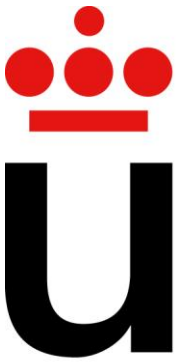


Universidad
Rey Juan Carlos

Procesadores Lenguajes – Práctica Obligatoria

DAVID GARCÍA CAVA Y DIEGO PICAZO GARCÍA

DAVID GARCÍA CAVA Y DIEGO PICAZO GARCÍA

Contenido

Autores

La realización de la práctica obligatoria ha sido llevada a cabo por *David García Cava* y por *Diego Picazo García*. La elaboración de dicho traductor ha sido en conjunto, ya que así hemos podido compartir diferentes puntos de vista acerca de cómo plantear distintos problemas que nos han ido surgiendo (como por ejemplo quitar la recursividad por la izquierda en un gran número de producciones) durante la realización de la práctica, ya que en situaciones concretas es mejor contar con la opinión de otro compañero que con documentación abstracta.

Descripción de la práctica

La práctica ha consistido en la creación de un procesador de lenguaje de programación (C) mediante la implementación de una gramática en ANTLR4 compuesta principalmente por un analizador léxico y sintáctico.

La práctica incluye la presentación de ocho casos de prueba, cuatro de los cuáles son correctos y cuatro son incorrectos. Estos casos de prueba se utilizan para comprobar el comportamiento del procesador y verificar que cumple con los objetivos de la práctica.

Los casos de prueba correctos incluyen tanto ejemplos simples como más complejos de código que contienen funciones, variables y estructuras de control como *if-else* (entre otras). Además, se ha incluido el ejemplo de prueba proporcionado por los responsables de la práctica para demostrar que el procesador cumple con los objetivos de esta.

Los casos de prueba incorrectos muestran errores en el código que son detectados por el procesador, ya estén contemplados o no de forma explícita, pero que en ciertas ocasiones no impiden que se genere el código HTML correspondiente, lo que significa que nuestro procesador ofrece recuperación de errores frente a problemas léxicos, notificándonos por pantalla como advertencia, mientras que frente a problemas sintácticos se ha seguido un criterio básico donde si los errores no evitan llegar al final de fichero, entonces se produce recuperación de errores, y en el caso contrario, se aborta y se notifica el problema.

Por último, un par de anotaciones acerca del funcionamiento de la aplicación:

- **Funcionalidad de forma gráfica y portable con el JAR y un archivo de input cualquiera.**
- **El archivo de input y el JAR pueden tener ubicaciones distintas y cualesquiera.**
- **Es necesario introducir rutas absolutas cuando se solicitan.**
- **El archivo HTML de salida se genera en la misma ruta donde se encuentra el fichero de input que se introdujo.**

Casos de prueba

A continuación, se muestran cuatro casos de prueba correctos y otros cuatro incorrectos permitiendo la observación del comportamiento del procesador, como indica el enunciado de la práctica.

- ## Caso correcto 1:

  Como primer caso correcto, hemos decidido incluir el ejemplo de prueba que nos proporcionaron en la práctica para demostrar que nuestro procesador cumple con los objetivos mínimos sin ningún tipo de problema.

- ## Caso correcto 2:

  Como segundo caso correcto, hemos decidido añadir más contenido (funciones, declaración de funciones, bucles, condiciones anidadas…) cambiando un poco el ejemplo de prueba para ver si de verdad está funcionando todo como debería.

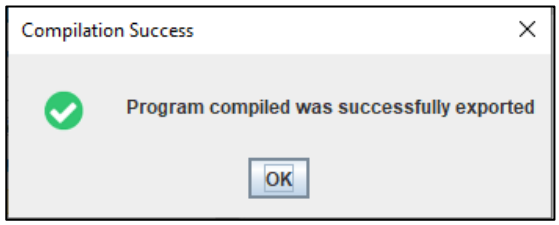
- ## Caso correcto 3:

  Como tercer caso, sencillamente añadimos un if-*else* (incluidos en la parte opcional), así como una estructura de datos personalizada en un *struct*, y comentarios al código para probar todo tipo de nuevas funcionalidades.

- ## Caso correcto 4:

  Y el último, un tanto más avanzado, con estructuras de control anidadas, así como bucles de distinto tipo junto con comentarios y funciones, englobando un poco todo lo visto hasta ahora.

Cabe destacar que todos los casos correctos pasan de forma satisfactoria los validadores del procesador, sin ninguna advertencia adicional, generando código HTML correspondiente y avisando al usuario con un mensaje de éxito.



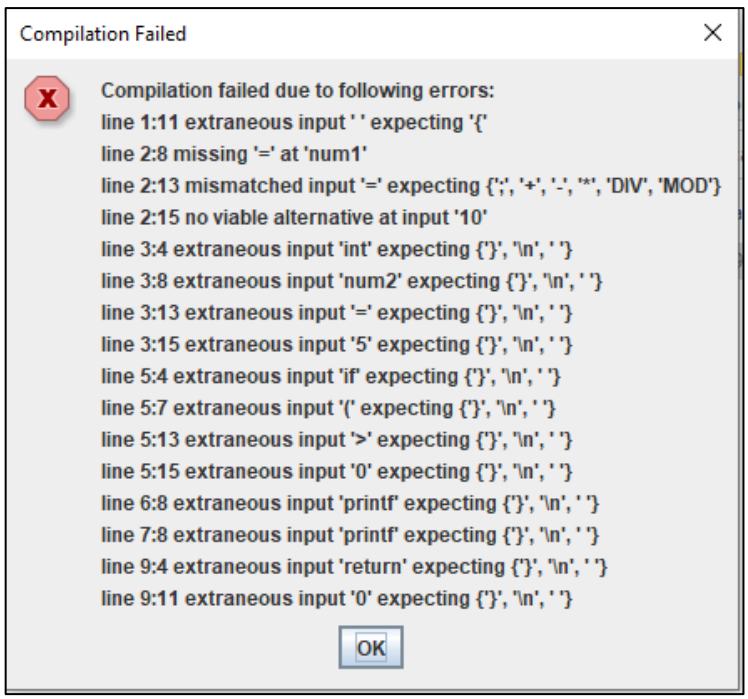
De forma adicional, se han considerado como casos erróneos aquellas entradas al procesador que, a pesar de generar un código HTML final, ha provocado advertencias adicionales, así como recuperación de errores, dentro de la misma categoría de aquellos errores sintácticos o léxicos de los que el procesador no puede de ninguna forma recuperarse, saltando algún tipo de excepción:

- ## Caso incorrecto 1:

  Como primer caso, nos encontramos sin poder realizar recuperación de errores ya que la acumulación de fallos en distintas funciones y operadores nos llevan a abortar la compilación.

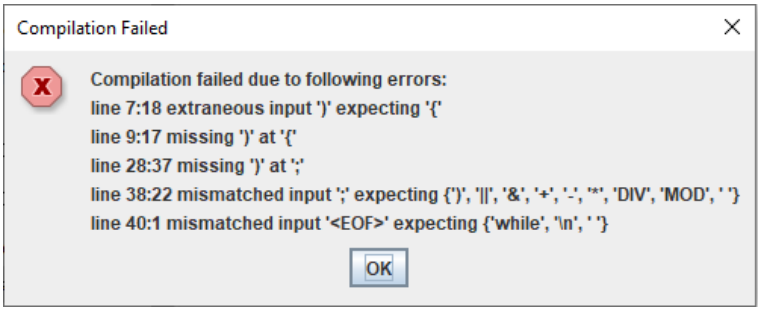
- ## Caso incorrecto 2:

Otro caso incorrecto, en el cual suceden errores varios que impiden la compilación del código entre ellos la falta de operadores, así como de cierres de condiciones. No hay recuperación de errores porque es un error crítico.



- ## Caso incorrecto 3:

Como tercer caso incorrecto, ofrecemos un caso donde la recuperación de errores es imposible, provocado por la cantidad de errores y la complejidad de éstos produce que no se llegue al fin de fichero, lo que lleva a que la compilación falle y se arrojen los errores oportunos.



- ## Caso incorrecto 4:

Como último caso incorrecto, hemos seleccionado un caso que genera un error crítico , al no permitir (debido a las especificaciones de la gramática) introducir un *if* con una sola acción sin la necesidad de poner llaves como se podría hacer en cualquier lenguaje de alto nivel de programación. Además, la compilación falla debido al número de errores que se generan, siendo como máximo 5 aceptables.

Consideraciones

Hemos realizado las siguientes consideraciones de cara a la práctica, sobre todo de cara a la parte opcional:

- **Se han añadido reglas adicionales y modificado varias, así como recursiones para favorecer la compilación e interpretación del código. Se han añadido las respectivas anotaciones en el código.**
- **Respecto a la parte opcional, se ha logrado cubrir su totalidad, destacando nuestra interpretación de los Ids y referencias a cabeceras y declaraciones como si fueran conjuntos dentro de otros, es decir, tal y como hemos seguido en el enunciado, si una variable se encontraba dentro del método Main, entonces tendría una referencia "programa_principal:Main:nombre_de_la_variable" correspondiente.**
- **Cabe destacar que la build funciona de forma estable en Windows 10 probado, y se puede encontrar en la ruta out/artifacts/PracticaPL_jar.**
- **Los códigos de prueba se encuentran en la ruta src/test/resources.**