Diseño e Implementación de un Analizador Léxico y Analizador Semántico para el Lenguaje AVISMO

Aramis E. Matos

Lenier Gerena

Angel Berrios Pellot

Segundo Semestre, 2022-2023

Tabla de Contenido

1 Introducción **2**

2 Analizador Léxico **4**

2.1 Gramatica del Lenguaje AVISMO 4

2.2 Diseño del del Analizador Léxico 12

2.2.1 Autómatas Finitos Deterministas 12

2.2.2 Tabla de Símbolos 13

2.3 Implementación del Analizador Léxico 14

3 Implementación del Analizador Sintáctico **18**

4 Conclusiones y Recomendaciones **19**

Referencias Bibliográficas **22**

Capítulo 1

Introducción

La visualización molecular puede ser considerada como una de las áreas mas importante dentro de la bioinformática. Entre sus aplicaciones mas relevantes se destacan el diseño de nuevo fármacos... (Narciso Farias, Rios, Hidrobo, & Vicuña, 2012). Este enunciado fue escrito hace mas de una década. Sin embargo, hoy día en un mundo pospandemia, reconocemos que tan sabio fue. El desarollo de la vacuna contra el COVID-19 tan rápido fue gracias a herramientas de visualización como el Ambiente de visualización Molecular (AVISMO) (Narciso Farias et al., 2012) El propósito de este proyecto es definir el automata de estado finito del lenguaje AVISMO, los patrones el cual caracterizan los lexemas del lenguaje, los atributos de los lexemas y la implementación del analizador léxico y sintáctico del lenguaje AVISO en C++. La implementación léxica se desarolló utilizando GNU Flex (*Flex - a Scanner Generator*, n.d.) con la implementación de Calc++ por bwasti (Wasti, 2020) como base. La implementación sintáctica se desarolló

en GNU Bison (*Bison - GNU Project - Free Software Foundation*, n.d.) con la implementación de Calc++ por bwasti (Wasti, 2020) como base.

Capítulo 2

Analizador Léxico

2.1 Gramatica del Lenguaje AVISMO

- <SENTENCIAS> ::= <FIN_DE_LINEA> <SENTENCIAS> | <SENTENCIA> <FIN_DE_LINEA>
- <FIN_DE_LINEA> ::= ":" | ";"
- <SENTENCIA> ::= "defina" <ID> "como" <TIPO> | <ID> "=" <MODELO_MOLECULAR> | <OPERACION> "(" <ID> ")"
- <ID> ::= "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z" | "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x" | "y" | "z" | <LETRA> <IDCONT>

- <IDCONT> ::= "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z" | "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x" | "y" | "z" | <LETRA> <IDCONT> | "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" | <DIGITO> <IDCONT>

- <LETRA> ::= "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z" | "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x" | "y" | "z"

- <DIGITO> ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

- <TIPO> ::= "modelo"

- <OPERACION> ::= "graficar2d" |"graficar3d" | "pesomolecular"

- <MODELO_MOLECULAR> ::= "H" | "Li" | "Na" | "K" | "Rb" | "Cs" | "Fr" | "Be" | "Mg" | "Ca" | "Sr" | "Ba" | "Ra" | "Sc" | "Y" | "Ti" | "Zr" | "Hf" | "Db" | "V" | "Nb" | "Ta" | "Ji" | "Cr" | "Mo" | "W" | "Rf" | "Mn" | "Tc" | "Re" | "Bh" | "Fe" | "Ru" | "Os" | "Hn" | "Co" | "Rh" | "Ir" | "Mt" | "Ni" | "Pd" | "Pt" | "Cu" | "Ag" | "Au" | "Zn" | "Cd" | "Hg" | "B" | "Al" | "Ga" | "In" | "Ti" | "C" | "Si" | "Ge" | "Sn" | "Pb" | "N" | "P" | "As" | "Sb" | "Bi" | "O" | "S" | "Se" | "Te" | "Po" | "F" | "Cr" | "Br" | "I" | "At" | "He" | "Ne" | "Ar" | "Kr" | "Xe" | "Rn" | <ELEMENTO_QUIMICO> <VALENCIA> | <ELE-

MENTO> <GRUPO_FUNCIONAL> | <COMPUESTO> <ELEMENTO> <GRUPO_FUNCIONAL> | <COMPUESTO> <COMPUESTO>

- <COMPUESTO> ::= "H" | "Li" | "Na" | "K" | "Rb" | "Cs" | "Fr" | "Be" | "Mg" | "Ca" | "Sr" | "Ba" | "Ra" | "Sc" | "Y" | "Ti" | "Zr" | "Hf" | "Db" | "V" | "Nb" | "Ta" | "Ji" | "Cr" | "Mo" | "W" | "Rf" | "Mn" | "Tc" | "Re" | "Bh" | "Fe" | "Ru" | "Os" | "Hn" | "Co" | "Rh" | "Ir" | "Mt" | "Ni" | "Pd" | "Pt" | "Cu" | "Ag" | "Au" | "Zn" | "Cd" | "Hg" | "B" | "Al" | "Ga" | "In" | "Ti" | "C" | "Si" | "Ge" | "Sn" | "Pb" | "N" | "P" | "As" | "Sb" | "Bi" | "O" | "S" | "Se" | "Te" | "Po" | "F" | "Cr" | "Br" | "I" | "At" | "He" | "Ne" | "Ar" | "Kr" | "Xe" | "Rn" | <ELEMENTO_QUIMICO> <VALENCIA> | <ELEMENTO> <GRUPO_FUNCIONAL> | <ELEMENTO> <GRUPO_FUNCIONAL> <ENLACE> | <ELEMENTO> <ENLACE>
- <COMPUESTOS> ::= <COMPUESTO> <COMPUESTO> | <COMPUESTOS>
- <ELEMENTO> ::= "H" | "Li" | "Na" | "K" | "Rb" | "Cs" | "Fr" | "Be" | "Mg" | "Ca" | "Sr" | "Ba" | "Ra" | "Sc" | "Y" | "Ti" | "Zr" | "Hf" | "Db" | "V" | "Nb" | "Ta" | "Ji" | "Cr" | "Mo" | "W" | "Rf" | "Mn" | "Tc" | "Re" | "Bh" | "Fe" | "Ru" | "Os" | "Hn" | "Co" | "Rh" | "Ir" | "Mt" | "Ni" | "Pd" | "Pt" | "Cu" | "Ag" | "Au" | "Zn" | "Cd" | "Hg" | "B" | "Al" | "Ga" | "In" | "Ti" | "C" | "Si" | "Ge" | "Sn" | "Pb" | "N" | "P" | "As" | "Sb" | "Bi" | "O" | "S" | "Se" | "Te" | "Po" | "F" | "Cr" | "Br" | "I" | "At" | "He" | "Ne" | "Ar" | "Kr" | "Xe" | "Rn" | <ELEMENTO_QUIMICO> <VALENCIA>
- <ELEMENTO_QUIMICO> ::= "H" | "Li" | "Na" | "K" | "Rb" | "Cs" | "Fr" |

"Be" | "Mg" | "Ca" | "Sr" | "Ba" | "Ra" | "Sc" | "Y" | "Ti" | "Zr" | "Hf" | "Db" | "V" | "Nb" | "Ta" | "Ji" | "Cr" | "Mo" | "W" | "Rf" | "Mn" | "Tc" | "Re" | "Bh" | "Fe" | "Ru" | "Os" | "Hn" | "Co" | "Rh" | "Ir" | "Mt" | "Ni" | "Pd" | "Pt" | "Cu" | "Ag" | "Au" | "Zn" | "Cd" | "Hg" | "B" | "Al" | "Ga" | "In" | "Ti" | "C" | "Si" | "Ge" | "Sn" | "Pb" | "N" | "P" | "As" | "Sb" | "Bi" | "O" | "S" | "Se" | "Te" | "Po" | "F" | "Cr" | "Br" | "I" | "At" | "He" | "Ne" | "Ar" | "Kr" | "Xe" | "Rn"

- <VALENCIA> ::= "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
- <GRUPO_FUNCIONAL> ::= <GRUPO_FUNCIONAL_INFERIOR> <GRUPO_FUNCIONAL_SUPERIOR> | <GRUPO_FUNCIONAL_SUPERIOR> <GRUPO_FUNCIONAL_INFERIOR> | "(" <MODELO_GRUPO_FUNCIONAL> ")" | "[" <MODELO_GRUPO_FUNCIONAL> "]"
- <GRUPO_FUNCIONAL_SUPERIOR> ::= "[" <MODELO_GRUPO_FUNCIONAL> "]"
- <GRUPO_FUNCIONAL_INFERIOR> ::= "(" <MODELO_GRUYPO_FUNCIONAL> ")
- <MODELO_GRUYPO_FUNCIONAL> ::= <ENLACE> <MODELO_MOLECULAR> | "H" | "Li" | "Na" | "K" | "Rb" | "Cs" | "Fr" | "Be" | "Mg" | "Ca" | "Sr" | "Ba" | "Ra" | "Sc" | "Y" | "Ti" | "Zr" | "Hf" | "Db" | "V" | "Nb" | "Ta" | "Ji" | "Cr" | "Mo" | "W" | "Rf" | "Mn" | "Tc" | "Re" | "Bh" | "Fe" | "Ru" | "Os" | "Hn" | "Co" | "Rh" | "Ir" | "Mt" | "Ni" | "Pd" | "Pt" | "Cu" | "Ag" | "Au" | "Zn" | "Cd" | "Hg" | "B" | "Al" | "Ga" | "In" | "Ti" | "C" | "Si" | "Ge" | "Sn" | "Pb" | "N" | "P"

| "As" | "Sb" | "Bi" | "O" | "S" | "Se" | "Te" | "Po" | "F" | "Cr" | "Br" | "I" | "At" | "He" | "Ne" | "Ar" | "Kr" | "Xe" | "Rn" | <ELEMENTO_QUIMICO> <VALENCIA> | <ELEMENTO> <GRUPO_FUNCIONAL> | <COMPUESTO> <ELEMENTO> | <COMPUESTO> <COMPUESTO> <COMPUESTOS>

En la tabla 2.1, en la columna de patrones, note que cuando dice $\{TOKEN\}$ donde *TOKEN* se refiere a el patrón asociado a *token*. Por ejemplo, si un patrón dice $\{ELEMENTO\_QUIMICO\}$, esto significa que inserta el patrón asociado al *token ELEMENTO_QUIMICO*. Esto no significa que el analizador léxico espera un *token* de por si, sencillamente se hizo con el propósito de evitar redundancias.

Token	Patrón	Lexema	Atributos
<FIN_DE_LINEA>	; | :	:	Simbolo reservado
<PALABRA _RESERVADA>	defina | como	defina	Palabra reservada
<ID>	[A-Za-z][A-Za-z0-9]*	var1	Modelo molecular asociado
<IDCONT>	*[A-Za-z0-9]+*	1ar	ID asociado
<LETRA>	[A-Za-z]	a	ID asociado
<DIGITO>	[0-9]	7	Valor numérico, lexema asociado
<TIPO>	modelo	modelo	ID asociado
<OPERACION>	graficar2d | graficar3d | pesomolecular	pesomolecular	ID asociado
<MODELO _MOLECULAR>	({ELEMENTO _QUIMICO} | {ELEMENTO _QUIMICO} {VALENCIA} | {ELEMENTO} {GRUPO _FUNCIONAL} | {ELEMENTO} {GRUPO _FUNCIONAL} {ENLACE} | {ELEMENTO} {ENLACE})	CH3(CH3)CHH	ID asociado

<COMPUESTO>	COMPUESTO ({ELEMENTO _QUIMICO} | {ELEMENTO _QUIMICO} {VALENCIA} | {ELEMENTO} {GRUPO_FUNCIONAL} | {ELEMENTO} {GRUPO _FUNCIONAL} {ENLACE} | {ELEMENTO} {ENLACE})	CH3::	Modelo molecular asociado, enlaces, valencias
<COMPUESTOS>	{COMPUESTO}+	CH3::(OH)3	Modelo molecular asociado, enlaces, valencias
<ELEMENTO>	{ELEMENTO _QUIMICO} {VALENCIA}?	Ag3	Elemento, valencia
<ELEMENTO _QUIMICO>	("H" | "Li" | "Na" | "K" | "Rb" | "Cs" | "Fr" | "Be" | "Mg" | "Ca" | "Sr" | "Ba" | "Ra" | "Sc" | "Y" | "Ti" | "Zr" | "Hf" | "Db" | "V" | "Nb" | "Ta" | "Ji" | "Cr" | "Mo" | "W" | "Rf" | "Mn" | "Tc" | "Re" | "Bh" | "Fe" | "Ru" | "Os" | "Hn" | "Co" | "Rh" | "Ir" | "Mt" | "Ni" | "Pd" | "Pt" | "Cu" | "Ag" | "Au" | "Zn" | "Cd" | "Hg" | "B" | "Al" | "Ga" | "In" | "Ti" | "C" | "Si" | "Ge" | "Sn" | "Pb" | "N" | "P" | "As" | "Sb" | "Bi" | "O" | "S" | "Se" | "Te" | "Po" | "F" | "Cr" | "Br" | "I" | "At" | "He" | "Ne" | "Ar" | "Kr" | "Xe" | "Rn")	I	Elemento
<VALENCIA>	[1-9]	2	Valor

<GRUPO _FUNCIONAL>	({GRUPO _FUNCIONAL _INFERIOR} {GRUPO _FUNCIONAL _SUPERIOR} | {GRUPO _FUNCIONAL _SUPERIOR} {GRUPO _FUNCIONAL_INFERIOR} | "(" {MODELO _GRUPO _FUNCIONAL} ")" | "[" MODELO _GRUPO _FUNCIONAL "]")	(CH3){Ag2}	Grupos funcionales, grupo funcional inferior, grupo funcional superior
<GRUPO _FUNCIONAL _INFERIOR>	"[" {MODELO _GRUPO _FUNCIONAL} "]"	[CVHe3]	Elementos, valencias
<GRUPO _FUNCIONAL _SUPERIOR>	"(" {MODELO _GRUPO _FUNCIONAL} ")"	(CVHe3)	Elementos, valencias
<MODELO _GRUPO _FUNCIONAL>	({ELEMENTO _QUIMICO}+ {VALENCIA}?)+ | ({ELEMENTO}+ {ENLACE} {ELEMENTO}+)+	FeH=C3Si4	Elementos, enlaces, valencias
<ENLACE>	("-" | "=" | ":" | "::")	-	Valencia

Tabla 2.1: Tabla de Componentes Léxicos de AVISMO

2.2 Diseño del del Analizador Léxico

2.2.1 Autómatas Finitos Deterministas

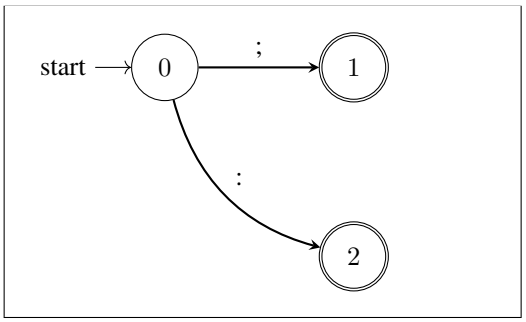

Figura 2.1: Automata del patrón para el token <FIN_DE_LINEA>

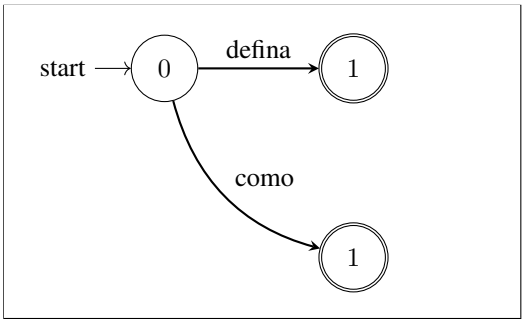

Figura 2.2: Automata del patrón para el token <PALABRAS_RESERVADA>

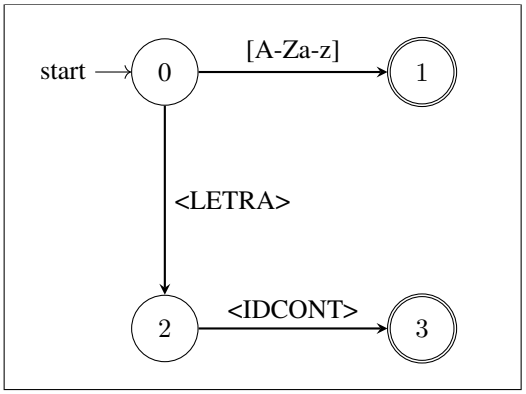

Figura 2.3: Automata del patrón para el token <ID>

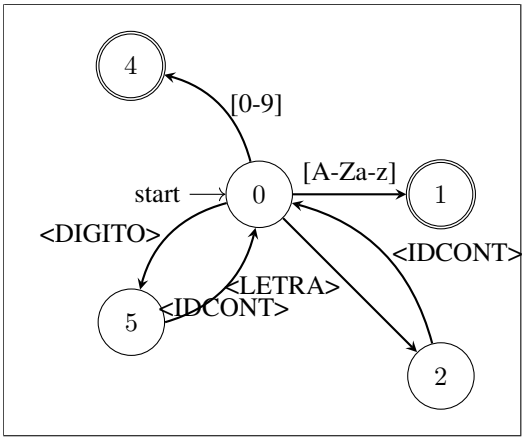

Figura 2.4: Automata del patrón para el token <IDCONT>

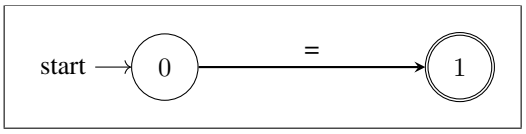

Figura 2.5: Automata del patrón para el token <ASIGNACION>

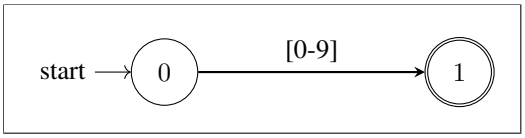

Figura 2.6: Automata del patrón para el token <LETRA>

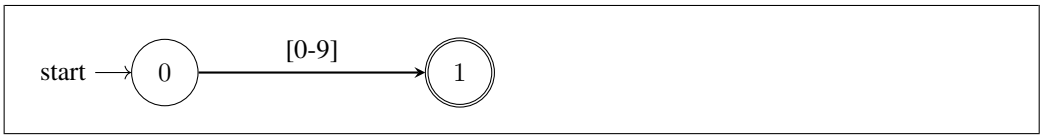

Figura 2.7: Automata del patrón para el token <DIGITO>

2.2.2 Tabla de Símbolos

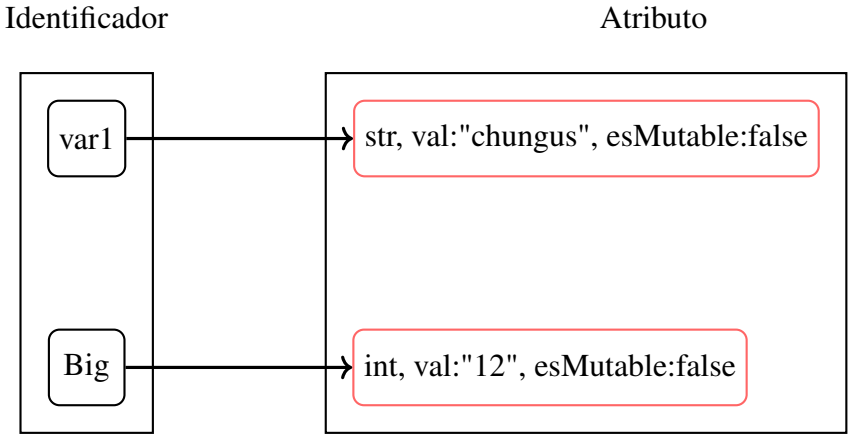

Figura 2.8: Tabla de símbolos implementada como un diccionario

2.3 Implementación del Analizador Léxico

Como se ha mencionado anteriormente, la implementación lexica del proyecto fue inspirada por la implementación de Calc++ por bwasti y adaptada para la gramática de AVISMO (Wasti, 2020).

Para compilar el programa, primero que todo se tiene que ejecutar *make clean* por la línea dentro del directorio *code*, Esto se hace con el propósito de evitar errores de compilación. Entonces, se ejecuta el comando *make*, esto compilará todas las dependencias necesarias, en particular, los archivos de contiene el léxico de Flex (con todos los archivos con extensión .cc, .hh y .ll). Además, el comando anterior compila todos los programas en código objeto (.o) que se crean para el programa *driver* (el cual está encargado de abrir archivos de entrada e instanciar el analizador), parser y scanner. Posterior a esto es que se puede ejecutar *./avismo fileName.txt* el cual para nuestro caso seria el siguiente:

./avismo test_prog.txt

Al ejecutarse el comando anterior, el programa procede a leer **cada caracter** del programa e identificar si una serie de caracteres sigue un patrón que forma parte del lenguaje AVISMO. Al encontrar un patrón reconocido, tales como un identificador o modelo molecular, lo clasifica con un *token* correspondiente, lo emprime en el archivo de output.txt y lo devuelve al analizador sintáctico. Note, que el patrón de identificador reconoce palabra reservadas también. Esto crea ambiguedad semántica debido a que la gramatica no tiene un mecanismo para diferenciar entre una palabra reservada y un identificador. Por esta razón, si una

serie de caracteres se identifica como un lexema de categoría identificador, se compara con los valores ya existentes del diccionario *variables*, el cual es un miembro de la clase *driver*. Al inicializar un objeto *driver*, cuyo constructor está localizado en *driver.cc*, este se encarga de abrir el archivo de palabras reservadas (*keywords.txt*) y añadir las palabras reservadas antes que cualquier variable se pueda inicializar. Mas aún, a las palabras reservadas se les asigna el valor de la cadena vacía. Esto se hace con el propósito de poder diferenciar entre palabras reservadas e identificadores, ya que al nivel sintáctico, no es posible asignarle a una identificador una cadena vacía, como se puede apreciar a continuación:

```
{ID} {
  std::string text(yytext);
  if (drv.variables.find(text) != drv.variables.end() &&
     drv.variables[text] == "") {
    if (drv.variables[text] == "") {
      format_output("PALABRA_RESERVADA",yytext,loc);
      return yy::parser::make_PALABRA_RESERVADA (yytext,loc)
         ;
    }
  }
       format_output("ID",yytext,loc);
       return yy::parser::make_ID(text,loc);
}
```

Con el propósito de visualizar los lexemas generados por el *scanner*, colocado en

el archivo *scanner.ll*, se utiliza la siguiente función:

```
std::ofstream& file("output.txt");
void format_output (std::string token,const char* yytext,
   yy::location& loc) {
   file << "(" << "<" << token << ">," << std::string(
      yytext) << "," << loc << ")" << std::endl;
}
```

En el caso de un error léxico, se ejecuta el siguiente código:

```
. {
  file << "CARACTER INVALIDO " << std::string(yytext) <<
     "," << loc << std::endl;
}
```

Los patrones que se utilizan para este archivo *scanner.ll* (líneas 30-71) son una adaptación de la gramática en la tabla 2.1. Note las variables *yytext* y *loc* en el código anterior. *loc* contiene la referencia a la memoria de la variable *location* de la clase *driver*, cuyo propósito es retornar el valor de la linea en donde se encuentra un lexema dado. *yytext* contiene el lexema que fue aceptado por un patrón. *loc* y *yytext* se utilizan para imprimir la aceptación de una cadena de caracteres como un lexema de un *token* o para gestión de errores. Por ejemplo, el siguiente código genera una entrada en el archivo *output.txt* que decalara la cadena de caracteres que se aceptó como una sentencia.

```
format_output("SENTENCIAS",yytext,loc);
```

Todas las definiciones de patrones (con la exepción del patrón que maneja la gestión de errores) contienen la función de *format_output* para imprimir la aceptación de una serie de caracteres como un lexema de un patrón en particular. Además, como se ha mencionado anteriormente, una vez se hace la aceptación, se devuelve el *token* al analizador sintáctico.

```
{SENTENCIAS} {
   format_output("SENTENCIAS",yytext,loc);\\
   return yy::parser::make_SENTENCIAS(yytext,loc);\\
}
```

Capítulo 3

Implementación del Analizador Sintáctico

Capítulo 4

Conclusiones y Recomendaciones

List of Figures

2.1 Automata del patrón para el token <FIN_DE_LINEA> 12
2.2 Automata del patrón para el token <PALABRAS_RESERVADA> 12
2.3 Automata del patrón para el token <ID> 12
2.4 Automata del patrón para el token <IDCONT> 12
2.5 Automata del patrón para el token <ASIGNACION> 13
2.6 Automata del patrón para el token <LETRA> 13
2.7 Automata del patrón para el token <DIGITO> 13
2.8 Tabla de símbolos implementada como un diccionario 13

List of Tables

2.1 Tabla de Componentes Léxicos de AVISMO 11

Referencias Bibliográficas

Bison - GNU Project - Free Software Foundation. (n.d.). https://www.gnu.org/software/bison/.

Flex - a scanner generator. (n.d.). https://ftp.gnu.org/old-gnu/Manuals/flex-2.5.4/html_mono/flex.html.

Narciso Farias, F., Rios, A., Hidrobo, F., & Vicuña, O. (2012, May). Una gramática libre de contexto para el lenguaje del ambiente de visualización molecular - AVISMO..

Wasti, B. (2020, June). *Bwasti/bison-example-calc-*.