Universidad Simón Bolívar

Departamento de Computación y Tecnología de la Información

Traductores e Interpretadores

Trimestre Abril-Julio 2018

PROYECTO: FASE I

Aurivan Castro 14-10205

Sandra Vera 14-11130

Sartenejas, 16 de mayo de 2018

Desarrolo de Analizador Lexicográfico

Con el objetivo de realizar un interpretador de un nuevo lenguaje, en el presente trabajo se expone la manera de abordar el desarrollo de un analizador lexicográfico, generando tokens o errores estáticos dependiendo del archivo de entrada para el analizador.

Planteamiento general

Se ha solicitado la creación de un analizador lexicográfico para el lenguaje “Fortran”, del cual se proveen tanto las palabras claves como los caracteres que deben ser reconocidos como token. Se requiere, además, que el programa sea capaz de reconocer ID de variables declaradas dentro del input dado.

Como requerimiento adicional, el programa debe poder ejecutarse por medio de un script que reciba como argumento el archivo que se va a procesar.

De la implementación

Debido a que el presente proyecto fue desarrollado en Python, fue necesaria la librería PLY. Los tokens fueron definidos con expresiones regulares, uno por uno, o dentro de un diccionario en el caso de las palabras reservadas, dependiendo de su uso en el lenguaje especificado en el enunciado. Las palabras reservadas se colocaron aparte con la finalidad de que se reconocieran por sí solas y no repetidas (Ex. “while” debe ser reconocida como palabra reservada, pero “whilewhile” no).

Aunado a lo anterior, algunos tokens requirieron funciones para su definición, puesto que las expresiones regulares por sí solas no bastaban para definirlos. Entre ellos se encuentran t_TkNum para lo digitos y t_TkId para los ID. Aparte de esto, fue necesario desarrollar funciones auxiliares para cumplir los requerimientos del enunciado, por ejemplo, una para listar los tokens según su tipo, a manera de imprimirlos una vez procesado todo el input.

A medida que se van reconociendo los tokens, el main.py los va agregando a una lista llamada tokensList. En caso de existir errores, tales como caracteres especiales no permitidos, se agrega un string con el error correspondiente en la lista tokError. En caso de que la lista de errores tenga al menos un error, se imprime ésta; y de lo contrario, se imprime la lista de tokens.

Acerca de la ejecución del programa

A manera de facilitar la corrida del programa, se creó un script que acepta el nombre del archivo a procesar y ejecuta el lexer. Por ejemplo, si el input está en 'input.txt', entonces el programa puede correrse de la siguiente manera:

```
./Lex input.txt
```

De darse problemas en este punto, significa que el script no tiene permisos de ejecución. Para concederle los permisos necesarios, puede ejecutarse el siguiente comando:

```
chmod 777 Lex
```

Obstáculos y otras circunstancias

Puesto que la librería PLY hace la mayor parte del trabajo, no se presentaron sino un par de problemas al momento de implementar. En primer lugar, dado que Python usar el backslash ("\") para reconocer caracteres especiales en expresiones regulares, al principio se encontraron muchos inconvenientes para reconocer el TkConjuncion ("/\") y el TkDisyuncion ("\/"). Luego de arregladas las expresiones regulares que definen estos dos tokens, a pesar de reconocer de manera correcta estos tokens por sí solos, los encontraba a su vez en input como "\/\" en vez de lanzar un error. Una vez discutido con el preparador de la materia, se concluye que es una falla del lenguaje.

En segunda instancia, se presentó la disyuntiva del reconocimiento de input tales como "45hola" como dos tokens, TkNum(45) y TkId("hola"), o como un error. Después de discusiones con otros equipos de trabajo y el preparador, se ha llegado al acuerdo de que la primera opción es la correcta, pues se trata de un análisis sólo al nivel lexicográfico.

Conclusiones y recomendaciones

Un analizador lexicográfico representa la fase inicial del procesamiento de un código dado en un particular lenguaje, encargado de reconocer y retornar los diferentes tokens definidos para el mismo. Debe a su vez retornar error en caso de existir caracteres no reconocidos.

De la experiencia adquirida en esta fase, puede afirmarse que el programa ha sido, en general, sencillo de implementar, sobre todo con el apoyo de la librería dada. Sin embargo, con el objeto de evitar errores arriba mencionados (respecto al uso de "\") es recomendable probar con otros lenguajes y librerías, con el objeto de reforzar la calidad del compilador en desarrollo.