UNIVERSIDAD TECNOLÓGICA DE PEREIRA

SEBASTIAN ARDILA AGUDELO

DANNY ALEJANDRO VILLEGAS OCAMPO

Compiladores

08/03/2016

PEREIRA – RISARALDA – COLOMBIA

DOCUMENTACIÓN DEL ANALIZADOR LÉXICO

En esta documento se hará una descripción para la segunda entrega, la cual corresponde al analizador sintáctico, comenzaremos por especificar que el proyecto está divido en 3 partes, primero encontramos el analizador léxico donde están definidos los tokens, la gramática y sus respectivas reglas, la segunda parte consta del analizador sintáctico con las producciones que se tradujeron a partir de la gramática del lenguaje que está construido en la forma Backus-Naur a forma normal, en dicha analizador se encuentran todos los llamados que se hacen a nuestra clase AST. La tercera parte consta posteriormente de la clase AST, donde podemos encontrar las clases principales que nos mostrarán específicamente el árbol gramatical.

Hablando un poco más detalladamente de la clase AST, dentro del código fuente encontramos la clase “Node”, la cual nos imprimirá en la función “postorden” todos los hijos y su respectivo recorrido uno por uno, si llegado el caso y se llega a un punto donde solo se tiene un solo hijo, imprimirá dicho hijo y evitará seguir con el ciclo.

Por otro lado encontramos la clase ID, el cual es un identificador del tipo de dato. También tenemos una clase Null, la cual imprime que el nodo produce vacío o que el nodo es nulo.

La identación de todo el programa AST, fue desarrollado con identacion para distinguir entre los padres y sus respectivos hijos.

En el analizador sintáctico se hace el llamado a la clase AST para la construcción del mismo, específicamente en la linea 1471, la identificamos con el nombre de “imprimirPostOrden(" ")”. Antes de esto se genera el archivo ParseOut para saber cuantos estados generamos y el numero de estados, en nuestro caso depende de la cantidad de test que hayamos construido para probar el analizador sintáctico.

Ya para finalizar la explicación debemos mencionar, que se incluimos una clase adicional que se llama “buscarFicheros()”, la cual lista los test que realizamos, los muestra en pantalla y posteriormente podemos elegir una de ellas. En la tupla llamada precedencia se incluyó la prioridad de algunos operandos.

**Problemas encontrados:**

Respecto a problemas encontrados, podemos mencionar que al a hora de imprimir algunos hijos en la clase AST, tuvimos problemas porque dichos hijos estaban definidos en unicode y otros en float, por lo cual fue necesario agregar esos tipos de datos y e igualar los datos que nos estaban llegando en la función “Node”a los ya mencionados tipos de datos (unicode y float) para su respectiva impresión.

**Bugs:**

A la hora de crear la función que lee los archivos, no nos funcionaba al principio porque se perdía el formato “unicode”a pesar de se hubiesen definido las librerías, para ello, fue necesario importar la librearía “codecs” e implementar una linea de código para que funcionara