Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computadores

Compiladores e Interpretes(CE1108)

Proyecto LOGOTEC

Primer Avance

Profesor:

Marco Hernandez Vasquez

Estudiantes:

Fabiola Meléndez Sequeira

Jeremy Serracin Oporta

Alisson Redondo Moya

Braulio Retana Murillo

II Semestre

Link al repositorio de Github:

https://github.com/JeremySO202/LogoTEC/tree/master

TABLA DE CONTENIDOS

1. Decisiones de Diseño

Este primer avance se concentra en el desarrollo del frontend, el cual en terminos generales se centra en la definición de la gramatica libre de contexto. Para lo anterior se desarrolla inicialmente el analisis lexico donde se establecen los tokens, palabras reservadas, simbolos, identificadores y tipos. En las tablas [*(1.1), (1.2)*] se puede observar de una mejor manera cada una de las partes mencionadas anteriormente.

Una vez definido el analisis lexico se realiza el analisis sintactico donde se construye el BNF o estructura del lenguaje, en este se define la inicialización del programa, la estructura o cuerpo de un procedimiento, la estructura de comentarios, definición de instrucciones, operaciones, ordenes y establecimiento de metricas especificas del lenguaje. En las tablas [*(1.3), (1.4), (1.5*] se puede observar cada una de las partes desarrolladas en esta etapa.

Una vez definida esta gramatica se procede a comprobar el lenguaje desarrollado. Con la ayuda del IDE de Intellij Idea junto con ANTLR se realizan las pruebas de estructura por medio del arbol de parseo. Se verifica que cada una de las instrucciones esté identificando de manera correcta cada una de las partes que la constituyen. En la figura 1.1 se puede observar una prueba realizada para las instrucciones de INIC y SUMA la figura 1.2 es una prueba realizada para las instrucciones AVANZA y GIRADERECHA.

Algunos detalles a mencionar sobre el lenguaje son los siguientes:

- Existen dos tipos de comentarios en el programa. El primer tipo de comentario es el comentario inicial el cual es obligatorio cada que se inicialice un programa, este tiene una estructura: {_ - _} y debe de contener un nombre seguido de su funcionalidad. El siguiente tipo de comentario es un comentario libre sin estructura definida, este no es obligatorio y se puede utilizar en cualquier parte

del codigo. Cabe mencionar que estos comentarios solamente pueden contener letras, numeros y espacios y terminan cuando se salta de linea. No se permiten nin tipo de palabra reservada ni caracter especial. Lo anterior se establece así para tener una mejor ligibilidad de los mismos.

- El programa permite utilizar procedimientos e instrucciones, lo cual no limita a la persona que utilice el lenguaje a tener que crear un procedimiento para poder utilizar una instrucción. Lo anterior se decidió así para poder tener un flujo de programa más abierto y menos limitado.

- Los espacios y saltos de linea no son ignorados por el lenguaje sino que estos forman parte de la estructura y definición de cada instrucción.

Token	**Lexema / Expresión Regular**
HAZ	'HAZ'
INIC	'INIC'
INC	'INC'
AVANZA	'AVANZA' | 'AV'
RETROCEDE	'RETROCEDE' | 'RE'
GIRADERECHA	'GIRADERECHA' | 'GD'
GIRAIZQUIERDA	'GIRAIZQUIERDA' | 'GI'
PONPOS	'PONPOS'
PONXY	'PONXY'
PONRUMBO	'PONRUMBO'
RUMBO	'RUMBO'
PONX	'PONX'
PONY	'PONY'
CENTRO	'CENTRO'
BAJALAPIZ	'BAJALAPIZ' | 'BL'
SUBELAPIZ	'SUBELAPIZ' | 'SB'
PONCOLORLAPIZ	'PONCOLORLAPIZ' | 'PONCL'
OCULTATORTUGA	'OCULTATORTUGA' | 'OT'
ESPERA	'ESPERA'
EJECUTA	'EJECUTA'
REPITE	'REPITE'
SI	'SI'
HAZ_HASTA	'HAZ.HASTA'
HASTA	'HASTA'
HAZ_MIENTRAS	'HAZ.MIENTRAS'
MIENTRAS	'MIENTRAS'
Y	'Y'
O	'O'
IGUALES	'IGUALES?'
MAYORQUE	'MAYORQUE?'
MENORQUE	'MENORQUE?'

Table 1.1: Palabras Reservadas

Token	Lexema / Expresión Regular
DIFERENCIA	'DIFERENCIA'
AZAR	'AZAR'
PRODUCTO	'PRODUCTO'
POTENCIA	'POTENCIA'
DIVISION	'DIVISION'
SUMA	'SUMA'
PARA	'PARA'
FIN	'FIN'
BOOLEAN	'TRUE' | 'FALSE'
COLOR	'NEGRO' | 'AZUL' | 'ROJO'
IGUAL	'='
SEMICOLON	';'
SQUARE_OPEN	'['
SQUARE_CLOSE	']'
BRACKET_OPEN	'{'
BRACKET_CLOSE	'}'
PAR_OPEN	'('
PAR_CLOSE	')'
GUION	'-'
COMILLA	'"'
STRING	COMILLA (LETRAS | NUMERO | GUION_BAJO)+ COMILLA
ID	(LETRAS | GUION_BAJO) (LETRAS | GUION_BAJO | NUMERO)*
NUMERO	DIGITO+
LETRAS	[a-zA-Z]+
GUION_BAJO	'_'
DIGITO	[0-9]
ESPACIO	' '
SALTO_LINEA	'\n'
WS	[\t\r]+ -> skip

Table 1.2: Palabras Reservadas

Token / Regla	**Definición / Expresión**
PROGRAM	(estructura_comentario) (procedimientos | ordenes)* EOF
ESTRUCTURA_COMENTARIO	'//' ESPACIO* BRACKET_OPEN (ID | NUMERO | ESPACIO)* GUION (ID | NUMERO | ESPACIO)* BRACKET_CLOSE SALTO_LINEA
COMENTARIO	'//' (ID | DIGITO | ESPACIO)* SALTO_LINEA+
PROCEDIMIENTOS	(SALTO_LINEA* procedimiento SALTO_LINEA*)+
PROCEDIMIENTO	PARA ESPACIO ID ESPACIO SQUARE_OPEN parametros SQUARE_CLOSE ordenes SALTO_LINEA* FIN
PARAMETROS	(ESPACIO*ID (ESPACIO ID)* ESPACIO*)?
ORDENES	orden+
ORDEN	(SALTO_LINEA* ESPACIO* instruccion SALTO_LINEA* ESPACIO*)

Table 1.3: Tabla de Reglas Sintácticas Principales

Token / Regla	Definición / Expresión
HAZ	HAZ ESPACIO ID ESPACIO* (numeric_val | STRING | boolean_val) SEMICOLON
INIC	INIC ESPACIO ID ESPACIO IGUAL ESPACIO (numeric_val | STRING | boolean_val | operacion) SEMICOLON
INC	INC ESPACIO SQUARE_OPEN numeric_val (ESPACIO numeric_val)? SQUARE_CLOSE SEMICOLON
AVANZA	AVANZA ESPACIO numeric_val SEMICOLON
RETROCEDE	RETROCEDE ESPACIO numeric_val SEMICOLON
GIRADERECHA	GIRADERECHA ESPACIO numeric_val SEMICOLON
GIRAIZQUIERDA	GIRAIZQUIERDA ESPACIO numeric_val SEMICOLON
PONPOS	PONPOS ESPACIO SQUARE_OPEN numeric_val ESPACIO numeric_val SQUARE_CLOSE SEMICOLON
PONXY	PONXY ESPACIO numeric_val ESPACIO numeric_val SEMICOLON
PONRUMBO	PONRUMBO ESPACIO numeric_val SEMICOLON
PONX	PONX ESPACIO numeric_val SEMICOLON
PONY	PONY ESPACIO numeric_val SEMICOLON
CENTRO	CENTRO SEMICOLON
BAJALAPIZ	BAJALAPIZ SEMICOLON
SUBELAPIZ	SUBELAPIZ SEMICOLON
PONCOLOR	PONCOLORLAPIZ ESPACIO COLOR SEMICOLON
OCULTATORTUGA	OCULTATORTUGA SEMICOLON
ESPERA	ESPERA ESPACIO numeric_val SEMICOLON
EJECUTA	EJECUTA ESPACIO SQUARE_OPEN ordenes SQUARE_CLOSE SEMICOLON
REPITE	REPITE ESPACIO numeric_val ESPACIO SQUARE_OPEN ordenes SQUARE_CLOSE SEMICOLON
SI	SI ESPACIO PAR_OPEN boolean_val PAR_CLOSE ESPACIO SQUARE_OPEN ordenes SQUARE_CLOSE (ESPACIO SQUARE_OPEN ordenes SQUARE_CLOSE)? SEMICOLON

Table 1.4: Tabla de Instrucciones y Definiciones del Lenguaje

Token / Regla	**Definición / Expresión**
HAZ_HASTA	HAZ_HASTA ESPACIO SQUARE_OPEN ordenes SQUARE_CLOSE ESPACIO PAR_OPEN boolean_val PAR_CLOSE SEMICOLON
HASTA	HASTA ESPACIO PAR_OPEN boolean_val PAR_CLOSE ESPACIO SQUARE_OPEN ordenes SQUARE_CLOSE SEMICOLON
HAZ_MIENTRAS	HAZ_MIENTRAS ESPACIO SQUARE_OPEN ordenes SQUARE_CLOSE ESPACIO PAR_OPEN boolean_val PAR_CLOSE SEMICOLON
MIENTRAS	MIENTRAS ESPACIO PAR_OPEN boolean_val PAR_CLOSE ESPACIO SQUARE_OPEN ordenes SQUARE_CLOSE SEMICOLON
Y	Y ESPACIO boolean_val ESPACIO boolean_val
O	O ESPACIO boolean_val ESPACIO boolean_val
BOOLEAN_VAL	ID | BOOLEAN | op_logica | op_comparativa
OP_COMPARATIVA	IGUALES ESPACIO numeric_val ESPACIO numeric_val | MAYORQUE ESPACIO numeric_val ESPACIO numeric_val | MENORQUE ESPACIO numeric_val ESPACIO numeric_val
OPERACION	op_algebraicas
OP_ALGEBRAICAS	DIFERENCIA | AZAR | PRODUCTO | POTENCIA | DIVISION | SUMA
NUMERIC_VAL	ID | NUMERO | DIGITO | rumbo | PAR_OPEN operacion PAR_CLOSE
DIFERENCIA	DIFERENCIA ESPACIO numeric_val (ESPACIO numeric_val)+
AZAR	AZAR ESPACIO numeric_val
PRODUCTO	PRODUCTO ESPACIO numeric_val (ESPACIO numeric_val)+
POTENCIA	POTENCIA ESPACIO numeric_val ESPACIO numeric_val
DIVISION	DIVISION ESPACIO numeric_val ESPACIO numeric_val
SUMA	SUMA ESPACIO numeric_val (ESPACIO numeric_val)+
RUMBO	RUMBO

Table 1.5: Tabla de Instrucciones y Definiciones del Lenguaje

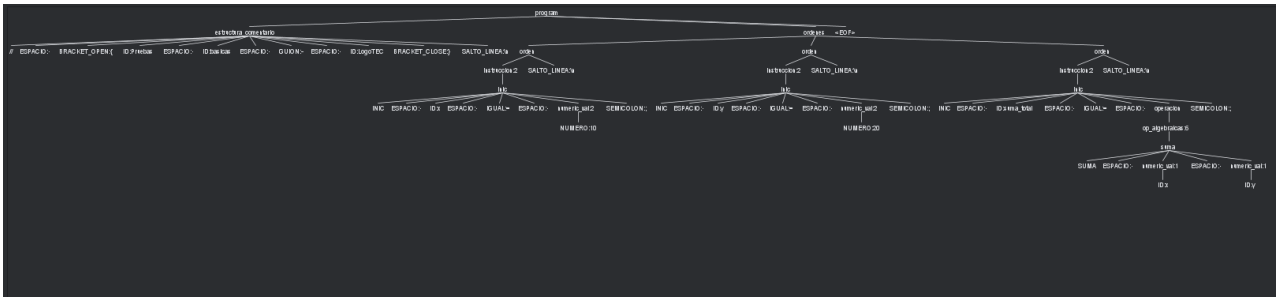

Figure 1.1: Arbol de Parseo para instrucción INIC y SUMA

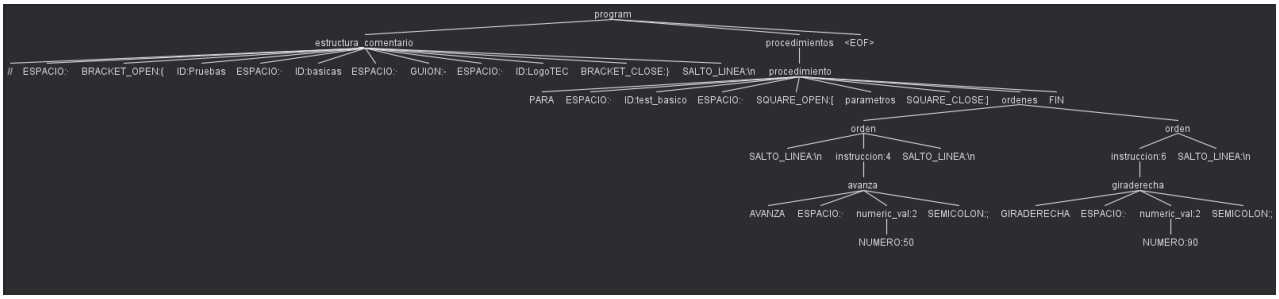

Figure 1.2: Arbol de Parseo para procedimiento, AVANZA y GIRADERECHA

2. Problemas Encontrados

Unos de los problemas encontrados en el desarrollo de este trabajo fue el uso de espacios en ciertas expresiones regulares, en la imagen 2.1 se puede observar un ejemplo de esto.

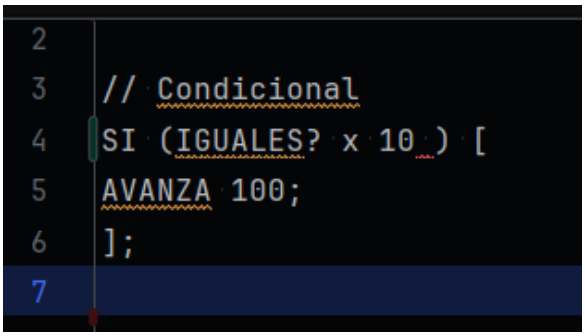

Figure 2.1: Uso incorrecto de espacios

Se puede observar como en la linea 4 cuando se coloca un espacio justo despues del 10 el interprete detecta un error, mas especificamente el espacio despues del 10, esto es debido a que como se menciono previamente el codigo realizado para este trabajo

si toma en cuenta tanto los espacios como los saltos de linea. Este problema no solo se aplica para los espacios sino que tambien para los saltos de linea por lo que un mal uso de estos provocaria los mismos resultados que los de la imagen 2.1.

References

[1] A. V. Aho, M. S. Lam, R. Sethi, and J. D. Ullman, *Compilers: Principles, Techniques, and Tools*, 2nd ed. Boston: Pearson, 2007.

[2] D. Jurafsky and J. H. Martin, *Speech and Language Processing*, 3rd ed. Draft, 2023.

[3] Universidad Europea de Madrid, *Análisis semántico / La tabla de símbolos*, Apuntes en línea. Disponible en: `https://www.cartagena99.com/recursos/alumnos/apuntes/ININF2_M4_U5_T2.pdf`.

[4] Jaime Pavlich-Mariscal, *Proceso de interpretacion de un lenguaje de programación*, YouTube, 8 abr 2016. [Online]. Available: `https://youtu.be/WrlgULIJqEw?si=r92cBolEYuXCYr3t`

[5] Jaime Pavlich-Mariscal, *Implementación del analizador léxico*, YouTube, 19 abr 2016. [Online]. Available: `https://youtu.be/PP20rtarbuc?si=8WkyFg2kxCoavot_`

[6] Jaime Pavlich-Mariscal, *Implementación del analizador sintáctico*, YouTube, 12 jul 2016. [Online]. Available: `https://youtu.be/7ptttGTQQmk?si=1biklvVCMDfUO8_l`