

# Construcción de Compiladores - Laboratorio Guiado No. 2

Gabriel Brolo, Bidkar Pojoy

Julio de 2024

## 1. Introducción

En este laboratorio, aprenderemos a usar ANTLR para compilar un lenguaje de programación. ANTLR (ANother Tool for Language Recognition) es una herramienta poderosa para generar analizadores léxicos y sintácticos. Vamos a crear un lenguaje simple y compilaremos un programa usando ANTLR.

## 2. Instrucciones

En este lab se van a familiarizar con ANTLR. Los hemos ayudado proporcionando un Dockerfile que facilita la generación de un ambiente de ANTLR. Les dejo unos pasos para correrlo y también para correr la gramática con Python, ya que es más fácil para pruebas pequeñas y sin tener que tener un entorno de Java para compilar las cosas.

- En el root de este lab deben crear un container interactivo en dónde van a correr los comandos de ANTLR:

### Comando Docker

```
docker build --rm . -t lab2-image &&  
docker run --rm -ti -v "$(pwd)/program":/program lab2-image
```

- En este caso creamos un volumen de Docker en `/program` que es en donde se encuentra la gramática de ANTLR. Asimismo, se encuentra un archivo `Driver.py` que sirve como un Main para importar las clases de Lexer y Parser generadas por ANTLR y poder analizar archivos para validarlos sintácticamente con la gramática que proporcionamos.
  - Aquí es donde después, conforme avance el curso, ustedes van a generar inclusive otras clases con ANTLR, ya sean Visitors o Listeners y van a aplicar un análisis semántico.
- Dentro de esta carpeta también se encuentra el archivo `program_test.txt` el cual será el que vamos a parsear para analizar si es válido con nuestra gramática o no.
- Luego que corran el comando de Docker, deberán de generar los archivos de Lexer y Parser de ANTLR con el siguiente comando:

### Comando ANTLR

```
antlr -Dlanguage=Python3 MiniLang.g4
```

- Luego, usan el Driver para analizar el archivo de prueba:

Comando Driver

```
Driver.py program_test.txt
```

- Si el archivo se encuentra sintácticamente correcto, no verán un output en la consola. Por el contrario si hay un error sintáctico/léxico, ANTLR se los hará saber.
- Y listo, con esto ya tienen un playground inicial para probar archivos con ANTLR.
- Para sus proyectos, se recomienda que extiendan este ambiente para sostener una arquitectura más robusta, este solo es un ejemplo básico.

### 3. Sin usar Docker

Si deciden no usar Docker, deberán instalar ANTLR manualmente en sus computadoras y ejecutar el jar, para lo que deberán tener el runtime de Java instalado para poder ejecutar el jar.

### 4. Actividades con ANTLR

1. Cree un programa que asigne un valor a una variable.
2. Cree un programa que realice una operación aritmética simple.
3. Experimente con expresiones más complejas.
4. Modifique el lenguaje para incluir la asignación de variables con expresiones aritméticas.
5. Agregue manejo de errores al compilador para detectar tokens inválidos en el programa fuente.
6. Cree un programa que utilice paréntesis para cambiar la precedencia de operadores.
7. Extienda el lenguaje para soportar comentarios de una sola línea.
8. Agregue operadores de comparación (`==`, `!=`, `<`, `>`, `<=`, `>=`) al lenguaje.
9. Cree un programa que utilice operadores de comparación.
10. Extienda el lenguaje para soportar estructuras de control como `'if'` y `'while'`.
11. Cree un programa que utilice una estructura `'if'`.
12. Cree un programa que utilice una estructura `'while'`.
13. Agregue soporte para funciones definidas por el usuario.
14. Cree un programa que defina y llame a una función.
15. Implemente un sistema de tipos básico que, además de incluir enteros, también incluya cadenas (yo sé que todavía no sabe que es un sistema de tipos, de manera formal, pero para este laboratorio, busque, investigue, haga algo simple, no necesitamos *rocket science todavía* y la respuesta es bastante trivial y a nivel de producción en la gramática).

## 5. Refreshando la memoria... y atando cabos con ANTLR

1. ¿Qué es un analizador léxico?

**Respuesta:** Un analizador léxico, o lexer, es un componente del compilador que convierte una secuencia de caracteres de entrada en una secuencia de tokens.

2. ¿Qué es un analizador sintáctico?

**Respuesta:** Un analizador sintáctico, o parser, es un componente del compilador que analiza una secuencia de tokens para determinar su estructura gramatical de acuerdo con una gramática formal.

3. ¿Qué son las reglas léxicas en ANTLR?

**Respuesta:** Las reglas léxicas, o lexer rules, definen cómo los caracteres de entrada se agrupan en tokens.

4. ¿Qué son las reglas sintácticas en ANTLR?

**Respuesta:** Las reglas sintácticas, o parser rules, definen cómo se estructuran los tokens para formar construcciones sintácticas válidas en el lenguaje de programación.

5. ¿Cómo maneja ANTLR los errores léxicos?

**Respuesta:** ANTLR puede personalizar el manejo de errores léxicos sobrescribiendo los métodos de la clase base del lexer para reportar y recuperar errores.

6. ¿Cómo maneja ANTLR los errores sintácticos?

**Respuesta:** ANTLR permite manejar errores sintácticos definiendo un ‘error handler’ personalizado o utilizando el manejo de errores por defecto que proporciona mensajes de error detallados.

7. ¿Qué es un árbol de sintaxis abstracta (AST)?

**Respuesta:** Un AST es una representación jerárquica de la estructura sintáctica de un programa, que abstrae detalles de la sintaxis concreta y se centra en la estructura lógica del código.

8. ¿Cómo se puede visualizar un AST generado por ANTLR?

**Respuesta:** Utilizando la opción ‘-gui’ de ‘grun’, se puede visualizar el AST generado por ANTLR en una interfaz gráfica.

9. ¿Qué es la precedencia de operadores y cómo se maneja en ANTLR?

**Respuesta:** La precedencia de operadores determina el orden en que se evalúan las operaciones. En ANTLR, se maneja definiendo las reglas sintácticas de manera que las operaciones con mayor precedencia se evalúan primero.

10. ¿Qué es una gramática libre de contexto?

**Respuesta:** Una gramática libre de contexto es un tipo de gramática formal en la que las reglas de producción se pueden aplicar independientemente del contexto de los símbolos no terminales.

## 6. Entregables

- Documento PDF con capturas de pantalla de la ejecución del ambiente en Docker, o de su propio entorno, para realizar las actividades solicitadas.
- El documento debe de incluir el enlace a un repositorio privado de Github con todo el código utilizado para la realización de este laboratorio guiado. Recuerde que debe de compartir todos los repositorios utilizados en este curso con el catedrático y los auxiliares. La información de los usuarios se encuentra en Canvas.

## 7. Rúbrica de Evaluación

Tarea	Puntos
Crear un programa que asigne un valor a una variable	5
Crear un programa que realice una operación aritmética simple	5
Experimentar con expresiones más complejas	10
Modificar el lenguaje para incluir la asignación de variables con expresiones aritméticas	10
Agregar manejo de errores al compilador para detectar tokens inválidos	10
Crear un programa que utilice paréntesis para cambiar la precedencia	5
Extender el lenguaje para soportar comentarios	10
Agregar operadores de comparación al lenguaje	10
Crear un programa que utilice operadores de comparación	5
Extender el lenguaje para soportar estructuras de control	10
Crear un programa que utilice una estructura 'if'	5
Crear un programa que utilice una estructura 'while'	5
Agregar soporte para funciones definidas por el usuario	10
Crear un programa que defina y llame a una función	5
Implementar un sistema de tipos básico	10
<b>Total</b>	<b>100</b>