

Procesadores Lenguajes – Práctica Obligatoria

DAVID GARCÍA CAVA Y DIEGO PICAZO GARCÍA

DAVID GARCÍA CAVA Y DIEGO PICAZO GARCÍA

Contenido

[Autores 2](#_Toc134986786)

[Descripción de la práctica 2](#_Toc134986787)

[Casos de prueba 2](#_Toc134986788)

[Caso correcto 1: 2](#_Toc134986789)

[Caso correcto 2: 2](#_Toc134986790)

[Caso correcto 3: 2](#_Toc134986791)

[Caso correcto 4: 2](#_Toc134986792)

[Caso incorrecto 1: 3](#_Toc134986793)

[Caso incorrecto 2: 3](#_Toc134986794)

[Caso incorrecto 3: 3](#_Toc134986795)

[Caso incorrecto 4: 3](#_Toc134986796)

# 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 abstract****a.**

# 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 la misma.**

**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. (Importante: No usar la ‘X’ para salir, usar el botón de ‘Cancel’)**
* **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…) a vuestro ejemplo de prueba para ver si de verdad estaba funcionando todo como debería.**

## Caso correcto 3:

**Como tercer caso, sencillamente añadimos un if-else (incluidos en la parte opcional) y comentarios al código para probar este tipo de 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.**

Interfaz de usuario gráfica, Texto, Aplicación, Word, Correo electrónico

Descripción generada automáticamente

**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:

**REDACTAR**

## Caso incorrecto 2:

**Otro caso incorrecto, operador ‘%’, el cual no estaba implementado en la práctica, al meter una porción de código de una práctica de gestión de hilos y procesos de otra asignatura, nos salta este fallo. Pero al no ser un error crítico, se alcanza el fin de fichero y se genera el HTML correspondiente.**

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

## 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.**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

## Caso incorrecto 4: