UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería

Proyecto 1

Samuel Argueta - 211024

Alejandro Martinez - 21430

GUATEMALA, 2 de septiembre de 2024

ÍNDICE

Video del funcionamiento

https://youtu.be/EYSZOiw0vXg

Ejecución del Compilador

Para ejecutar el compilador en un entorno de desarrollo integrado (IDE):

- Usa el comando: python3 Main.py.

Para ejecutar las pruebas preconfiguradas:

- Usa el comando: python3 Test.py.

Además, se puede pasar un argumento opcional --render para decidir si se guarda el árbol sintáctico como una imagen PNG:

- Ejemplo: python3 Test.py --render=False.

Arquitectura del Proyecto

El análisis semántico se realiza mediante el Semantic Analyzer, que es la implementación de la clase Visitor de ANTLR. Esta clase es responsable de rastrear los alcances, construir mapas, y eventualmente generar la Tabla de Símbolos.
El archivo Main.py se encarga de inicializar la interfaz gráfica de usuario (GUI), la cual incluye:

- Un editor de texto con resaltado de sintaxis para ingresar el código.
- Un registro que muestra toda la información relevante del backend y las operaciones del compilador.
- Tres tablas de símbolos, una para funciones, otra para variables y una última para clases, cada una en diferentes pestañas.
- El árbol sintáctico generado.

El código ingresado se preprocesa utilizando el Compiscript Lexer, luego pasa por el Compiscript Parser para generar el árbol sintáctico, que se puede guardar como una imagen (ej. Output/Syntax-Graph.png). Posteriormente, este árbol se pasa al Semantic Analyzer, que visita cada nodo para generar y mostrar los datos en la GUI.

Variables del Analizador Semántico

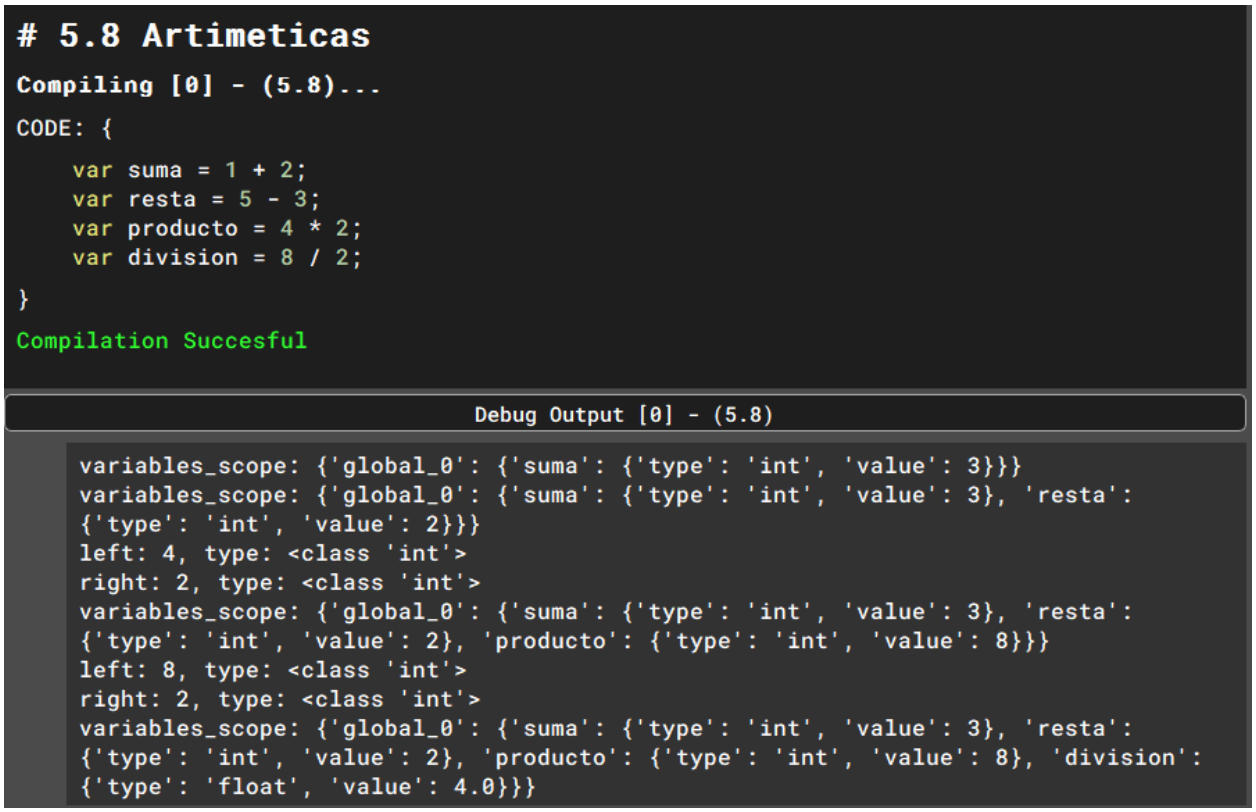
El archivo Semantic_Analyzer.py contiene variables clave para rastrear el alcance y almacenar la información necesaria durante la compilación, como:

- inside_loop: Para rastrear si se está dentro de un bucle.
- inside_block_fun_if: Para rastrear si se está dentro de un bloque de función o de una condición if.
- global_variables y local_variables: Para rastrear variables globales y locales.
- declared_functions: Para rastrear funciones declaradas.
- table_functions, table_variables, table_classes: Tres tablas de símbolos para funciones, variables y clases respectivamente.

Ejemplos de Salida del Análisis Semántico

El documento también incluye ejemplos visuales de la salida del análisis semántico, con imágenes que muestran el éxito o fracaso del proceso de compilación.
Para probar diferentes entradas, puedes ejecutar el archivo Test.py con ejemplos de prueba en los archivos Tests/Small_Tests.py y Tests/Large_Tests.py

Success



```
# 5.10 Logicos
Compiling [2] - (5.10)...
CODE: {
    var y = true and false ; // false
    var o = true or false ; // true
    var no = ! true ; // false
}
Compilation Succesful
```

Success (With Warning For Failure)

```
# 7.2 Ambito de las Variables
Compiling [10] - (7.2)...
CODE: {
    {
    var a = " dentro del bloque " ;
        print a ; // Imprime : dentro del bloque
    }
    print a ; // Error : a no esta definida
}
Compilation ''Succesful''(Should fail and did fail) {
    Variable 'a' no declarada en el ámbito scope1.
}
```

Failure

12.4 Funcion Recursiva

Compiling [32] - (12.4)...

CODE: {

```
    fun factorial ( n ) {
        if ( n <= 1) return 1;
        return n * factorial ( n - 1) ;
    }
        print " Factorial de 5: " + factorial (5) ; // Salida : Factorial de 5: 120
    fun fibonacci ( n ) {
        if ( n <= 1) return n ;
        return fibonacci ( n - 1) + fibonacci ( n - 2) ;
    }
    print " Fibonacci de 10: " + fibonacci (10) ; // Salida : Fibonacci de 10: 55
}
```

Compilation Failed {

```
    invalid literal for int() with base 10: 'n'
}
```

Debug Output [32] - (12.4)

```
variables_scope: {'global_0': {'suma': {'type': 'int', 'value': 3}}}
variables_scope: {'global_0': {'suma': {'type': 'int', 'value': 3}, 'resta':
{'type': 'int', 'value': 2}}}
left: 4, type: <class 'int'>
right: 2, type: <class 'int'>
variables_scope: {'global_0': {'suma': {'type': 'int', 'value': 3}, 'resta':
{'type': 'int', 'value': 2}, 'producto': {'type': 'int', 'value': 8}}}
left: 8, type: <class 'int'>
right: 2, type: <class 'int'>
variables_scope: {'global_0': {'suma': {'type': 'int', 'value': 3}, 'resta':
{'type': 'int', 'value': 2}, 'producto': {'type': 'int', 'value': 8}, 'division':
{'type': 'float', 'value': 4.0}}}
Datos después de eliminar paréntesis: ['3', '5']
variables_scope: {'global_0': {'menor': {'type': 'boolean', 'value': False}}}
Datos después de eliminar paréntesis: ['10', '10']
```