

Escuela de Ingeniería de Computación

IC-5701: Compiladores e Intérpretes

Profesor: Ing. Ignacio Trejos Zelaya

Proyecto #1: Compilador Triángulo Extendido

-Analizador Léxico

Estudiantes:

André Arroyo Piedra 2015073657

Isaac Chaverri Corrales 2013024299

Jafeth Leiva Mejía 2015058012

Esteban Quirós Alvarado 2014012892

Fecha de Entrega: 27 de septiembre del 2018

-Modificaciones realizadas en el analizador léxico.

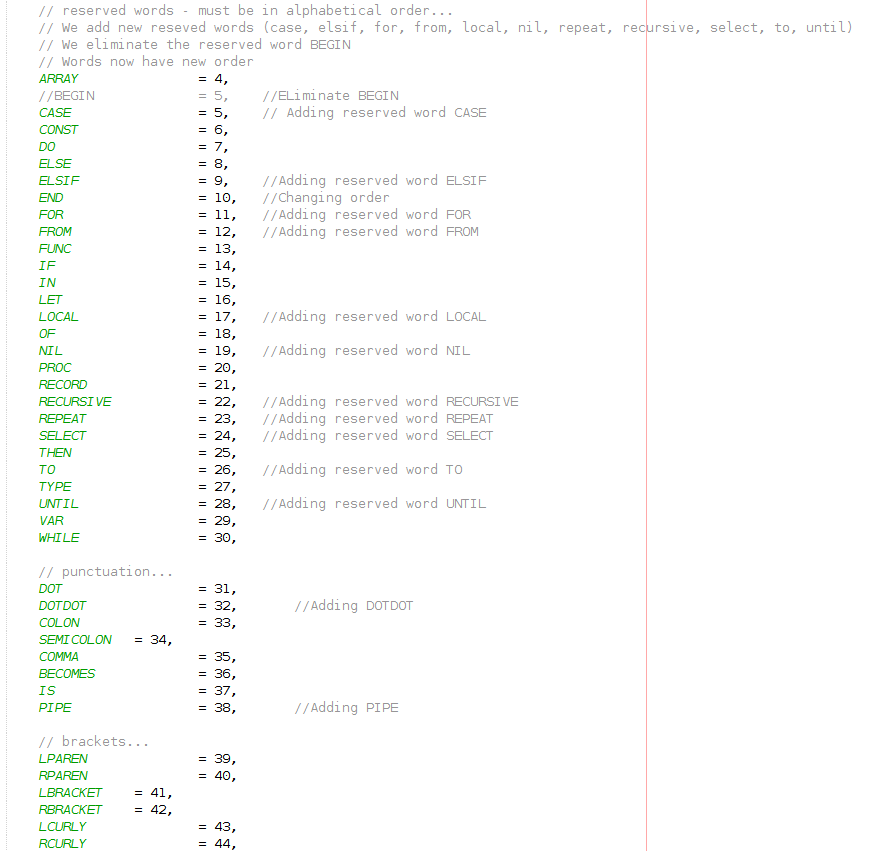
Se agregaron los siguientes tokens en Token.java:

* CASE
* ELSIF
* FOR
* FROM
* LOCAL
* NIL
* RECURSIVE
* REPEAT
* SELECT
* TO
* UNTIL
* PIPE
* DOTDOT

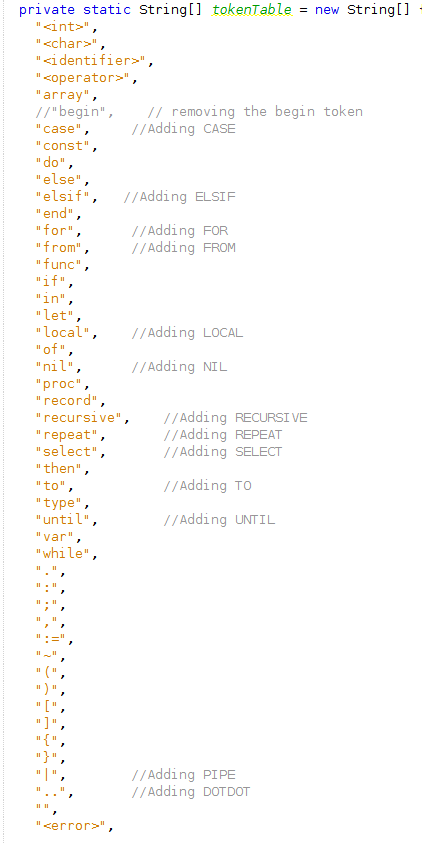
Se eliminó el token BEGIN.

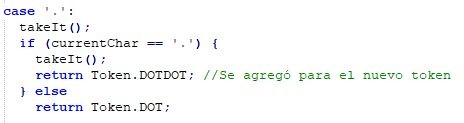
Adicionalmente se cambió el orden del token END.

Se adjunta imagen de como quedo el archivo Token:



Además, en la tabla de Tokens se ordenan alfabéticamente como se ve a continuación:



Archivo Scanner: se añadió el case PIPE con el símbolo “|”, se modificó el DOT (“.”) para el caso de recibir DOTDOT (“..”).

Archivo Parser: se modifica según las especificaciones dadas por el profesor. Y con estos cambios el lenguaje pasa a llamarse Triangulo Extendido.

Primero se elimina el comando vacío.

Posteriormente se siguen las indicaciones y se eliminan las siguientes alternativas:

* "begin" Command "end"
* "let" Declaration "in" single-Command
* "if" Expression "then" single-Command "else" single-Command
* "while" Expression "do" single-Command

El siguiente paso a realizar es añadir las nuevas alternativas:

* "nil"
* "repeat" "while" Expression "do" Command "end"
* "repeat" "until" Expression "do" Command "end"
* "repeat" "do" Command "while" Expression "end"
* "repeat" "do" Command "until" Expression "end"
* "repeat" "for" Identifier "from" Expression "to" Expression "do" Command "end"
* "let" Declaration "in" Command "end"
* "if" Expression "then" Command ("elsif" Expression "then" Command) \* "else" Command "end"
* "select" Expression "from" Cases "end"

### Declaration

### Se modifica de acuerdo a las instrucciones de profesor de la siguiente manera:

### 

### Se añade la siguiente regla:

### 

### Se añaden la siguiente regla:

### 

### 

### 

### Proc-Func y Proc Funcs:

### 

### 

### Se modifica el proc de la siguiente manera:

### 

### En single-declaration se añade la siguiente variable:

### 

### Se modifica el Type-denoter con respecto a la regla de arreglos:

### 

### Como principal regla se añade el case:

### 

### 

### Se añade el código:

### 

### 

### 

### Para los errores detectados por el compilador se usa el mismo patrón que está en el mismo:

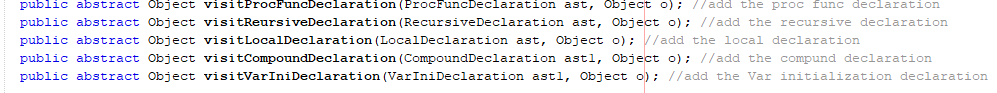
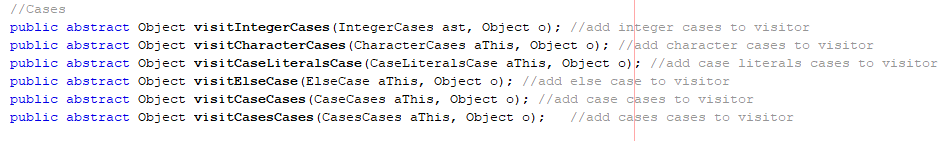
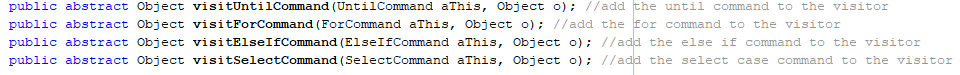
### 

### Modelaje de árboles:

Para cada tipo de comando, expresión, declaración, etc. que esté presente en el analizador sintáctico es necesario de generar un árbol que represente la estructura de este.

Para cada nodo, se dará un objeto de tipo AST para la formación del árbol. Para los nuevos comandos se tuvo que generar la clase utilizada por el árbol correspondiente, esto para que en el Visitor.java fuera posible recibir un objeto de ese tipo como parámetro y con ello conocer qué tipo de árbol era (cantidad de nodos).

Además, se modifica el Visitor.java:



Discusión y análisis de resultados:

Se da como concluido el primer proyecto, que conllevaba el analizador léxico. Se agregaron las nuevas reglas, tokens, comandos, expresiones, declaraciones, especificados por el profesor. De acuerdo con esto ahora el lenguaje no es Triangulo, sino Triangulo Extendido. Que con estas hace el lenguaje mucho más atractivo a la hora de realizar un programa.

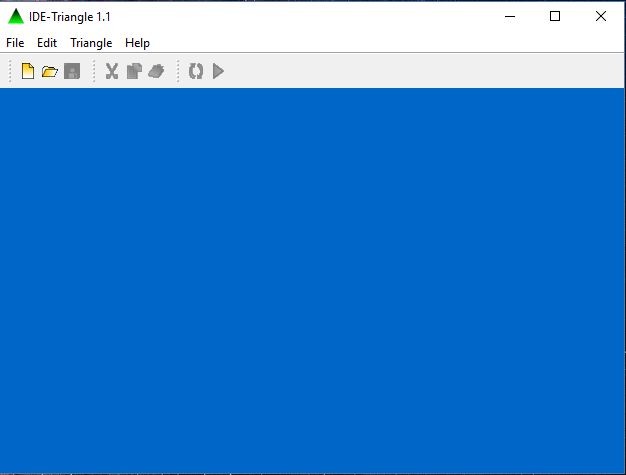
Reflexión:

Al inicio del proyecto es complicado entender muchas de las cosas que pasan, ya que no es código propio y esto hace que el grado de entendimiento suba. Se tuvo que ir poco a poco para poder entender el funcionamiento. Ya hecho esto resultó más fácil el poder modificar y agregar todo lo necesario. En el caso del “case” fue el de mayor dificultad, se tuvo que realizar consultas al profesor para poder comprender como resolver este caso.

Grupo de trabajo:

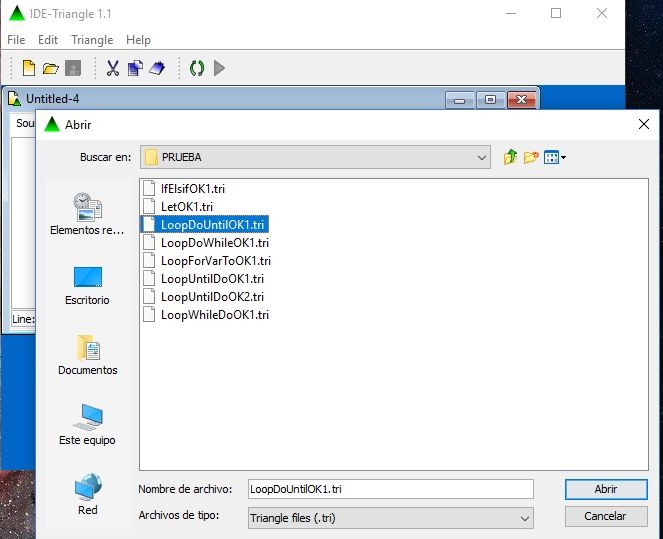
|  |  |
| --- | --- |
| Miembro | Actividades |
| André Arroyo Piedra |  |
| Isaac Chaverri Corrales |  |
| Jafeth Leiva Mejía | Trabajo en el entendimiento del parser.java al inicio del proyecto al igual que buscar información para la realización de las reglas, expresiones, comandos. Documentación pedida en la especificación. |
| Esteban Quirós Alvarado |  |

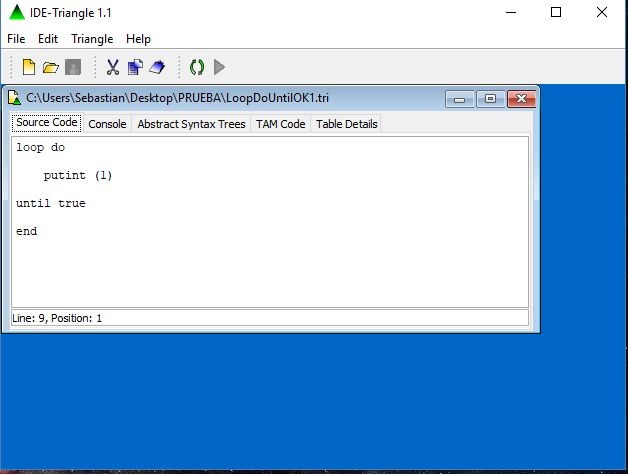
### ¿Como se debe de ejecutar el programa?

Para ejecutar el programa debe de darse doble click al IDE-Triangle.jar y en este punto se abrirá esto:

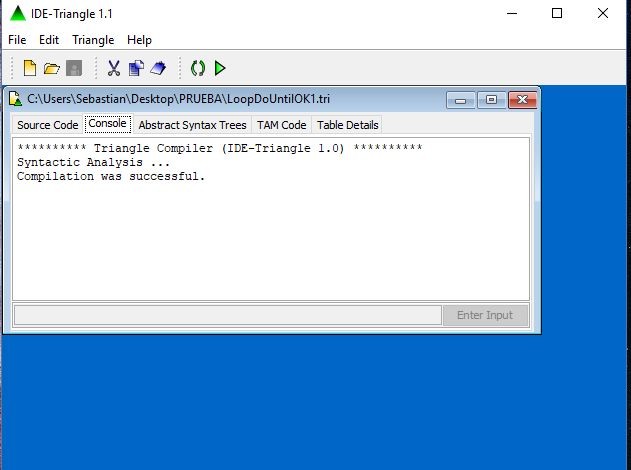
Para abrir un archivo fuente debe de irse a file -> open

Ahí se abrirá un explorador de archivos donde se selecciona el archivo a compilar.



Le damos en abrir y se cargará a la interfaz.

Para compilarlo le daremos en Triangle -> Compile y nos dirá si fue succesful o no



Y si le damos en la pestaña Abstract Syntax Trees nos mostrará el árbol.

