# Técnicas de Compilación

Francisco Ameri Lopez Lozano

Trabajo Final — 2025

## Resume

Diseñar e implementar un compilador para un subconjunto del lenguaje C++ utilizando la herramienta ANTLR4, aplicando los conceptos teóricos y prácticos vistos en la materia de Técnicas de Compilación.

## Descripción

El estudiante deberá desarrollar un compilador completo que sea capaz de analizar, verificar y generar código para programas escritos en un subconjunto del lenguaje C++. El compilador deberá implementar todas las fases del proceso de compilación: análisis léxico, análisis sintáctico, análisis semántico, generación de código intermedio y optimización.

## Modalidad de Trabajo

- Trabajo en Equipo: El proyecto deberá realizarse en equipos de dos (2) estudiantes.
- **Control de Versiones**: Todo el código del proyecto debe estar alojado en un repositorio de GitHub público.
- Commits Frecuentes: Se requiere que ambos integrantes realicen commits regulares que reflejen el progreso del desarrollo. No se evaluarán repositorios que contengan un solo commit o muy pocos commits concentrados cerca de la fecha de entrega.
- Distribución Equitativa: Ambos integrantes deben participar activamente en el desarrollo, lo cual debe reflejarse en la historia de commits del repositorio.

# Funcionalidades Requeridas

## 1. Análisis Léxico

- Implementar un analizador léxico utilizando ANTLR4 que reconozca los tokens del lenguaje.
- Identificar y reportar errores léxicos.
- Generar una tabla de tokens.

## 2. Análisis Sintáctico

- Implementar un analizador sintáctico utilizando ANTLR4 que verifique la estructura gramatical del programa.
- Construir un árbol de sintaxis abstracta (AST).
- Identificar y reportar errores sintácticos.
- Visualizar el árbol sintáctico generado.

## 3. Análisis Semántico

- Implementar un analizador semántico que verifique la coherencia semántica del programa.
- Construir y mantener una tabla de símbolos.
- Verificar tipos de datos y compatibilidad en operaciones.
- Verificar el ámbito de las variables y funciones.
- Reportar errores semánticos (con detalles específicos).
- Distinguir entre errores (críticos) y warnings (no críticos).

## 4. Generación de Código Intermedio

- Implementar un generador de código de tres direcciones.
- Manejar expresiones aritméticas y lógicas.
- Manejar estructuras de control (if-else, bucles).
- Manejar llamadas a funciones y retorno de valores.

## 5. Optimización de Código

- Implementar al menos tres técnicas de optimización, que pueden incluir:
    - Propagación de constantes
    - Eliminación de código muerto

    - Simplificación de expresiones
    - Eliminación de subexpresiones comunes
    - Optimización de bucles

### 6. Salidas del Compilador

- Generar archivos de salida para el código intermedio y optimizado.
- Implementar un sistema de reporte de errores y warnings que utilice colores para diferenciarlos (verde para éxito, amarillo para warnings, rojo para errores).

## Subconjunto del Lenguaje C++ a Implementar

### Tipos de Datos

- int
- char
- double
- void (para funciones)

### Estructuras de Control

- Condicionales (if-else)
- Bucles (for, while)
- Sentencias de control de bucle (break, continue)

### Elementos del Lenguaje

- Declaración de variables
- Declaración de funciones
- Expresiones aritméticas y lógicas
- Llamadas a funciones
- Retorno de valores
- Asignaciones

## Evaluación

Se evaluará el trabajo considerando:

1. Corrección y completitud de la implementación
2. Manejo adecuado de errores y warnings
3. Calidad del código generado
4. Efectividad de las optimizaciones implementadas
5. Calidad de la documentación y del informe técnico
6. Ejemplos de prueba que demuestren las capacidades del compilador
7. **Historial de commits en GitHub: Se evaluará la frecuencia, distribución y calidad de los commits realizados por ambos integrantes del equipo**
8. Participación equitativa: Ambos miembros deben demostrar participación activa en el desarrollo

# Entregables

## 1. Repositorio de Código

- Código fuente del compilador
- Gramática ANTLR4 utilizada
- Ejemplos de programas de prueba
- URL del repositorio GitHub donde se ha desarrollado el proyecto

## 2. Informe Técnico (en formato PDF)

El informe debe incluir:

- Portada: Incluir título del trabajo, nombres de los integrantes, materia, profesor y fecha.
- Introducción: Descripción general del compilador desarrollado y sus objetivos.
- Análisis del Problema: Especificación del subconjunto de C++ implementado.
- Diseño de la Solución:
    - Arquitectura general del compilador
    - Descripción de cada fase de compilación
    - Decisiones de diseño tomadas
- Implementación:
    - Detalles técnicos de la implementación
    - Descripción de la gramática ANTLR4
    - Detalles de la tabla de símbolos
    - Algoritmos utilizados en cada fase
    - Técnicas de optimización implementadas
- Ejemplos y Pruebas:

    - Casos de prueba significativos
    - Salidas generadas
    - Análisis de resultados
- Dificultades Encontradas y Soluciones Aplicadas
- Conclusiones
- Referencias Bibliográficas
- Anexos (si son necesarios)

### 3. Manual de Usuario

- Instrucciones de instalación
- Guía de uso del compilador
- Ejemplos de compilación de programas
- Interpretación de los mensajes de error y warning

## Fecha de Entrega

[Fecha específica según el calendario académico]

## Restricciones y Consideraciones

- El compilador debe ser desarrollado en Java, utilizando ANTLR4 como herramienta de generación de analizadores léxicos y sintácticos.
- Se debe realizar una demostración del funcionamiento del compilador con ejemplos representativos.
- El código debe estar adecuadamente comentado y seguir buenas prácticas de programación.
- **No se aceptarán entregas que no cumplan con los requisitos de trabajo en equipo y gestión de versiones** (repositorio con commits regulares de ambos integrantes).
- La última actualización del repositorio debe realizarse antes de la fecha y hora límite de entrega.
- El informe técnico debe ser claro, completo y reflejar el trabajo realizado por ambos integrantes. La calidad del informe será un componente importante en la evaluación final.

*Nota: La entrega de este trabajo requiere demostrar tanto el dominio técnico como la capacidad de trabajar colaborativamente utilizando herramientas de control de*

*versiones profesionales. El historial de commits debe reflejar el desarrollo incremental del proyecto, con contribuciones equilibradas de ambos integrantes del equipo. El informe técnico debe demostrar la comprensión profunda de los conceptos de compilación y la capacidad de comunicar adecuadamente las decisiones técnicas tomadas.*