

Universidad Mariano Gálvez de Guatemala

San Juan Sacatepéquez

Facultad de Ingeniería en Sistemas de Información

Compiladores

Ing. Miguel Catalán



Alexis Vidal Juarez Siguantay 7590-14-3421

Guatemala 23 de mayo del 2025

SimpleScript – Documentación Técnica

1. Definición del Lenguaje y Gramática

El lenguaje inventado **SimpleScript** está diseñado para la enseñanza de estructuras de control, expresiones lógicas y manipulación de variables. Su sintaxis es una mezcla entre pseudocódigo y estructuras similares a Pascal.

Características principales:

- Tipos de datos: entero, decimal, cadena, booleano
- Palabras reservadas: DEFINE, PRINT, FUNCTION, RETURN, IF, ELSE, ELSEIF, WHILE, LOOP, DO, THEN, END, AND, OR, NOT

Tokens (JFlex)

- Identificadores: `[a-zA-Z_][a-zA-Z0-9_]*`
- Números enteros: `[0-9]+`
- Decimales: `[0-9]+\.[0-9]+`
- Cadenas: `"([^\n"]*)"`
- Booleanos: `true, false`
- Operadores: `+, -, *, /, =, ==, !=, <, <=, >, >=`
- Delimitadores: `(,), ;, ,`

2. Implementación del Análisis Léxico y Sintáctico

Se implementó un analizador léxico en **JFlex** y un parser sintáctico en **CUP**, capaces de validar archivos .ss con estructuras del lenguaje SimpleScript.

Componentes implementados:

AnalizadorLexico.flex

<https://github.com/ajurezs3/compiladorTraductor/blob/main/analizador/src/main/jflex/SimpleScript.flex>

Parser.cup

<https://github.com/ajurezs3/compiladorTraductor/blob/main/analizador/src/main/cup/SimpleScript.cup>

Pruebas realizadas:

Agregar SimpleScript code:

```
FUNCTION sumar(a, b)
RETURN a + b;
END
DEFINE x = 10;
DEFINE y = 20;
DEFINE resultado = sumar(x, y);
PRINT "La suma es: ", resultado;
```

Enviar

Codigo valido.

Agregar SimpleScript code:

```
x = 5 + 3 * 2;
```

Enviar

Codigo valido.

Agregar SimpleScript code:

```
FUNCTION factorial(n)
  DEFINE resultado = 1;
  WHILE n > 1 DO
    resultado = resultado * n;
    n = n - 1;
  END
  RETURN resultado;
END

DEFINE num = 5;
DEFINE fact = factorial(num);
PRINT "El factorial de ", num, " es ", fact;
```

Enviar

Codigo valido.

Agregar SimpleScript code:

```
IF fact > 100 THEN
  PRINT "El resultado es mayor que 100";
ELSE
  PRINT "El resultado es menor o igual a 100";
END
DEFINE a = true;
DEFINE b = false;
IF a AND NOT b THEN
  PRINT "La condición lógica es verdadera";
ELSE
  PRINT "La condición lógica es falsa";
END
|
```

Enviar

Codigo valido.

3. Desarrollo de la interfaz web

Esta interfaz permite a los usuarios escribir y validar código en el lenguaje SimpleScript desde un entorno web amigable. Está conectada con el compilador desarrollado en Java (JFlex + CUP) a través de la ejecución de un archivo .jar, y muestra los resultados directamente en la vista

4. Versión Previa

Tecnologías utilizadas

- **Laravel 9** – framework PHP para backend y ruteo
- **Blade** – sistema de plantillas de Laravel
- **Tailwind CSS** – framework de estilos CSS moderno
- **DataTables** – tabla dinámica para ver historial de entradas
- **Java 21** – backend del compilador
- **JFlex y CUP** – herramientas para análisis léxico y sintáctico

Funcionalidad principal

Formulario para ingreso de código

- Vista principal (dashboard.blade.php) contiene un textarea
- El código es enviado al backend vía POST
- El controlador valida el contenido y lo redirige al analizador Java

Ejecución del compilador Java

- Laravel usa la clase `Symfony\Component\Process\Process`
- Se ejecuta `java -jar analizador.jar` y se le pasa el código por stdin
- Si el análisis es correcto, Laravel muestra el mensaje en verde
- Si hay errores, se muestra en rojo con detalle de línea/columna

Historial de códigos

- Los códigos analizados se almacenan en la base de datos (codes table)
- Se listan en una tabla HTML mejorada con **DataTables**
- Permite búsqueda y paginación automática