el código LaTeX.

3.1. Componentes del Proyecto

- **Scanner (Flex)**: Define los tokens que se reconocerán en el formato de entrada.
- **Parser (Bison)**: Toma los tokens definidos por Flex y los analiza según las reglas de sintaxis, creando una representación interna que puede ser procesada para generar el LaTeX.
- **Generador de Código LaTeX**: A partir de la representación interna generada por el parser, se produce el código LaTeX final que puede ser compilado.

4. Scanner

El scanner es responsable de leer el archivo de entrada y dividirlo en **tokens** que se utilizan para la interpretación. Estos tokens son definidos en un archivo de **Flex**.

Token	Descripción
TOKEN_PARAGRAPH	Representa un párrafo en el documento.
TOKEN_TITLE	Representa el título del documento.
TOKEN_DATE	Representa la fecha del documento.
TOKEN_SUBTITLE	Representa un subtítulo en el documento.
TOKEN_CHAPTER	Representa un capítulo en el documento.
TOKEN_ABSTRACT	Representa un resumen del documento.
TOKEN_AUTHOR	Representa el autor del documento.
TOKEN_LIST	Indica el inicio de una lista en el documento.
TOKEN_LINK	Representa un enlace (link) en el documento.
TOKEN_FONT	Indica un cambio de fuente en el documento.
TOKEN_TABLE	Indica una tabla en el documento.
TOKEN_DIAGRAM	Representa un diagrama en el documento.
TOKEN_INDEX	Indica la sección de índice en el documento.
TOKEN_IMG	Representa una imagen en el documento.
TOKEN_QUOTE	Indica una cita en el documento.
TOKEN_FOOT	Representa una nota al pie (footnote) en el documento.
TOKEN_LINEBREAK	Indica un salto de línea.
TOKEN_ENTER	Representa un retorno de carro o nueva línea.
TOKEN_DATE_FORMAT	Indica un formato de fecha válido.
TOKEN_NUMBER	Representa un número.
TOKEN_IMG_PATH	Representa la ruta de una imagen.
TOKEN_AT	Representa el símbolo `@`.

Token	Descripción
TOKEN_COMMA	Representa una coma.
TOKEN_DEF	Representa el símbolo :.
TOKEN_L_TAG	Representa el símbolo <.
TOKEN_R_TAG	Representa el símbolo >.
TOKEN_L_PAREN	Representa el símbolo (.
TOKEN_R_PAREN	Representa el símbolo).
TOKEN_L_BRACE	Representa el símbolo {.
TOKEN_R_BRACE	Representa el símbolo }.
TOKEN_UNDERSCORE	Representa el símbolo _.
TOKEN_HYPHEN	Representa el símbolo -.
TOKEN_SLASH	Representa el símbolo /.
TOKEN_WILDCARD	Representa el símbolo *.
TOKEN_WAVE	Representa el símbolo ~.

5. Parser

El parser procesa los tokens que el scanner ha identificado y los interpreta según las **reglas sintácticas definidas en el archivo de Bison (.y)**. Esta gramática define cómo los tokens se combinan para formar una estructura válida.

Por ejemplo, el token `@title:` debe ir seguido de un texto para representar el título. El parser asegura que estos elementos se estructuren de forma coherente para generar un documento bien formado.

6. Implementación

El código de **MarkEngine** está escrito en **C** y consta de varios archivos que gestionan la lectura de entrada, el análisis léxico y sintáctico, y la generación de LaTeX. Los archivos principales incluyen:

- Archivos Flex y Bison para la definición de tokens y gramáticas.
- Archivos de implementación en C que procesan la entrada y generan el código LaTeX.

La herramienta **convierte las reglas definidas** en código LaTeX estructurado, el cual luego puede ser compilado para producir un documento en formato PDF.

7. Manual de uso

7.1. Requisitos:

- **C** (para la programación).
- **Flex** y **Bison** (para el análisis léxico y sintáctico).
- **LaTeX** (para generar documentos en formato LaTeX).
- **gcc** o un compilador similar (para compilar el código).
- **pdflatex** (para convertir LaTeX a PDF).

7.2. Instalación:

1. Clonar el repositorio del proyecto.
2. Instalar las dependencias necesarias, como Flex, Bison y LaTeX.
3. Para instalar LaTeX:

```
sudo apt-get update
sudo apt-get install texlive-full
```

4. Compilar el código:

```
make
```

5. Ejecutar el programa:

```
./compiler name_file.txt
```

   Esto generará un archivo de salida en formato `.pdf` con el documentado final.

8. Formato de Entrada

El formato de entrada está estructurado con tokens precedidos por el símbolo `@`. A continuación se describen algunos de los tokens más importantes:

- `@title`: El título del documento.
- `@author`: El autor del documento.
- `@date`: La fecha del documento.
- `@paragraph`: Un párrafo en el documento.
- `@list`: Lista de elementos.
- `@img`: Ruta de una imagen, etc.

9. Tokens y Ejemplos

Cada token tiene una sintaxis definida, que el usuario debe seguir para que el programa funcione correctamente. A continuación, se presentan algunos ejemplos:

Título del Documento
Sintaxis: `@title: <texto>`
Ejemplo:

```
@title: some title
```

Autor del Documento
Sintaxis: `@author: <texto>`
Ejemplo:

```
@author: some author
```

Texto General
Sintaxis: `@text: <texto>`
Ejemplo:

```
@text: some text
```

Fecha del Documento
Sintaxis: `@date: <fecha>`
Formato: dd-mm-yyyy
Ejemplo:

```
@date: 04-12-2001
```

Subtítulo
Sintaxis: `@subtitle: <texto>`
Ejemplo:

```
@subtitle: some subtitle
```

Capítulo
Sintaxis: `@chapter: <texto>`
Ejemplo:

```
@chapter: some chapter
```

Resumen
Sintaxis: `@abstract: <texto>`
Ejemplo:

```
@abstract: some text
```

Índice
Sintaxis: `@index:`
Ejemplo:

```
@index:
```

Párrafo
Sintaxis: `@paragraph: <texto>`
Ejemplo:

```
@paragraph: some subtitle
```

Lista
Sintaxis: `@list: <texto1>, <texto2>`
Ejemplo:

```
@list: some text1, some text2
```

Imagen
Sintaxis: `@img:<ruta>`
Ejemplo:

```
@img:/example/to/img.jpg
```

Cita
Sintaxis: `@quote: {<texto>/<autor>/<año>}`
Ejemplo:

```
@quote: {Some quote/Some author/1889}
```

Nota al Pie
Sintaxis: `@foot: <texto>`
Ejemplo:

```
@foot: some text
```

Tabla
Sintaxis: `@table: <columnas>`
Ejemplo:

```
@table: <column1,column2>
    (value1,value2)
    (value3,value4)
```

Nota Importante:

Es fundamental que, al utilizar esta herramienta, se respete la sintaxis exactamente como se indica. Esto significa que cualquier espacio adicional, carácter o formato que no esté especificado puede generar problemas al generar el documento en formato LaTeX. Para evitar inconvenientes, es necesario seguir estrictamente las pautas tal como se presentan.

Este manual proporciona instrucciones detalladas sobre cómo utilizar el lenguaje de marcado para generar documentos en formato LaTeX de manera correcta.

Uso de la Sintaxis del Lenguaje de Marcado: Al iniciar un documento, es crucial que sigas el siguiente orden al definir su estructura:

Comienza por colocar el título del documento, seguido, de manera opcional, por el autor y la fecha. A continuación, incluye el token @begin para iniciar la definición del contenido. Luego, coloca todos los tokens que sean necesarios para estructurar tu documento (como capítulos, párrafos, imágenes, listas, entre otros). Finalmente, cierra el documento utilizando el token @end para completar la estructura. Recuerda que el formato debe seguir este esquema de manera rigurosa para garantizar que el archivo LaTeX se genere correctamente.

10. Ejemplo Completo

Un ejemplo de un archivo de entrada completo que utiliza los tokens definidos:

```
@title: some title
@author: some author
@date: 04-12-2001

@begin:

@new_page:

@subtitle: some subtitle

@chapter: some chapter

@abstract: some text

@index:

@paragraph: some subtitle

@list: some text1, some text2

@foot: some text

@table: <column1,column2>
(value1,value2)
(value3,value4)
```

```
@

@img:images/image2.jpeg

@quote:{"some quote."/ Niels Bohr/ 1927}
@end
```

El codigo latex que se genera es:

```
\documentclass{article}

 \usepackage{graphicx}

 \title{some title}

 \author{some author}

 \date{04-12-2001}

 \begin{document}
 \maketitle
 \newpage

 \subsection{some subtitle}

 \section{some chapter}

 \begin{abstract}
 some text
 \end{abstract}

 \tableofcontents

 some subtitle

 \begin{itemize}
 \item some text1
```

```
\item some text2
\end{itemize}
\footnote{some text}

\begin{table}[h]
\centering
\begin{tabular}{cc}
\hline
column1 & column2 \\ \hline
value1 & value2 \\
value3 & value4 \\
\\ \hline
\end{tabular}

\end{table}

\begin{figure}[h]
\includegraphics[width=0.5\textwidth]{images/image2.jpeg}
\centering
\end{figure}

\begin{quote}
"some quote."
\end{quote}
\begin{flushright}
Niels Bohr
year: 1927\end{flushright}


\end{document}
```

El documento obtenido por latex:

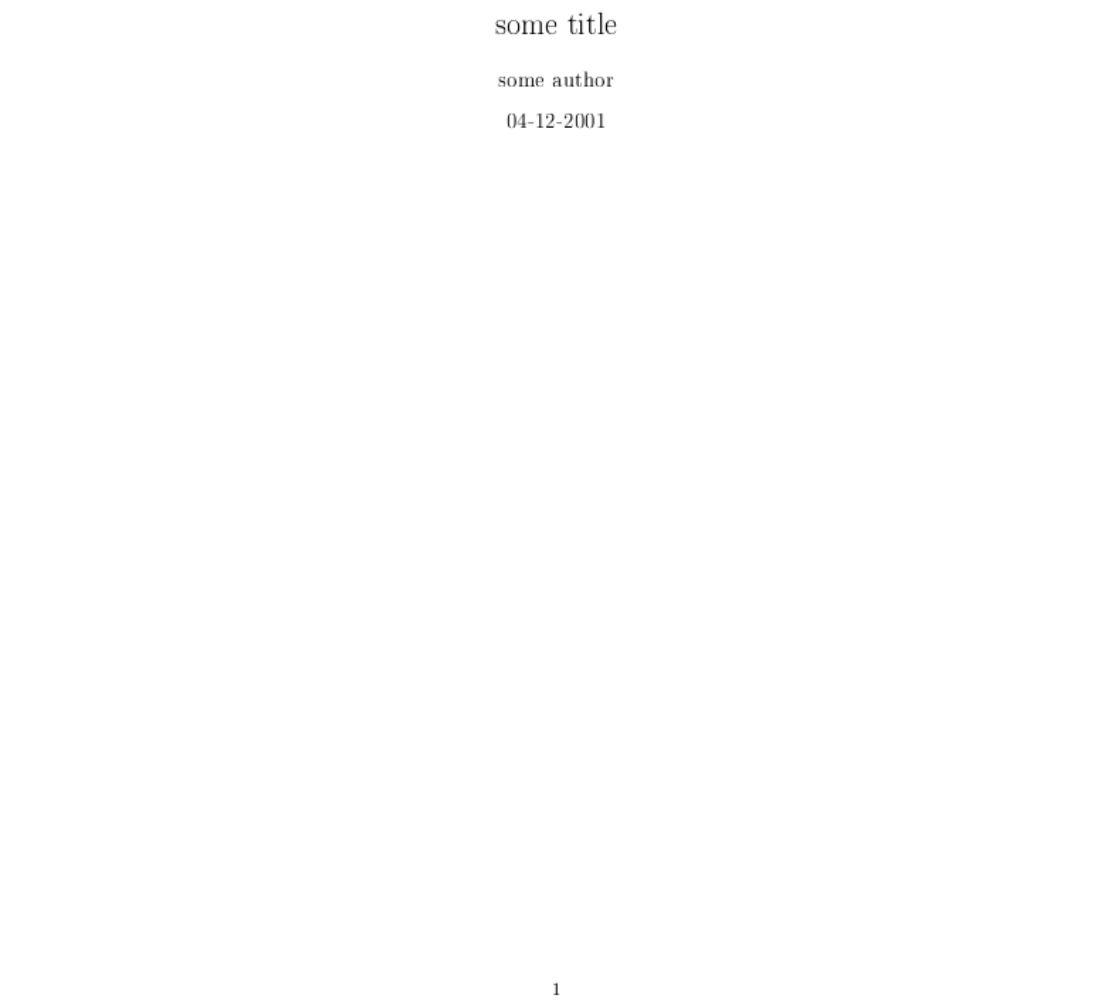

Figura 1: Página 1 del ejemplo realizado

Última actualización: Noviembre 2024

0.1 some subtitle

1 some chapter

Abstract

some text

Contents

- some text1
- some text2

[1]

column1	column2
value1	value2
value3	value4

"some quote."

Niels Bohr year: 1927

[1] some text

Figura 2: Página 2 del ejemplo realizado

Última actualización: Noviembre 2024

11. Conclusiones

El proyecto **MarkEngine** proporciona una herramienta para la generación de documentos LaTeX. Utilizando un conjunto de reglas definidas por el usuario, el sistema transforma el texto de entrada en un formato LaTeX válido, lo que facilita la creación de documentos científicos y técnicos de manera rápida y profesional.
El uso de Flex y Bison permite una gran flexibilidad y control sobre el proceso de análisis léxico y sintáctico, y la capacidad de personalizar los tokens y las reglas de producción hace que el sistema sea extensible y adaptable a diferentes necesidades.