Tecnológico de Monterrey

Instituto Tecnológico de Estudios Superiores Monterrey

CAMPUS MONTERREY

TC3002B | Desarrollo de aplicaciones avanzadas de ciencias computacionales

Grupo 502

Patito

Alumno:

Erick Alfredo García Huerta - A01708119

Entrega 1

En la primera iteración de este proyecto se planeó usar PLY (Python Lex-Yacc), una herramienta basada en Python para la realización de este proyecto. Sin embargo, tras haber interactuado con la herramienta y revisado la otra opción que se había tomado en cuenta, se decidió ir con ANTLR, esto debido a la disponibilidad de recursos sobre su funcionamiento, así como su fácil manipulación y visualización con plugins en PyCharm. Otro elemento a tomar en cuenta es que también es compatible con Python y su sintaxis es también bastante fácil de entender.

ANTLR (Another Tool for Language Recognition) es una poderosa herramienta que permite generar analizadores léxicos y sintácticos para el procesamiento de lenguajes, tanto naturales como de programación. Es ampliamente utilizada para crear compiladores, intérpretes y analizadores de lenguajes personalizados. ANTLR convierte una gramática definida por el usuario en código fuente que puede analizar y comprender un lenguaje, facilitando la creación de parsers. Sus principales características incluyen el soporte para expresiones regulares y reglas gramaticales, la generación automática de código en múltiples lenguajes de programación (como Java, Python y C#), y un robusto manejo de errores. Además, ofrece flexibilidad mediante el uso de "listeners" y "visitors", que permiten manipular y recorrer el árbol de análisis generado por el parser, lo que facilita el desarrollo de herramientas personalizadas de procesamiento de lenguajes.

Las reglas gramaticales para el lenguaje Patito se definieron en el archivo **patito.g4**, siguiendo un formato estructurado que incluye tanto expresiones regulares como reglas gramaticales.

- Expresiones regulares: Se utilizan para definir los tokens básicos del lenguaje, como identificadores (ID), enteros (INT_TOK), flotantes (FLOAT_TOK), y palabras reservadas como Programa, Vars, Inicio, y Fin. Estas reglas son fundamentales para que el lexer identifique correctamente las unidades mínimas del lenguaje:

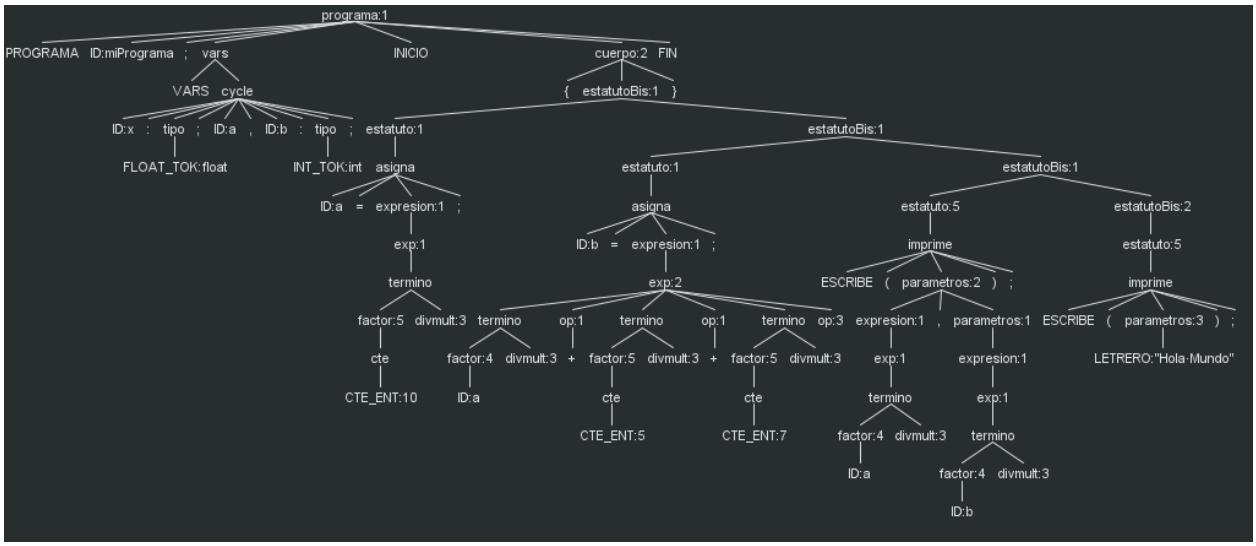
```
// Palabras reservadas
INICIO: 'Inicio' ;
FIN: 'Fin' ;
ESCRIBE: 'Escribe' ;
NULA: 'Nula' ;
PROGRAMA: 'Programa' ;
SI: 'Si' ;
SINO: 'Sino' ;
MIENTRAS: 'Mientras' ;
HAZ: 'Haz' ;
VARS: 'Vars' ;

// Tokens para tipos de datos
INT_TOK: 'int' ;
FLOAT_TOK:  'float' ;
ID: [a-zA-Z] [a-zA-Z0-9_]*;
LETRERO: '"' ~[\r\n"]* '"';
CTE_ENT: '-'? [0-9]+;
CTE_FLOT: '-'? [0-9]+ '.' [0-9]+;
```

```
// Ignorar espacios en blanco
WS: [ \t\r\n]+ -> skip ;
```

- Reglas gramaticales: Las reglas de la gramática definen la estructura completa del programa. La regla inicial programa especifica cómo comienza un programa en Patito, incluyendo la declaración de variables, funciones, y el cuerpo principal del programa. Cada elemento está cuidadosamente definido, con alternativas (|) y operadores que permiten flexibilidad en la estructura del código:

```
// Regla inicial del programa
programa: PROGRAMA ID ';' vars? funcs* INICIO cuerpo FIN
        | PROGRAMA ID ';' INICIO cuerpo FIN
        | PROGRAMA ID ';' vars? INICIO cuerpo FIN
        | PROGRAMA ID ';' funcs* INICIO cuerpo* FIN
        | PROGRAMA ID ';' funcs* funcsBis
        ;
```

A continuación se presenta una proyección del árbol sintáctico



Este fue el caso de *test case* utilizado:

```
Programa miPrograma;
Vars
   x: float;
   a, b: int;
Inicio {
   a = 10;
   b = a + 5 + 7;
   Escribe(a, b);
   Escribe("Hola Mundo");
}
Fin
```

Funcionamiento

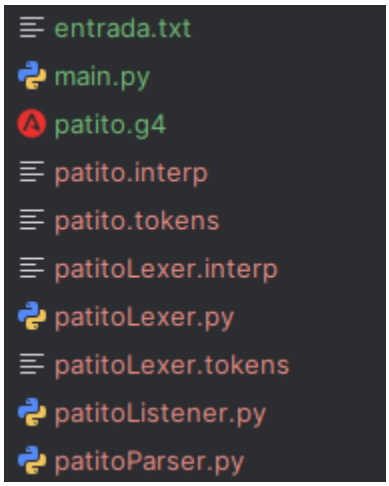
Se generaron los archivos necesarios para el funcionamiento del proyecto haciendo uso de las herramientas que ofrece ANTLR (Versión 4.3.2) siguiendo los siguientes pasos:

1. Tener la versión más actualizada de Java 11 en el equipo.
2. Utilizar en la consola el comando teniendo en el mismo directorio el archivo .g4 y el ejecutable .jar de ANTLR:

   java -jar antlr-4.13.2-complete.jar -Dlanguage=Python3 patito.g4

Para el funcionamiento del proyecto también es necesario

3. Tener la versión más actualizada de Python3 en el equipo
4. Contar con los siguientes documentos en el mismo directorio



5. Escribir en la consola en la ubicación del directorio del proyecto el siguiente comando:

   python main.py entrada.txt

6. Se desplegará en la consola los resultados.

```
(programa Programa miPrograma ; (vars Vars (cycle x : (tipo float) ; a , b : (tipo int) ;)) Inicio (cuerpo { (estatutoBis (estatuto (asigna a = (expresion (ex
p (termino (factor (cte 10)) divmult))) ;)) (estatutoBis (estatuto (asigna b = (expresion (exp (termino (factor a) divmult) (op +) (termino (factor (cte 5)) d
ivmult) (op +) (termino (factor (cte 7)) divmult) op)) ;)) (estatutoBis (estatuto (imprime Escribe ( (parametros (expresion (exp (termino (factor a) divmult))
) , (parametros (expresion (exp (termino (factor b) divmult))))) ) ;)) (estatutoBis (estatuto (imprime Escribe ( (parametros "Hola Mundo") ) ;)))))) }) Fin)
```

NOTA: Si se desea modificar los elementos de prueba se pueden hacer en el documento entrada.txt o si se planea usar un archivo distinto, reemplazar entrada.txt en el comando de python con el nuevo documento.