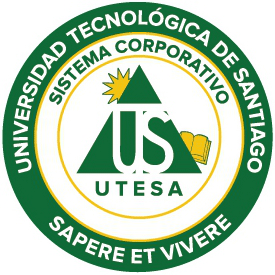
**UNIVERSIDAD TECNOLÓGICA DE SANTIAGO, UTESA**

SISTEMA CORPORATIVO

FACULTAD DE INGENIERIA Y ARQUITECTURA

CARRERA DE INGENIERIA EN SISTEMAS COMPUTACIONALES 

**ASIGNATURA:**

Compiladores

INF-920-001

**TAREA SEMANA 3:**

Análisis Sintáctico

**PRESENTADO A:**

Iván Mendoza

**PRESENTADO POR:**

Liván Herrera (2-16-0686)

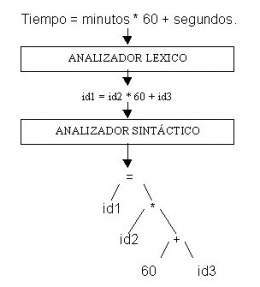
**Santiago de los Caballeros**

**República Dominicana**

**Febrero, 2022**

*Leer Capítulo 4 del Libro Compiladores, Principios, Técnicas. Tarea de investigación sobre Análisis Sintáctico.*

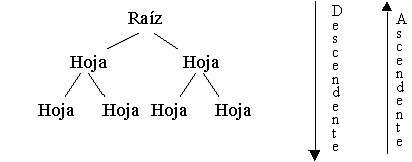
**1-) INVESTIGAR:**

**Análisis Sintáctico**

Este análisis es de jerarquía y/o concordancia que existe en una cadena de palabras las cuales se agrupan formando un sintagma y/u oraciones.

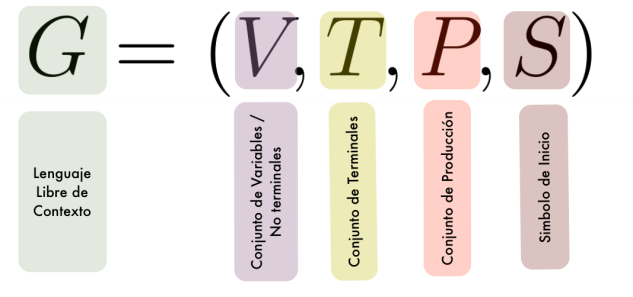
**Análisis Ascendente y descendente**

* Análisis Ascendente: Este tipo de análisis se basa en fabricar el árbol sintáctico de abajo hacia arriba, o sea, desde los tokens hacia el axioma inicial.
* Análisis Descendente: Este tipo de análisis busca descubrir entre las producciones de la gramática busca la derivación por la izquierda del símbolo inicial en una cadena de entrada.

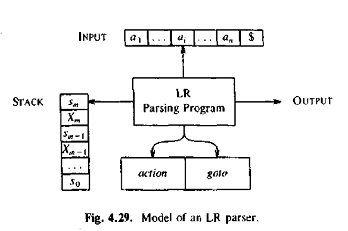


**Gramática de contexto libre**

Esta gramática consiste de varios elementos como: los terminales, los no terminales, el símbolo inicial y las producciones. Donde:

* Los terminales son aquellos símbolos básicos con los cuales se pueden formar las cadenas.
* Los no terminales denotan conjunto de cadenas mediante variables sintácticas.
* El símbolo inicial viene siendo el no terminal, y el conjunto de cadenas que este denota sería el lenguaje generado por la gramática.
* Las producciones están encargadas de describir como se pueden combinar los terminales y los no terminales para posteriormente formar o generar cadenas.

**Analizador Sintáctico LR**

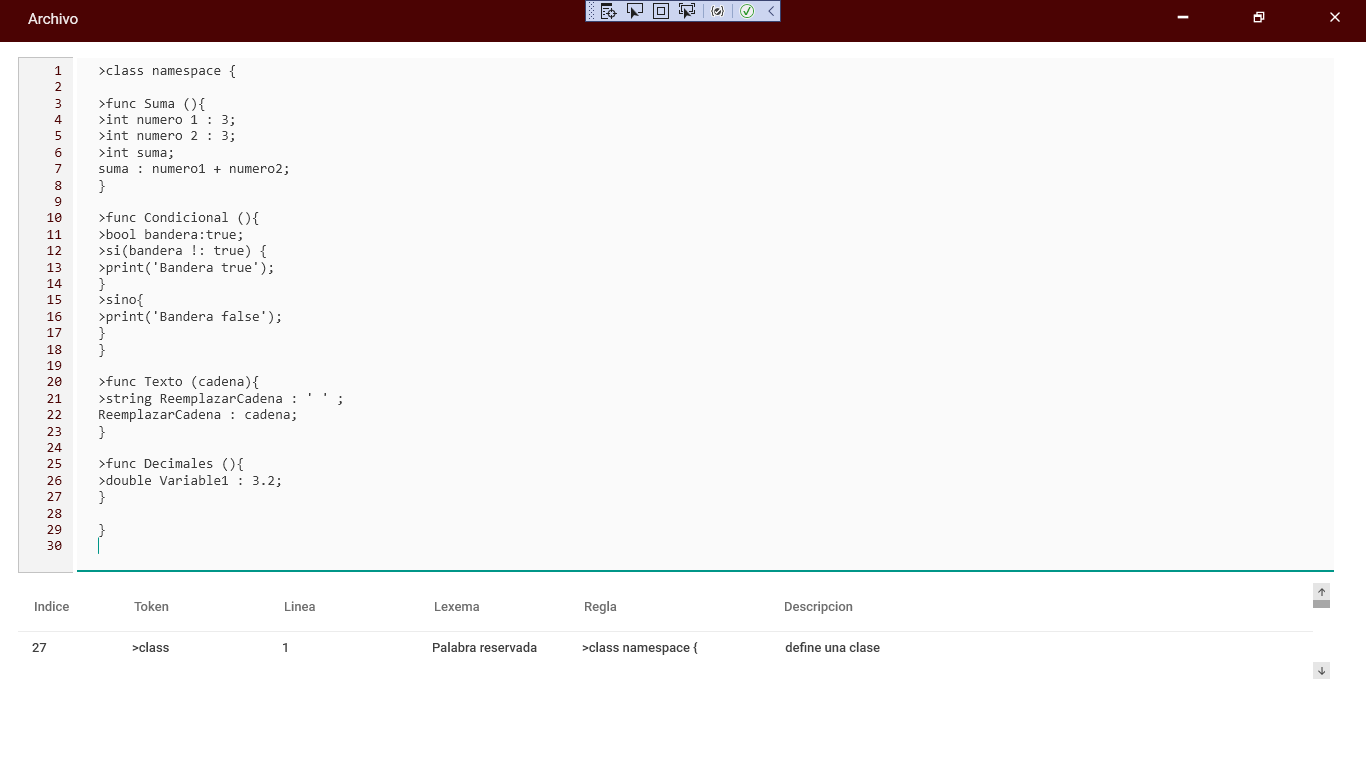
Este tipo de analizador pertenece al grupo de analizadores ascendentes, porque construye su árbol sintáctico desde las hojas hacia la raíz. Además, estos utilizan el método o técnica de análisis por desplazamiento reducción.

**Generadores de Analizadores Sintácticos**

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Tipo** | **Lenguaje** |
| **YACC**: Yet  Another Compiler  – Compiler | LALR | C |
| **Bison**: GNU version  libre de YACC | LALR | C |
| **AYACC** | LALR | Ada |
| **BYACC**: Berkeley  YACC | LALR | C |
| **PCYACC** | LALR | C, C++, Java,  Delphi, ... |
| **CUP**: constructor  of Useful Parser | LALR | Java |
| **COCO/R** | Descenso  recursivo LL(K) | C, C++, C#, Java |

**2-) EJERCICIOS:**

1. **Crear un analizador Sintáctico (Cualquier tipo, LL, LR o sin usar estos métodos) en el Lenguaje de programación de su preferencia.**



**CODIGO UTILIZADO**

>class namespace {

>func Suma (){

>int numero 1 : 3;

>int numero 2 : 3;

>int suma;

suma : numero1 + numero2;

}

>func Condicional (){

>bool bandera:true;

>si(bandera !: true) {

>print('Bandera true');

}

>sino{

>print('Bandera false');

}

}

>func Texto (cadena){

>string ReemplazarCadena : ' ' ;

ReemplazarCadena : cadena;

}

>func Decimales (){

>double Variable1 : 3.2;

}

}

1. **Subir el código a Github, enviar el código fuente y el ejecutable del proyecto.**

[**https://github.com/livanh1/AnalizadorSintactico**](https://github.com/livanh1/AnalizadorSintactico)