# TRABAJO PRÁCTICO 1

**Técnicas de Compilación**

**Fecha:** 20/06/2022

**Integrantes:** Szulman Pablo, Tello Eric.

**Temas:** Reglas de compilación, expresiones regulares, reglas semánticas y sintácticas.

**Profesor:** Maximiliano Andrés Eschoyez.

**Herramientas:** ANTLR – Visual Estudio Code – Java.

**Consigna:** Dado un archivo de entrada en lenguaje C, se debe generar como salida el Árbol Sintáctico (ANTLR) correcto y mostrar por consola o archivo el contenido de la Tabla de Símbolos de cada contexto. Esto es válido únicamente cuando el archivo de entrada es correcto. Para esta versión se deberá realizar un control de errores básicos y generar un reporte de lo encontrado para poder analizar archivos con posibles errores de codificación. El reporte de errores podrá realizarse por consola o archivo de texto. Los errores a detectar deben ser los siguientes: Errores sintácticos comunes:
• Falta de un punto y coma,
• Falta de apertura de paréntesis,
• Formato incorrecto en lista de declaración de variables Errores semánticos comunes:
• Doble declaración del mismo identificador,
• Uso de un identificador no declarado,
• Uso de un identificador sin inicializar,
• Identificador declarado pero no usado,
• Tipos de datos incompatibles.
Cada error reportado debe indicarse si es sintáctico o semántico.

# DESARROLLO Y CONCLUISIONES

Para encarar la segunda parte del trabajo hemos tenido que cambiar las expresiones regulares usadas, ya que nuestro árbol no estaba “armado” de manera correcta y había conflictos a la hora de tomar una entrada de texto, ya que se rompía nuestro .g4 en varios casos por no abordar de forma óptima nuestras primeras expresiones regulares.

Hemos tenido que pedir ayuda a un compañero para volver a generar nuestras expresiones regulares de forma óptima y desde cero ya que era imposible encarar dicho práctico con nuestras expresiones, teníamos mal la declaración de funciones, parámetros, entre otros.

Hemos comprendido la “funcionalidad de un compilador” y como desarrollar nuestro programa para adaptarlo a un lenguaje requerido, jugar con als expresiones regulares y métodos implementados y entender el recorrido de la interpretación de un lenguaje.

Una vez mejorado nuestro .g4, procedemos a crear nuestro propio Listener sobrescribiendo métodos para detectar el correcto funcionamiento (estructura) de un código.

La complicación mayor fue el uso de elementos propios de java como Listas, HashMap, implementación de métodos como super(), ya que no estamos acostumbrados al lenguaje, pero gracias a un amigo que tiene más experiencia nos pudo explicar y ayudar a diseñar nuestro métodos, junto con nuestra tablas de símbolos para detectar cada instrucción, o error común en el programa y mostrar también la línea donde ocurren los mismos.

Otra complicación fue la parte de los parámetros, las funciones y sus debidos “return”, pero después de mucho trabajo nos sentimos muy nutridos tanto en la utilización de java, como en la comprensión de cómo puede funcionar un compilador, las reglas que se siguen, como detectarlas, detectar contextos, y errores comunes.