

# Práctica 3

*Pulido Reséndiz Aarón Isai*

Lenguajes y Autómatas II

---

## Introducción

En esta práctica se verán y aplicarán conceptos vistos anteriormente en la materia. Se deberá de modificar un código ya definido para generar un programa que sea capaz de identificar y producir un árbol de derivación a partir de las operaciones básicas (suma, resta, multiplicación y división).

Así mismo, se deberán de efectuar varias pruebas para verificar la funcionalidad del programa.

# Desarrollo

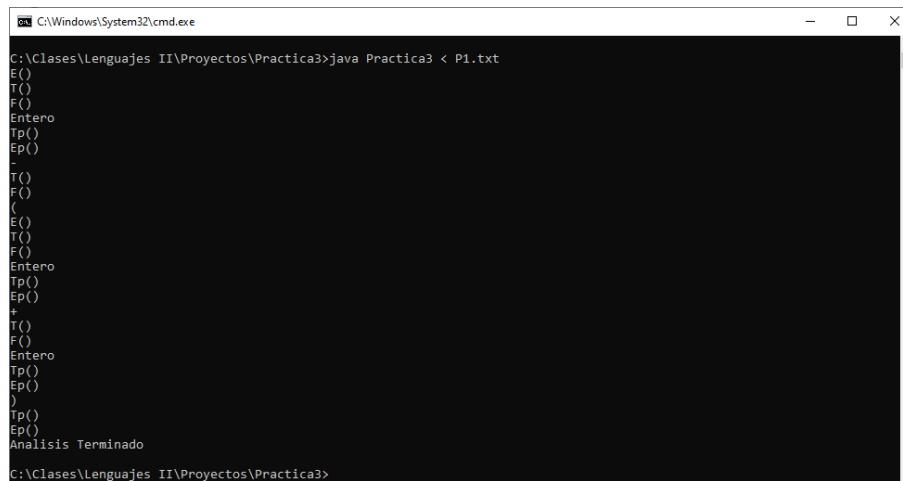
Para esta práctica se nos dio un código ya definido, el cual ya aceptaba las operaciones de suma y multiplicación, por lo cual solo se tuvieron que hacer unas pequeñas modificaciones para que funcionara con todas las operaciones básicas; teniendo que agregar solo que agregar las operaciones de resta y división. El código utiliza llamadas a métodos para poder “tomar decisiones” de que operaciones van a continuación. A continuación, se presenta el código de la práctica:

```
1  options
2  {
3      LOOKAHEAD=2; //Para determinar si es un número negativo o una resta
4  }
5  PARSER_BEGIN(Practica3)
6  {
7      public class Practica3 {
8          public static void main(String args[]) throws ParseException {
9              Practica3 parser = new Practica3(System.in);
10             parser.Inicial();
11             System.out.println("Análisis Terminado");
12         }
13     }
14 }
15 PARSER_END(Practica3)
16
17 SKIP :
18 { " " | "\t" | "\n" | "\r" }
19
20 TOKEN :
21 {
22     <OPSUM>: "+" |
23     <OPMUL>: "*" |
24     <OPDIV>: "/" | //Operandos
25     <OPRES>: "-" |
26     <NUMDEC>: ("0" | ("1"-"9")(<DIGDEC>)*> | //Se aceptan números enteros, donde el signo negativo es opcional (Si es un entero negativo)
27     <DIGDEC>: ["0"-"9"]> |
28     <IDENTIFICADOR>: ["A"-"Z", "a"-"z"](["A"-"Z", "a"-"z"] | ["0"-"9"] | "_" ){0,9}>
29 }
30
31 //Producciones {System.out.println("");}
32 void Inicial() :
33 {
34     {({System.out.println("E()");} E() )+ <EOF>
35 }
36
37 void E() :
38 {
39     {System.out.println("T()");} T() {System.out.println("Ep()");} Ep()
40 }
41
42 void Ep() :
43 {
44     [{System.out.println("+");} <OPSUM> {System.out.println("T()");} T() {System.out.println("Ep()");} Ep()
45     |
46     {System.out.println("-");} <OPRES> {System.out.println("T()");} T() {System.out.println("Ep()");} Ep() ] //Modificación para aceptar las restas
47 }
48
49 void T() :
50 {
51     {System.out.println("F()");} F() {System.out.println("Tp()");} Tp()
52 }
53
54 void Tp() :
55 {
56     [{System.out.println("*");} <OPMUL> {System.out.println("F()");} F() {System.out.println("Tp()");} Tp()
57     |
58     {System.out.println("/");} <OPDIV> {System.out.println("F()");} F() {System.out.println("Tp()");} Tp() ] //Modificación para aceptar las divisiones
59 }
60
61 void F() :
62 {
63     {System.out.println("Entero");} <NUMDEC>
64     |
65     {System.out.println("Identificador");} < IDENTIFICADOR>
66     |
67     {System.out.println("(");} "(" {System.out.println("E()");} E() {System.out.println(")");} ")"
68 }
69 }
```

**Figura 1.** Código de la Práctica 3.

Para comprobar la funcionalidad del programa generado, se realizaron 5 pruebas diferentes, dentro de las cuales algunas de ellas combinan diferentes operaciones, números positivos y números negativos. Adicionalmente, se intentó implementar las tabulaciones para que pareciera un árbol de derivación, pero no se tuvo éxito ya que solo algunas partes del árbol se representaban de manera correcta; esto se puede ver en la prueba P6. A continuación, se mostrarán las pruebas y resultados:

$$1) P1 = 9 - (8 + 9)$$



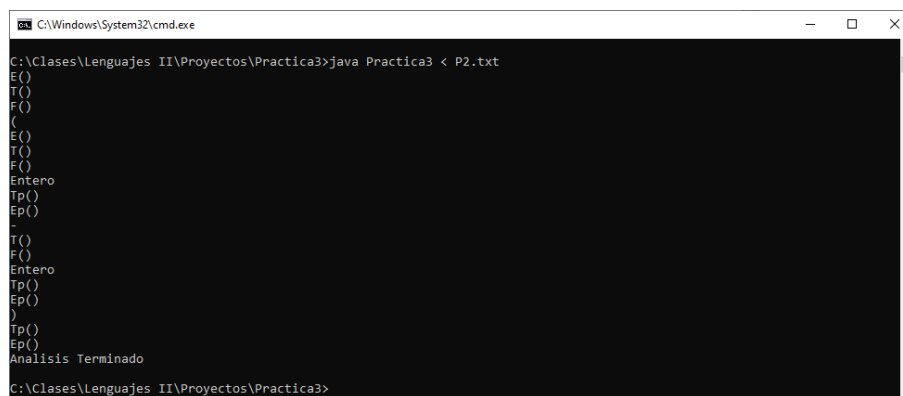
```

C:\Windows\System32\cmd.exe
C:\Clases\Lenguajes II\Proyectos\Practica3>java Practica3 < P1.txt
E()
T()
F()
Entero
Tp()
Ep()
-
T()
F()
(
E()
T()
F()
Entero
Tp()
Ep()
+
T()
F()
Entero
Tp()
Ep()
)
Tp()
Ep()
Analisis Terminado
C:\Clases\Lenguajes II\Proyectos\Practica3>

```

**Figura 2.** Árbol de derivación de la Prueba #1.

$$2) P2 = (4 - 7)$$



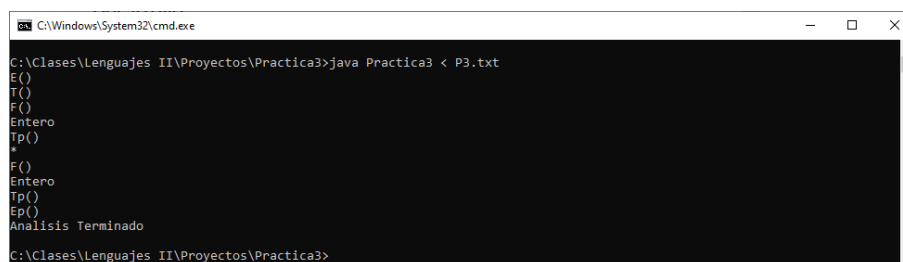
```

C:\Windows\System32\cmd.exe
C:\Clases\Lenguajes II\Proyectos\Practica3>java Practica3 < P2.txt
E()
T()
F()
(
E()
T()
F()
Entero
Tp()
Ep()
-
T()
F()
Entero
Tp()
Ep()
)
Tp()
Ep()
Analisis Terminado
C:\Clases\Lenguajes II\Proyectos\Practica3>

```

**Figura 3.** Árbol de derivación de la Prueba #2.

$$3) P3 = 9 * -5$$



```

C:\Windows\System32\cmd.exe
C:\Clases\Lenguajes II\Proyectos\Practica3>java Practica3 < P3.txt
E()
T()
F()
Entero
Tp()
*
F()
Entero
Tp()
Ep()
Analisis Terminado
C:\Clases\Lenguajes II\Proyectos\Practica3>

```

**Figura 4.** Árbol de derivación de la Prueba #3.

4)  $P4 = 7 + (-4 - -7)$

```

C:\Windows\System32\cmd.exe
C:\Clases\Lenguajes II\Proyectos\Practica3>java Practica3 < P4.txt
E()
T()
F()
Entero
Tp()
Ep()
+
T()
F()
(
E()
T()
F()
Entero
Tp()
Ep()
-
T()
F()
Entero
Tp()
Ep()
)
Tp()
Ep()
Analisis Terminado
C:\Clases\Lenguajes II\Proyectos\Practica3>

```

**Figura 5.** Árbol de derivación de la Prueba #4.

5)  $P5 = 5 / (9 * 7)$

```

C:\Windows\System32\cmd.exe
C:\Clases\Lenguajes II\Proyectos\Practica3>java Practica3 < P5.txt
E()
T()
F()
Entero
Tp()
/
F()
(
E()
T()
F()
Entero
Tp()
*
F()
Entero
Tp()
Ep()
)
Tp()
Ep()
Analisis Terminado
C:\Clases\Lenguajes II\Proyectos\Practica3>

```

**Figura 6.** Árbol de derivación de la Prueba #5.

6)  $P5 = 3 * var + 4$

```

C:\Windows\System32\cmd.exe
File "ParseException.java" is being rebuilt.
File "Token.java" is being rebuilt.
File "SimpleCharStream.java" is being rebuilt.
Parser generated with 0 errors and 1 warnings.
C:\Semestre Feb-Jul 2021\LENGUAJES Y AUTÓMATAS II\Proyectos\Practica3>javac *.java
C:\Semestre Feb-Jul 2021\LENGUAJES Y AUTÓMATAS II\Proyectos\Practica3>java Practica 3 < P6.txt
Error: no se ha encontrado o cargado la clase principal Practica
C:\Semestre Feb-Jul 2021\LENGUAJES Y AUTÓMATAS II\Proyectos\Practica3>java Practica3 < P6.txt
E()
T()
F()
Entero
Tp()
*
F()
Identificador
Tp()
Ep()
+
T()
F()
Entero
Tp()
Ep()
Analisis Terminado
C:\Semestre Feb-Jul 2021\LENGUAJES Y AUTÓMATAS II\Proyectos\Practica3>

```

**Figura 7.** Prueba fallida del Árbol de derivación de la Prueba #6.

## Conclusiones

Durante la pasada semana se repasaron los conceptos vistos dentro de la materia de Lenguajes y Autómatas I, los cuales fueron los GIC (Gramática Independiente del Contexto) y los Árboles de derivación. Estos se vieron de manera “teórica” para después aplicarlos dentro de un programa en JavaCC, lo cual fue interesante.

Dentro de la práctica se planteó una problemática, la cual fue que el programa debía de ser capaz de diferenciar entre un operador de resta y un signo de un número negativo, para lo cual se hizo uso de la opción *LOOKAHEAD*, mandándole un valor de 2; esto para que también contemplara el siguiente carácter y pudiera tomar la decisión de qué camino tomar dentro del árbol de derivación.

El único problema o impedimento que se tuvo durante la práctica fue que no se tuvo éxito al intentar implementar las tabulaciones para simular un árbol de derivación real (Como se muestra en la **Figura 7**), por lo cual se decidió mostrar el árbol de manera “plana”; esperando que en futuras clases se pueda resolver este problema.

Como conclusión final, esta práctica fue sencilla pero interesante para recordar ciertos conceptos de Lenguajes y Autómatas I, los cuales nos servirán a lo largo de la presente materia para el desarrollo de un lenguaje ensamblador.