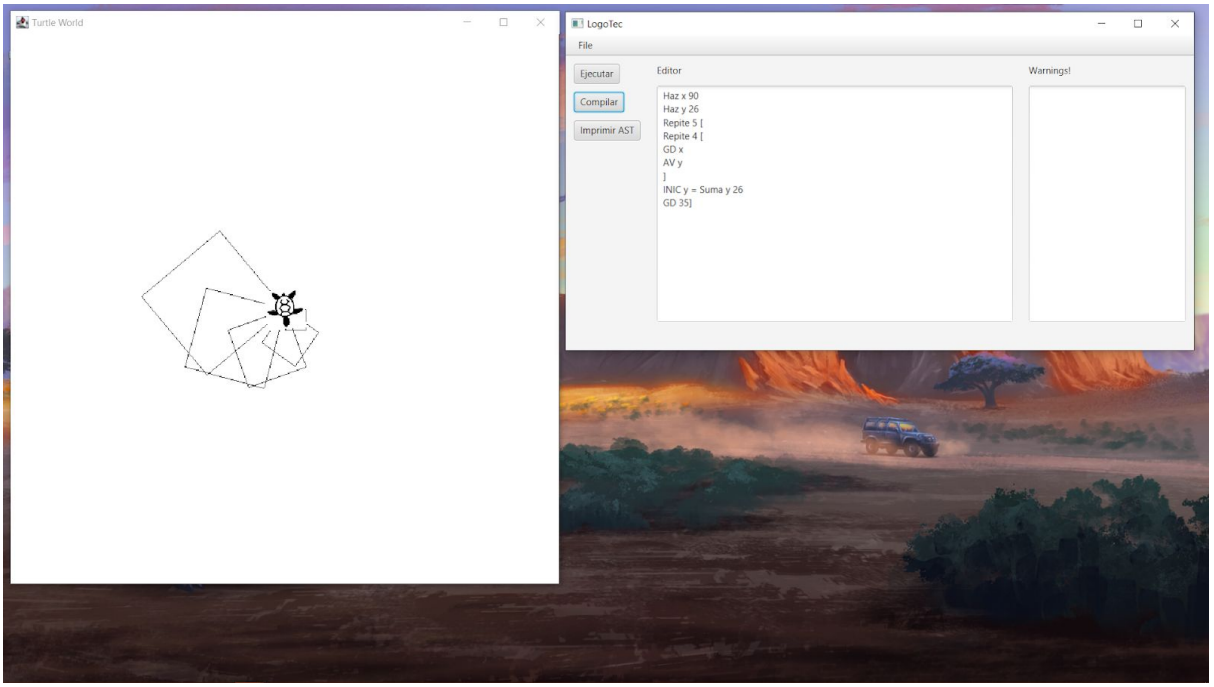
TEC | Tecnológico de Costa Rica

Área de Ingeniería en Computadores.
CE3104 – Lenguajes Compiladores e Intérpretes.
II Semestre 2020 Grupo 2.
Documentación externa correspondiente al proyecto LogoTEC.
Fecha: Lunes 18 de enero de 2021.
Miembros:

- José Fabián Mendoza Mata (2018319699).
- Sergio Ríos Campos (2019007977) .
- Santiago Brenes Torres (2019063875).
- Tomás Segura Monge (2018099729).

LogoTEC consiste en un lenguaje de programación inspirado en popular LOGO, original de los años 60 como un lenguaje con fines educativos. Al igual que su contraparte original, LogoTEC tiene la capacidad de ejecutar comandos para la declaración de variables así como ejecución de ciclos y condiciones booleanas. El principal objetivo es poder graficar diferentes figuras a partir de comandos de dirección como: “AVANZA”, “RETROCEDE”, “GIRADERECHA”, entre otros. Además, tiene la funcionalidad de desplegar el árbol de sintaxis del código escrito.


Turtle World
LogoTec
File
Ejecutar
Compilar
Imprimir AST
Editor
Warnings!
Haz x 90
Haz y 26
Repite 5 [
Repite 4 [
GD x
AV y
]
INIC y = Suma y 26
GD 35]

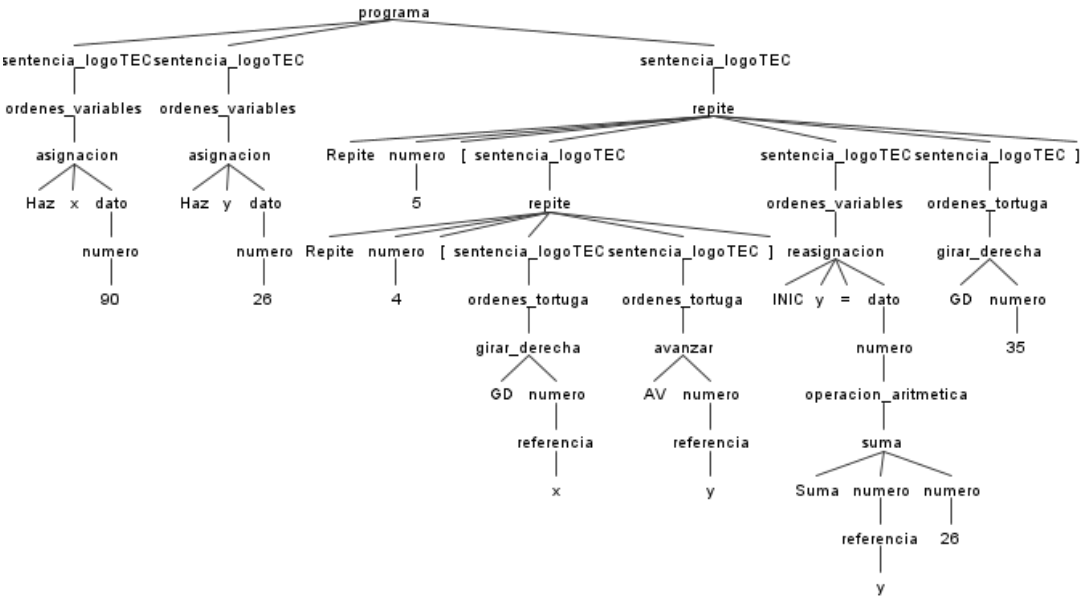
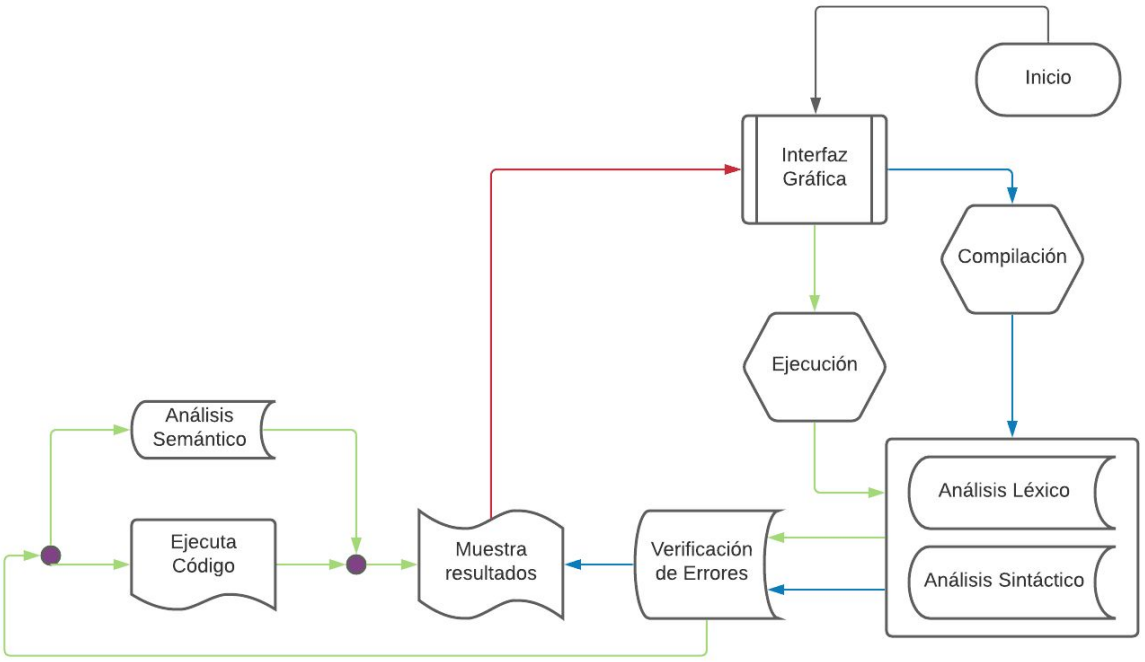

programa
sentencia_logoTEC
sentencia_logoTEC
sentencia_logoTEC
ordenes_variables
ordenes_variables
repite
asignacion
asignacion
Repite numero [sentencia_logoTEC
sentencia_logoTEC sentencia_logoTEC]
Haz x dato
Haz y dato
5
repite
ordenes_variables
ordenes_tortuga
numero
numero
Repite numero [sentencia_logoTEC
sentencia_logoTEC]
reasignacion
girar_derecha
90
26
4
ordenes_tortuga
ordenes_tortuga
INIC y = dato
GD numero
35
girar_derecha
avanzar
numero
GD numero
AV numero
operacion_aritmetica
referencia
referencia
suma
x
y
Suma numero numero
referencia
26
y

Diagrama de arquitectura de la solución.



Problemas conocidos.

En esta sección se detalla cualquier problema que no se ha podido solucionar en el trabajo.

Botón compilar.

En la especificación se definía que ambos botones debían tener funciones diferentes. El botón compilar se encargaba de procesar el código y analizarlo por lo que retornaba los errores de sintaxis pero sin ejecutar las acciones. El botón ejecutar se encargaría de efectuar todas las acciones restantes. El problema final es que no fue posible separar la generación de la interfaz del analizador léxico por lo que ambos botones ejecutan el código.

Tipo de dato Float.

Todas las sentencias matemáticas se ejecutan tomando en cuenta a los números como el tipo de dato Integer de Java. Esto se cumple con excepción de la función “REDONDEAR” que sí recibe correctamente valores Float.

No fue posible implementar una solución que permitiera ejecutar los valores numéricos de tipo Float e Integer a la vez ya que sufría errores con el código de Java. Se recomienda escribir una clase que funcione como filtro de modo que todos los números se trabajen como Float y el resultado lo retorne según el tipo más conveniente.

Problemas encontrados.

Librerías para realizar interfaces gráficas en Java.

Se planteó utilizar la librería swing, la cual es una librería preinstalada de Java para hacer GUI's, sin embargo, las interfaces que se podían crear tenían un aspecto antiguo, además de ser confusa en algunos casos, se procedió a investigar qué librerías podrían ser útiles, y se optó por usar JavaFX, librería simple con un paradigma fuerte orientado a objetos, además hace uso del poliformismo para crear interfaces fáciles de programar.

Se recomienda consultar la documentación oficial, además de buscar ayuda en vídeos o blogs que expliquen cómo instalar y configurar la librería, también para crear una interfaz básica con dicha librería.

El unir la interfaz gráfica con el compilador de ANTLR no supuso problemas con JavaFX. Se puede consultar la documentación oficial en el sitio de oracle: https://docs.oracle.com/javafx/2/

Cargar archivos al editor de la interfaz gráfica.

No se conseguía implementar un método sencillo para poder cargar archivos al editor de la interfaz gráfica, ya que ésta pedía que se le indicase un string con la ruta absoluta del archivo que se desea cargar, luego de investigar durante unas horas, se encontró una clase en la librería JavaFX llamada FileChooser, esta clase permite utilizar el buscador del sistema para obtener una ruta absoluta de un archivo, esto permite que la búsqueda y lectura del archivo al cargar sea más intuitiva para el usuario.

Pantalla de tortuga.

Al implementar la pantalla de dibujo se notó que para las instrucciones que implican movimiento de la tortuga, la imagen de esta se duplicaba, creando así una imagen nueva por cada instrucción, se optó primeramente por borrar el lienza en la posición de la tortuga con su alto y su ancho, sin embargo, esto presentaba problemas en ciertas ocasiones, ya que borraba también la líneas realizadas anteriormente. Se procedió a crear una nueva capa en la ventana donde se ubique solo la imagen de la tortuga y quedara separada de los trazos del lienzo, así, por cada movimiento de la tortuga se

limpia por completo la capa de la tortuga y se coloca nuevamente en la nueva posición según la instrucción dada por el usuario.

Uso de ANTLR.

El uso de la herramienta se facilitó gracias al soporte de un tutorial. Aún así, implementar el patrón intérprete requirió superar diversos retos como la implementación de sentencias complejas como ciclos y manejo de variables. Con el fin de poder manejar las variables en memoria se hizo uso de una estructura de datos HashMap en Java que funciona como una tabla de símbolos que asocia el nombre de variables con su valor respectivo.

El patrón intérprete funciona a partir de una interfaz ASTNode de la cual se derivan todas las clases que controlan cada una de las reglas léxicas definidas en el archivo G4 de ANTLR.

También es importante mencionar que la terminología general de ANTLR no representó mayor problema gracias a que es muy parecida a la vista en clases, de igual manera que los quices y los ejercicios realizados.

Se puede consultar la documentación de ANTLR en: https://www.antlr.org/.
El tutorial utilizado se puede encontrar en:
https://www.youtube.com/watch?v=WrlgULIJqEw&list=PL5BoUl9EDVnBojdOv9J9S9KZPJdOc6HTw.

Conclusiones y Recomendaciones del proyecto.

- El uso de ANTLR supuso una gran ayuda en la creación del compilador ya que, creando una gramática bien fundamentada con los requisitos del compilador, se puede ahorrar mucho tiempo en las diferentes etapas de la creación de éste, ya que ANTLR lo hace automáticamente, además, la creación del análisis semántico en ANTLR es sencillo ya que, permite especificar código Java en la ejecución de cada sentencia.

- ANTLR posee opciones de visualización del compilador, posee una vista del árbol sintáctico abstracto, además de un diagrama de sintaxis.
  Estos elementos pueden ser muy útiles durante el diseño del compilador.

- A pesar de que la herramienta ANTLR nos permite agilizar el proceso de creación de un compilador, eso no quiere decir que no tengamos que conocer en profundidad dicho proceso, para poder utilizarlo.
  Conceptos como flujos de tokens, árboles de parseo, sintaxis, expresiones regulares, tablas de símbolos, entre otros, son esenciales al crear el compilador, ya que se debe hacer uso de éstos conceptos durante la creación de la gramática y la semántica en el archivo con extensión .g4 de este proyecto.

Bibliografía consultada en todo el proyecto.

ANTLR. (*s. f.*). *Antlr*. Recuperado 18 de enero de 2021, de https://www.antlr.org/

Proceso de interpretacion de un lenguaje de programación. (2016, 8 *abril*). [*Vídeo*]. *YouTube*. https://www.youtube.com/watch?v=WrlgULIJqEw&list=PL5BoUl9EDVnBojdOv9J9S9KZPJdOc6HTw.