David Vacas Miguel (GII+GIS)

Pedro Redondo Rabanal (GII+GIS)

PROCESADORES DE LENGUAJES

Traducción dirigida por la sintaxis

**Índice**

1. Introducción (pág. 2)
2. Especificación de la traducción dirigida por la sintaxis (pág. 3-4)
3. Casos de prueba (pág. 5)
4. Conclusión (pág. 6)

**1. Introducción**

En esta práctica debemos realizar un procesador de una modificación del lenguaje Pascal. En esta segunda entrega hemos realizado la parte correspondiente al traductor dirigido por la sintaxis. Para la realización de la práctica hemos utilizado la última versión del enunciado; en el cual, se arreglaban ciertas erratas que había en este.

La aplicación resultante se ha comprobado su funcionamiento en Windows 10, con dos jdk distintos, uno version 1.8.0\_73 y otro 1.8.0\_60, y el archivo .html resultante en Chrome.

**2. Especificación de la traducción dirigida por la sintaxis**

Para la realización del traductor dirigido por sintaxis hemos creado una nueva clase llamada Nt; la cual, tiene dos atributos de tipo String: value y valueNoHtml.

* El atributo value se le asigna un valor cuando se inicializa una nueva instancia de Nt, en este valor se va guardando la estructura html como por ejemplo “<h1>Programa: "+valorid+"</h1>”.
* El atributo valueNoHtml es solamente utilizado en los identificadores existentes en las cabeceras de las funciones y procedimientos que vayamos a poner en la lista inicial, para que así no tengan éstas el ancla, sino que el ancla se encuentre en la cabecera que tenga seguidamente sus sentencias correspondientes.

Para la realización correcta de los sangrados de las sentencias hemos utilizado dos variables globales las cuales son dos contadores que vamos aumentando si nos encontramos algunas sentencias como ‘begin’, ‘case’, etc. Para poder aumentar dichos contadores cada vez que viésemos alguna de estas sentencias, hemos tenido que modificar en ciertos puntos el sintáctico añadiendo un estado auxiliar debido a que cup es ascendente. Una vez encontramos alguna otra sentencia como podría ser ‘end’ disminuimos dichos contadores.

Por otro lado, tenemos dos variables globales de tipo String llamadas tipos y funciones, donde vamos guardando las declaraciones de variables y tipos en tipos, y las de funciones y procedimientos en la variable funciones. Estas variables las utilizamos para separar dichas declaraciones para luego colocarlas en el orden correspondiente, en el caso de que no vengan; es decir, primero funciones y procedimientos y finalmente las declaraciones de variables y tipos seguidas de las sentencias del programa principal.

Para poder realizar las lista de enlaces en la parte superior, nos ayudamos de un método llamado listaCabeceras(); el cual, nos devuelve un String con el formato html correspondiente utilizando una variable global donde vamos guardando dichas cabeceras.

Finalmente, hemos realizado un método llamado writeHtml(String código); en el cual, creamos el fichero .html y en este escribimos el String resultante del código ya parseado junto con la estructura general de un html. Para poner en tittle y como nombre del fichero de salida, el nombre del fichero entrante, hemos creado una variable estática en la clase que junta el analizador léxico y sintáctico, en nuestro caso llamada PL.java, donde guardamos el nombre que proviene de los parámetros de entrada. Para poder utilizar la variable estática, creamos una nueva instancia de dicha clase y cogemos ese valor.

**3. Casos de prueba**

Los casos de prueba al ya estar enunciados y descritos en la memoria de la entrega anterior hemos decidido no añadirlos en ésta también, igualmente han sido añadidos en una carpeta con el resto de archivos necesarios.

**4. Conclusión**

Como comentario final, nos gustaría añadir que no hemos realizado de nuevo la memoria de la parte del sintáctico ya que son cambios mínimos y creemos que con la explicación dada en el punto dos es suficiente. Por otra pensamos gracias a esta parte hemos aprendido bastante mejor como funcionan los traductores dirigidos por la sintaxis ascendentes y como poder ajustarlos para que funcionen como esquemas de traducción y hagan acciones entre medias añadiendo estados auxiliares.