

PRACTICA NO.4 Léxico

Integrantes:

Patricia Fernanda Castellanos Ruiz

Mayra Esmeralda Castillo Montenegro

Jesús Eduardo Silva Vázquez

Descripción de la práctica:

Descripción de Patricia:

Investigué sobre la herramienta Flex, cómo funciona y como se utiliza, intente ayudar a terminar la tabla del autómata finito determinista que hicimos en equipo, pero me revolví toda y junto con mis compañeros lo resolvimos.

Descripción de Mayra:

Hice el AFD y junto a mis compañeros realizamos la tabla, además de eso, definimos las expresiones regulares para las distintas categorías léxicas presentadas en la práctica.

Descripción de Jesús:

Me encargué de instalar las herramientas necesarias para hacer funcionar Flex, hice un archivo en lenguaje C con las expresiones regulares de las categorías léxicas para el analizador léxico, además de eso realicé un código prueba y logré hacer el ejecutable de nuestro analizador léxico.

¿Qué es Flex?

Flex es una herramienta para generar **escáneres**: programas que reconocen patrones léxicos en un texto. flex lee los ficheros de entrada dados, o la entrada estándar si no se le ha indicado ningún nombre de fichero, con la descripción de un escáner a generar. La descripción se encuentra en forma de parejas de expresiones regulares y código C, denominadas **reglas**. flex genera como salida un fichero fuente en C, 'lex.yy.c', que define una rutina 'yylex()'. Este fichero se compila y se enlaza con la librería '-lfl' para producir un ejecutable. Cuando se arranca el fichero ejecutable, este analiza su entrada en busca de casos de las expresiones regulares. Siempre que encuentra uno, ejecuta el código C correspondiente.

La estructura de un archivo de lex es intencionadamente similar a la de un archivo del yacc; los archivos se dividen en tres secciones, separadas por líneas que contienen solamente dos símbolos "%", como sigue:

Sección de declaraciones

%%

Sección de reglas

%%

Sección de código en C

- La **sección de declaraciones** es el lugar para definir macros y para importar los archivos de cabecera escritos en C. También es posible escribir cualquier código de C aquí, que será copiado en el archivo fuente generado. Este código en C debe ir entre los símbolos %{ %}.

También se pueden incluir "atajos" para definir patrones de la Sección de Reglas, por ejemplo en vez del patrón [0-9]* (cero o más dígitos que reconocerían cualquier número natural), se puede definir en esta sección el "atajo": números [0-9]*, así, en la sección de código pondríamos el patrón {números} {acción_en_C;}. Con esto se clarifica la escritura del código en lex.

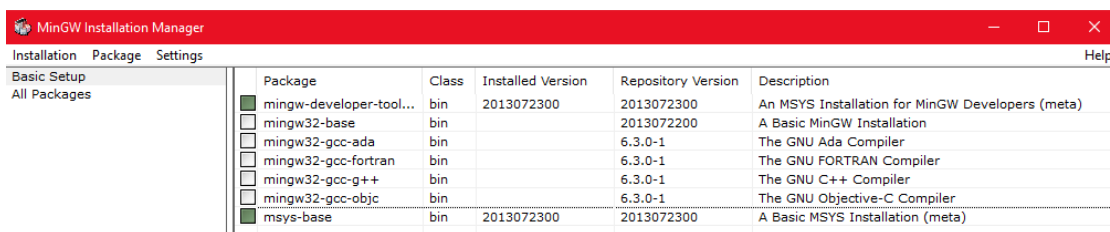
- La **sección de reglas** es la sección más importante; asocia patrones a sentencias de C. Los patrones son simplemente expresiones regulares. Cuando el *lexer* encuentra un texto en la entrada que es asociable a un patrón dado, ejecuta el código asociado de C. Ésta es la base de del funcionamiento de lex.
- La **sección de código C** contiene sentencias en C y funciones que serán copiadas en el archivo fuente generado. Estas sentencias contienen generalmente el código llamado por las reglas en la sección de las reglas. En programas grandes es más conveniente poner este código en un archivo separado y enlazarlo en tiempo de compilación.

¿Cómo funciona Flex?

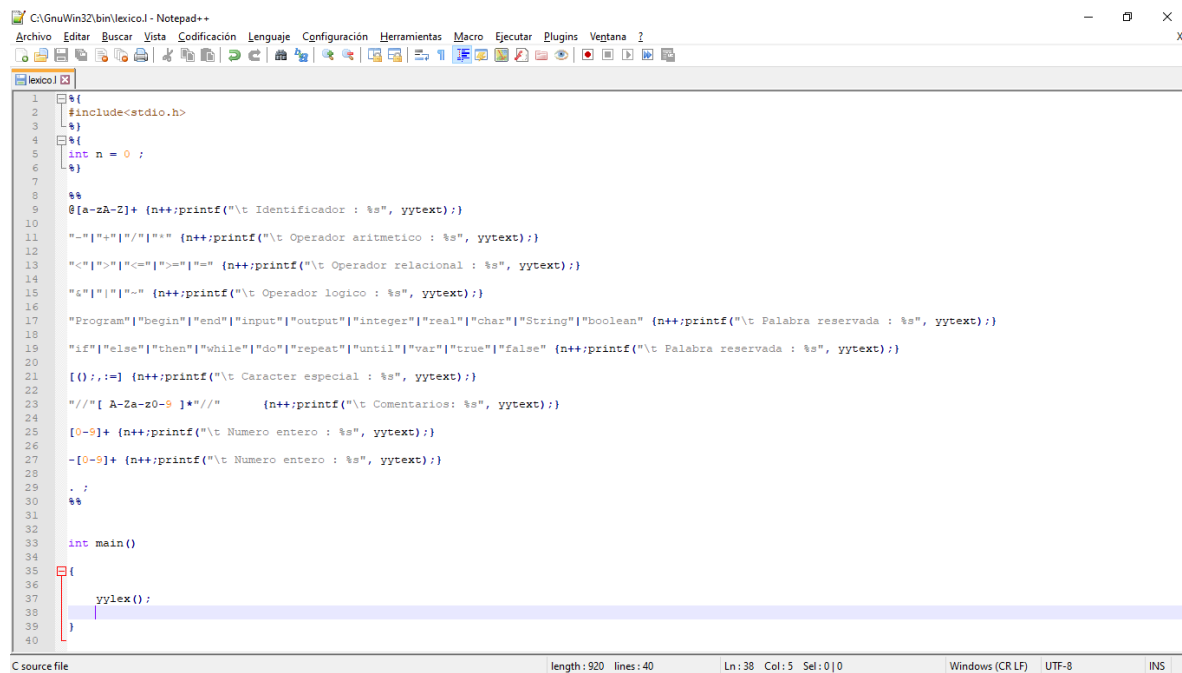
Para poder hacer funcionar Flex es indispensable contar con las siguientes herramientas:

- Procesador de textos.
- MinGW
- Flex (GnuWin32)

El primer paso consiste en instalar Flex y MinGW, es importante que al instalar MinGW dentro del “Installation Manager” sea instalado los paquetes del setup básico:



Esto debido a que la función de MinGW es crear archivos ejecutables dentro del cmd de Windows. Una vez que hayan instalados ambos programas, dentro del procesador de texto se debe hacer un código de programación en lenguaje C para que funcione como analizador léxico:

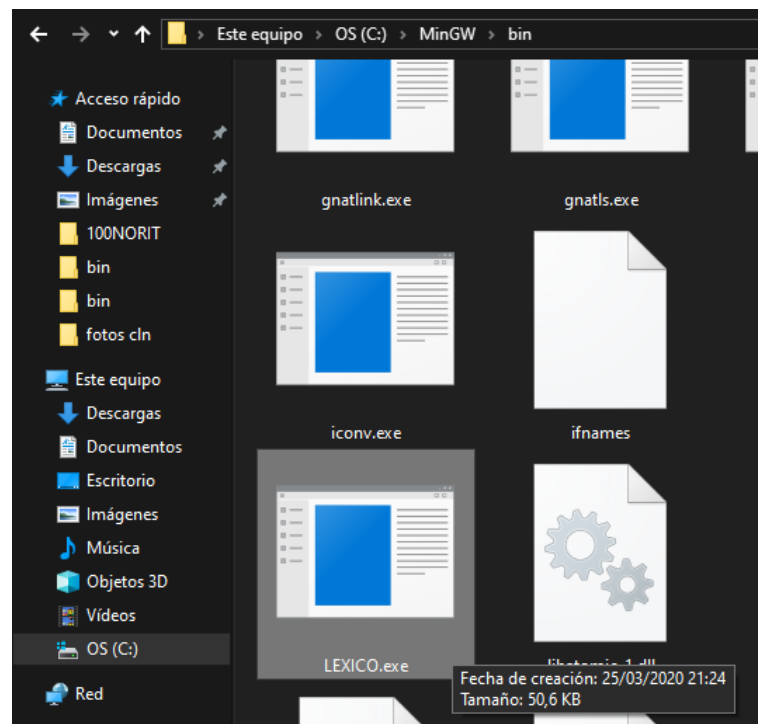


Es importante definir la extensión del archivo en tipo “.l” y guardarlo en C:\GnuWin32\bin\.

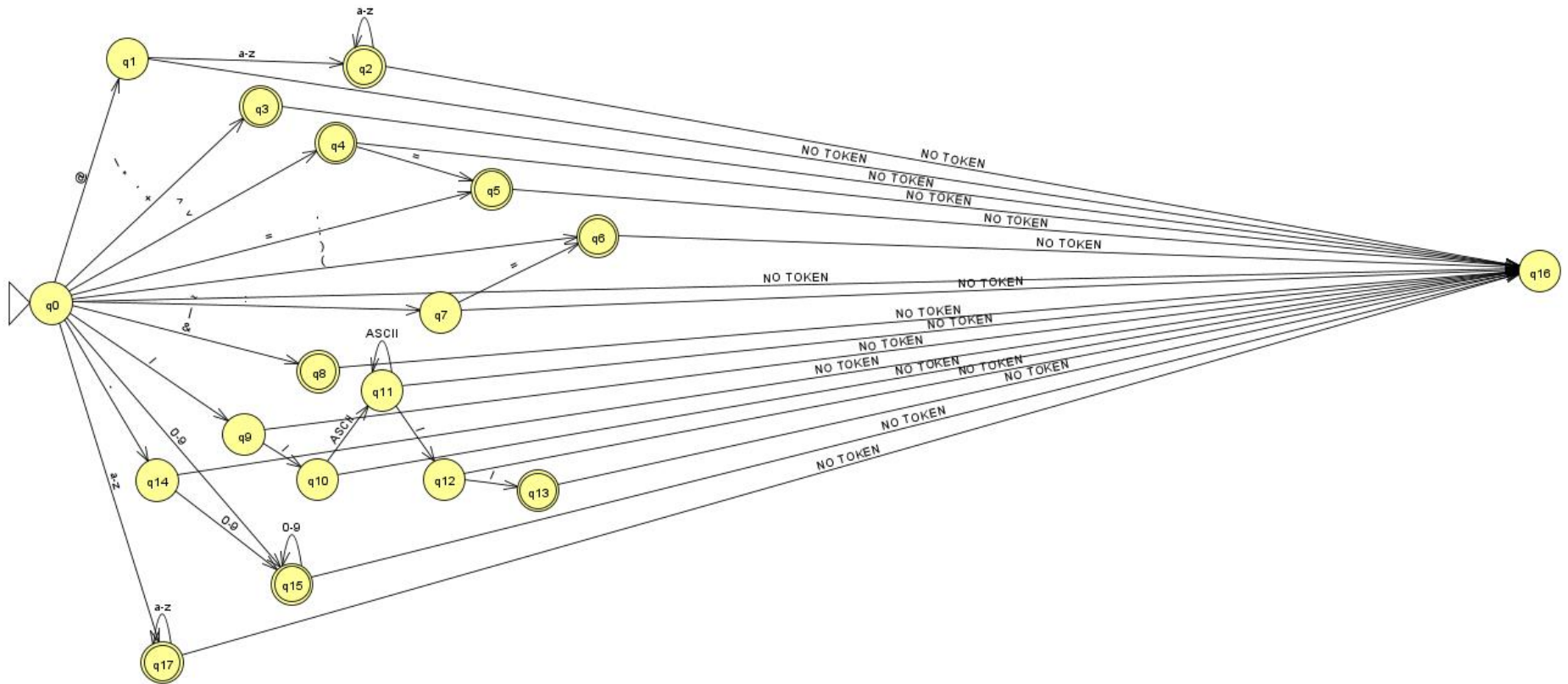
Ahora dentro del cmd de Windows se deben ingresar los siguientes comandos:

```
Simbolo del sistema
C:\>cd gnuwin32
C:\GnuWin32>cd bin
C:\GnuWin32\bin>FLEX lexico.1
C:\GnuWin32\bin>cd /
C:\>cd mingw
C:\MinGW>cd bin
C:\MinGW\bin>gcc lex.yy.c -lfl -o LEXICO
C:\MinGW\bin>cd /
C:\>cd gnuwin32
C:\GnuWin32>cd bin
C:\GnuWin32\bin>FLEX lexico.1
C:\GnuWin32\bin>cd /
C:\>cd mingw
C:\MinGW>cd bin
C:\MinGW\bin>gcc lex.yy.c -lfl -o LEXICO
```

La función del comando “FLEX” es convertir el archivo que creamos previamente en un archivo de tipo lex para análisis de expresiones regulares, creará un archivo llamado “lex.yy.c”, ese archivo debe ser reubicado hacia la carpeta C:\MinGW\bin\; “gcc -lfl” son comandos utilizados para generar un ejecutable (.exe) de nuestro código y “-o” para definir un nombre de nuestro ejecutable. Una vez terminados los comandos, se generará un archivo ejecutable como el siguiente:



Autómata Finito Determinista



	@	+	-	*	/	>	<	=	()	;	,	:	&		~	A-Z	ASCII*	NO TOKEN**
Q0	Q1	Q3	Q3,Q14	Q3	Q3,Q9	Q4	Q4	Q5	Q6	Q6	Q6	Q6	Q7	Q8	Q8	Q8	Q17	Q16	Q16
Q1	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q2	Q16	Q16
Q2	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q2	Q16	Q16
Q3	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16
Q4	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q5	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16
Q5	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16
Q6	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16
Q7	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q6	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16
Q8	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16
Q9	Q16	Q16	Q16	Q16	Q10	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16
Q10	Q11	Q11	Q11	Q11	Q11,Q16	Q11	Q11	Q11	Q11	Q11	Q11	Q11	Q11	Q11	Q11	Q11	Q11	Q11	Q16
Q11	Q11	Q11	Q11	Q11	Q11,Q12	Q11	Q11	Q11	Q11	Q11	Q11	Q11	Q11	Q11	Q11	Q11	Q11	Q11	Q16
Q12	Q16	Q16	Q16	Q16	Q13	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16
Q13	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16
Q14	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16
Q15	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16
Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16
Q17	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q16	Q17	Q16	Q16

***ASCII abarca todos los caracteres del teclado.**

****NO TOKEN se refiere a cualquier carácter que no genera token dentro del autómata, se incluye BCO TAB ENOL ENOF.**

Expresiones regulares de los componentes léxicos

Identificadores:

@[a-zA-Z]+

Operadores aritméticos:

"-" | "+" | "/" | "*"

Operadores relacionales:

"<" | ">" | ">=" | "<=" | "="

Operadores lógicos:

"&" | "|" | "~"

Palabras reservadas:

"Program" | "begin" | "end" | "input" | "output" | "integer" | "real" | "char" | "string" |
"boolean" | "if" | "else" | "then" | "while" | "do" | "repeat" | "until" | "var" | "true" U
"false"

Caracteres especiales:

"(" | ")" | "," | "." | "="

Comentarios:

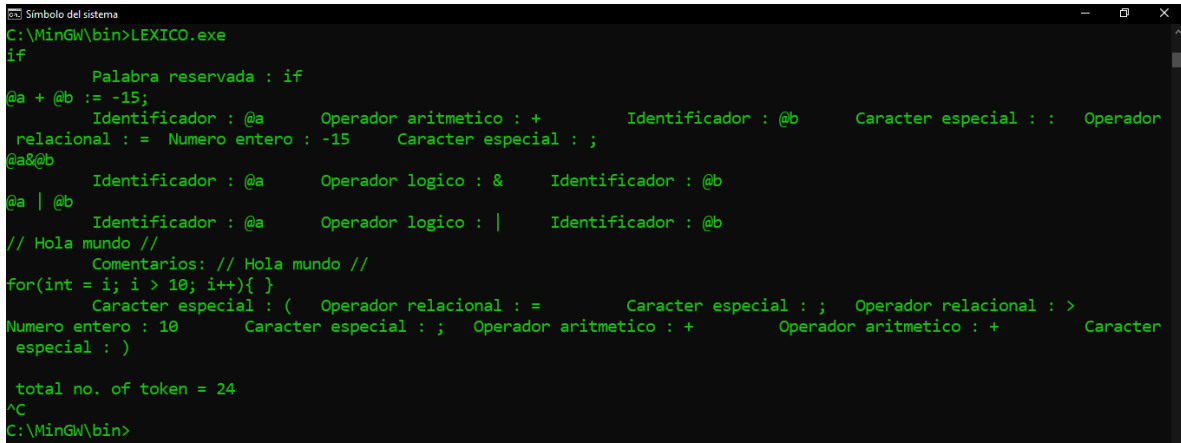
"//[A-Za-z0-9]*"//"

Numeros enteros:

[0-9]+ | -[0-9]+

Analizador de categorías léxicas realizado en Flex:

El programa es capaz de tanto analizar tokens que sean introducidos, como los de un archivo de texto, puede ser ejecutado dentro del cmd o en su propio ejecutable, a continuación se muestra la lectura de tokens a través de teclado, para terminar de introducir se presiona “Ctrl+C” y muestra el total de tokens analizados:



```
Simbolo del sistema
C:\MinGW\bin>LEXICO.exe
if
    Palabra reservada : if
@a + @b := -15;
    Identificador : @a    Operador aritmetico : +    Identificador : @b    Caracter especial : :    Operador
relacional : =    Numero entero : -15    Caracter especial : ;
@a&@b
    Identificador : @a    Operador logico : &    Identificador : @b
@a | @b
    Identificador : @a    Operador logico : |    Identificador : @b
// Hola mundo //
    Comentarios: // Hola mundo //
for(int = i; i > 10; i++){ }
    Caracter especial : (    Operador relacional : =    Caracter especial : ;    Operador relacional : >
Numero entero : 10    Caracter especial : ;    Operador aritmetico : +    Operador aritmetico : +    Caracter
especial : )

total no. of token = 24
^C
C:\MinGW\bin>
```

Para hacer la prueba de lectura del archivo, primeramente debe existir un archivo de texto con un código de programación en la ubicación C: \MinGW\bin\:

// Prueba del programa //

```
integer @a = 0
```

```
integer @b = 0
```

```
integer @c = 0
```

```
if( @a & @b <= 10){
```

```
@c = -5;
```

```
} else if (@a | @b = 5 ){
```

```
@c = 10 + @d
```

```
}
```


Dentro del cmd, nos posicionamos en la ubicación de nuestro ejecutable y hacemos lo siguiente:

```
Símbolo del sistema
C:\MinGW\bin>LEXICO.exe < codigo.txt
```

El comando “LEXICO.exe < código.txt” permitirá al ejecutable leer el texto dentro del archivo, sucediendo lo siguiente:

```
Símbolo del sistema
C:\MinGW\bin>LEXICO.exe < codigo.txt
Comentarios: // Prueba del programa //

Palabra reservada : integer      Identificador : @a      Operador relacional : =      Numero entero : 0
Palabra reservada : integer      Identificador : @b      Operador relacional : =      Numero entero : 0
Palabra reservada : integer      Identificador : @c      Operador relacional : =      Numero entero : 0

Palabra reservada : if  Caracter especial : (  Identificador : @a      Operador logico : &      Identificador :
@b  Operador relacional : <=      Numero entero : 10      Caracter especial : )

Identificador : @c      Operador relacional : =      Numero entero : -5      Caracter especial : ;

Palabra reservada : else      Palabra reservada : if  Caracter especial : (  Identificador : @a      Operador
logico : |      Identificador : @b      Operador relacional : =      Numero entero : 5      Caracter especial : )

Identificador : @c      Operador relacional : =      Numero entero : 10      Operador aritmetico : +
Identificador : @d

total no. of token = 39
```

Analizará los tokens y gráficamente los mostrará además de su total, cabe resaltar que componentes que no generen token no serán analizados, como se puede ver a continuación:

```
C:\MinGW\bin>LEXICO.exe
ñ
EOF
BCO
allsadla
%
-
Operador aritmetico : -
total no. of token = 1
```

Conclusiones

Conclusión de Patricia:

En esta práctica fue un reto para mí en lo personal ya que no conocía nada de la herramienta Flex y al hacer un autómata con conjunto con todos los componentes léxicos que se hicieron fue algo difícil pero no imposible, tengo practica con los componentes léxicos, junto con las expresiones pero se complicó más de lo que pensaba ya que me revolví a la hora de llenar la tabla.

Conclusión de Mayra:

El desarrollo del autómata finito determinista fue muy difícil de definir pero tampoco era imposible ya que solo se necesitaba mirar desde otra lógica para poderlo realizar y cumplir con los requisitos de que si es finito determinista. Aprendí y reforcé lo que había visto en la práctica 3 cuando usábamos JFlap para los autómatas.

Conclusión de Jesús:

Aprendí a cómo utilizar Flex y el crear un ejecutable de un código me podrá ser muy útil para otras asignaturas, además de eso, la programación en C no es muy difícil, solo que como observación se debe respetar mucho lo que escribas en las expresiones regular, ya que un mínimo detalle es capaz de alterar toda la lógica de tu código; pero por lo general, la práctica me pareció no difícil pero si muy laboriosa.

Bibliografía:

<http://es.tldp.org/Manuales-LuCAS/FLEX/flex-es-2.5.html>

[https://es.wikipedia.org/wiki/Lex_\(inform%C3%A1tica\)](https://es.wikipedia.org/wiki/Lex_(inform%C3%A1tica))

http://webdiis.unizar.es/asignaturas/LGA/material_2004_2005/Intro_Flex_Bison.pdf

http://repositori.uji.es/xmlui/bitstream/handle/10234/5998/Primera_Practica_IS17_Curso_06_07.pdf?sequence=1&isAllowed=y

<http://ri.uaemex.mx/bitstream/handle/20.500.11799/59149/selection.pdf?sequence=3&isAllowed=y>