PRÁCTICA OBLIGATORIA. PROCESADORES DE LENGUAJE.

Nombres de los autores:

Isaac Lozano Osorio

Jorge Prieto Gómez

Índice

1. Léxico

En la parte léxica de la práctica no hemos tenido especial problema simplemente hemos tenido que tener cuidado con la recuperación de errores. En la última línea del '*.flex*' reconocemos todo aquel elemento que no pertenezca al resto por lo cual es un error. También como no hay que reescribir los comentarios en el nuevo documento no le hemos asignado un token específico ni a el comentario de una línea ni al de múltiple.

2. Sintáctico

A partir de este punto vimos que al ser muchos archivos los que teníamos que modificar, se podía facilitar todo mucho más si utilizáramos un IDE, por lo que creamos un proyecto de NetBeans. Además, a la hora de compilar y generar los diferentes archivos, nos facilitó bastante el avance de la práctica el haber creado un fichero *BAT* con las directrices para compilar y generar tanto el *flex* como el *cup*. Además, redirigimos la salida a un fichero llamado '*sal.txt*' para que sea más sencillo observarla.

A la hora de tener la gramática en el fichero '*.cup'*, tuvimos unos errores que en los operandos, al no saber cuál tenía precedencia. Esto lo supimos realizando el árbol sintáctico. Una vez detectado el fallo buscamos en internet algunos métodos de solución y dimos con los *precedence*. Lo cual consiguió solucionar el error.

Otro error que tuvimos fue a la hora de introducir lambda dentro del CUP como Tvacia, siendo este un terminal, nos daba errores que simplemente se solucionaron quitando el terminal Tvacia.

Por último, hemos generado un archivo *'.bat'* para ejecutar todos los casos de prueba obteniendo sus salidas en los ficheros por si resulta más útil.

3. Semántico

Por último, se añadieron las diferentes acciones semánticas, y el diagrama de clases necesario para poder subir todo, tuvimos algunos problemas debido a un bucle que genera el *flex* con el *eofval,* ya que cuando leía el último token cerraba directamente la ejecución del programa no permitiendo que volviese a subir por el árbol, el problema se solucionó eliminando el *eofval*. Para poder tener todas las declaraciones en una misma línea se almacenan todos los tipos en un *ArrayList* en el cual dependiendo del tipo se almacenan y al final se imprimen todas, tanto para enteros como reales. Para comprobar si es una librería o un programa, se tiene un valor booleano dentro de donde se detectan las funciones, y en tal caso de que exista una con un main se pone el valor a false y se considera arriba que es un programa, si no existe esta palabra se quedará a true y se tendrá como librería. A la hora de las funciones, en la lista de parámetros también se han agrupado de igual manera que se hace en las declaraciones de las variables. Relacionado con la tabulación, para las variables y constantes, se entendió, que se debe dar una tabulación antes del var, si fuera después del var es tan sencillo como cambiar el \t. Para que el archivo tenga el mismo nombre, en vez de redirigir la salida en el *flex*, se redirige en el analizador, cogiendo *argv[1]*, quitando la extensión *.c* y añadiendo *.pas*, de esta manera se cumple que tenga el mismo nombre.

4. Casos de prueba

Casos de prueba correctos

Ejemplo: *ConstantesLiterales_Comentarios*

En este ejemplo probamos las constantes literales y comentarios, con varios tipos de ejemplos, separados en varias líneas los comentarios múltiples.

Un caso importante es utilizar un comentario dentro de una constante literal debería reconocer todo como la constante literal y no separar en comentario y constante literal

Ejemplo: *ConstantesNumericas*

Probamos todos los tipos de constantes numéricas existentes para su correcto funcionamiento

Ejemplo: *PalabrasReservadas_Identificadores*

Comprobación de todas las palabras reservadas e identificadores

Ejemplo: *VariosTipos*

Es un caso de ejemplo largo en el cual se prueban diversos casos que se darán en un futuro en nuestro código para ver el correcto funcionamiento de todos juntos.

Casos de prueba incorrectos

Ejemplo: *ErrorConstantesLiterales*

Errores en constantes literales

Ejemplo: *ErrorConstantesNumericas*

Errores en constantes numéricas

Ejemplo: *ErrorIdentificadores_PalabrasReservadas*

Errores en identificadores y palabras reservadas

Ejemplo: *ErrorNuevosTipos*

Errores generales

5. Cambios realizados

Para la entrega de junio hemos realizado la series de cambios que especificamos a continuación. Primero de todo hemos compactado correctamente las declaraciones de variables *int* y *float* manteniendo la estructura de agrupar correctamente las variables cuando son seguidas, es decir, si aparece, *int float int*, sería *int float int*, sin agrupar. Además, hemos comprobado que la variable en los *for* es la misma, quitado los parentesis cuando no eran necesarios en la lista de parámetros o llamadas a funciones, puesto los ":" delante de algunos = que les faltaban, tabulado correctamente el programa generado y puesto \t después de const.

También hemos añadido la recuperación de errores en *SENTLIST* y en *PARAMLIST*, añadido la línea y columnas en flex para detectar donde se ha fallado.

6. Bibliografía

http://pages.cs.wisc.edu/~fischer/cs536.f12/lectures/Lecture19.4up.pdf

http://spiegel.cs.rit.edu/~hpb/public_html/Lectures/20012/LP/all.pdf

http://cs.haifa.ac.il/courses/compilers/BILAL/Tutorials/JLex_CUP_tools.pdf