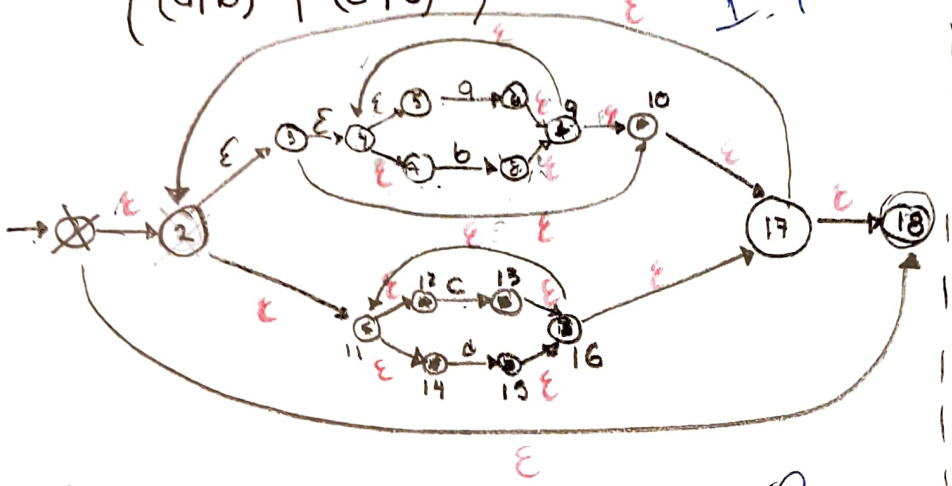
ASIGNATURA: Lenguajes Formales
ALUMNO: Murrieta Villegas Alfonso

GRUPO: 05 Ejercicio_núm.: 6
FECHA: 14/02/2019

▶ Exp.R $((a|b)^*|(c|d)^+)^*$ 1.9

1



Descripción

$Q = \{1,2,3,4,5,6,\ldots,18\}$

$\Sigma = \{a,b,c,d\}$

$q_0 = \{1\}$

$F = \{18\}$

$\delta =$

Transiciones 2.0

$\delta = \{$

a	b	c	d
1 a→ 6	1 b→ 8	1 c→ 13	1 d→ 15
5 a→ 6	7 b→ 8	12 c→ 13	14 d→ 15
6 a→ 6	8 b→ 8	13 c→ 13	15 d→ 15
* 8 a→ 6	* 6 b→ 8	15 c→ 13	13 d→ 15
13 a→ 6	13 b→ 8	8 c→ 13	8 d→ 15
15 a→ 6	15 b→ 8	6 c→ 13	6 d→ 15

$\}$

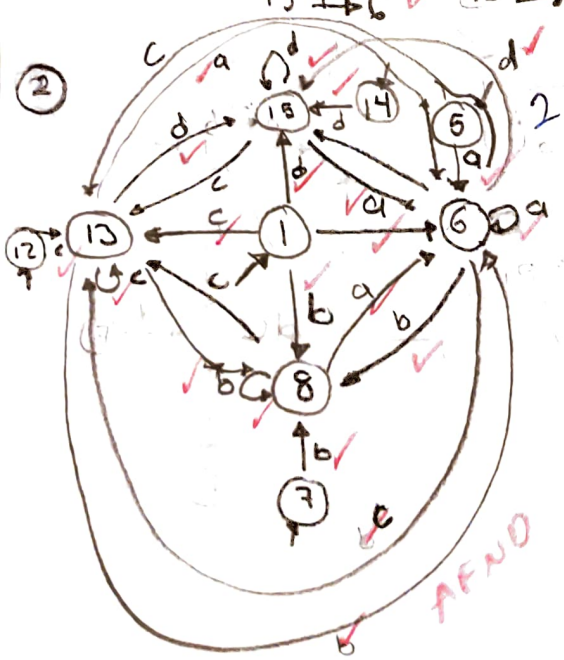
* Aceptación X

▶ $q_0 = \{1,5,7,12,14\}$

* Iniciales

▶ $F = \{6,8,13,15,1\}$

2

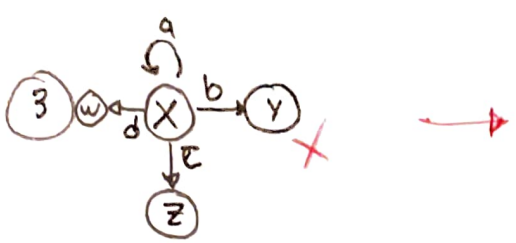


2.0

δ	a	b	c	d	1/0
X={1,5,7,12,14}	{6}	{8}	{13}	{15}	1
X={6}	{6}	{8}	{13}	{15}	1
X={8}	{6}	{8}	{13}	{15}	1
X={13}	{6}	{8}	{13}	{15}	1
X={15}	{6}	{8}	{13}	{15}	1

0.1

¿?



LENGUAJES FORMALES Y AUTÓMATAS

9

Nombre: Murrieta Villegas Alfonso Fecha: 21/02/2019

1] Defina los siguientes conceptos:

Símbolo, Lenguaje (1p), *Autómata (Estructura Matemática)* (1p), *Características del AFD* (1p)

2] Dibuje el diagrama de las operaciones que actúan sobre las expresiones regulares: (1p)

3] Desarrolle el AFND utilizando las reglas de Thompson de la siguiente E.R. (2p), posteriormente obtenga las transiciones para obtener el autómata sin transiciones vacías(2p), por último, por medio de las tablas de transición obtenga el AFD. (2p)

NOTA: No olvide escribir los elementos que conforman a el primer y último autómata.

$((ad)^*|(be)^*)^+$

1] • Símbolo: Es representación abstracta de una idea.

• Lenguaje: Es el conjunto de símbolos (gráfica) los cuales cumplen con ciertas reglas gramaticales (para la representación de cadenas y oraciones)

• Autómata: - Es un grafo usado con el fin de la representación de una expresión regular

$A = (\Sigma, Q, q_0, \delta, F)$ - Su estructura matemática son:

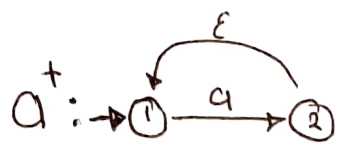
- Σ = Alfabeto
- q_0 = Inicial Símbolo
- F = símbolos finales
- Q = No terminales Símbolos
- δ = Transiciones
- símbolos Terminales

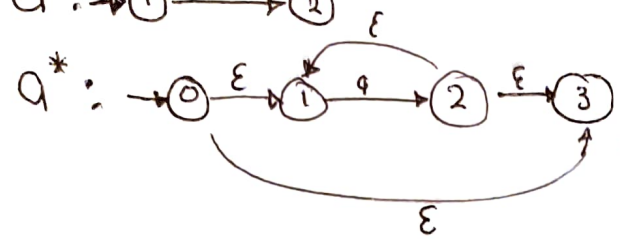
• Características de AFD

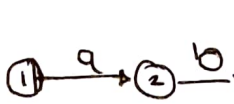
- Es programable debido a que:
  - Es deseable que no tenga estados equivalentes
  - No tiene ambigüedades
  - No tiene símbolos muertos
  - No tiene ciclos infinitos
  - No tiene inaccesibles
  - Un solo estado inicial
  - No tiene transiciones vacías
  - Puede tener muchos estados finales

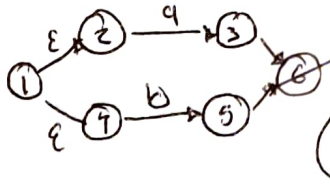
1

2)

a^+: 

a^*: 

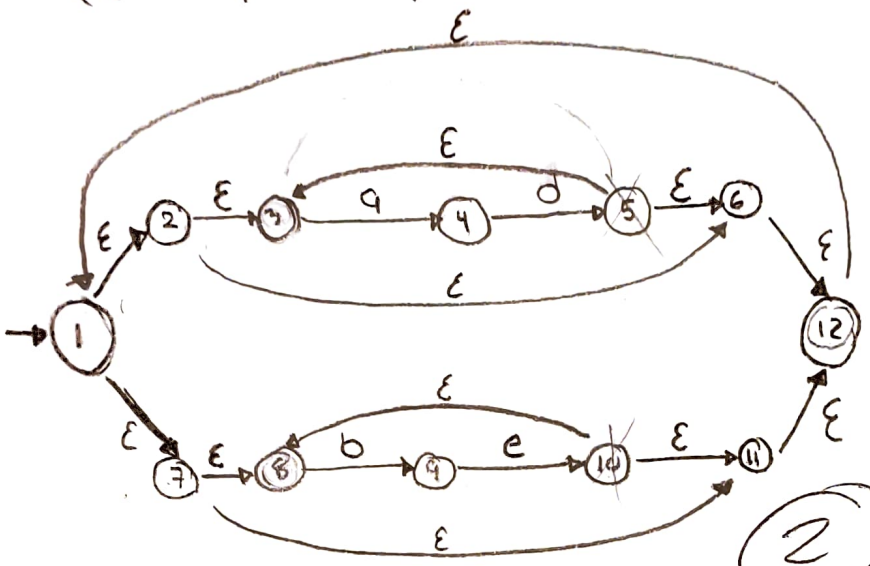
ab: 

a|b: 

1

3] E.R: $((ad)^* \mid (be)^*)^+$

• AFND



2

$\Sigma = \{a, b, d, e\}$

$= \{1, 2, 3, 4, 5, \ldots, 12\}$

$F_n = \{12\}$

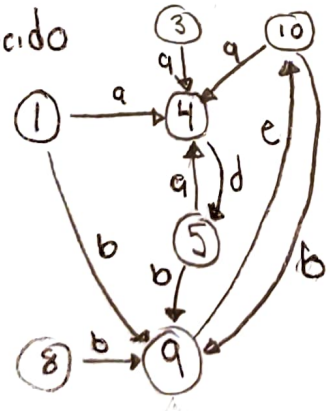
$q(0) = \{1\}$

$\delta = \{ \_\_ \}$

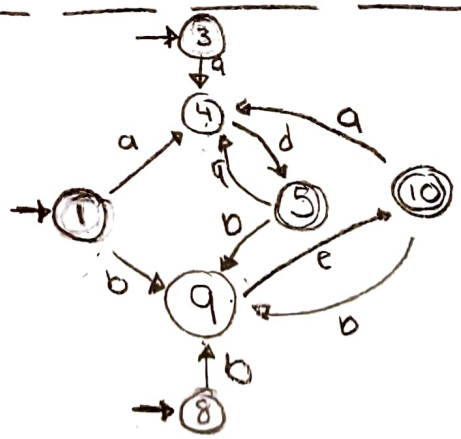
• Transiciones

$\delta\{$ $1 \xrightarrow{a} 4$, $3 \xrightarrow{a} 4$, $5 \xrightarrow{a} 4$, $*10 \xrightarrow{a} 4$; $4 \xrightarrow{d} 5$; $1 \xrightarrow{b} 9$, $8 \xrightarrow{b} 9$, $10 \xrightarrow{b} 9$, $*5 \xrightarrow{b} 9$; $9 \xrightarrow{e} 10$ $\}$

→ Reducido



;



2

continua en 3

• Iniciales

$= \{1, 3, 8\}$

• Finales

$= \{5, 10\}$

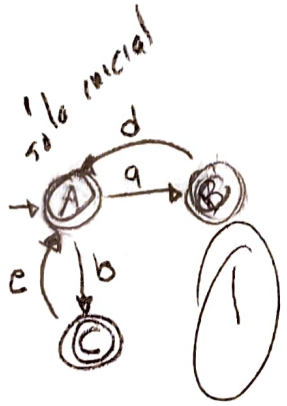
- Hurrieta Villegas Alfonso
- Lenguajes Formales y Autómatas

δ	a	b	d	e	0/1
A={1,3,8}	{4}D	{9}C	—	—	∅
B={4}	—	—	{5}A	—	∅
C={9}	—	—	—	{10}A	∅
~~A={5}~~	~~{4}~~	~~{9}~~	—	—	~~1~~
~~A={10}~~	~~{4}~~	~~{9}~~	—	—	~~1~~

TABLA DE TRANSICIONES

	a	b	d	e	
A	B	C	—	—	∅
B	—	—	A	—	∅
C	—	—	—	—	∅

- AFD



C, B no Finales.

- Características
  - = {A, B, C}
  - = {a, b, d, e}
  - q_0 = {A}
  - Finale = {B, C}

$\delta = \{ A \xrightarrow{q} B \quad A \xrightarrow{d} A \quad B \xrightarrow{b} A \quad A \xrightarrow{b} C \quad C \xrightarrow{e} A \}$

// Puede usarse la tabla

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Proyecto
Analizador Léxico con FLEX

SEMESTRE 2019-2

LENGUAJES FORMALES Y AUTÓMATAS

P R E S E N T A

Murrieta Villegas Alfonso

Cárdenas Cárdenas Jorge

Ciudad Universitaria, Cd. Mx., 21 de mayo de 2019

BNF DEL LENGUAJE C

1. declarationList→declarationList declaration|declaration
2. declaration→varDeclaration|funDeclaration
3. varDeclaration→typeSpecifier varDeclList;
4. scopedVarDeclaration→scopedTypeSpecifier varDeclList;
5. varDeclList→varDeclList, simpleExpression | simpleExpression
6. typeSpecifier→ int|bool|char
7. funDeclaration→ typeSpecifier (params)statement|(params)statement
8. params → params;paramTypeList|paramTypeList
9. paramTypeList→typeSpecifier
10. statement→expressionStmt|compoundStmt|selectionStmt|iterationStmt|returnStmt|breakStmt
11. compoundStmt→localDeclarations statementList
12. localDeclarations→localDeclarations scopedVarDeclaration
13. statementList→statementList statement
14. selectionStmt→if(simpleExpression)statement|if(simpleExpression)statementelsestatement
15. iterationStmt→while(simpleExpression)statement
16. returnStmt→return;
17. breakStmt→break;
18. expression → letters simpleExpression | letters | simpleExpression
19. simpleExpression→ sumExpression relop sumExpression|sumExpression | expression
20. relop→<=|<|>|>=|==|! =
21. sumExpression→sumExpression sumop term|term
22. sumop→+|−
23. term→ *|/|%
24. letters→a.b,c,…,z

Elementos del lenguaje:

NT: {declarationList, declaration, varDeclaration, funDeclaration, varDeclList, typeSpecifier, simpleExpression, params, paramTypeList, statement, compoundStmt, localDeclarations, statementList, selectionStmt, iterationStmt, returnStmt, breakStmt, expression, simpleExpression, relop, sumExpression, sumop, term, letters}

T: { a.b,c,…,z, +, −, *, /, %, <=, <, >, >=, == ,! =, return, break, while, if }

S: {declarationList }

F: {letters, term, sumop, relop, returnStmt, breakStmt, typeSpecifier }

BNF DEL LENGUAJE C

δ: {

1. declarationList→declarationList declaration|declaration
2. declaration→varDeclaration|funDeclaration
3. varDeclaration→typeSpecifier varDeclList;
4. varDeclList→varDeclList, simpleExpression | simpleExpression
5. typeSpecifier→ int|bool|char
6. funDeclaration→ typeSpecifier (params)statement|(params)statement
7. params → params;paramTypeList|paramTypeList
8. paramTypeList→typeSpecifier
9. statement→ selectionStmt|iterationStmt|returnStmt|breakStmt
10. statementList→statementList statement
11. selectionStmt→if(simpleExpression)statement|if(simpleExpression)statementelsestate ment
12. iterationStmt→while(simpleExpression)statement
13. returnStmt→return;
14. breakStmt→break;
15. expression → letters simpleExpression | letters | simpleExpression
16. simpleExpression→ sumExpression relop sumExpression|sumExpression | expression
17. relop→<=|<|>|>=|==|! =
18. sumExpression→sumExpression sumop term|term
19. sumop→+|−
20. term→ ∗|/|%
21. letters→a.b,c,…,z

}

Elementos del lenguaje:

NT: {declarationList, declaration, varDeclaration, funDeclaration, varDeclList, typeSpecifier, simpleExpression, params, paramTypeList, statement, compoundStmt, localDeclarations, statementList, selectionStmt, iterationStmt, returnStmt, breakStmt, expression, simpleExpression, relop, sumExpression, sumop, term, letters}

T: { a.b,c,…,z, +, −, ∗, /, %, <=, <, >, >=, == ,! =, return, break, while, if, int, bool, char }

S: {declarationList }

F: {letters, term, sumop, relop, returnStmt, breakStmt, typeSpecifier }

Significado de cada Terminal y no Terminal

→ Terminales

a,b,c,...,z : Elementos que sirven para generar patrones asociados a los identificadores

+, -, *, /, % : Elementos utilizados para realizar operaciones aritméticas

<=, <, >, =>, !=, == : Elementos utilizados para realizar operaciones de comparación e incluso de lógica.

return, break, while, if : Palabras reservadas del lenguaje asociados al regreso de valores, finalización de ciclos, comienzo de iteraciones y por último palabra reservada para condiciones.

→ No terminales

- declarationlist : Elemento "raíz" del cual parten todos los elementos o ramas del BNF.
- var declaration : Es la parte encargada de la definición de la declaración de las variables.
- fundeclaration : Es la parte encargada de la declaración de las funciones.
- typeSpecifier : Es la parte encargada de mencionar el especificador ya sea de la variable o función es el caso de int, bool, char, etc.

simpleExpression: Es la parte encargada de generar los patrones asociados a los nombres dados a las variables o funciones al momento de programar

params: Al igual que simpleExpression es generada mediante algún patrón y es la encargada de definir los parametros de funciones

paramTypeList: Es el encargado de llamar a params y typeSpecifier realmente es el que junta a ambos elementos

localDeclarations: Pobablemente sea el elemento encargado de definir elementos dentro de una función dada.

iterationStmt: Es el apartado encargado de definir los ciclos en el lenguaje.

selectionStmt: Es el apartado encargado de condicionar el código como es el caso de las palabras switch, if, else.

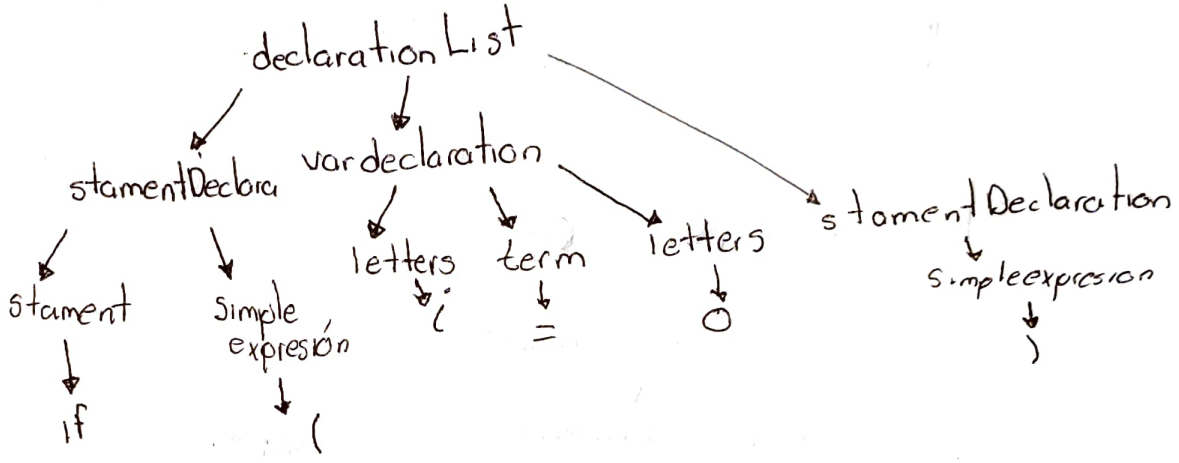
relop, sumop, term, letters, sumExpression: Son los elementos de generar todos los caracteres dados por el teclado qwerty, en este caso tanto sumop, relop y term se encargan de las operaciones de todo tipo.
Por otro lado, letters son los encargadas para le generación de patrones.

Char, Int, Bool: Son las variables o identificadores del tipo de variables

Árbol de derivación

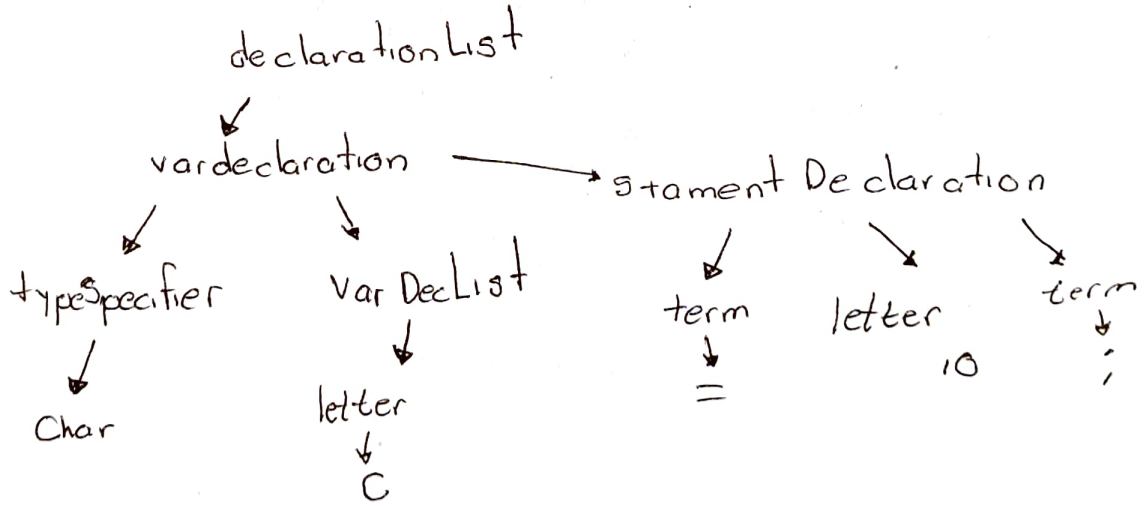
1] Cadena

→ if(i=0)



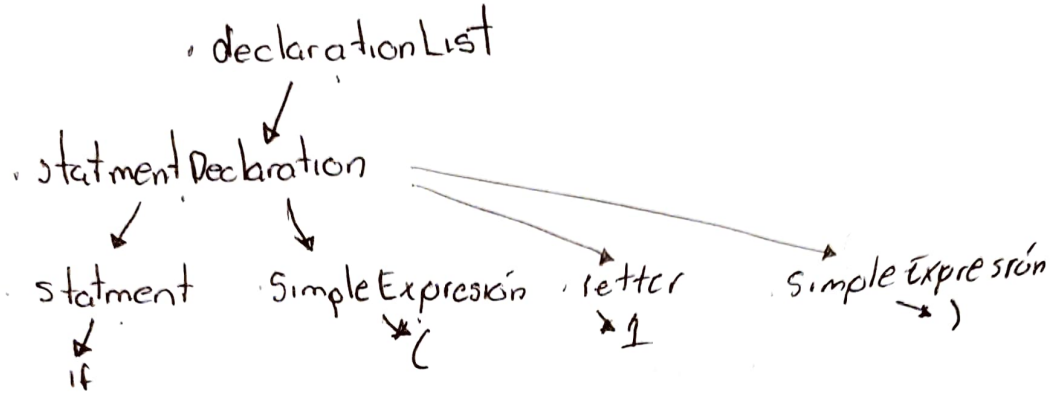
2] Cadena:

char c = 10;

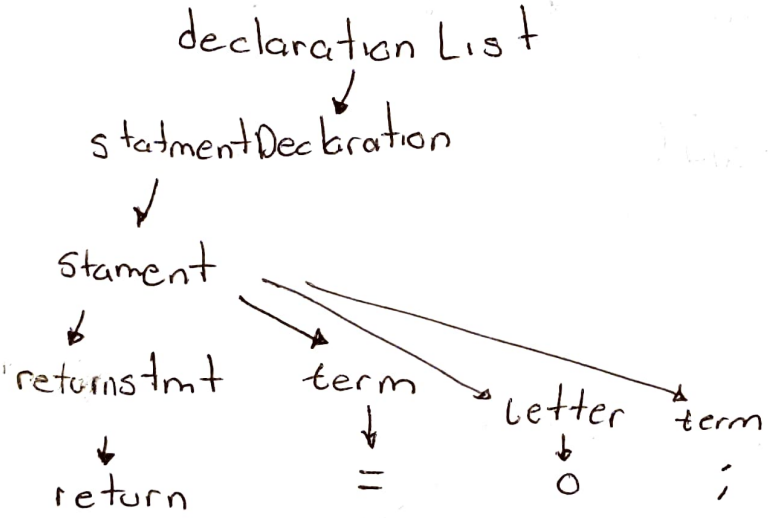


3] Cadera |

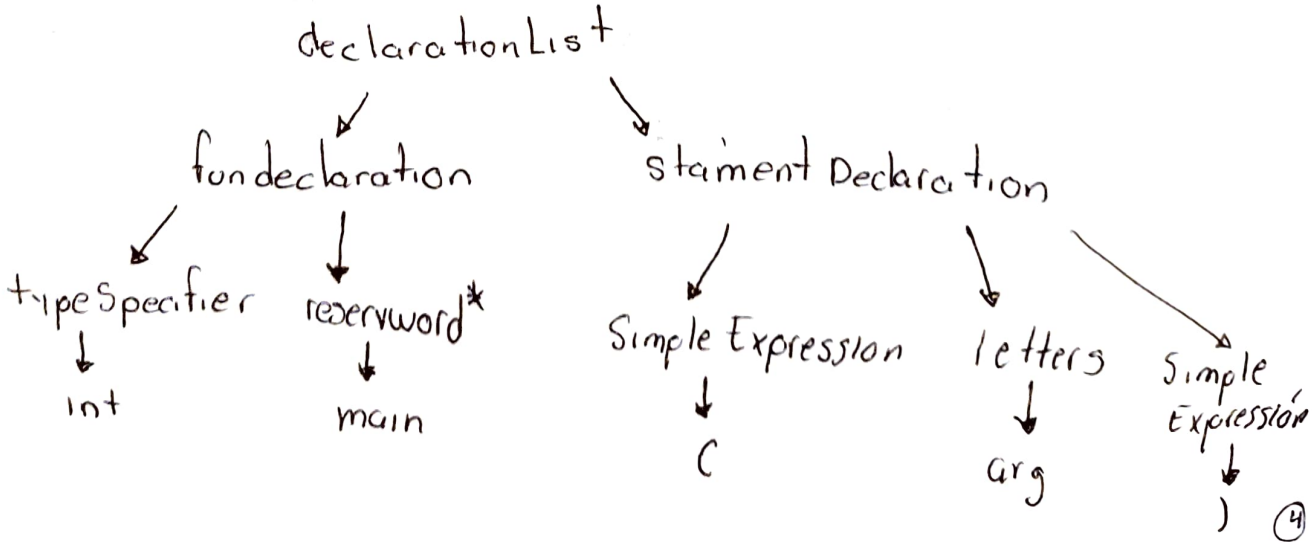
if (1)



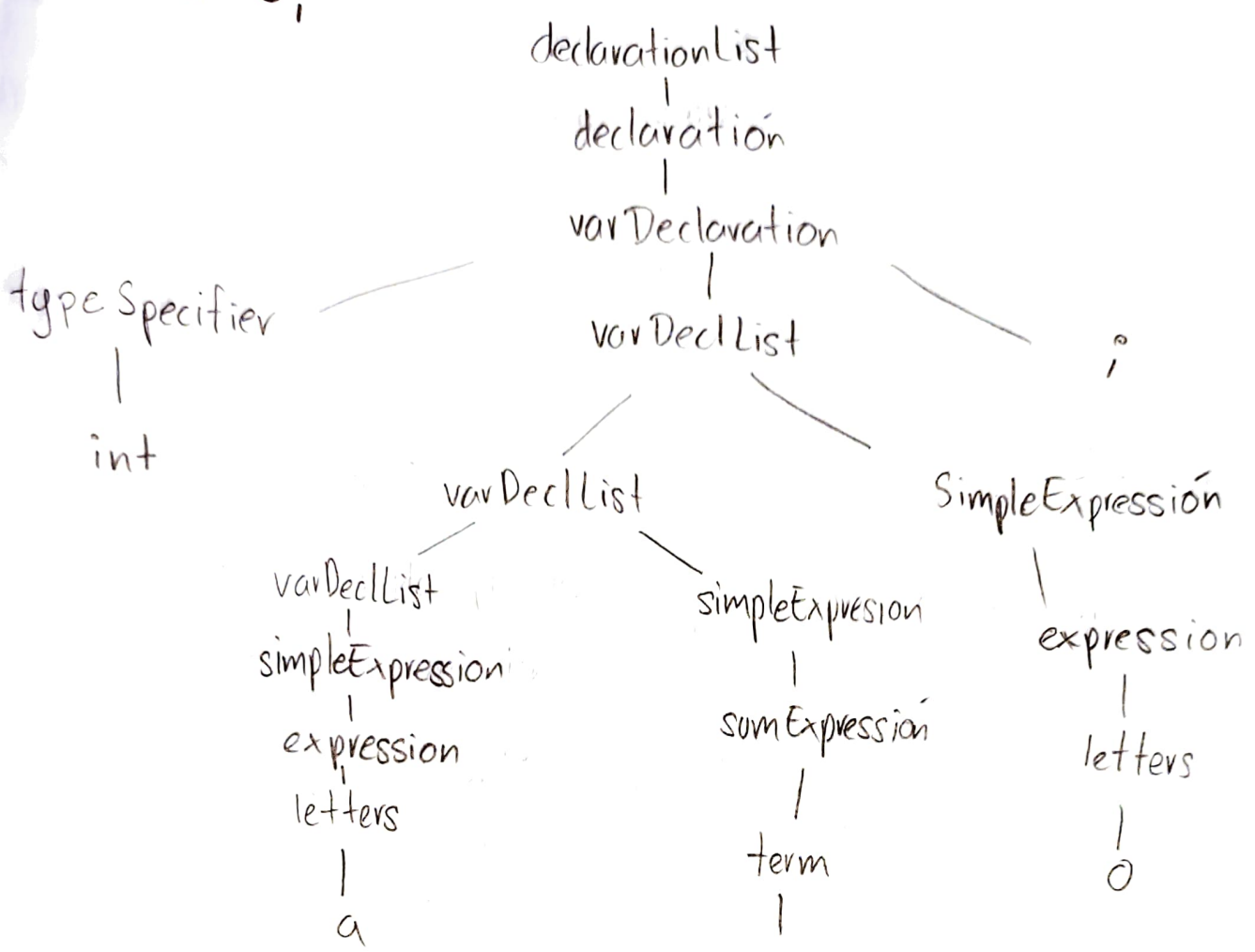
4] Cadena | return = 0;



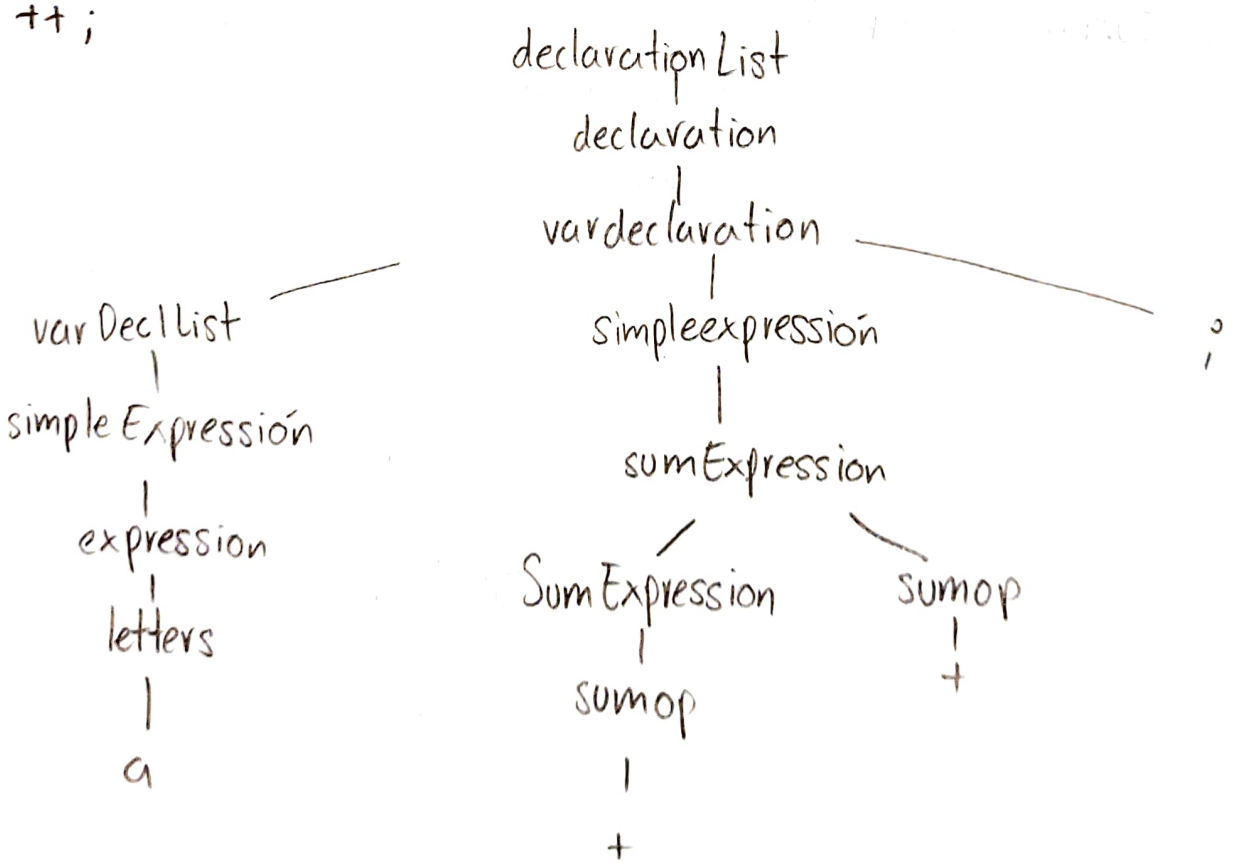
5] Cadena | int main (arg)



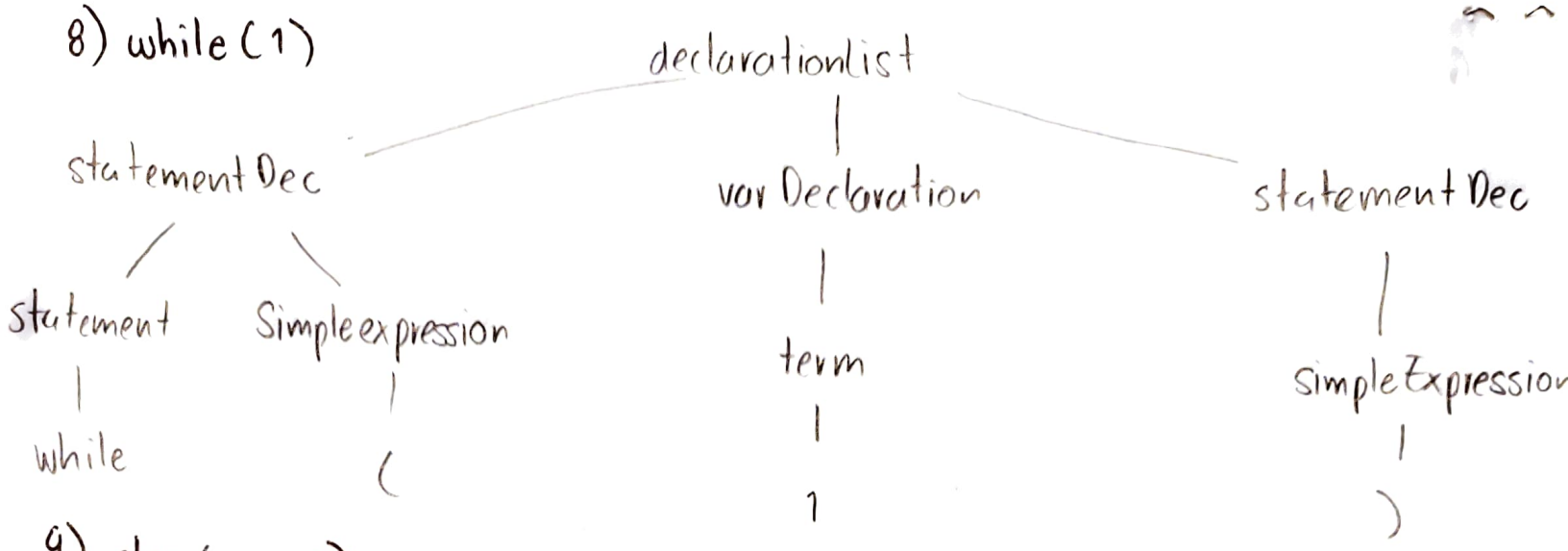
6) int a = 0;


declarationList
declaration
varDeclaration
typeSpecifier
int
varDeclList
;
varDeclList
SimpleExpression
varDeclList
simpleExpresion
expression
letters
a
simpleExpresion
sumExpression
term
=
expression
letters
0

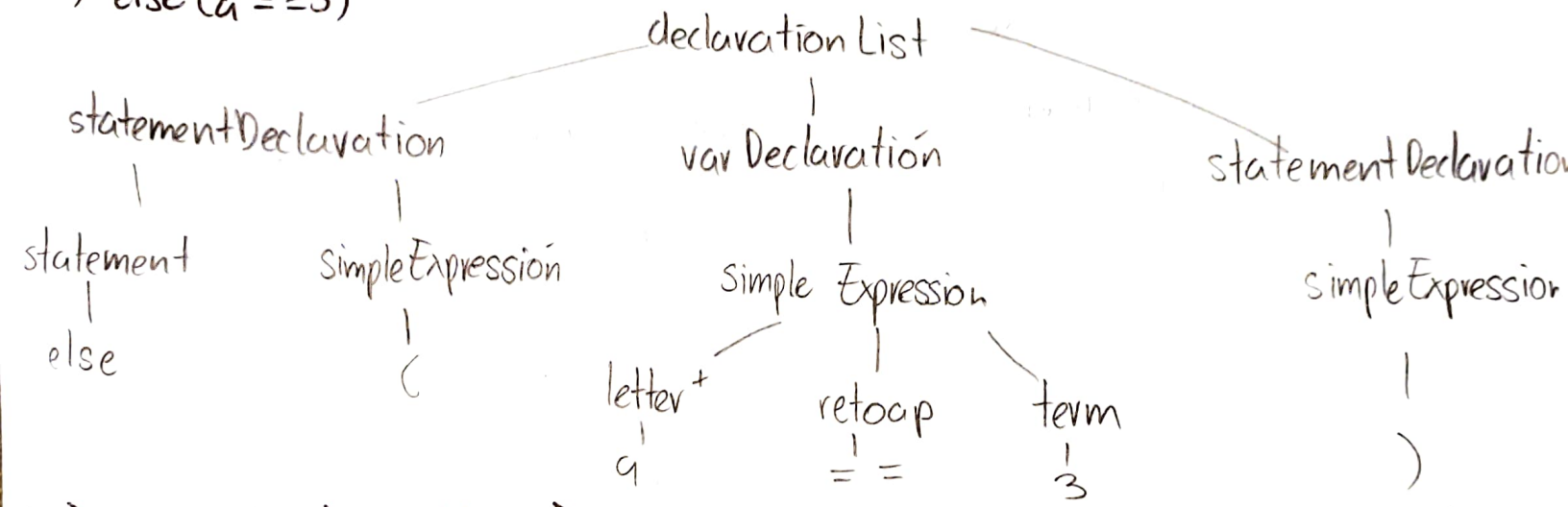
7) a ++;


declarationList
declaration
vardeclaration
varDeclList
simpleExpression
expression
letters
a
simpleexpression
sumExpression
;
SumExpression
sumop
+
sumop
+

8) while (1)



9) else (a==3)



10) char Function (char c)