Práctica 2
Analizador Sintáctico TL

Grupos: Máximo 2 personas por grupo, ambas personas deberán estar presentes el día de la entrega y estar en capacidad de alterar el programa dado que surjan nuevos requerimientos durante la sustentación).

Objetivo: desarrollar un analizador sintáctico LL(1) para el lenguaje TL.

Dado el código fuente de un programa en TL, su tarea consiste en realizar el análisis sintáctico sobre dicho código. Nos enfocaremos en los errores sintácticos generados. En este documento se especifica la manera correcta de mostrar las salidas.

En el caso que el programa se encuentre bien formado respecto a las reglas especificadas de TL, se debe escribir la siguiente salida:

El analisis sintactico ha finalizado correctamente.

Note que no se permiten tildes en el mensaje de salida.

En caso contrario, es decir, si se encontró un error sintáctico, se debe **abortar** el análisis y escribir únicamente el primer error encontrado.

Consideraciones gramaticales

- Un programa escrito en TL debe seguir la siguiente estructura

```
# Declaracion de cero o mas funciones

funcion id_nombre_funcion(argumento1, argumento2, ...)
 #algoritmo
 retorno
end funcion


# Modulo Principal
# Consiste de cero o mas instrucciones

#sentencias
```

Si un programa no respeta esta estructura deberá considerarse como un error sintáctico. El módulo principal podría no contener ninguna instrucción.

- Variables
- Operaciones Math
- Booleanos
- Arreglos
- Bucles
- Estructuras
- Funciones

- La estructura del condicional if se presenta a continuación: :

```
if ( condicion1 )
        bloque_sentencias 1
else if ( condicion2 )
        bloque_sentencias 2
else if ( condicion3 )
        bloque_sentencias 3
else if ( condicion4 )
        bloque_sentencias 4
else if ( condicion... )
        bloque_sentencias...
else
        bloque_sentencias N
```

  Donde los bloques “else if” son opcionales, al igual que el bloque del “else”. En caso de existir, sólo debe haber un “else”.

  Los bloques de sentencias van encerrados entre llaves “{“ y “}”. En caso que dentro del bloque sólo haya una sola instrucción, las llaves son opcionales. Esto aplica para todas las partes de la gramática donde pueda haber un bloque de sentencias. Por ejemplo: para los ciclos while.

- Además del ciclo for presentado antes, el lenguaje TL también soporta ciclos while. Su estructura es la siguiente:

```
while ( condicion )
        bloque_sentencias
```

- TL puede importar librerías de forma similar a como se hace en Python, por ejemplo:

```
importar IDENTIFICADOR
importar IDENTIFICADOR1.IDENTIFICADOR2      #Podría haber más ids
                                            # separados por “.”
desde IDENTIFICADOR1 importar IDENTIFICADOR2
```

- Así como la palabra reservada “log” sirve para escribir en pantalla, para leer desde consola se usa “leer”. Por ejemplo:

log("prueba de impresion"**)**
leer(num**)**
log(num**)**
leer(cadena**)**
log(cadena**)**

Nota: considere la palabra reservada “leer” como tal. En la práctica anterior no se había considerado como palabra reservada.

- La palabra reservada “retorno” sólo debe aparecer dentro de una función. Su sintaxis es la siguiente:
  retorno (expresión o valor a retornar **)**

- Una sentencia en TL también podría ser una instrucción atómica del lenguajes, es decir, cualquiera de los siguientes: número entero, número real, cadena, booleano, arreglo, acceso a un arreglo, variable, la palabra reservada “nil”, una estructura, o el acceso a uno de sus campos.

Especificaciones de entrada y salida

Errores Sintácticos

Los errores sintácticos se producen cuando el programa no respeta la estructura gramatical mencionada. Cuando se presenta alguno **se debe reportar y abortar el análisis**.

El formato para reportar un error sintáctico es el siguiente:

```
<fila:columna> Error sintactico. Encontrado: {lexema del token
encontrado}; se esperaba: {lista de símbolos/tokens esperados}.
```

Donde:

- `fila` y `columna` son los números de línea y columna donde se detectó el error.
- `lexema del token encontrado`: corresponde al lexema encontrado que no se esperaba encerrado entre comillas simples.
- `lista de símbolos/tokens esperados`: corresponde a la lista de tokens esperados o los lexemas esperados cuando estos tokens representen un único símbolo del lenguaje. Los elementos de la lista deben ir separados por comas y encerrados entre comillas simples. Por ejemplo: ';', ')', ','.

Consideraciones para reporte de errores

- Cuando es necesario reportar palabras reservadas se debe hacer imprimiendo la palabra encontrada en minúscula.

- Para los lexemas de los tipos de datos primitivos se debe hacer con la palabra valor_tipodedato. Mostrado en la siguiente tabla:

token	lo que se imprime
token_integer	valor_entero
token_string	valor_string
token_float	valor_float

- Cuando se reportan símbolos del lenguaje se deben imprimir sin ninguna alteración. Ej. ';', '%', '!', '<', '<='.
- Cuando hay más de un símbolo/token esperado la lista debe reportarse **lexicográficamente ordenada** (basado en ASCII) y separada por comas.
- Cuando se reporta un id se debe hacer con la palabra 'identificador'. Ej: "<12, 15> Error sintactico: se encontro 'identificador'; se esperaba '+'."
- Cuando se esperaba fin de archivo o se encuentra fin de archivo. Lo que se debe imprimir es 'EOF'.

Ejemplos

Entrada (in01.txt)	**Salida (out01.txt)**
`if { a>0 }`	`<1:4> Error sintactico. Encontrado: '{'; se esperaba: '('.`

Entrada (in02.txt)	**Salida (out02.txt)**

`if ( a>0 ) {`
`  log("Mayor")`
`else`
`  log("Menor o igual")`
`}`
	`El analisis sintactico ha finalizado correctamente.`

NOTA: como el lenguaje TL no posee un caracter especial como delimitador de sentencias como ";" en lenguaje C/C++/Java, considere capturar el caracter de nueva línea como un token adicional desde su analizador léxico.