Reflexión actividad integradora 3.4

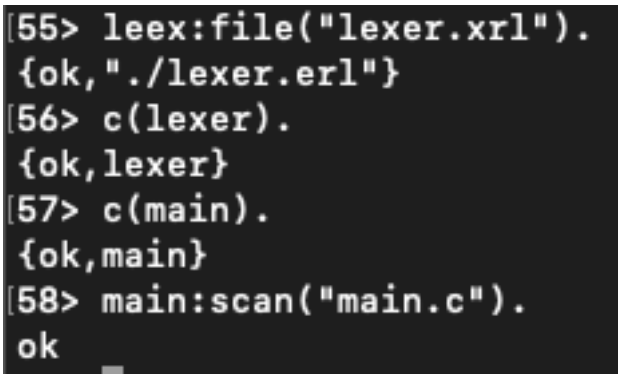
Considerando que se utilizó para la solución de esta primera parte de las actividades integradoras una solución con el uso de un DFA (autómata finito determinista), la complejidad de ejecución de este es de tipo O(|n|). Esto significa que se trata de la cardinalidad de la entrada, de la longitud de la cadena a analizar. Esto sucede ya que se usa el reconocimiento de varias categorías léxicas varias veces en el programa.

Podemos hacer una comparación ya que la creación de un DFA es mucho más costosa que la de un NFA (autómata finito no determinístico), pero los tiempos de ejecución igual cambian. La complejidad en ejecución de este último es de tipo $O(|x| * (n + m))$, donde ***n*** es el número de estados y ***m*** el número de aristas. Pudiéndose observar que tiene una mayor complejidad que el DFA utilizado en la solución. Por lo tanto, aunque la complejidad de su creación es mayor, esto se compensa en la ejecución de éste.

Por otro lado, se puede considerar la complejidad del algoritmo según cada grupo léxico. Podemos identificar varios grupos como operadores, literales, comentarios, palabras reservadas, entre otros. En el grupo de los operadores podemos identificar símbolos con una longitud de 1, por lo tanto, se tiene una complejidad $O(1)$ o constante; mismo caso para los símbolos especiales. En el caso de las palabras clave o reservadas (como, *while*, *if*, etc…), identificadores o variables, literales numéricas (como, "12", "0.45", etc), comentarios ("//abc\n"), como literales de cadena ("abc"); su complejidad depende directamente de la longitud de la cadena, es decir del número de caracteres que se analizan. Por lo tanto, estos grupos tienen la misma complejidad de $O(|n|)$.

El análisis léxico es el corazón y la base de los programas, por lo que una correcta identificación de los tokens es vital para el funcionamiento correcto del programa. Por lo tanto, si se comete un error entre identificar una palabra reservada o una variable; los siguientes pasos dejan de funcionar correctamente. Asimismo, los programadores debemos garantizar que el software desarrollado cumpla con las especificaciones y haga su función de manera correcta y utilizando el menor número de recursos posible. Por ejemplo, el desarrollo de equipo médico, donde se debe tener cuidado ya que una falla del código puede resultar en el costo de vidas humanas. Por lo tanto, se debe cuidar la calidad de los productos desarrollados desde las partes más básicas como el análisis, y en este caso, el resaltado de los tokens.

Para correr el programa se debe tener instalado *erlang* en la computadora, y abrir una terminal en el directorio donde se encuentren los archivos. Primero, se transforma a .erl el .xrl con "leex:file("lexer.xrl").", posteriormente, se compila con "c(lexer).". Después se compila el main con "c(main).". Finalmente se manda a llamar la función scan dentro del archivo main, con el nombre del archivo en c para leer; por ejemplo: "main:scan("main.c").".
Ejemplo para correrlo en la terminal:



El archivo de salida se creó en la misma carpeta con el nombre “salida.html”.