



INSTITUTO TECNOLÓGICO DE JIQUILPAN

INGENIERÍA EN SISTEMAS COMPUTACIONALES

AUTOMATAS II

UNIDAD 2:

Generación de código intermedio

“Actividad 9”

Generador de código intermedio

PRESENTAN:

SALVADOR RIVAS LEPEZ

EILEEN GIZELLE VEGA HERNÁNDEZ

DOCENTE:

FERNANDO CARRANZA CAMPOS

JIQUILPAN, MICHOACÁN, 16 DE DICIEMBRE DEL 2021

Analizador Semántico LR

Gramática Empleada

$G(T, N, P, S) = \{ \quad T = \{id, int, float, char, ,, ,, ,, +, -, *, /, (,)\}, \quad N = \{P, Tipo, V, A, Exp, Term, T, F\},$
 $P = \{ \quad P \rightarrow Tipo \ id \ V \mid A$
 $\quad Tipo \rightarrow int \mid float \mid char$
 $\quad V \rightarrow ,id \ V \mid ;P$
 $\quad A \rightarrow id=Exp$
 $\quad Exp \rightarrow Term \ E$
 $\quad E \rightarrow +Term \ E \mid -Term \ E \mid \epsilon$
 $\quad Term \rightarrow F \ T$
 $\quad T \rightarrow *FT \mid /FT \mid \epsilon$
 $\quad F \rightarrow id \mid (Exp)$
 $\} S = \{P\} \}$

1. Expandir gramática	2. Añadir no terminal previo al inicial	3. Enumerar las producciones
P →Tipo id V P →A Tipo → int Tipo → float Tipo → char V →,id V V →;P A → id=Exp Exp →Term E E →+Term E E →-Term E E → ε Term →FT T →*FT T →/FT T → ε F →id F →(Exp)	P' →P P →Tipo id V P →A Tipo → int Tipo → float Tipo → char V →,id V V →;P A → id=Exp Exp →Term E E →+Term E E →-Term E E → ε Term →FT T →*FT T →/FT T → ε F →id F →(Exp)	P0> P'→P P1> P→Tipo id V P2> P→A P3> Tipo→ int P4> Tipo→ float P5> Tipo→ char P6> V→,id V P7> V→;P P8> A→ id=Exp P9> Exp→Term E P10> E→+Term E P11> E→-Term E P12> E→ ε P13> Term→FT P14> T→*FT P15> T→/FT P16> T→ ε P17> F→id P18> F→(Exp)

4. Elaborar tabla de primeros y siguientes

	PRIMEROS	SIGUIENTES
P'	int , float, char , id	\$
P	int , float, char , id	\$
Tipo	int , float , char	id
V	, , ;	\$
A	id	\$
Exp	id , (\$,)
E	+ , - , ε	\$,)
Term	id , (+ , - , \$,)
T	*, / , ε	+ , - , \$,)
F	id , (*, / , + , - , \$,)

5. Hacer funciones de cerradura e ir a

$I_0 = \text{Cerradura}(P' \rightarrow P \bullet) =$ { $P' \rightarrow \bullet P$ >I1 $P \rightarrow \bullet \text{Tipo id V}$ >I2 $P \rightarrow \bullet A$ >I3 $\text{Tipo} \rightarrow \bullet \text{int}$ >I4 $\text{Tipo} \rightarrow \bullet \text{float}$ >I5 $\text{Tipo} \rightarrow \bullet \text{char}$ >I6 $A \rightarrow \bullet \text{id} = \text{Exp}$ >I7 }	$I_1 = \text{Cerradura}(P' \rightarrow P \bullet) = \text{lr_a}(I_0, P)$ { $P' \rightarrow P \bullet$ >P0 $\text{Sig}(P') = \{\\$ \}$ }	$I_2 = \text{Cerradura}(P \rightarrow \text{Tipo} \bullet \text{id V}) = \text{lr_a}(I_0, \text{Tipo})$ { $P \rightarrow \text{Tipo} \bullet \text{id V}$ >I8 }
$I_3 = \text{Cerradura}(P \rightarrow A \bullet) = \text{lr_a}(I_0, A)$ { $P \rightarrow A \bullet$ >P2 $\text{Sig}(P) = \{\\$ \}$ }	$I_4 = \text{Cerradura}(\text{Tipo} \rightarrow \text{int} \bullet) = \text{lr_a}(I_0, \text{int})$ { $\text{Tipo} \rightarrow \text{int} \bullet$ >P3 $\text{Sig}(\text{Tipo}) = \{\text{id}\}$ }	$I_5 = \text{Cerradura}(\text{Tipo} \rightarrow \text{float} \bullet) = \text{lr_a}(I_0, \text{float})$ { $\text{Tipo} \rightarrow \text{float} \bullet$ >P4 $\text{Sig}(\text{Tipo}) = \{\text{id}\}$ }
$I_6 = \text{Cerradura}(\text{Tipo} \rightarrow \text{char} \bullet) = \text{lr_a}(I_0, \text{char})$ { $\text{Tipo} \rightarrow \text{char} \bullet$ >P5 $\text{Sig}(\text{Tipo}) = \{\text{id}\}$ }	$I_7 = \text{Cerradura}(A \rightarrow \text{id} \bullet = \text{Exp}) = \text{lr_a}(I_0, \text{id})$ { $A \rightarrow \text{id} \bullet = \text{Exp}$ >I9 }	$I_8 = \text{Cerradura}(P \rightarrow \text{Tipo id} \bullet V) = \text{lr_a}(I_2, \text{id})$ { $P \rightarrow \text{Tipo id} \bullet V$ >I10 $V \rightarrow \bullet, \text{id V}$ >I11 $V \rightarrow \bullet; P$ >I12 }
$I_9 = \text{Cerradura}(A \rightarrow \text{id} = \bullet \text{Exp}) = \text{lr_a}(I_7, =)$ { $A \rightarrow \text{id} = \bullet \text{Exp}$ >I13 $\text{Exp} \rightarrow \bullet \text{Term E}$ >I14 $\text{Term} \rightarrow \bullet \text{FT}$ >I15 $F \rightarrow \bullet \text{id}$ >I16 $F \rightarrow \bullet (\text{Exp})$ >I17 }	$I_{10} = \text{Cerradura}(P \rightarrow \text{Tipo id V} \bullet) = \text{lr_a}(I_8, V)$ { $P \rightarrow \text{Tipo id V} \bullet$ >P1 $\text{Sig}(P) = \{\\$ \}$ }	$I_{11} = \text{Cerradura}(V \rightarrow \bullet, \text{id V}) = \text{lr_a}(I_8, ,)$ { $V \rightarrow \bullet, \text{id V}$ >I18 }

<p>I12=Cerradura($V \rightarrow ; \bullet P$) =lr_a(I8,;)</p> <pre>{ V → ; • P >I19 P → • Tipo id V >I2 P → • A >I3 Tipo → • int >I4 Tipo → • float >I5 Tipo → • char >I6 A → • id=Exp >I7 }</pre>	<p>I13=Cerradura($A \rightarrow id = \bullet \text{Exp}$) =lr_a(I9,Exp)</p> <pre>{ A → id=Exp • >P8 Sig(Tipo)={ \$ } }</pre>	<p>I14=Cerradura($\text{Exp} \rightarrow \text{Term} \bullet E$) =lr_a(I9,Term)</p> <pre>{ Exp → Term • E >I20 E → • +Term E >I21 E → • -Term E >I22 E → • ε >P12 Sig(E)={ \$,) } }</pre>
<p>I15=Cerradura($\text{Term} \rightarrow F \bullet T$) =lr_a(I9,F)</p> <pre>{ Term → F • T >I23 T → • *FT >I24 T → • /FT >I25 T → • ε >P6 Sig(T)={ + , - , \$,) } }</pre>	<p>I16=Cerradura($F \rightarrow id \bullet$) =lr_a(I9,id)</p> <pre>{ F → id • >P17 Sig(F)={ *, / , + , - , \$,) } }</pre>	<p>I17=Cerradura($F \rightarrow (\bullet \text{Exp})$) =lr_a(I9,())</p> <pre>{ F → (• Exp) >I26 Exp → • Term E >I14 Term → • FT >I15 F → • id >I16 F → • (Exp) >I17 }</pre>
<p>I18=Cerradura($V \rightarrow , id \bullet V$) =lr_a(I11,id)</p> <pre>{ V → , id • V >I27 V → • , id V >I11 V → • ; P >I12 }</pre>	<p>I19=Cerradura($V \rightarrow ; P \bullet$) =lr_a(I12,P)</p> <pre>{ V → ; P • >P7 Sig(V)={ \$ } }</pre>	<p>I20=Cerradura($\text{Exp} \rightarrow \text{Term } E \bullet$) =lr_a(I14,E)</p> <pre>{ Exp → Term E • >P9 Sig(Exp)={ \$,) } }</pre>
<p>I21=Cerradura($E \rightarrow + \bullet \text{Term } E$) =lr_a(I14,+)</p> <pre>{ E → + • Term E >I28 Term → • FT >I15 F → • id >I16 F → • (Exp) >I17 }</pre>	<p>I22=Cerradura($E \rightarrow - \bullet \text{Term } E$) =lr_a(I14,-)</p> <pre>{ E → - • Term E >I29 Term → • FT >I15 F → • id >I16 F → • (Exp) >I17 }</pre>	<p>I23=Cerradura($\text{Term} \rightarrow FT \bullet$) =lr_a(I15,T)</p> <pre>{ Term → FT • >P13 Sig(Term)={ + , - , \$,) } }</pre>
<p>I24=Cerradura($T \rightarrow * \bullet FT$) =lr_a(I15,*)</p> <pre>{ T → * • FT >I30 F → • id >I16 F → • (Exp) >I17 }</pre>	<p>I25=Cerradura($T \rightarrow / \bullet FT$) =lr_a(I15,/)</p> <pre>{ T → / • FT >I31 F → • id >I16 F → • (Exp) >I17 }</pre>	<p>I26=Cerradura($F \rightarrow (\text{Exp} \bullet)$) =lr_a(I17,Exp)</p> <pre>{ F → (Exp •) >I32 }</pre>
<p>I27=Cerradura($V \rightarrow , id V \bullet$) =lr_a(I18, V)</p> <pre>{ V → , id V • >P6 Sig(V) = { \$ } }</pre>	<p>I28=Cerradura($E \rightarrow + \text{Term} \bullet E$) =lr_a(I21, Term)</p> <pre>{ E → +Term • E >I33 E → • +Term E >I21 E → • -Term E >I22 E → • ε >P12 Sig(E) = { \$,) } }</pre>	<p>I29=Cerradura($E \rightarrow - \text{Term} \bullet E$) =lr_a(I22, Term)</p> <pre>{ E → -Term • E >I34 E → • +Term E >I21 E → • -Term E >I22 E → • ε >P12 Sig(E) = { \$,) } }</pre>

<p>I30=Cerradura($T \rightarrow *F \bullet T$) =lr_a(I24, F)</p> <p>{</p> <p>$T \rightarrow *F \bullet T$ >I35</p> <p>$T \rightarrow \bullet *FT$ >I24</p> <p>$T \rightarrow \bullet /FT$ >I25</p> <p>$T \rightarrow \bullet \epsilon$ >P16</p> <p>Sig(T) = {+ , - , \$,)}</p> <p>}</p>	<p>I31=Cerradura($T \rightarrow /F \bullet T$) =lr_a(I25, F)</p> <p>{</p> <p>$T \rightarrow /F \bullet T$ >I36</p> <p>$T \rightarrow \bullet *FT$ >I24</p> <p>$T \rightarrow \bullet /FT$ >I25</p> <p>$T \rightarrow \bullet \epsilon$ >P16</p> <p>Sig(T) = {+ , - , \$,)}</p> <p>}</p>	<p>I32=Cerradura($F \rightarrow (Exp) \bullet$) =lr_a(I26,))</p> <p>{</p> <p>$F \rightarrow (Exp) \bullet$ >P18</p> <p>Sig(F) = {*, /, + , - , \$,)}</p> <p>}</p>
<p>I33=Cerradura($E \rightarrow +TermE \bullet$) =lr_a(I28, E)</p> <p>{</p> <p>$E \rightarrow +TermE \bullet$ >P10</p> <p>Sig(E) = { \$,) }</p> <p>}</p>	<p>I34=Cerradura($E \rightarrow -TermE \bullet$) =lr_a(I29, E)</p> <p>{</p> <p>$E \rightarrow -TermE \bullet$ >P11</p> <p>Sig(E) = { \$,) }</p> <p>}</p>	<p>I35=Cerradura($T \rightarrow *FT \bullet$) =lr_a(I30, T)</p> <p>{</p> <p>$T \rightarrow *FT \bullet$ >P14</p> <p>Sig(T) = {+ , - , \$,)}</p> <p>}</p>
<p>I36=Cerradura($T \rightarrow /FT \bullet$) =lr_a(I31, T)</p> <p>{</p> <p>$T \rightarrow /FT \bullet$ >P15</p> <p>Sig(T) = {+ , - , \$,)}</p> <p>}</p>		

7. Tabla de análisis sintáctico

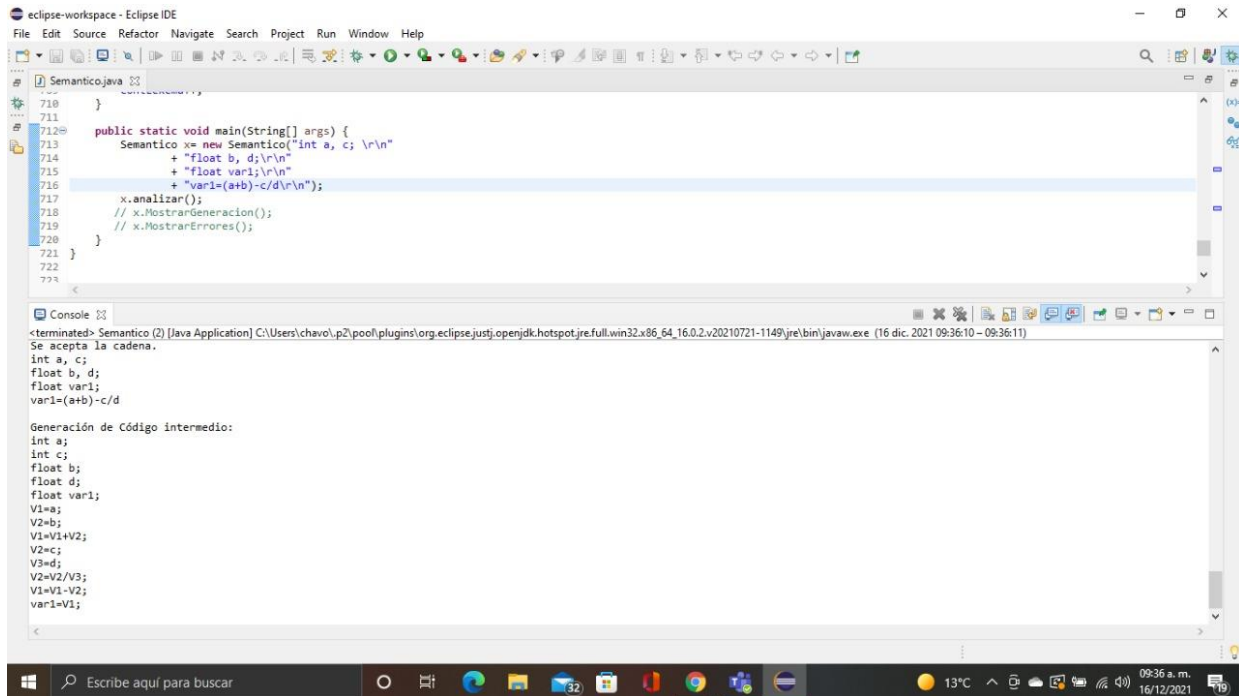
Analizar int var1, var2; float var3; var1=(var2+var2)*var3

Entrada	Pila	Acción/Reducción	Pila semántica
int id, id; float id; id=(id+id)*id\$	\$ I0		
int	\$ I0	I0 con int desplaza a I4	
id	\$ I0 int I4	I4 con id P3, busco edo	
Id	\$ I0 Tipo I2	I2 con id desplaza a I8	
,	\$ I0 Tipo I2 id I8	I8 con , desplaza a I11	
id	\$ I0 Tipo I2 id I8 , I11	I11 con id desplaza a I18	
;	\$ I0 Tipo I2 id I8 , I11 id I18	I18 con ; desplaza a I12	
float	\$ I0 Tipo I2 id I8 , I11 id I18 ; I12	I12 con float desplaza a I5	
id	\$ I0 Tipo I2 id I8 , I11 id I18 ; I12 float I5	I5 con id P4, busco edo	
id	\$ I0 Tipo I2 id I8 , I11 id I18 ; I12 Tipo I2	I2 con id desplaza a I8	
;	\$ I0 Tipo I2 id I8 , I11 id I18 ; I12 Tipo I2 id I8	I8 con ; desplaza a I12	
id	\$ I0 Tipo I2 id I8 , I11 id I18 ; I12 Tipo I2 id I8 ; I12	I12 con id desplaza a I7	
=	\$ I0 Tipo I2 id I8 , I11 id I18 ; I12 Tipo I2 id I8 ; I12 id I7	I7 con = desplaza a I9	
(\$ I0 Tipo I2 id I8 , I11 id I18 ; I12 Tipo I2 id I8 ; I12 id I7 = I9	I9 con (desplaza a I17	
Id	\$ I0 Tipo I2 id I8 , I11 id I18 ; I12 Tipo I2 id I8 ; I12 id I7 = I9 (I17	I17 con id desplaza a I16	
+	\$ I0 Tipo I2 id I8 , I11 id I18 ; I12 Tipo I2 id I8 ; I12 id I7 = I9 (I17 id I16	I16 con + P17, busco edo	1
+	\$ I0 Tipo I2 id I8 , I11 id I18 ; I12 Tipo I2 id I8 ; I12 id I7 = I9 (I17 F I15	I15 con + P16, busco edo	1
+	\$ I0 Tipo I2 id I8 , I11 id I18 ; I12 Tipo I2 id I8 ; I12 id I7 = I9 (I17 F I15 T I23	I23 con + P13, busco edo	1
+	\$ I0 Tipo I2 id I8 , I11 id I18 ; I12 Tipo I2 id I8 ; I12 id I7 = I9 (I17 Term I14	I14 con + desplaza a I21	1
id	\$ I0 Tipo I2 id I8 , I11 id I18 ; I12 Tipo I2 id I8 ; I12 id I7 = I9 (I17 Term I14 + I21	I21 con id desplaza a I16	1
)	\$ I0 Tipo I2 id I8 , I11 id I18 ; I12 Tipo I2 id I8 ; I12 id I7 = I9 (I17 Term I14 + I21 id I16	I16 con) P17, busco edo	1 1
)	\$ I0 Tipo I2 id I8 , I11 id I18 ; I12 Tipo I2 id I8 ; I12 id I7 = I9 (I17 Term I14 + I21 F I15	I15 con) P16, busco edo	1 1
)	\$ I0 Tipo I2 id I8 , I11 id I18 ; I12 Tipo I2 id I8 ; I12 id I7 = I9 (I17 Term I14 + I21 F I15 T I23	I23 con) P13, busco edo	1 1
)	\$ I0 Tipo I2 id I8 , I11 id I18 ; I12 Tipo I2 id I8 ; I12 id I7 = I9 (I17 Term I14 + I21 Term I28	I28 con) P12, busco edo	1 1
)	\$ I0 Tipo I2 id I8 , I11 id I18 ; I12 Tipo I2 id I8 ; I12 id I7 = I9 (I17 Term I14 + I21 Term I28 E I33	I33 con) P10, busco edo	1 1
)	\$ I0 Tipo I2 id I8 , I11 id I18 ; I12 Tipo I2 id I8 ; I12 id I7 = I9 (I17 Term I14 E I20	I20 con) P9, busco edo	1 1
)	\$ I0 Tipo I2 id I8 , I11 id I18 ; I12 Tipo I2 id I8 ; I12 id I7 = I9 (I17 Exp I26	I26 con) desplaza a I32	1
*	\$ I0 Tipo I2 id I8 , I11 id I18 ; I12 Tipo I2 id I8 ; I12 id I7 = I9 (I17 Exp I26) I32	I32 con * P18, busco edo	1
*	\$ I0 Tipo I2 id I8 , I11 id I18 ; I12 Tipo I2 id I8 ; I12 id I7 = I9 F I15	I15 con * desplaza a I24	1
id	\$ I0 Tipo I2 id I8 , I11 id I18 ; I12 Tipo I2 id I8 ; I12 id I7 = I9 F I15 * I24	I24 con id desplaza a I16	1
\$	\$ I0 Tipo I2 id I8 , I11 id I18 ; I12 Tipo I2 id I8 ; I12 id I7 = I9 F I15 * I24 id I16	I16 con \$ P17, busco edo	1 2
\$	\$ I0 Tipo I2 id I8 , I11 id I18 ; I12 Tipo I2 id I8 ; I12 id I7 = I9 F I15 * I24 F I30	I30 con \$ P16, busco edo	1 2
\$	\$ I0 Tipo I2 id I8 , I11 id I18 ; I12 Tipo I2 id I8 ; I12 id I7 = I9 F I15 * I24 F I30 T I35	I35 con \$ P14, busco edo	1 2
\$	\$ I0 Tipo I2 id I8 , I11 id I18 ; I12 Tipo I2 id I8 ; I12 id I7 = I9 F I15 T I23	I23 con \$ P13, busco edo	2
\$	\$ I0 Tipo I2 id I8 , I11 id I18 ; I12 Tipo I2 id I8 ; I12 id I7 = I9 Term I14	I14 con \$ P12, busco edo	2
\$	\$ I0 Tipo I2 id I8 , I11 id I18 ; I12 Tipo I2 id I8 ; I12 id I7 = I9 Term I14 E I20	I20 con \$ P9, busco edo	2
\$	\$ I0 Tipo I2 id I8 , I11 id I18 ; I12 Tipo I2 id I8 ; I12 id I7 = I9 Exp I13	I13 con \$ P8, busco edo	2
\$	\$ I0 Tipo I2 id I8 , I11 id I18 ; I12 Tipo I2 id I8 ; I12 A I3	I3 con \$ P2, busco edo	2
\$	\$ I0 Tipo I2 id I8 , I11 id I18 ; I12 Tipo I2 id I8 ; I12 P I19	I19 con \$ P7, busco edo	2
\$	\$ I0 Tipo I2 id I8 , I11 id I18 ; I12 Tipo I2 id I8 V I10	I10 con \$ P1, busco edo	2
\$	\$ I0 Tipo I2 id I8 , I11 id I18 ; I12 P I19	I19 con \$ P7, busco edo	2
\$	\$ I0 Tipo I2 id I8 , I11 id I18 V I27	I27 con \$ P6, busco edo	2
\$	\$ I0 Tipo I2 id I8 V I10	I10 con \$ P1, busco edo	2
\$	\$ I0 P I1	I1 con \$ P0, se acepta.	2

Reglas semánticas

+,- *, /	Int 1	Float 2	Char 3
Int 1	1	2	-1
Float 2	2	2	-1
Char 3	-1	-1	3

Ejemplo deCodigo fuente e intermedio



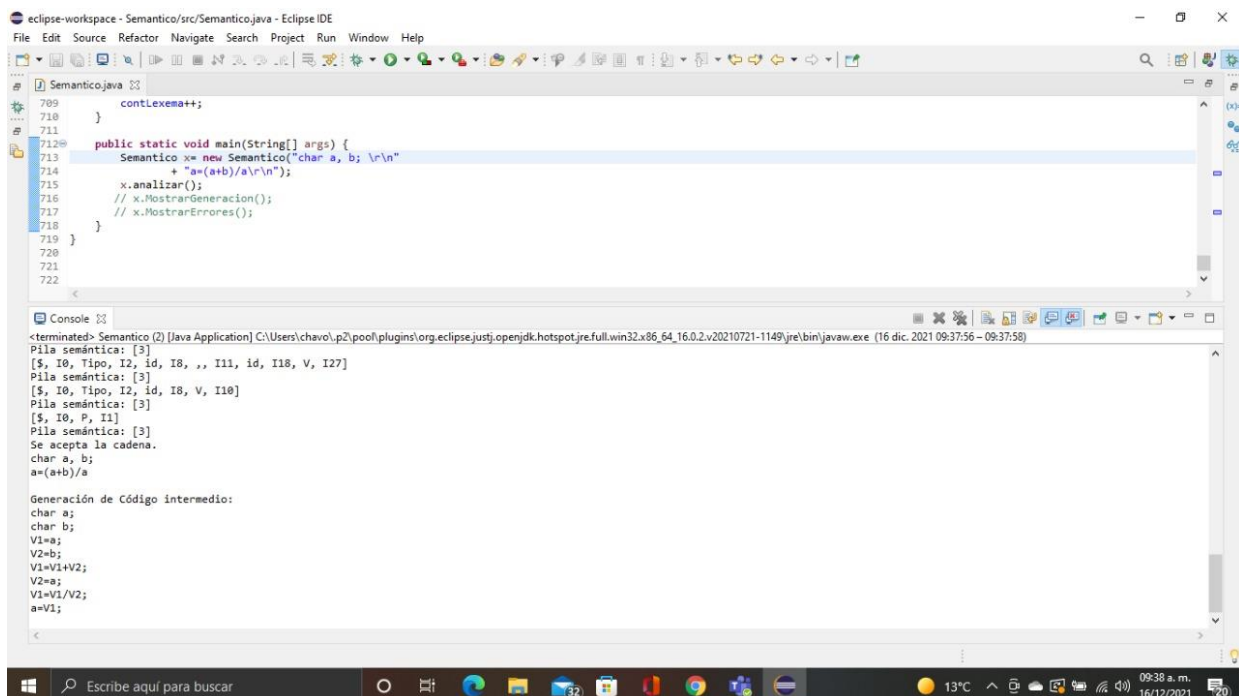
```
710 }
711
712 public static void main(String[] args) {
713     Semantico x= new Semantico("int a, c; \r\n"
714         + "float b, d;\r\n"
715         + "float var1;\r\n"
716         + "var1=(a+b)-c/d\r\n");
717     x.analizar();
718     // x.MostrarGeneracion();
719     // x.MostrarErrores();
720 }
721
722
723
```

<terminated> Semantico (2) [Java Application] C:\Users\chavo\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.16.0.2.v20210721-1149\jre\bin\javaw.exe (16 dic. 2021 09:36:10 – 09:36:11)

Se acepta la cadena.
int a, c;
float b, d;
float var1;
var1=(a+b)-c/d

Generación de Código intermedio:

```
int a;  
int c;  
float b;  
float d;  
float var1;  
V1=a;  
V2=b;  
V1=V1+V2;  
V2=c;  
V3=d;  
V2=V2/V3;  
V1=V1-V2;  
var1=V1;
```



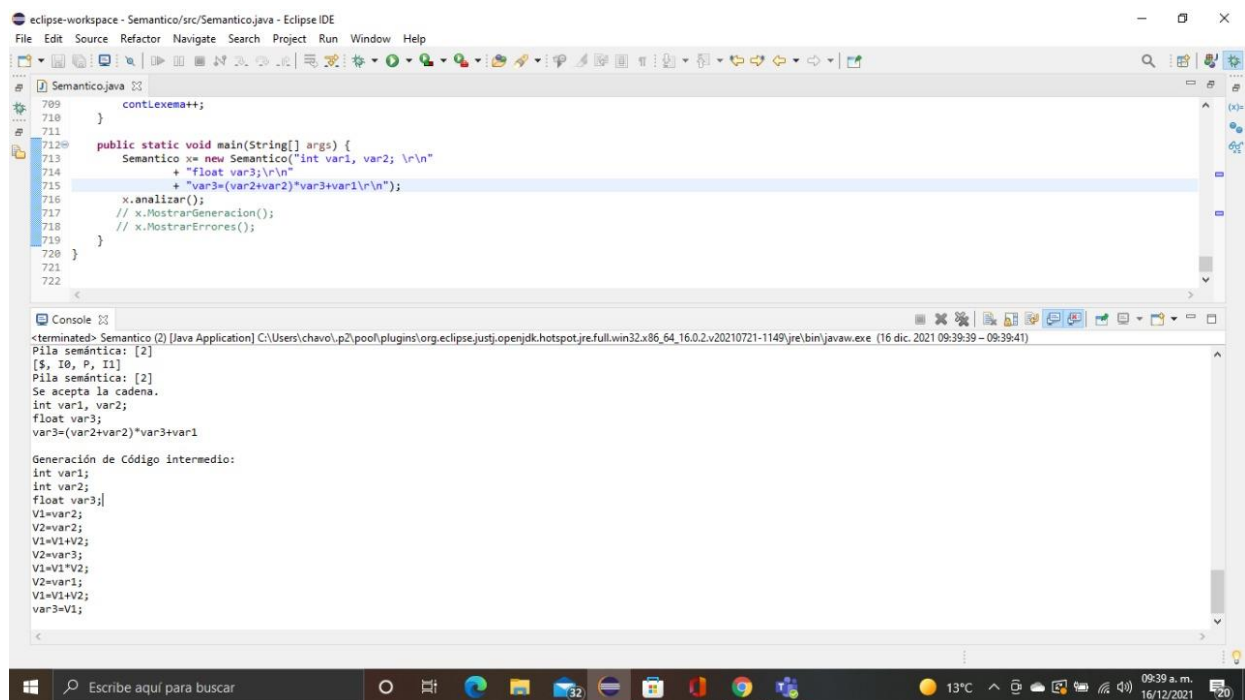
```
709     contLexema++;
710 }
711
712 public static void main(String[] args) {
713     Semantico x= new Semantico("char a, b; \r\n"
714         + "a=(a+b)/a\r\n");
715     x.analizar();
716     // x.MostrarGeneracion();
717     // x.MostrarErrores();
718 }
719
720
721
722
```

<terminated> Semantico (2) [Java Application] C:\Users\chavo\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.16.0.2.v20210721-1149\jre\bin\javaw.exe (16 dic. 2021 09:37:56 – 09:37:58)

Pila semántica: [3]
[5, I0, Tipo, I2, id, I8, ,, I11, id, I18, V, I27]
Pila semántica: [3]
[5, I0, Tipo, I2, id, I8, V, I10]
Pila semántica: [3]
[5, I0, P, I1]
Pila semántica: [3]
Se acepta la cadena.
char a, b;
a=(a+b)/a

Generación de Código intermedio:

```
char a;  
char b;  
V1=a;  
V2=b;  
V1=V1+V2;  
V2=a;  
V1=V1/V2;  
a=V1;
```

Código Java

```

import java.util.ArrayList;
import java.util.LinkedList;
import java.util.Stack;

```

```

class Reservado {

    /* tipo de datos */
    public static String tipoDatos[] = {"int", "float", "char"};
    public static String palabraReservada[] = { "int", "float", "char"};

    public boolean isReservado(String lexema)
    {
        for(String pal : palabraReservada)
        {
            if(pal.equalsIgnoreCase(lexema))
                return true;
        }
        return false;
    }
}

class Lexema {

    private int numero;
    private String lexema;
    private int fila;
    private String tipo;
    private String comp;
    private String td;
    public Lexema() {
    }
}

```

```

public Lexema(int numero, String lexema, int fila, String tipo, String comp, String td) {
    this.numero = numero;
    this.lexema = lexema;
    this.fila = fila;
    this.tipo = tipo;
    this.comp = comp;
    this.td = td;
    verificarReservado(lexema);
}

public void verificarReservado(String texto) {
    for (String item : Reservado.tipoDatos) {
        if (item.equalsIgnoreCase(texto)) {
            this.tipo = "TIPO DE DATO";
            return;
        }
    }
}

public int getNumero() {
    return numero;
}

public void setNumero(int numero) {
    this.numero = numero;
}

public String getLexema() {
    return lexema;
}

public void setLexema(String lexema) {
    this.lexema = lexema;
}

public int getFila() {
    return fila;
}

public void setFila(int fila) {
    this.fila = fila;
}

public String getTipo() {
    return tipo;
}

public void setTipo(String tipo) {
    this.tipo = tipo;
}

public String getComp() {
    return comp;
}

public void setComp(String comp) {
    this.comp = comp;
}

```

```

    }
    public String getTD() {
        return td;
    }

    public void setTD(String td) {
        this.td = td;
    }
    @Override
    public String toString() {
        return "Lexema{" + "numero=" + numero + ", lexema=" + lexema + ", fila=" + fila + ",  

        tipo=" + tipo + '}';
    }
}

public class Semantico {
    Stack<String> pila = new Stack<String>();
    public String tok;
    public String texto;
    public int linea;
    public ArrayList<Lexema> listaLexemas; //contiene los lexemas
    public ArrayList<Error> listaErrores; //contiene los errores producidos
    public Reservado reser = new Reservado();
    public ArrayList<String> errores = new ArrayList<String>();
    String dato;
    public int contError, contLexema, contLineas;
    boolean comentario;
    boolean activarSemantico = false, existe = false, datogen = false;
    String lexemaSem = "", cadenaGen = "", var="", datofinal="";

    int contid = 0, contgen = 0;
    Stack<String> pilaSint = new Stack<String>();
    Stack<Integer> pilaSem = new Stack<Integer>();
    Stack<String> pilaGen = new Stack<String>();
    LinkedList<String> listaGen = new LinkedList<>();

    String valores[] = {"id", "int", "float", "char", ",", ";", "+", "-", "*", "/", "(",  

    ")", "=", "$", "P", "Tipo", "V", "A", "Exp", "E", "Term", "T", "F"};
    String edos[] = {"I0", "I1", "I2", "I3", "I4", "I5", "I6", "I7", "I8", "I9", "I10",  

    "I11", "I12", "I13", "I14", "I15", "I16", "I17", "I18", "I19", "I20", "I21", "I22", "I23",  

    "I24", "I25", "I26", "I27", "I28", "I29", "I30", "I31", "I32", "I33", "I34", "I35", "I36"};

    String tabla[][] = {
        {"I7", "I4", "I5", "I6", "@", "@", "@", "@", "@", "@", "@", "@", "@", "@",  

        "@", "I1", "I2", "@", "I3", "@", "@", "@", "@", "@", "@",  

        {"@", "@", "@", "@", "@", "@", "@", "@", "@", "@", "@", "@", "P0",  

        "@", "@", "@", "@", "@", "@", "@", "@", "@",  

        {"I8", "@", "@", "@", "@", "@", "@", "@", "@", "@", "@", "@", "@",  

        "@", "@", "@", "@", "@", "@", "@", "@", "@",  

        {"@", "@", "@", "@", "@", "@", "@", "@", "@", "@", "@", "@", "P2",  

        "@", "@", "@", "@", "@", "@", "@", "@", "@",  

        {"P3", "@", "@", "@", "@", "@", "@", "@", "@", "@", "@", "@", "@",  

        "@", "@", "@", "@", "@", "@", "@", "@", "@",  

        {"P4", "@", "@", "@", "@", "@", "@", "@", "@", "@", "@", "@", "@",  

        "@", "@", "@", "@", "@", "@", "@", "@", "@"}
    }
}

```

[illegible]


```

        pilaSem.push(T1);
    }
    else
    {
        System.out.println("Error semántico en línea "+
linea+". No coinciden los tipos.");
        return;
    }
}
else
{
    System.out.println("Error semántico en línea "+
linea+". No hubo declaración de variable.");
    return;
}
String signo = "";
switch(vtabla)
{
    case "P10":
        signo = "+";
        break;
    case "P11":
        signo = "-";
        break;
    case "P14":
        signo = "*";
        break;
    case "P15":
        signo = "/";
        break;

}
cadenaGen = "V"+(contgen-1)+"="+ "V"+(contgen-
1)+signo+"V"+contgen+"";

listaGen.add(cadenaGen);
cadenaGen="";
contgen--;
}
String reduccion[] = Reducciones(vtabla);
if(reduccion[1].equals("e"))
{
    String edoanterior = pilaSint.peek();
    pilaSint.push(reduccion[0]);
    col = BuscarValores(pilaSint.peek());
    ren = BuscarEdos(edoanterior);
    vtabla = tabla[ren][col];
    pilaSint.push(vtabla);
    Analisis(token);
}
else
{
    String nr[] = reduccion[1].split(" ");
    for(int i=0; i<(nr.length*2); i++)
    {
        pilaSint.pop();
    }
    String edoanterior = pilaSint.peek();

```

```

        pilaSint.push(reduccion[0]);
        col = BuscarValores(pilaSint.peek());
        ren = BuscarEdos(edoanterior);
        vtabla = tabla[ren][col];
        pilaSint.push(vtabla);
        Analisis(token);
    }
}
// }

}

public void IdSem(String token)
{
    if(token.equals("id"))
    {
        if(existe)
        {
            contgen++;
            for (Lexema item : listaLexemas) {
                if(item.getLexema().equals(lexemaSem))
                {
                    dato = item.getTD();
                    break;
                }
            }
            switch(dato)
            {
                case "int":
                    pilaSem.push(1);
                    break;
                case "float":
                    pilaSem.push(2);
                    break;
                case "char":
                    pilaSem.push(3);
                    break;
            }
            cadenaGen = "V"+contgen+"="+lexemaSem+";";
            listaGen.add(cadenaGen);
        }
    }
}

}

public boolean VerificarFinalPila()
{
    int d=0;
    switch(datofinal)
    {
        case "int":
            d = 1;
            break;
        case "float":
            d = 2;
            break;
    }
}

```

```

        case "char":
            d = 3;
            break;
    }
    if(d == pilaSem.peek())
        return true;
    return false;
}
public String[] Reducciones(String p)
{
    String red[] = new String[2];
    switch(p)
    {
        case "P0":
            red[0] = "P'";
            red[1] = "P";
            break;
        case "P1":
            red[0] = "P";
            red[1] = "Tipo id V";
            break;
        case "P2":
            red[0] = "P";
            red[1] = "A";
            break;
        case "P3":
            red[0] = "Tipo";
            red[1] = "int";
            break;
        case "P4":
            red[0] = "Tipo";
            red[1] = "float";
            break;
        case "P5":
            red[0] = "Tipo";
            red[1] = "char";
            break;
        case "P6":
            red[0] = "V";
            red[1] = ", id V";
            break;
        case "P7":
            red[0] = "V";
            red[1] = "; P";
            break;
        case "P8":
            red[0] = "A";
            red[1] = "id = Exp";
            break;
        case "P9":
            red[0] = "Exp";
            red[1] = "Term E";
            break;
        case "P10":
            red[0] = "E";
            red[1] = "+ Term E";
            break;
    }
}

```



```

        case "P11":
            red[0] = "E";
            red[1] = "- Term E";
            break;
        case "P12":
            red[0] = "E";
            red[1] = "e";
            break;
        case "P13":
            red[0] = "Term";
            red[1] = "F T";
            break;
        case "P14":
            red[0] = "T";
            red[1] = "* F T";
            break;
        case "P15":
            red[0] = "T";
            red[1] = "/ F T";
            break;
        case "P16":
            red[0] = "T";
            red[1] = "e";
            break;
        case "P17":
            red[0] = "F";
            red[1] = "id";
            break;
        case "P18":
            red[0] = "F";
            red[1] = "( Exp )";
            break;
    }
    return red;
}

public int BuscarValores(String valor)
{
    for(int i=0; i<valores.length; i++)
    {
        if(valores[i].equalsIgnoreCase(valor))
            return i;
    }
    return -1;
}

public int BuscarEdos(String edo)
{
    for(int i=0; i<edos.length; i++)
    {
        if(edos[i].equalsIgnoreCase(edo))
            return i;
    }
    return -1;
}

public Semantico() {

```

```

        this.texto = "";
        this.listaLexemas = new ArrayList<>();
        this.listaErrores = new ArrayList<>();
        this.contLineas = 0;
        this.contLexema = 0;
        this.contError = 0;
        this.linea = 0;
    }

    public Semantico(String texto) {
        this.texto = texto;
        this.listaLexemas = new ArrayList<>();
        this.listaErrores = new ArrayList<>();
        this.contLineas = 0;
        this.contLexema = 0;
        this.contError = 0;
        this.linea = 0;
    }

    /* metodo que analiza todo el texto caracter por caracter */
    public void analizar() {
        Sintactico();
        System.out.println(texto);
        int longitud = texto.length();
        char c;
        int estado = 0;
        String lexema = "";
        for (int i = 0; i < longitud; i++) {
            c = texto.charAt(i);
            switch (estado) {
                case 0:
                    if (c >= '0' && c <= '9') { //numeros
                        lexema = lexema + c;
                        estado = 1;
                    } else if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || c == '$'
|| c == 'o') { //letras
                        lexema = lexema + c;
                        estado = 2;
                    } else if (c == ';') {
                        dato = "";
                        activarSemantico = false;
                        datogen = false;
                        Analisis(";");
                        addLexema(";", linea, "DELIMITADOR", ";", "");
                        estado = 0;

                    } else if (c == ' ') {
                        estado = 0;
                    } else if (c == '\n') {
                        estado = 0;
                        linea++;
                        contLineas++;
                    } else if (c == '=') {
                        addLexema("=", linea, "IGUAL", "=", "");
                        Analisis("=");
                        activarSemantico = true;
                    }
                }
            }
        }
    }

```

```

        var = lexemaSem;
        if(!BuscarDatoDeclarado(var))
        {
            System.out.println("Error semántico en línea "+ linea+". No hubo
declaración de variable.");
            return;
        }
        estado = 0;

    } else if (c == '+') {
        addLexema(" " + c, linea, "OPERADOR MATEMATICO", "+", "");
        Analisis("+");
        estado = 0;

    }else if (c == '-') {
        addLexema(" " + c, linea, "OPERADOR MATEMATICO", "-", "");
        Analisis("-");
        estado = 0;

    }else if (c == '*') {
        addLexema(" " + c, linea, "OPERADOR MATEMATICO", "*", "");
        Analisis("*");
        estado = 0;

    }else if (c == '/') {
        addLexema(" " + c, linea, "OPERADOR MATEMATICO", "/", "");
        Analisis("/");
        estado = 0;

    } else if (c == '(') {
        addLexema("(", linea, "PARENTESIS ABIERTO", "(", "");
        Analisis("(");
        estado = 0;
    } else if (c == ')') {
        addLexema(")", linea, "PARENTESIS CERRADO", ")", "");
        Analisis(")");
        estado = 0;
    } else if (c == ',') {
        addLexema(",", linea, "COMA", ",", "");
        Analisis(",");
        estado = 0;
    } else if (c == '>' || c == '<') {
        lexema = lexema + c;
        estado = 6;
    } else if (c == '"') {
        lexema = lexema + c;
        estado = 5;
    } else if (c == '\\t' || c == '\\b') {
        estado = 0;
    } else {
        contError++;
        //listaErrores.add(new Error(contError, "Caracter raro: " + c, linea,
"Lexicos"));

        lexema = "";
        estado = 404;
    }
    break;

```

```

case 1: //numero
    if (c >= '0' && c <= '9') {
        lexema = lexema + c;
        estado = 1;
    } else if (c == '.') {
        lexema = lexema + c;
        estado = 4;
    } else {
        addLexema(lexema, linea, "NÚMERO", "num", "");
        dato = "int";
        Analisis("num");
        lexema = "";
        estado = 0;
        i--;
    }
    break;
case 2: //variables
    if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z')) {
        lexema = lexema + c;
        estado = 2;
    } else if (c >= '0' && c <= '9') {
        lexema = lexema + c;
        estado = 2;
    } else {
        if(reser.isReservado(lexema))
        {
            addLexema(lexema, linea, "PALABRA RESERVADA", lexema, "");
            Analisis(lexema);
            datogen = true;
            if(lexema.equals("int"))
            {
                dato = "int";
            }
            else if(lexema.equals("float"))
            {
                dato = "float";
            }
            else if(lexema.equals("char"))
            {
                dato = "char";
            }
        }
        else
        {
            for (Lexema item : listaLexemas) {
                if(item.getLexema().equals(lexema))
                {
                    dato = item.getTD();
                    existe = true;
                    break;
                }
            }
            lexemaSem = lexema;
            addLexema(lexema, linea, "VARIABLE", "id", dato);
            Analisis("id");
        }
    }
}

```

```

        existe = false;
        if(datogen)
        {
            cadenaGen = dato + " " +cadenaGen + lexema+"";
            listaGen.add(cadenaGen);
            cadenaGen = "";
        }
    }

    lexema = "";
    estado = 0;
    i--;
}
break;
case 4://numero decimal
    if (c >= '0' && c <= '9') {
        lexema = lexema + c;
        estado = 4;
    } else {
        addLexema(lexema, linea, "float", "float", "");
        dato = "float";
        lexema = "";
        estado = 0;
        i--;
    }
    break;
case 5: // cadena
    if (c == '"') {
        lexema = lexema + c;
        addLexema(lexema, linea, "String", "string", "");
        Analisis("litcad");
        lexema = "";
        estado = 0;
    } else if (c == '\n') {
        contError++;
        // listaErrores.add(new Error(contError, "Cadena no valida: " +
lexema, linea, "Lexicos"));
        lexema = "";
        estado = 404;
        i--;
    } else {
        lexema = lexema + c;
        estado = 5;
    }
    break;
case 6: // <= >=
    if (c == '=' || c == '>') {
        lexema = lexema + c;
        addLexema(lexema, linea, "OPERADOR RELACIONAL", ">", "");
        Analisis(">=");
    } else {
        addLexema(lexema, linea, "OPERADOR RELACIONAL", "<", "");
        Analisis("<=");
        i--;
    }
    lexema = "";

```

```

        estado = 0;
        break;
    case 404: //error
        if (c == ' ' || c == ';' || c == '/' || c == '\t' || c == '\n') {
            estado = 0;
            lexema = "";
            i--;
        } else {
            estado = 404;
        }
        break;
    }
}
Analisis("$");

}

public boolean BuscarDatoDeclarado(String lexema)
{
    for (Lexema item : listaLexemas) {
        if(item.getLexema().equals(lexema))
        {
            datofinal = item.getTD();
            if(!datofinal.equals(""))
            {
                return true;
            }
        }
    }
    return false;
}

public void MostrarGeneracion()
{
    System.out.println(texto);
    System.out.println("Generación de Código intermedio: ");
    for(int i=0; i<listaGen.size(); i++)
        System.out.println(listaGen.get(i));
}

/* metodo que agrega el lexema a la lista */
public void addLexema(String cadena, int linea, String tipo, String comp, String td) {
    listaLexemas.add(new Lexema(contLexema, cadena, linea, tipo, comp, td));
    contLexema++;
}

public static void main(String[] args) {
    Semantico x= new Semantico("int a, c; \r\n"
        + "float b, d;\r\n"
        + "float var1;\r\n"
        + "var1=(a+b)-c/d\r\n");
    x.analizar();
    // x.MostrarGeneracion();
    // x.MostrarErrores();
}
}

```

