**Universidad Simón Bolívar**

**CI3725: Traductores e Interpretadores**

**Integrantes: Gustavo Gutiérrez 11-10428**

**José Pascarella 11-10743**

**Informe Etapa 1**

**Interprete de BOT**

**Introducción**

Para esta primera etapa del proyecto se pidió desarrollar un *Lexer* para el lenguaje BOT especificado en el enunciado del Proyecto. Para la realización de esta tarea se dio a escoger entre un conjunto de lenguajes de programación y librerías. El objetivo de este informe es exponer el lenguaje y las herramientas utilizadas así como explicar brevemente la implementación del Lexer. Además se incluyen las respuestas de la revisión Teórico-Práctica.

**Herramientas Escogidas**

Se decidió utilizar *Ruby* como lenguaje de programación para el desarrollo del proyecto. Las razones de esta elección fueron meramente académicas, pues existe interés por parte de los integrantes del equipo de aprender a utilizar dicho lenguaje. Debido a esto y al hecho de que no existen librerías recomendadas por los profesores del laboratorio para Ruby el Lexer fue implementado desde cero.

**Diseño de la Solución**

El Lexer implementado toma como entrada el contenido del archivo a procesar como un solo string e itera sobre este hasta que se encuentre vacío. El proceso de cada iteración es el siguiente:

- Se compara el inicio del string con las expresiones regulares de cada uno de los tokens posibles.

- Una vez reconocido el caso se crea el objeto Token correspondiente y se agrega a la lista de Tokens. Se elimina el substring que ya fue procesado, se actualizan los contadores de línea y columna y se continúa con la siguiente iteración.

- En caso que ninguna regla concuerde se reporta un error y se elimina el carácter del inicio del string para seguir iterando.

- Existe una regla que reconoce un error. Esta corresponde al caso borde de los Identificadores no válidos (aquellos que empiezan con [0-9\_]). En caso de caer en esta regla se reporta un error y se consume el identificador completo.

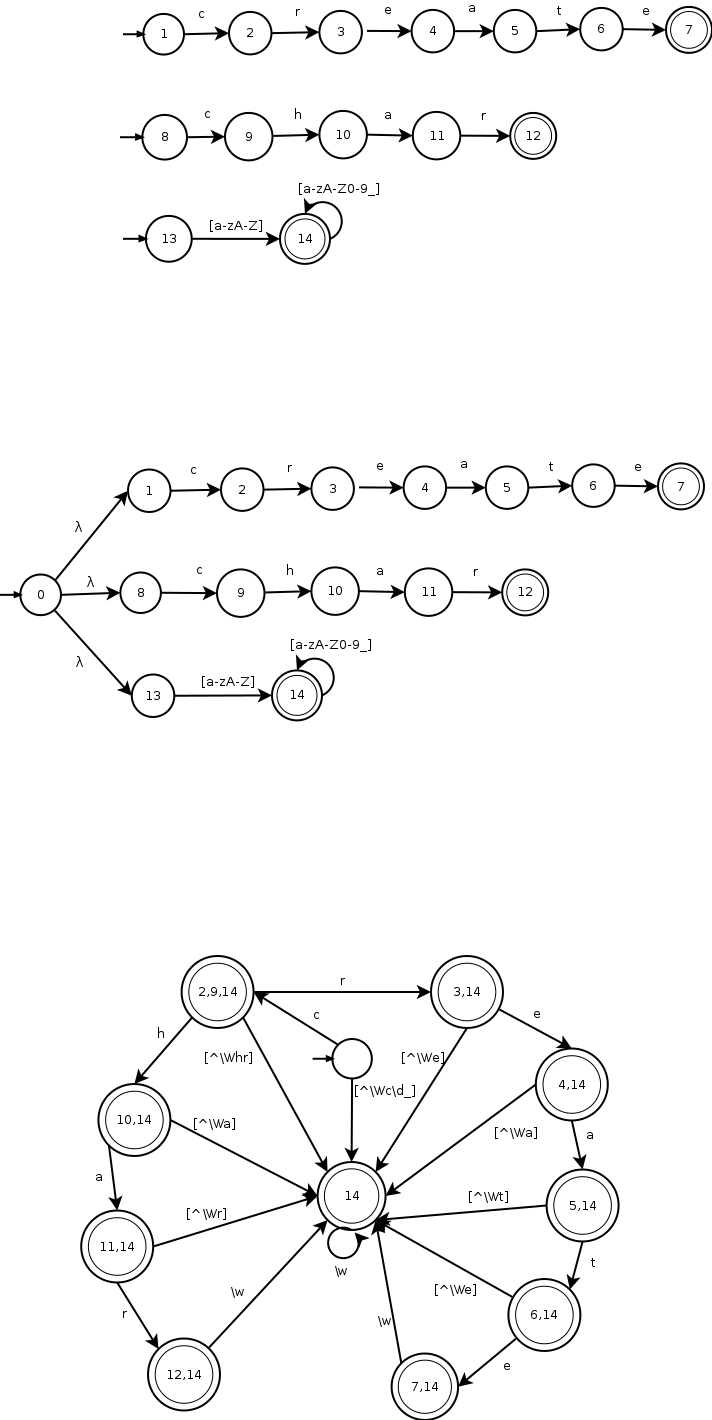
Una vez procesado el string completo se devuelve la lista de tokens en caso de que no haya existido un error. Si se encontraron errores se devuelve *nill.*

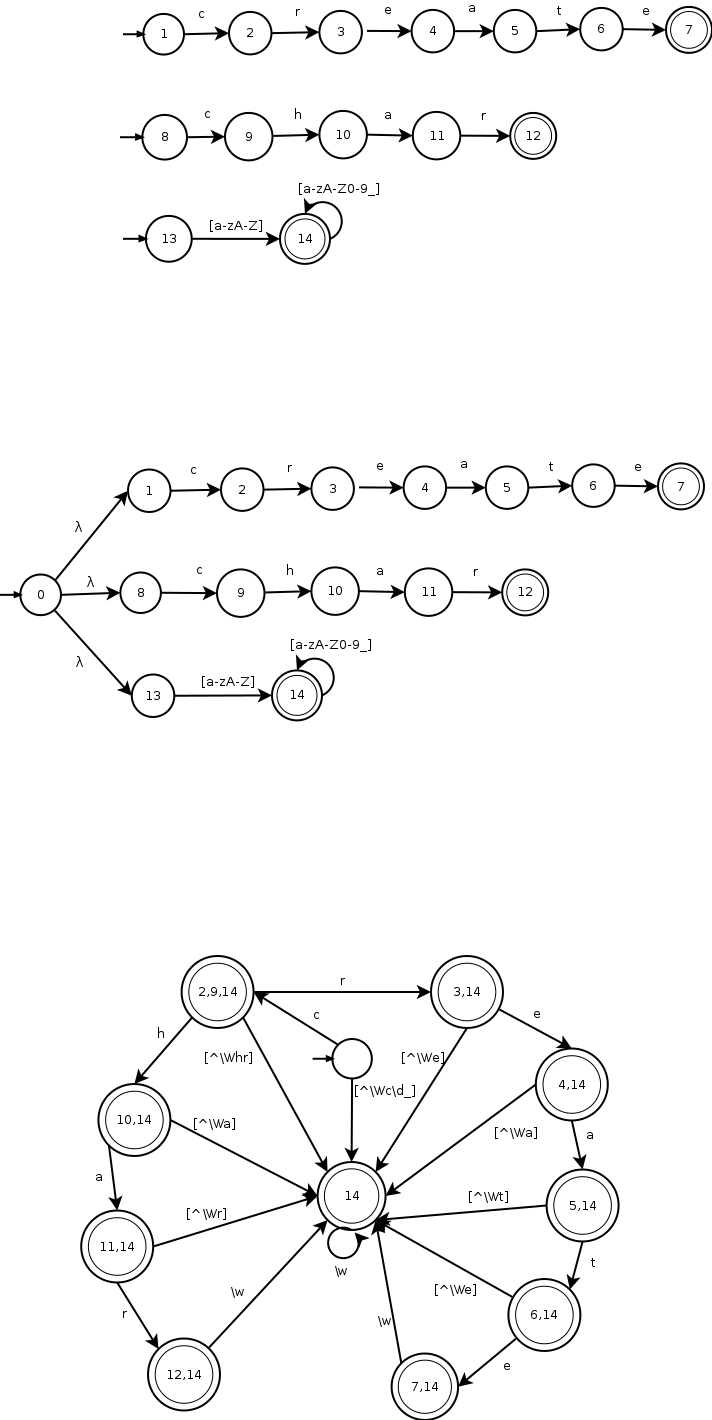
**Respuestas Teórico-Prácticas**

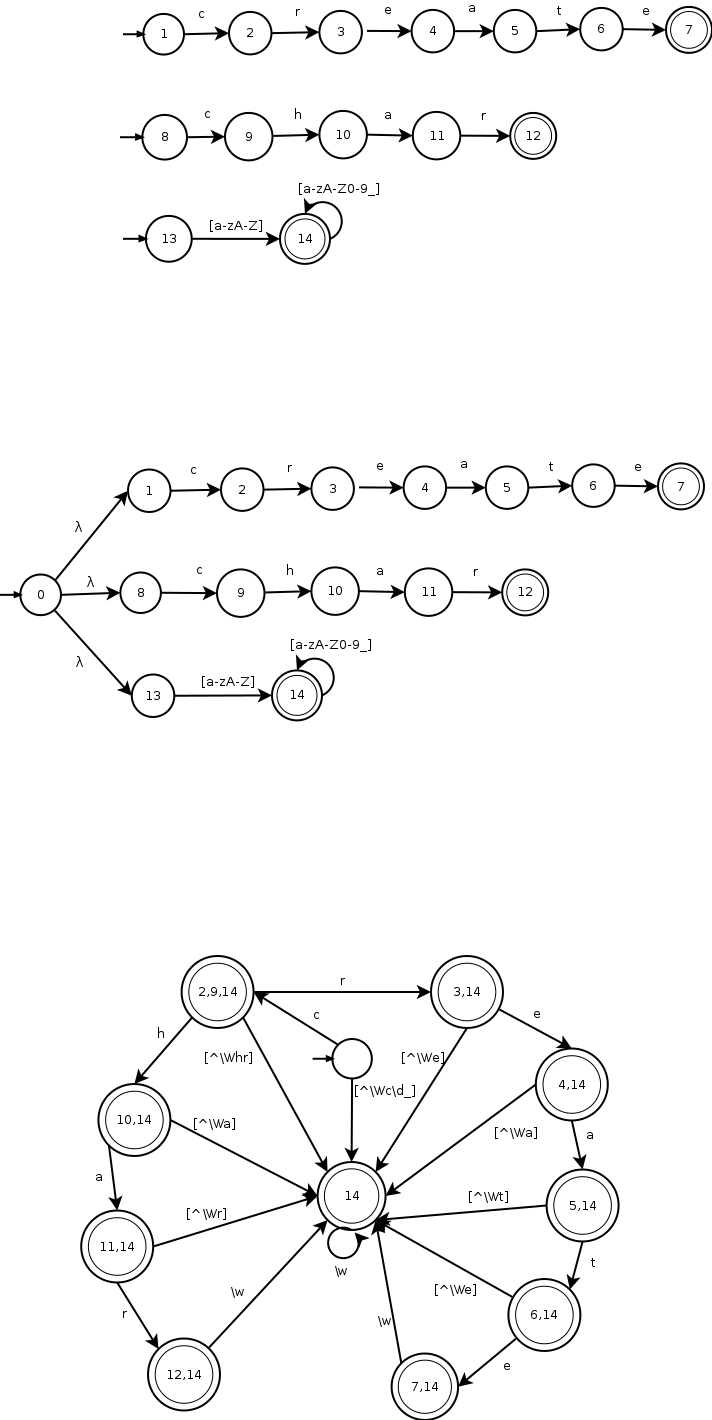
1. A) create

B) char

C) [a-zA-Z][a-zA-Z0-9\_]

1. 



1. El estado 7 corresponde al primer lenguaje; el estado 12 corresponde al segundo lenguaje; finalmente el estado 14 corresponde al tercer lenguaje.
2. Existen dos palabras que generan conflicto. Éstas son *créate* que puede ser reconocida en L1 y en L3 y la palabra *char* que puede ser reconocida en los lenguajes L2 y L3
3. 
4. En el autómata *Mdet* se refleja la ambigüedad en los estados finales [12,14] y [7,14] debido a que estos conjuntos de estados contienen estados finales de *M*  que corresponden a dos lenguajes distintos.
5. - [14] -> L3

- [2, 9, 14] -> L3

- [10, 14] -> L3

- [11, 14] -> L3

- [12, 14] -> L2

- [3, 14] -> L3

- [4, 14] -> L3

- [5, 14] -> L3

- [6, 14] -> L3

- [7, 14] -> L1

1. Al realizar el algoritmo de *Mmin* sobre el autómata de la respuesta 6 se obtiene el mismo autómata. Esto quiere decir que dicha solución era la solución mínima. Del mismo modo la relación de los estados finales y los lenguajes a los que representan es la misma que la expuesta en la pregunta 8.
2. A la hora de implementar el Lexer también se presentó el problema de tokens que pudieran ser reconocidos por más de un patrón. Para solucionar las ambigüedades se evaluaron primero las expresiones regulares que correspondieran a las soluciones de mayor prioridad, en otras palabras, se estableció un orden lineal.