PROCESADORES DEL LENGUAJE

ENTREGA 3

Francisco Jesús Beltrán Moreno

Índice

Descripción de la gramática ampliada

	Reglas
R0	X ::= CompilationUnit
R1	CompilationUnit ::= ImportClauseList LibraryDecl
R2	ImportClauseList ::= ImportClauseList ImportClause
R3	ImportClauseList ::= lambda
R4	ImportClause ::= **import identifier semicolon**
R5	LibraryDecl ::= **library identifier lbrace** FunctionList **rbrace**
R6	FunctionList ::= FunctionList FunctionDecl
R7	FunctionList ::= lambda
R8	FunctionDecl ::= Access FunctionType **identifier** ArgumentDecl FunctionBody
R9	Access ::= **public**
R10	Access ::= **private**
R11	FunctionType ::= Type
R12	FunctionType ::= **void**
R13	Type ::= **int**
R14	Type ::= **char**
R15	Type ::= **boolean**
R16	ArgumentDecl ::= **lparen rparen**
R17	ArgumentDecl ::= **lparen** ArgumentList **rparen**
R18	ArgumentList ::= Argument
R19	ArgumentList ::= ArgumentList **comma** Argument
R20	Argument ::= Type **identifier**
R21	FunctionBody ::= **lbrace** StatementList **rbrace**
R22	StatementList ::= StatementList Statement
R23	StatementList ::= lambda
R24	Statement ::= Decl **semicolon**
R25	Statement ::= IdStm **semicolon**
R26	Statement ::= IfStm

	Reglas
R27	Statement ::= WhileStm
R28	Statement ::= ReturnStm
R29	Statement ::= NoStm
R30	Statement ::= BlockStm
R31	Decl ::= Type IdList
R32	IdList ::= **identifier**
R33	IdList ::= **identifier** **assign** Expr
R34	IdList ::= IdList **comma** **identifier**
R35	IdList ::= IdList **comma** **identifier** **assign** Expr
R36	IfStm ::= **if** **lparen** Expr **rparen** Statement
R37	IfStm ::= **if** **lparen** Expr **rparen** Statement **else** Statement
R38	WhileStm ::= **while** **lparen** Expr **rparen** Statement
R39	ReturnStm ::= **return** Expr **semicolon**
R40	ReturnStm ::= **return** **semicolon**
R41	NoStm ::= **semicolon**
R42	IdStm ::= **identifier** **assign** Expr
R43	IdStm ::= **identifier** FunctionCall
R44	IdStm ::= **identifier** **dot** **identifier** FunctionCall
R45	BlockStm ::= **lbrace** StatementList **rbrace**
R46	Expr ::= AndExpr
R47	Expr ::= Expr **or** AndExpr
R48	AndExpr ::= RelExpr
R49	AndExpr ::= AndExpr **and** RelExpr
R50	RelExpr ::= SumExpr
R51	RelExpr ::= SumExpr **eq** SumExpr
R52	RelExpr ::= SumExpr **ne** SumExpr
R53	RelExpr ::= SumExpr **gt** SumExpr
R54	RelExpr ::= SumExpr **ge** SumExpr
R55	RelExpr ::= SumExpr **lt** SumExpr

	Reglas
R56	RelExpr ::= SumExpr **le** SumExpr
R57	SumExpr ::= **not** ProdExpr
R58	SumExpr ::= **minus** ProdExpr
R59	SumExpr ::= **plus** ProdExpr
R60	SumExpr ::= ProdExpr
R61	SumExpr ::= SumExpr **minus** ProdExpr
R62	SumExpr ::= SumExpr **plus** ProdExpr
R63	ProdExpr ::= Factor
R64	ProdExpr ::= ProdExpr **prod** Factor
R65	ProdExpr ::= ProdExpr **div** Factor
R66	ProdExpr ::= ProdExpr **mod** Factor
R67	Factor ::= Literal
R68	Factor ::= Reference
R69	Factor ::= **lparen** Expr **rparen**
R70	Literal ::= **integer_literal**
R71	Literal ::= **char_literal**
R72	Literal ::= **true**
R73	Literal ::= **false**
R74	Reference ::= **identifier**
R75	Reference ::= **identifier** FunctionCall
R76	Reference ::= **identifier dot identifier** FunctionCall
R77	FunctionCall ::= **lparen rparen**
R78	FunctionCall ::= **lparen** ExprList **rparen**
R79	ExprList ::= Expr
R80	ExprList ::= ExprList **comma** Expr
R81	Statement ::= SwitchStm
R82	Statement ::= ForStm
R83	Statement ::= DoWhileStm
R84	Statement ::= ContinueStm

	Reglas
R85	Statement ::= BreakStm
R86	BreakStm ::= **break semicolon**
R87	ContinueStm ::= **continue semicolon**
R88	SwitchStm ::= **switch lparen** Expr **rparen lbrace** ClauseList **rbrace**
R89	ClauseList ::= lambda
R90	ClauseList ::= ClauseList CaseClause
R91	ClauseList ::= ClauseList DefaultClause
R92	CaseClause ::= **case integer_literal colon** StatementList
R93	DefaultClause ::= **default colon** StatementList
R94	DoWhileStm ::= **do** Statement **while lparen** Expr **rparen semicolon**
R95	ForStm ::= **for lparen** ForInit **semicolon** ForCond **semicolon** ForUpdate **rparen** Statement
R96	ForInit ::= Decl
R97	ForInit ::= IdStmList
R98	ForInit ::= lambda
R99	ForCond ::= Expr
R100	ForCond ::= lambda
R101	ForUpdate ::= IdStmList
R102	ForUpdate ::= lambda
R103	IdStmList ::= IdStm
R104	IdStmList ::= IdStmList **comma** IdStm

Cálculo del autómata reconocedor de prefijos viables

Estado	Reglas	Transiciones
0	X : * CompilationUnit CompilationUnit : * ImportClauseList LibraryDecl ImportClauseList : * ImportClauseList ImportClause ImportClauseList : *	CompilationUnit → 1 ImportClauseList → 2 R3
1	X : CompilationUnit *	R0
2	CompilationUnit : ImportClauseList * LibraryDecl ImportClauseList : ImportClauseList * ImportClause ImportClause : * **import identifier semicolon** LibraryDecl : * **library identifier lbrace** FunctionList **rbrace**	LibraryDecl → 3 ImportClause → 4 **import** → 5 **library** → 6
3	CompilationUnit : ImportClauseList LibraryDecl *	R1
4	ImportClauseList : ImportClauseList ImportClause *	R2
5	ImportClause : **import** * **identifier semicolon**	**identifier** → 7
6	LibraryDecl : **library** * **identifier lbrace** FunctionList **rbrace**	**identifier** → 8
7	ImportClause : **import identifier** * **semicolon**	**semicolon** → 9
8	LibraryDecl : **library identifier** * **lbrace** FunctionList **rbrace**	**lbrace** → 10
9	ImportClause : **import identifier semicolon** *	R4
10	LibraryDecl : library **identifier lbrace** * FunctionList **rbrace** FunctionList : * FunctionList FunctionDecl FunctionList : *	FunctionList → 11 R7
11	LibraryDecl : **library identifier lbrace** FunctionList * **rbrace** FunctionList : FunctionList * FunctionDecl FunctionDecl : * Access FunctionType **identifier** ArgumentDecl FunctionBody Access : * **public** Access : * **private**	**rbrace** → 12 FunctionDecl → 13 Access → 14 **public** → 15 **private** → 16
12	LibraryDecl : **library identifier lbrace** FunctionList **rbrace** *	R5
13	FunctionList : FunctionList FunctionDecl *	R6

Estado	Reglas	Transiciones
14	FunctionDecl : Access * FunctionType **identifier** ArgumentDecl FunctionBody FunctionType : * Type FunctionType : * **void** Type : * **int** Type : * **char** Type : * **boolean**	FunctionType → 17 Type → 18 **void** → 19 **int**→ 20 **char** → 21 **boolean** → 22
15	Access : **public** *	R9
16	Access : **private** *	R10
17	FunctionDecl : Access FunctionType * **identifier** ArgumentDecl FunctionBody	**identifier** → 23
18	FunctionType : Type *	R11
19	FunctionType : **void** *	R12
20	Type : **int** *	R13
21	Type : **char** *	R14
22	Type : **boolean** *	R15
23	FunctionDecl : Access FunctionType **identifier** * ArgumentDecl FunctionBody ArgumentDecl : * **lparen rparen** ArgumentDecl : * **lparen** ArgumentList **rparen**	ArgumentDecl → 24 **lparen** → 25
24	FunctionDecl : Access FunctionType **identifier** ArgumentDecl * FunctionBody FunctionBody : * **lbrace** StatementList **rbrace**	FunctionBody → 26 **lbrace** → 27
25	ArgumentDecl : **lparen** * **rparen** ArgumentDecl : **lparen** * ArgumentList **rparen** ArgumentList : * Argument ArgumentList : * ArgumentList **comma** Argument Argument : * Type **identifier** Type : * **int** Type : * **char** Type : * **boolean**	**rparen** → 28 Argument → 30 ArgumentList → 29 Type → 31 **int**→ 20 **char** → 21 **boolean** → 22
26	FunctionDecl : Access FunctionType **identifier** ArgumentDecl FunctionBody *	R8

Estado	Reglas	Transiciones
27	FunctionBody : **lbrace** * StatementList **rbrace** StatementList : * StatementList Statement StatementList : *	StatementList → 32 R23
28	ArgumentDecl : **lparen** **rparen** *	R16
29	ArgumentDecl : **lparen** ArgumentList * **rparen** ArgumentList : ArgumentList * **comma** Argument	**rparen** → 33 **comma** → 34
30	ArgumentList : Argument *	R18
31	Argument : Type * **identifier**	**identifier** → 35
32	FunctionBody : **lbrace** StatementList * **rbrace** StatementList : StatementList * Statement Statement : * Decl **semicolon** Statement : * IdStm **semicolon** Statement : * IfStm Statement : * SwitchStm Statement : * DoWhileStm Statement : * ForStm Statement : * BreakStm Statement : * ContinueStm Statement : * WhileStm Statement : * ReturnStm Statement : * NoStm Statement : * BlockStm Decl : * Type IdList Type : * **int** Type : * **char** Type : * **boolean** IfStm : * **if** **lparen** Expr **rparen** Statement IfStm : * **if** **lparen** Expr **rparen** Statement **else** Statement SwitchStm : * **switch** **lparen** Expr **rparen** **lbrace** ClauseList **rbrace** DoWhileStm : * **do** Statement **while** **lparen** Expr **rparen** **semicolon** ForStm : * **for** **lparen** ForInit **semicolon** ForCond **semicolon** ForUpdate **rparen** Statement BreakStm : * **break** **semicolon** ContinueStm : * **continue** **semicolon** WhileStm : * **while** **lparen** Expr **rparen** Statement ReturnStm : * **return** Expr **semicolon** ReturnStm : * **return** **semicolon** NoStm : * **semicolon** IdStm : * **identifier** **assign** Expr IdStm : * **identifier** FunctionCall IdStm : * **identifier** **dot** **identifier** FunctionCall BlockStm : * **lbrace** StatementList **rbrace**	**rbrace** → 36 Statement → 37 Decl → 38 IdStm → 39 IfStm → 40 SwitchStm → 131 DoWhileStm → 139 ForStm → 144 BreakStm → 145 ContinueStm → 146 WhileStm → 41 ReturnStm → 42 NoStm → 43 BlockStm → 44 Type → 45 **int**→ 20 **char** → 21 **boolean** → 22 **if** → 46 **switch** → 147 **dowhile** → 148 **for** → 149 **break** → 150 **continue** → 151 **while** → 47 **return** → 48 **semicolon** → 49 **identifier** → 50 **lbrace** → 51
33	ArgumentDecl : **lparen** ArgumentList **rparen** *	R17
34	ArgumentList : ArgumentList **comma** * Argument Argument : * Type **identifier** Type : * **int** Type : * **char** Type : * **boolean**	Argument → 52 Type → 31 **int**→ 20 **char** → 21 **boolean** → 22
35	Argument : Type **identifier** *	R20

Estado	Reglas	Transiciones
36	FunctionBody : **lbrace** StatementList **rbrace** *	R21
37	StatementList : StatementList Statement *	R22
38	Statement : Decl * **semicolon**	**semicolon** → 80
39	Statement : IdStm * **semicolon**	**semicolon** → 106
40	Statement : IfStm *	R26
41	Statement : WhileStm *	R27
42	Statement : ReturnStm *	R28
43	Statement : NoStm *	R29
44	Statement : BlockStm *	R30
45	Decl : Type * IdList IdList : * **identifier** IdList : * **identifier assign** Expr IdList : * IdList **comma identifier** IdList : * IdList **comma identifier assign** Expr	IdList → 53 **identifier** → 54
46	IfStm : **if** * **lparen** Expr **rparen** Statement IfStm : **if** * **lparen** Expr **rparen** Statement **else** Statement	**lparen** → 55
47	WhileStm : **while** * **lparen** Expr **rparen** Statement	**lparen** → 56

Estado	Reglas	Transiciones
48	ReturnStm : **return** * Expr **semicolon** ReturnStm : **return** * **semicolon** Expr : * AndExpr Expr : * Expr **or** AndExpr AndExpr : * RelExpr AndExpr : * AndExpr **and** RelExpr RelExpr : * SumExpr RelExpr : * SumExpr **eq** SumExpr RelExpr : * SumExpr **ne** SumExpr RelExpr : * SumExpr **gt** SumExpr RelExpr : * SumExpr **ge** SumExpr RelExpr : * SumExpr **lt** SumExpr RelExpr : * SumExpr **le** SumExpr SumExpr : * **not** ProdExpr SumExpr : * **minus** ProdExpr SumExpr : * **plus** ProdExpr SumExpr : * ProdExpr SumExpr : * SumExpr **minus** ProdExpr SumExpr : * SumExpr **plus** ProdExpr ProdExpr : * Factor ProdExpr : * ProdExpr **prod** Factor ProdExpr : * ProdExpr **div** Factor ProdExpr : * ProdExpr **mod** Factor Factor : * Literal Factor : * Reference Factor : * **lparen** Expr **rparen** Literal : * **integer_literal** Literal : * **char_literal** Literal : * **true** Literal : * **false** Reference : * **identifier** Reference : * **identifier** FunctionCall Reference : * **identifier dot identifier** FunctionCall	Expr → 57 **semicolon** → 58 AndExpr → 59 RelExpr → 60 SumExpr → 61 **not** → 62 **minus** → 63 **plus** → 64 ProdExpr → 65 Factor → 66 Literal → 67 Reference → 68 **lparen** → 69 **integer_literal** →70 **char_literal** → 71 **true** → 72 **false** → 73 **identifier** → 74
49	NoStm : **semicolon** *	R41
50	IdStm : **identifier** * **assign** Expr IdStm : **identifier** * FunctionCall IdStm : **identifier** * **dot identifier** FunctionCall FunctionCall : * **lparen rparen** FunctionCall : * **lparen** ExprList **rparen**	**assign** → 75 FunctionCall → 76 **dot** → 77 **lparen** → 78

Estado	Reglas	Transiciones
51	BlockStm : **lbrace** * StatementList **rbrace** StatementList : * StatementList Statement StatementList : *	StatementList → 79 R23
52	ArgumentList : ArgumentList **comma** Argument *	R19
53	Decl : Type IdList * IdList : IdList * **comma identifier** IdList : IdList * **comma identifier assign** Expr	R31 **comma** → 81
54	IdList : **identifier** * IdList : **identifier** * **assign** Expr	R32 **assign** →82

Estado	Reglas	Transiciones
55	IfStm : **if** **lparen** * Expr **rparen** Statement IfStm : **if** **lparen** * Expr **rparen** Statement **else** Statement Expr : * AndExpr Expr : * Expr **or** AndExpr AndExpr : * RelExpr AndExpr : * AndExpr **and** RelExpr RelExpr : * SumExpr RelExpr : * SumExpr **eq** SumExpr RelExpr : * SumExpr **ne** SumExpr RelExpr : * SumExpr **gt** SumExpr RelExpr : * SumExpr **ge** SumExpr RelExpr : * SumExpr **lt** SumExpr RelExpr : * SumExpr **le** SumExpr SumExpr : * **not** ProdExpr SumExpr : * **minus** ProdExpr SumExpr : * **plus** ProdExpr SumExpr : * ProdExpr SumExpr : * SumExpr **minus** ProdExpr SumExpr : * SumExpr **plus** ProdExpr ProdExpr : * Factor ProdExpr : * ProdExpr **prod** Factor ProdExpr : * ProdExpr **div** Factor ProdExpr : * ProdExpr **mod** Factor Factor : * Literal Factor : * Reference Factor : * **lparen** Expr **rparen** Literal : * **integer_literal** Literal : * **char_literal** Literal : * **true** Literal : * **false** Reference : * **identifier** Reference : * **identifier** FunctionCall Reference : * **identifier dot identifier** FunctionCall	Expr → 83 AndExpr → 59 RelExpr → 60 SumExpr → 61 **not** → 62 **minus** → 63 **plus** → 64 ProdExpr → 65 Factor → 66 Literal → 67 Reference → 68 **lparen** → 69 **integer_literal** → 70 **char_literal** → 71 **true** →72 **false** → 73 **identifier** → 74

Estado	Reglas	Transiciones
56	WhileStm : **while** **lparen** * Expr **rparen** Statement Expr : * AndExpr Expr : * Expr **or** AndExpr AndExpr : * RelExpr AndExpr : * AndExpr **and** RelExpr RelExpr : * SumExpr RelExpr : * SumExpr **eq** SumExpr RelExpr : * SumExpr **ne** SumExpr RelExpr : * SumExpr **gt** SumExpr RelExpr : * SumExpr **ge** SumExpr RelExpr : * SumExpr **lt** SumExpr RelExpr : * SumExpr **le** SumExpr SumExpr : * **not** ProdExpr SumExpr : * **minus** ProdExpr SumExpr : * **plus** ProdExpr SumExpr : * ProdExpr SumExpr : * SumExpr **minus** ProdExpr SumExpr : * SumExpr **plus** ProdExpr ProdExpr : * Factor ProdExpr : * ProdExpr **prod** Factor ProdExpr : * ProdExpr **div** Factor ProdExpr : * ProdExpr **mod** Factor Factor : * Literal Factor : * Reference Factor : * **lparen** Expr **rparen** Literal : * **integer_literal** Literal : * **char_literal** Literal : * **true** Literal : * **false** Reference : * **identifier** Reference : * **identifier** FunctionCall Reference : * **identifier dot identifier** FunctionCall	Expr → 84 AndExpr → 59 RelExpr → 60 SumExpr → 61 **not** → 62 **minus** → 63 **plus** → 64 ProdExpr → 65 Factor → 66 Literal → 67 Reference → 68 **lparen** → 69 **integer_literal** →70 **char_literal** → 71 **true** → 72 **false** → 73 **identifier** → 74
57	ReturnStm : **return** Expr * **semicolon** Expr : Expr * **or** AndExpr	**semicolon** → 85 **or** → 86
58	ReturnStm : **return** **semicolon** *	R40
59	Expr : AndExpr * AndExpr : AndExpr * **and** RelExpr	R46 **and** →87
60	AndExpr : RelExpr *	R48

Estado	Reglas	Transiciones
61	RelExpr : SumExpr * RelExpr : SumExpr * **eq** SumExpr RelExpr : SumExpr * **ne** SumExpr RelExpr : SumExpr * **gt** SumExpr RelExpr : SumExpr * **ge** SumExpr RelExpr : SumExpr * **lt** SumExpr RelExpr : SumExpr * **le** SumExpr SumExpr : SumExpr * **minus** ProdExpr SumExpr : SumExpr * **plus** ProdExpr	R50 **eq** → 88 **ne** → 89 **gt** → 90 **ge** → 91 **lt** → 92 **le** → 93 **minus** → 94 **plus** → 95
62	SumExpr : **not** * ProdExpr ProdExpr : * Factor ProdExpr : * ProdExpr **prod** Factor ProdExpr : * ProdExpr **div** Factor ProdExpr : * ProdExpr **mod** Factor Factor : * Literal Factor : * Reference Factor : * **lparen** Expr **rparen** Literal : * **integer_literal** Literal : * **char_literal** Literal : * **true** Literal : * **false** Reference : * **identifier** Reference : * **identifier** FunctionCall Reference : * **identifier dot identifier** FunctionCall	ProdExpr → 96 Factor → 66 Literal → 67 Reference → 68 **lparen** → 69 **integer_literal** →70 **char_literal** → 71 **true** → 72 **false** → 73 **identifier** → 74
63	SumExpr : **minus** * ProdExpr ProdExpr : * Factor ProdExpr : * ProdExpr **prod** Factor ProdExpr : * ProdExpr **div** Factor ProdExpr : * ProdExpr **mod** Factor Factor : * Literal Factor : * Reference Factor : * **lparen** Expr **rparen** Literal : * **integer_literal** Literal : * **char_literal** Literal : * **true** Literal : * **false** Reference : * **identifier** Reference : * **identifier** FunctionCall Reference : * **identifier dot identifier** FunctionCall	ProdExpr → 97 Factor → 66 Literal → 67 Reference → 68 **lparen** → 69 **integer_literal** →70 **char_literal** → 71 **true** → 72 **false** → 73 **identifier** → 74

Estado	Reglas	Transiciones
64	SumExpr : **plus** * ProdExpr ProdExpr : * Factor ProdExpr : * ProdExpr **prod** Factor ProdExpr : * ProdExpr **div** Factor ProdExpr : * ProdExpr **mod** Factor Factor : * Literal Factor : * Reference Factor : * **lparen** Expr **rparen** Literal : * **integer_literal** Literal : * **char_literal** Literal : * **true** Literal : * **false** Reference : * **identifier** Reference : * **identifier** FunctionCall Reference : * **identifier dot identifier** FunctionCall	ProdExpr → 98 Factor → 66 Literal → 67 Reference → 68 **lparen** → 69 **integer_literal** →70 **char_literal** → 71 **true** → 72 **false** → 73 **identifier** → 74
65	SumExpr : ProdExpr * ProdExpr : ProdExpr * **prod** Factor ProdExpr : ProdExpr * **div** Factor ProdExpr : ProdExpr * **mod** Factor	R60 **prod** → 99 **div** → 100 **mod** → 101
66	ProdExpr : Factor *	R63
67	Factor : Literal *	R67
68	Factor : Reference *	R68

Estado	Reglas	Transiciones
69	Factor : **lparen** * Expr **rparen** Expr : * AndExpr Expr : * Expr **or** AndExpr AndExpr : * RelExpr AndExpr : * AndExpr **and** RelExpr RelExpr : * SumExpr RelExpr : * SumExpr **eq** SumExpr RelExpr : * SumExpr **ne** SumExpr RelExpr : * SumExpr **gt** SumExpr RelExpr : * SumExpr **ge** SumExpr RelExpr : * SumExpr **lt** SumExpr RelExpr : * SumExpr **le** SumExpr SumExpr : * **not** ProdExpr SumExpr : * **minus** ProdExpr SumExpr : * **plus** ProdExpr SumExpr : * ProdExpr SumExpr : * SumExpr **minus** ProdExpr SumExpr : * SumExpr **plus** ProdExpr ProdExpr : * Factor ProdExpr : * ProdExpr **prod** Factor ProdExpr : * ProdExpr **div** Factor ProdExpr : * ProdExpr **mod** Factor Factor : * Literal Factor : * Reference Factor : * **lparen** Expr **rparen** Literal : * **integer_literal** Literal : * **char_literal** Literal : * **true** Literal : * **false** Reference : * **identifier** Reference : * **identifier** FunctionCall Reference : * **identifier dot identifier** FunctionCall	Expr → 102 AndExpr → 59 RelExpr → 60 SumExpr → 61 **not** → 62 **minus** → 63 **plus** → 64 ProdExpr → 65 Factor → 66 Literal → 67 Reference → 68 **lparen** → 69 **integer_literal** →70 **char_literal** → 71 **true** → 72 **false** → 73 **identifier** → 74
70	Literal : **integer_literal** *	R70
71	Literal : **char_literal** *	R71
72	Literal : **true** *	R72
73	Literal : **false** *	R73

Estado	Reglas	Transiciones
74	Reference : **identifier** * Reference : **identifier** * FunctionCall Reference : **identifier** * **dot identifier** FunctionCall FunctionCall : * **lparen rparen** FunctionCall : * **lparen** ExprList **rparen**	R74 FunctionCall → 103 **dot** → 104 **lparen** → 78
75	IdStm : **identifier assign** * Expr Expr : * AndExpr Expr : * Expr **or** AndExpr AndExpr : * RelExpr AndExpr : * AndExpr **and** RelExpr RelExpr : * SumExpr RelExpr : * SumExpr **eq** SumExpr RelExpr : * SumExpr **ne** SumExpr RelExpr : * SumExpr **gt** SumExpr RelExpr : * SumExpr **ge** SumExpr RelExpr : * SumExpr **lt** SumExpr RelExpr : * SumExpr **le** SumExpr SumExpr : * **not** ProdExpr SumExpr : * **minus** ProdExpr SumExpr : * **plus** ProdExpr SumExpr : * ProdExpr SumExpr : * SumExpr **minus** ProdExpr SumExpr : * SumExpr **plus** ProdExpr ProdExpr : * Factor ProdExpr : * ProdExpr **prod** Factor ProdExpr : * ProdExpr **div** Factor ProdExpr : * ProdExpr **mod** Factor Factor : * Literal Factor : * Reference Factor : * **lparen** Expr **rparen** Literal : * **integer_literal** Literal : * **char_literal** Literal : * **true** Literal : * **false** Reference : * **identifier** Reference : * **identifier** FunctionCall Reference : * **identifier dot identifier** FunctionCall	Expr → 105 AndExpr → 59 RelExpr → 60 SumExpr → 61 **not** → 62 **minus** → 63 **plus** → 64 ProdExpr → 65 Factor → 66 Literal → 67 Reference → 68 **lparen** → 69 **integer_literal** →70 **char_literal** → 71 **true** → 72 **false** → 73 **identifier** → 74
76	IdStm : **identifier** FunctionCall *	R43
77	IdStm : **identifier dot** * **identifier** FunctionCall	**identifier** → 107

Estado	Reglas	Transiciones
78	FunctionCall : **lparen** * **rparen** FunctionCall : **lparen** * ExprList **rparen** ExprList : * Expr ExprList : * ExprList **comma** Expr Expr : * AndExpr Expr : * Expr **or** AndExpr AndExpr : * RelExpr AndExpr : * AndExpr **and** RelExpr RelExpr : * SumExpr RelExpr : * SumExpr **eq** SumExpr RelExpr : * SumExpr **ne** SumExpr RelExpr : * SumExpr **gt** SumExpr RelExpr : * SumExpr **ge** SumExpr RelExpr : * SumExpr **lt** SumExpr RelExpr : * SumExpr **le** SumExpr SumExpr : * **not** ProdExpr SumExpr : * **minus** ProdExpr SumExpr : * **plus** ProdExpr SumExpr : * ProdExpr SumExpr : * SumExpr **minus** ProdExpr SumExpr : * SumExpr **plus** ProdExpr ProdExpr : * Factor ProdExpr : * ProdExpr **prod** Factor ProdExpr : * ProdExpr **div** Factor ProdExpr : * ProdExpr **mod** Factor Factor : * Literal Factor : * Reference Factor : * **lparen** Expr **rparen** Literal : * **integer_literal** Literal : * **char_literal** Literal : * **true** Literal : * **false** Reference : * **identifier** Reference : * **identifier** FunctionCall Reference : * **identifier dot identifier** FunctionCall	**rparen** → 108 ExprList → 109 Expr → 110 AndExpr → 59 RelExpr → 60 SumExpr → 61 **not** → 62 **minus** → 63 **plus** → 64 ProdExpr → 65 Factor → 66 Literal → 67 Reference → 68 **lparen** → 69 **integer_literal** →70 **char_literal** → 71 **true** → 72 **false** → 73 **identifier** → 74

Estado	Reglas	Transiciones
79	BlockStm : **lbrace** StatementList * **rbrace** StatementList : StatementList * Statement Statement : * Decl **semicolon** Statement : * IdStm **semicolon** Statement : * IfStm Statement : * SwitchStm Statement : * DoWhileStm Statement : * ForStm Statement : * BreakStm Statement : * ContinueStm Statement : * WhileStm Statement : * ReturnStm Statement : * NoStm Statement : * BlockStm Decl : * Type IdList Type : * **int** Type : * **char** Type : * **boolean** IfStm : * **if** **lparen** Expr **rparen** Statement IfStm : * **if** **lparen** Expr **rparen** Statement **else** Statement SwitchStm : * **switch** **lparen** Expr **rparen** **lbrace** ClauseList **rbrace** DoWhileStm : * **do** Statement **while** **lparen** Expr **rparen** **semicolon** ForStm : * **for** **lparen** ForInit **semicolon** ForCond **semicolon** ForUpdate **rparen** Statement BreakStm : * **break** **semicolon** ContinueStm : * **continue** **semicolon** WhileStm : * **while** **lparen** Expr **rparen** Statement ReturnStm : * **return** Expr **semicolon** ReturnStm : * **return** **semicolon** NoStm : * **semicolon** IdStm : * **identifier** **assign** Expr IdStm : * **identifier** FunctionCall IdStm : * **identifier** **dot** **identifier** FunctionCall BlockStm : * **lbrace** StatementList **rbrace**	**rbrace** → 111 Statement → 37 Decl → 38 IdStm → 39 IfStm → 40 SwitchStm → 131 DoWhileStm → 139 ForStm → 144 BreakStm → 145 ContinueStm → 146 WhileStm → 41 ReturnStm → 42 NoStm → 43 BlockStm → 44 Type → 45 **int**→ 20 **char** → 21 **boolean** → 22 **if** → 46 **switch** → 147 **dowhile** → 148 **for** → 149 **break** → 150 **continue** → 151 **while** → 47 **return** → 48 **semicolon** → 49 **identifier** → 50 **lbrace** → 51
80	Statement : Decl **semicolon** * Decl : Type IdList **semicolon** *	R24 R31
81	IdList : IdList **comma** * **identifier** IdList : IdList **comma** * **identifier** **assign** Expr	**identifier** → 112

Estado	Reglas	Transiciones
82	IdList : **identifier** **assign** * Expr Expr : * AndExpr Expr : * Expr **or** AndExpr AndExpr : * RelExpr AndExpr : * AndExpr **and** RelExpr RelExpr : * SumExpr RelExpr : * SumExpr **eq** SumExpr RelExpr : * SumExpr **ne** SumExpr RelExpr : * SumExpr **gt** SumExpr RelExpr : * SumExpr **ge** SumExpr RelExpr : * SumExpr **lt** SumExpr RelExpr : * SumExpr **le** SumExpr SumExpr : * **not** ProdExpr SumExpr : * **minus** ProdExpr SumExpr : * **plus** ProdExpr SumExpr : * ProdExpr SumExpr : * SumExpr **minus** ProdExpr SumExpr : * SumExpr **plus** ProdExpr ProdExpr : * Factor ProdExpr : * ProdExpr **prod** Factor ProdExpr : * ProdExpr **div** Factor ProdExpr : * ProdExpr **mod** Factor Factor : * Literal Factor : * Reference Factor : * **lparen** Expr **rparen** Literal : * **integer_literal** Literal : * **char_literal** Literal : * **true** Literal : * **false** Reference : * **identifier** Reference : * **identifier** FunctionCall Reference : * **identifier** **dot** **identifier** FunctionCall	Expr → 113 AndExpr → 59 RelExpr → 60 SumExpr → 61 **not** → 62 **minus** → 63 **plus** → 64 ProdExpr → 65 Factor → 66 Literal → 67 Reference → 68 **lparen** → 69 **integer_literal** →70 **char_literal** → 71 **true** → 72 **false** → 73 **identifier** → 74
83	IfStm : **if** **lparen** Expr * **rparen** Statement IfStm : **if** **lparen** Expr * **rparen** Statement else Statement Expr : Expr * **or** AndExpr	**rparen** → 114 **or** → 86
84	WhileStm : **while** **lparen** Expr * **rparen** Statement Expr : Expr * **or** AndExpr	**rparen** → 115 **or** → 86
85	ReturnStm : **return** Expr **semicolon** *	R39

Estado	Reglas	Transiciones
86	Expr : Expr **or** * AndExpr AndExpr : * RelExpr AndExpr : * AndExpr **and** RelExpr RelExpr : * SumExpr RelExpr : * SumExpr **eq** SumExpr RelExpr : * SumExpr **ne** SumExpr RelExpr : * SumExpr **gt** SumExpr RelExpr : * SumExpr **ge** SumExpr RelExpr : * SumExpr **lt** SumExpr RelExpr : * SumExpr **le** SumExpr SumExpr : * **not** ProdExpr SumExpr : * **minus** ProdExpr SumExpr : * **plus** ProdExpr SumExpr : * ProdExpr SumExpr : * SumExpr **minus** ProdExpr SumExpr : * SumExpr **plus** ProdExpr ProdExpr : * Factor ProdExpr : * ProdExpr **prod** Factor ProdExpr : * ProdExpr **div** Factor ProdExpr : * ProdExpr **mod** Factor Factor : * Literal Factor : * Reference Factor : * **lparen** Expr **rparen** Literal : * **integer_literal** Literal : * **char_literal** Literal : * **true** Literal : * **false** Reference : * **identifier** Reference : * **identifier** FunctionCall Reference : * **identifier dot identifier** FunctionCall	AndExpr → 116 RelExpr → 60 SumExpr → 61 **not** → 62 **minus** → 63 **plus** → 64 ProdExpr → 65 Factor → 66 Literal → 67 Reference → 68 **lparen** → 69 **integer_literal** →70 **char_literal** → 71 **true** → 72 **false** → 73 **identifier** → 74

Estado	Reglas	Transiciones
87	AndExpr : AndExpr **and** * RelExpr RelExpr : * SumExpr RelExpr : * SumExpr **eq** SumExpr RelExpr : * SumExpr **ne** SumExpr RelExpr : * SumExpr **gt** SumExpr RelExpr : * SumExpr **ge** SumExpr RelExpr : * SumExpr **lt** SumExpr RelExpr : * SumExpr **le** SumExpr SumExpr : * **not** ProdExpr SumExpr : * **minus** ProdExpr SumExpr : * **plus** ProdExpr SumExpr : * ProdExpr SumExpr : * SumExpr **minus** ProdExpr SumExpr : * SumExpr **plus** ProdExpr ProdExpr : * Factor ProdExpr : * ProdExpr **prod** Factor ProdExpr : * ProdExpr **div** Factor ProdExpr : * ProdExpr **mod** Factor Factor : * Literal Factor : * Reference Factor : * **lparen** Expr **rparen** Literal : * **integer_literal** Literal : * **char_literal** Literal : * **true** Literal : * **false** Reference : * **identifier** Reference : * **identifier** FunctionCall Reference : * **identifier dot identifier** FunctionCall	RelExpr → 117 SumExpr → 61 **not** → 62 **minus** → 63 **plus** → 64 ProdExpr → 65 Factor → 66 Literal → 67 Reference → 68 **lparen** → 69 **integer_literal** →70 **char_literal** → 71 **true** → 72 **false** → 73 **identifier** → 74

Estado	Reglas	Transiciones
88	RelExpr : SumExpr **eq** * SumExpr SumExpr : * **not** ProdExpr SumExpr : * **minus** ProdExpr SumExpr : * **plus** ProdExpr SumExpr : * ProdExpr SumExpr : * SumExpr **minus** ProdExpr SumExpr : * SumExpr **plus** ProdExpr ProdExpr : * Factor ProdExpr : * ProdExpr **prod** Factor ProdExpr : * ProdExpr **div** Factor ProdExpr : * ProdExpr **mod** Factor Factor : * Literal Factor : * Reference Factor : * **lparen** Expr **rparen** Literal : * **integer_literal** Literal : * **char_literal** Literal : * **true** Literal : * **false** Reference : * **identifier** Reference : * **identifier** FunctionCall Reference : * **identifier dot identifier** FunctionCall	SumExpr → 118 **not** → 62 **minus** → 63 **plus** → 64 ProdExpr → 65 Factor → 66 Literal → 67 Reference → 68 **lparen** → 69 **integer_literal** → 70 **char_literal** → 71 **true** → 72 **false** → 73 **identifier** → 74
89	RelExpr : SumExpr **ne** * SumExpr SumExpr : * **not** ProdExpr SumExpr : * **minus** ProdExpr SumExpr : * **plus** ProdExpr SumExpr : * ProdExpr SumExpr : * SumExpr **minus** ProdExpr SumExpr : * SumExpr **plus** ProdExpr ProdExpr : * Factor ProdExpr : * ProdExpr **prod** Factor ProdExpr : * ProdExpr **div** Factor ProdExpr : * ProdExpr **mod** Factor Factor : * Literal Factor : * Reference Factor : * **lparen** Expr **rparen** Literal : * **integer_literal** Literal : * **char_literal** Literal : * **true** Literal : * **false** Reference : * **identifier** Reference : * **identifier** FunctionCall Reference : * **identifier dot identifier** FunctionCall	SumExpr → 119 **not** → 62 **minus** → 63 **plus** → 64 ProdExpr → 65 Factor → 66 Literal → 67 Reference → 68 **lparen** → 69 **integer_literal** →70 **char_literal** → 71 **true** → 72 **false** → 73 **identifier** → 74

Estado	Reglas	Transiciones
90	RelExpr : SumExpr **gt** * SumExpr SumExpr : * **not** ProdExpr SumExpr : * **minus** ProdExpr SumExpr : * **plus** ProdExpr SumExpr : * ProdExpr SumExpr : * SumExpr **minus** ProdExpr SumExpr : * SumExpr **plus** ProdExpr ProdExpr : * Factor ProdExpr : * ProdExpr **prod** Factor ProdExpr : * ProdExpr **div** Factor ProdExpr : * ProdExpr **mod** Factor Factor : * Literal Factor : * Reference Factor : * **lparen** Expr **rparen** Literal : * **integer_literal** Literal : * **char_literal** Literal : * **true** Literal : * **false** Reference : * **identifier** Reference : * **identifier** FunctionCall Reference : * **identifier dot identifier** FunctionCall	SumExpr → 120 **not** → 62 **minus** → 63 **plus** → 64 ProdExpr → 65 Factor → 66 Literal → 67 Reference → 68 **lparen** → 69 **integer_literal** → 70 **char_literal** → 71 **true** → 72 **false** → 73 **identifier** → 74
91	RelExpr : SumExpr **ge** * SumExpr SumExpr : * **not** ProdExpr SumExpr : * **minus** ProdExpr SumExpr : * **plus** ProdExpr SumExpr : * ProdExpr SumExpr : * SumExpr **minus** ProdExpr SumExpr : * SumExpr **plus** ProdExpr ProdExpr : * Factor ProdExpr : * ProdExpr **prod** Factor ProdExpr : * ProdExpr **div** Factor ProdExpr : * ProdExpr **mod** Factor Factor : * Literal Factor : * Reference Factor : * **lparen** Expr **rparen** Literal : * **integer_literal** Literal : * **char_literal** Literal : * **true** Literal : * **false** Reference : * **identifier** Reference : * **identifier** FunctionCall Reference : * **identifier dot identifier** FunctionCall	SumExpr → 121 **not** → 62 **minus** → 63 **plus** → 64 ProdExpr → 65 Factor → 66 Literal → 67 Reference → 68 **lparen** → 69 **integer_literal** → 70 **char_literal** → 71 **true** → 72 **false** → 73 **identifier** → 74

Estado	Reglas	Transiciones
92	RelExpr : SumExpr **lt** * SumExpr SumExpr : * **not** ProdExpr SumExpr : * **minus** ProdExpr SumExpr : * **plus** ProdExpr SumExpr : * ProdExpr SumExpr : * SumExpr **minus** ProdExpr SumExpr : * SumExpr **plus** ProdExpr ProdExpr : * Factor ProdExpr : * ProdExpr **prod** Factor ProdExpr : * ProdExpr **div** Factor ProdExpr : * ProdExpr **mod** Factor Factor : * Literal Factor : * Reference Factor : * **lparen** Expr **rparen** Literal : * **integer_literal** Literal : * **char_literal** Literal : * **true** Literal : * **false** Reference : * **identifier** Reference : * **identifier** FunctionCall Reference : * **identifier dot identifier** FunctionCall	SumExpr → 122 **not** → 62 **minus** → 63 **plus** → 64 ProdExpr → 65 Factor → 66 Literal → 67 Reference → 68 **lparen** → 69 **integer_literal** → 70 **char_literal** → 71 **true** → 72 **false** → 73 **identifier** → 74
93	RelExpr : SumExpr **le** * SumExpr SumExpr : * **not** ProdExpr SumExpr : * **minus** ProdExpr SumExpr : * **plus** ProdExpr SumExpr : * ProdExpr SumExpr : * SumExpr **minus** ProdExpr SumExpr : * SumExpr **plus** ProdExpr ProdExpr : * Factor ProdExpr : * ProdExpr **prod** Factor ProdExpr : * ProdExpr **div** Factor ProdExpr : * ProdExpr **mod** Factor Factor : * Literal Factor : * Reference Factor : * **lparen** Expr **rparen** Literal : * **integer_literal** Literal : * **char_literal** Literal : * **true** Literal : * **false** Reference : * **identifier** Reference : * **identifier** FunctionCall Reference : * **identifier dot identifier** FunctionCall	SumExpr → 123 **not** → 62 **minus** → 63 **plus** → 64 ProdExpr → 65 Factor → 66 Literal → 67 Reference → 68 **lparen** → 69 **integer_literal** →70 **char_literal** → 71 **true** → 72 **false** → 73 **identifier** → 74

Estado	Reglas	Transiciones
94	SumExpr : SumExpr **minus** * ProdExpr ProdExpr : * Factor ProdExpr : * ProdExpr **prod** Factor ProdExpr : * ProdExpr **div** Factor ProdExpr : * ProdExpr **mod** Factor Factor : * Literal Factor : * Reference Factor : * **lparen** Expr **rparen** Literal : * **integer_literal** Literal : * **char_literal** Literal : * **true** Literal : * **false** Reference : * **identifier** Reference : * **identifier** FunctionCall Reference : * **identifier dot identifier** FunctionCall	ProdExpr → 124 Factor → 66 Literal → 67 Reference → 68 **lparen** → 69 **integer_literal** → 70 **char_literal** → 71 **true** → 72 **false** → 73 **identifier** → 74
95	SumExpr : SumExpr **plus** * ProdExpr ProdExpr : * Factor ProdExpr : * ProdExpr **prod** Factor ProdExpr : * ProdExpr **div** Factor ProdExpr : * ProdExpr **mod** Factor Factor : * Literal Factor : * Reference Factor : * **lparen** Expr **rparen** Literal : * **integer_literal** Literal : * **char_literal** Literal : * **true** Literal : * **false** Reference : * **identifier** Reference : * **identifier** FunctionCall Reference : * **identifier dot identifier** FunctionCall	ProdExpr → 125 Factor → 66 Literal → 67 Reference → 68 **lparen** → 69 **integer_literal** → 70 **char_literal** → 71 **true** → 72 **false** → 73 **identifier** → 74
96	SumExpr : **not** ProdExpr * ProdExpr : ProdExpr * **prod** Factor ProdExpr : ProdExpr * **div** Factor ProdExpr : ProdExpr * **mod** Factor	R57 **prod** → 99 **div** → 100 **mod** → 101
97	SumExpr : **minus** ProdExpr * ProdExpr : ProdExpr * **prod** Factor ProdExpr : ProdExpr * **div** Factor ProdExpr : ProdExpr * **mod** Factor	R58 **prod** → 99 **div** → 100 **mod** → 101

Estado	Reglas	Transiciones
98	SumExpr : **plus** ProdExpr * ProdExpr : ProdExpr * **prod** Factor ProdExpr : ProdExpr * **div** Factor ProdExpr : ProdExpr * **mod** Factor	R59 **prod** → 99 **div** → 100 **mod** → 101
99	ProdExpr : ProdExpr **prod** * Factor Factor : * Literal Factor : * Reference Factor : * **lparen** Expr **rparen** Literal : * **integer_literal** Literal : * **char_literal** Literal : * **true** Literal : * **false** Reference : * **identifier** Reference : * **identifier** FunctionCall Reference : * **identifier dot identifier** FunctionCall	Factor → 126 Literal → 67 Reference → 68 **lparen** → 69 **integer_literal** → 70 **char_literal** → 71 **true** → 72 **false** → 73 **identifier** → 74
100	ProdExpr : ProdExpr **div** * Factor Factor : * Literal Factor : * Reference Factor : * **lparen** Expr **rparen** Literal : * **integer_literal** Literal : * **char_literal** Literal : * **true** Literal : * **false** Reference : * **identifier** Reference : * **identifier** FunctionCall Reference : * **identifier dot identifier** FunctionCall	Factor → 127 Literal → 67 Reference → 68 **lparen** → 69 **integer_literal** → 70 **char_literal** → 71 **true** → 72 **false** → 73 **identifier** → 74
101	ProdExpr : ProdExpr **mod** * Factor Factor : * Literal Factor : * Reference Factor : * **lparen** Expr **rparen** Literal : * **integer_literal** Literal : * **char_literal** Literal : * **true** Literal : * **false** Reference : * **identifier** Reference : * **identifier** FunctionCall Reference : * **identifier dot identifier** FunctionCall	Factor → 128 Literal → 67 Reference → 68 **lparen** → 69 **integer_literal** → 70 **char_literal** → 71 **true** → 72 **false** → 73 **identifier** → 74
102	Factor : **lparen** Expr * **rparen** Expr : Expr * **or** AndExpr	**rparen** → 129 **or** → 86

Estado	Reglas	Transiciones
103	Reference : **identifier** FunctionCall *	R75
104	Reference : **identifier** **dot** * **identifier** FunctionCall	**identifier** → 130
105	IdStm : **identifier** **assign** Expr * Expr : Expr * **or** AndExpr	R42 **or** → 86
106	Statement : IdStm **semicolon** * IdStm : **identifier** FunctionCall semicolon *	R25 R43
107	IdStm : **identifier** **dot** **identifier** * FunctionCall FunctionCall : * **lparen** **rparen** FunctionCall : * **lparen** ExprList **rparen**	FunctionCall → 132 **lparen** → 78
108	FunctionCall : **lparen** **rparen** *	R77
109	FunctionCall : **lparen** ExprList * **rparen** ExprList : ExprList * **comma** Expr	**rparen**→133 **comma** → 134
110	ExprList : Expr * Expr : Expr * **or** AndExpr	R79 **or** → 86
111	BlockStm : **lbrace** StatementList **rbrace** *	R45
112	IdList : IdList **comma** **identifier** * IdList : IdList **comma** **identifier** * **assign** Expr	R34 **assign** → 135
113	IdList : **identifier** **assign** Expr * Expr : Expr * **or** AndExpr	R33 **or** → 86

Estado	Reglas	Transiciones
114	IfStm : **if** **lparen** Expr **rparen** * Statement IfStm : **if** **lparen** Expr **rparen** * Statement else Statement Statement : * Decl **semicolon** Statement : * IdStm **semicolon** Statement : * IfStm Statement : * SwitchStm Statement : * DoWhileStm Statement : * ForStm Statement : * BreakStm Statement : * ContinueStm Statement : * WhileStm Statement : * ReturnStm Statement : * NoStm Statement : * BlockStm Decl : * Type IdList Type : * **int** Type : * **char** Type : * **boolean** IfStm : * **if** **lparen** Expr **rparen** Statement IfStm : * **if** **lparen** Expr **rparen** Statement **else** Statement SwitchStm : * **switch** **lparen** Expr **rparen** **lbrace** ClauseList **rbrace** DoWhileStm : * **do** Statement **while** **lparen** Expr **rparen** **semicolon** ForStm : * **for** **lparen** ForInit **semicolon** ForCond **semicolon** ForUpdate **rparen** Statement BreakStm : * **break** **semicolon** ContinueStm : * **continue** **semicolon** WhileStm : * **while** **lparen** Expr **rparen** Statement ReturnStm : * **return** Expr **semicolon** ReturnStm : * **return** **semicolon** NoStm : * **semicolon** IdStm : * **identifier** **assign** Expr IdStm : * **identifier** FunctionCall IdStm : * **identifier** **dot** **identifier** FunctionCall BlockStm : * **lbrace** StatementList **rbrace**	Statement → 136 Decl → 38 IdStm → 39 IfStm → 40 SwitchStm → 131 DoWhileStm → 139 ForStm → 144 BreakStm → 145 ContinueStm → 146 WhileStm → 41 ReturnStm → 42 NoStm → 43 BlockStm → 44 Type → 45 **int**→ 20 **char** → 21 **boolean** → 22 **if** → 46 **switch** → 147 **dowhile** → 148 **for** → 149 **break** → 150 **continue** → 151 **while** → 47 **return** → 48 **semicolon** → 49 **identifier** → 50 **lbrace** → 51

Estado	Reglas	Transiciones
115	WhileStm : **while lparen** Expr **rparen** * Statement Statement : * Decl **semicolon** Statement : * IdStm **semicolon** Statement : * IfStm Statement : * SwitchStm Statement : * DoWhileStm Statement : * ForStm Statement : * BreakStm Statement : * ContinueStm Statement : * WhileStm Statement : * ReturnStm Statement : * NoStm Statement : * BlockStm Decl : * Type IdList Type : * **int** Type : * **char** Type : * **boolean** IfStm : * **if lparen** Expr **rparen** Statement IfStm : * **if lparen** Expr **rparen** Statement **else** Statement SwitchStm : * **switch lparen** Expr **rparen lbrace** ClauseList **rbrace** DoWhileStm : * **do** Statement **while lparen** Expr **rparen semicolon** ForStm : * **for lparen** ForInit **semicolon** ForCond **semicolon** ForUpdate **rparen** Statement BreakStm : * **break semicolon** ContinueStm : * **continue semicolon** WhileStm : * **while lparen** Expr **rparen** Statement ReturnStm : * **return** Expr **semicolon** ReturnStm : * **return semicolon** NoStm : * **semicolon** IdStm : * **identifier assign** Expr IdStm : * **identifier** FunctionCall IdStm : * **identifier dot identifier** FunctionCall BlockStm : * **lbrace** StatementList **rbrace**	Statement → 137 Decl → 38 IdStm → 39 IfStm → 40 SwitchStm → 131 DoWhileStm → 139 ForStm → 144 BreakStm → 145 ContinueStm → 146 WhileStm → 41 ReturnStm → 42 NoStm → 43 BlockStm → 44 Type → 45 **int**→ 20 **char** → 21 **boolean** → 22 **if** → 46 **switch** → 147 **dowhile** → 148 **for** → 149 **break** → 150 **continue** → 151 **while** → 47 **return** → 48 **semicolon** → 49 **identifier** → 50 **lbrace** → 51
116	Expr : Expr **or** AndExpr * AndExpr : AndExpr * **and** RelExpr	R47 **and** → 87
117	AndExpr : AndExpr **and** RelExpr *	R49
118	RelExpr : SumExpr **eq** SumExpr * SumExpr : SumExpr * **minus** ProdExpr SumExpr : SumExpr * **plus** ProdExpr	R51 **minus** → 94 **plus** → 95
119	RelExpr : SumExpr **ne** SumExpr * SumExpr : SumExpr * **minus** ProdExpr SumExpr : SumExpr * **plus** ProdExpr	R52 **minus** → 94 **plus** → 95
120	RelExpr : SumExpr **gt** SumExpr * SumExpr : SumExpr * **minus** ProdExpr SumExpr : SumExpr * **plus** ProdExpr	R53 **minus** → 94 **plus** → 95
121	RelExpr : SumExpr **ge** SumExpr * SumExpr : SumExpr * **minus** ProdExpr SumExpr : SumExpr * **plus** ProdExpr	R54 **minus** → 94 **plus** → m95

Estado	Reglas	Transiciones
122	RelExpr : SumExpr **lt** SumExpr * SumExpr : SumExpr * **minus** ProdExpr SumExpr : SumExpr * **plus** ProdExpr	R55 **minus** → 94 **plus** → 95
123	RelExpr : SumExpr **le** SumExpr * SumExpr : SumExpr * **minus** ProdExpr SumExpr : SumExpr * **plus** ProdExpr	R56 **minus** → 94 **plus** → 95
124	SumExpr : SumExpr **minus** ProdExpr * ProdExpr : ProdExpr * **prod** Factor ProdExpr : ProdExpr * **div** Factor ProdExpr : ProdExpr * **mod** Factor	R61 **prod** → 99 **div** → 100 **mod** → 101
125	SumExpr : SumExpr **plus** ProdExpr * ProdExpr : ProdExpr * **prod** Factor ProdExpr : ProdExpr * **div** Factor ProdExpr : ProdExpr * **mod** Factor	R62 **prod** → 99 **div** → 100 **mod** → 101
126	ProdExpr : ProdExpr **prod** Factor *	R64
127	ProdExpr : ProdExpr **div** Factor *	R65
128	ProdExpr : ProdExpr **mod** Factor *	R66
129	Factor : **lparen** Expr **rparen** *	R69
130	Reference : **identifier** **dot** **identifier** * FunctionCall FunctionCall : * **lparen** **rparen** FunctionCall : * **lparen** ExprList **rparen**	FunctionCall → 138 **lparen** → 78
131	SwitchStm : SwitchStm * IdStm : **identifier** **assign** Expr *	R81 R42
132	IdStm : **identifier** **dot** **identifier** FunctionCall *	R44
133	FunctionCall : **lparen** ExprList **rparen** *	R78

Estado	Reglas	Transiciones
134	ExprList : ExprList **comma** * Expr Expr : * AndExpr Expr : * Expr **or** AndExpr AndExpr : * RelExpr AndExpr : * AndExpr **and** RelExpr RelExpr : * SumExpr RelExpr : * SumExpr **eq** SumExpr RelExpr : * SumExpr **ne** SumExpr RelExpr : * SumExpr **gt** SumExpr RelExpr : * SumExpr **ge** SumExpr RelExpr : * SumExpr **lt** SumExpr RelExpr : * SumExpr **le** SumExpr SumExpr : * **not** ProdExpr SumExpr : * **minus** ProdExpr SumExpr : * **plus** ProdExpr SumExpr : * ProdExpr SumExpr : * SumExpr **minus** ProdExpr SumExpr : * SumExpr **plus** ProdExpr ProdExpr : * Factor ProdExpr : * ProdExpr **prod** Factor ProdExpr : * ProdExpr **div** Factor ProdExpr : * ProdExpr **mod** Factor Factor : * Literal Factor : * Reference Factor : * **lparen** Expr **rparen** Literal : * **integer_literal** Literal : * **char_literal** Literal : * **true** Literal : * **false** Reference : * **identifier** Reference : * **identifier** FunctionCall Reference : * **identifier dot identifier** FunctionCall	Expr → 140 AndExpr → 59 RelExpr → 60 SumExpr → 61 **not** → 62 **minus** → 63 **plus** → 64 ProdExpr → 65 Factor → 66 Literal → 67 Reference → 68 **lparen** → 69 **integer_literal** →70 **char_literal** → 71 **true** → 72 **false** → 73 **identifier** → 74

Estado	Reglas	Transiciones
135	IdList : IdList **comma** **identifier** **assign** * Expr Expr : * AndExpr Expr : * Expr **or** AndExpr AndExpr : * RelExpr AndExpr : * AndExpr **and** RelExpr RelExpr : * SumExpr RelExpr : * SumExpr **eq** SumExpr RelExpr : * SumExpr **ne** SumExpr RelExpr : * SumExpr **gt** SumExpr RelExpr : * SumExpr **ge** SumExpr RelExpr : * SumExpr **lt** SumExpr RelExpr : * SumExpr **le** SumExpr SumExpr : * **not** ProdExpr SumExpr : * **minus** ProdExpr SumExpr : * **plus** ProdExpr SumExpr : * ProdExpr SumExpr : * SumExpr **minus** ProdExpr SumExpr : * SumExpr **plus** ProdExpr ProdExpr : * Factor ProdExpr : * ProdExpr **prod** Factor ProdExpr : * ProdExpr **div** Factor ProdExpr : * ProdExpr **mod** Factor Factor : * Literal Factor : * Reference Factor : * **lparen** Expr **rparen** Literal : * **integer_literal** Literal : * **char_literal** Literal : * **true** Literal : * **false** Reference : * **identifier** Reference : * **identifier** FunctionCall Reference : * **identifier** **dot** **identifier** FunctionCall	Expr → 141 AndExpr → 59 RelExpr → 60 SumExpr → 61 **not** → 62 **minus** → 63 **plus** → 64 ProdExpr → 65 Factor → 66 Literal → 67 Reference → 68 **lparen** → 69 **integer_literal** →70 **char_literal** → 71 **true** → 72 **false** → 73 **identifier** → 74
136	IfStm : **if** **lparen** Expr **rparen** Statement * IfStm : **if** **lparen** Expr **rparen** Statement * **else** Statement	R36 **else** → 142
137	WhileStm : **while** **lparen** Expr **rparen** Statement *	R38
138	Reference : **identifier** **dot** **identifier** FunctionCall *	R76
139	Statement: DoWhileStm * IdStm : **identifier** **dot** **identifier** FunctionCall *	R83 R44
140	ExprList : ExprList **comma** Expr * Expr : Expr * **or** AndExpr	R80 **or** → 86

Estado	Reglas	Transiciones
141	IdList : IdList **comma identifier assign** Expr * Expr : Expr * **or** AndExpr	R35 **or** → 86
142	IfStm : **if lparen** Expr **rparen** Statement **else** * Statement Statement : * Decl **semicolon** Statement : * IdStm **semicolon** Statement : * IfStm Statement : * SwitchStm Statement : * DoWhileStm Statement : * ForStm Statement : * BreakStm Statement : * ContinueStm Statement : * WhileStm Statement : * ReturnStm Statement : * NoStm Statement : * BlockStm Decl : * Type IdList Type : * **int** Type : * **char** Type : * **boolean** IfStm : * **if lparen** Expr **rparen** Statement IfStm : * **if lparen** Expr **rparen** Statement **else** Statement SwitchStm : * **switch lparen** Expr **rparen lbrace** ClauseList **rbrace** DoWhileStm : * **do** Statement **while lparen** Expr **rparen semicolon** ForStm : * **for lparen** ForInit **semicolon** ForCond **semicolon** ForUpdate **rparen** Statement BreakStm : * **break semicolon** ContinueStm : * **continue semicolon** WhileStm : * **while lparen** Expr **rparen** Statement ReturnStm : * **return** Expr **semicolon** ReturnStm : * **return semicolon** NoStm : * **semicolon** IdStm : * **identifier assign** Expr IdStm : * **identifier** FunctionCall IdStm : * **identifier dot identifier** FunctionCall BlockStm : * **lbrace** StatementList **rbrace**	Statement → 143 Decl → 38 IdStm → 39 IfStm → 40 SwitchStm → 131 DoWhileStm → 139 ForStm → 144 BreakStm → 145 ContinueStm → 146 WhileStm → 41 ReturnStm → 42 NoStm → 43 BlockStm → 44 Type → 45 **int**→ 20 **char** → 21 **boolean** → 22 **if** → 46 **switch** → 147 **dowhile** → 148 **for** → 149 **break** → 150 **continue** → 151 **while** → 47 **return** → 48 **semicolon** → 49 **identifier** → 50 **lbrace** → 51
143	IfStm : **if lparen** Expr **rparen** Statement **else** Statement *	R37
144	Statement : ForStm *	R82
145	Statement : BreakStm *	R85
146	Statement : ContinueStm *	R84
147	SwitchStm : **switch** * **lparen** Expr **rparen lbrace** ClauseList **rbrace**	**lparen** → 152

Estado	Reglas	Transiciones
148	DoWhileStm : **do** * Statement **while lparen** Expr **rparen semicolon** Statement : * Decl **semicolon** Statement : * IdStm **semicolon** Statement : * IfStm Statement : * SwitchStm Statement : * DoWhileStm Statement : * ForStm Statement : * BreakStm Statement : * ContinueStm Statement : * WhileStm Statement : * ReturnStm Statement : * NoStm Statement : * BlockStm Decl : * Type IdList Type : * **int** Type : * **char** Type : * **boolean** IfStm : * **if lparen** Expr **rparen** Statement IfStm : * **if lparen** Expr **rparen** Statement **else** Statement SwitchStm : * **switch lparen** Expr **rparen lbrace** ClauseList **rbrace** DoWhileStm : * **do** Statement **while lparen** Expr **rparen semicolon** ForStm : * **for lparen** ForInit **semicolon** ForCond **semicolon** ForUpdate **rparen** Statement BreakStm : * **break semicolon** ContinueStm : * **continue semicolon** WhileStm : * **while lparen** Expr **rparen** Statement ReturnStm : * **return** Expr **semicolon** ReturnStm : * **return semicolon** NoStm : * **semicolon** IdStm : * **identifier assign** Expr IdStm : * **identifier** FunctionCall IdStm : * **identifier dot identifier** FunctionCall BlockStm : * **lbrace** StatementList **rbrace**	Statement → 167 Decl → 38 IdStm → 39 IfStm → 40 SwitchStm → 131 DoWhileStm → 139 ForStm → 144 BreakStm → 145 ContinueStm → 146 WhileStm → 41 ReturnStm → 42 NoStm → 43 BlockStm → 44 Type → 45 **int**→ 20 **char** → 21 **boolean** → 22 **if** → 46 **switch** → 147 **dowhile** → 148 **for** → 149 **break** → 150 **continue** → 151 **while** → 47 **return** → 48 **semicolon** → 49 **identifier** → 50 **lbrace** → 51

Estado	Reglas	Transiciones
149	ForStm : **for** * **lparen** ForInit **semicolon** ForCond **semicolon** ForUpdate **rparen** Statement	**lparen** → 173
150	BreakStm : **break** * **semicolon**	**semicolon** → 188
151	ContinueStm : **continue** * **semicolon**	**semicolon** → 189
152	SwitchStm : **switch lparen** * Expr **rparen lbrace** ClauseList **rbrace** Expr : * AndExpr Expr : * Expr **or** AndExpr AndExpr : * RelExpr AndExpr : * AndExpr **and** RelExpr RelExpr : * SumExpr RelExpr : * SumExpr **eq** SumExpr RelExpr : * SumExpr **ne** SumExpr RelExpr : * SumExpr **gt** SumExpr RelExpr : * SumExpr **ge** SumExpr RelExpr : * SumExpr **lt** SumExpr RelExpr : * SumExpr **le** SumExpr SumExpr : * **not** ProdExpr SumExpr : * **minus** ProdExpr SumExpr : * **plus** ProdExpr SumExpr : * ProdExpr SumExpr : * SumExpr **minus** ProdExpr SumExpr : * SumExpr **plus** ProdExpr ProdExpr : * Factor ProdExpr : * ProdExpr **prod** Factor ProdExpr : * ProdExpr **div** Factor ProdExpr : * ProdExpr **mod** Factor Factor : * Literal Factor : * Reference Factor : * **lparen** Expr **rparen** Literal : * **integer_literal** Literal : * **char_literal** Literal : * **true** Literal : * **false** Reference : * **identifier** Reference : * **identifier** FunctionCall Reference : * **identifier dot identifier** FunctionCall	Expr → 153 AndExpr → 59 RelExpr → 60 SumExpr → 61 **not** → 62 **minus** → 63 **plus** → 64 PordExpr → 65 Factor → 66 Literal → 67 Reference → 68 **lparen** → 69 **integer_literal** → 70 **char_literal** → 71 **true** → 72 **false** → 73 **identifier** → 74
153	SwitchStm : **switch lparen** Expr * **rparen lbrace** ClauseList **rbrace** Expr : Expr * **or** AndExpr	**rparen** → 154 **or** → 86
154	SwitchStm : **switch lparen** Expr **rparen** * **lbrace** ClauseList **rbrace**	**lbrace** → 155

Estado	Reglas	Transiciones
155	SwitchStm : **switch lparen** Expr **rparen lbrace** * ClauseList **rbrace** ClauseList : ClauseList * CaseClause ClauseList : ClauseList * DefaultClause ClauseList : *	ClauseList → 156 R89
156	SwitchStm : **switch lparen** Expr **rparen lbrace** ClauseList * **rbrace** ClauseList : ClauseList * CaseClause ClauseList : ClauseList * DefaultClause CaseClasue : * **case integer_literal colon** StatementList DefaultClause **:** * **default colon** StatementList	**rbrace** → 157 CaseClause → 158 DefaultClause → 159 **case** → 160 **default** → 161
157	SwitchStm : **switch lparen** Expr **rparen lbrace** ClauseList **rbrace** *	R88
158	ClauseList : ClauseList CaseClause *	R90
159	ClauseList : ClauseList DefaultClause *	R91
160	CaseClause : **case** * **integer_literal colon** StatementList	**integer_literal** → 162
161	DefaultClause **: default** * **colon** StatementList	**colon** → 163
162	CaseClause : **case integer_literal** * **colon** StatementList	**colon** → 164
163	DefaultClause **: default colon** * StatementList StatementList : * StatementList Statement StatementList : *	StatementList → 165 R23
164	CaseClause : **case integer_literal colon** * StatementList StatementList : * StatementList Statement StatementList : *	StatementList → 166 R23

Estado	Reglas	Transiciones
165	DefaultClause **: default colon** StatementList * StatementList : StatementList * Statement Statement : * Decl **semicolon** Statement : * IdStm **semicolon** Statement : * IfStm Statement : * SwitchStm Statement : * DoWhileStm Statement : * ForStm Statement : * BreakStm Statement : * ContinueStm Statement : * WhileStm Statement : * ReturnStm Statement : * NoStm Statement : * BlockStm Decl : * Type IdList Type : * **int** Type : * **char** Type : * **boolean** IfStm : * **if lparen** Expr **rparen** Statement IfStm : * **if lparen** Expr **rparen** Statement **else** Statement SwitchStm : * **switch lparen** Expr **rparen lbrace** ClauseList **rbrace** DoWhileStm : * **do** Statement **while lparen** Expr **rparen semicolon** ForStm : * **for lparen** ForInit **semicolon** ForCond **semicolon** ForUpdate **rparen** Statement BreakStm : * **break semicolon** ContinueStm : * **continue semicolon** WhileStm : * **while lparen** Expr **rparen** Statement ReturnStm : * **return** Expr **semicolon** ReturnStm : * **return semicolon** NoStm : * **semicolon** IdStm : * **identifier assign** Expr IdStm : * **identifier** FunctionCall IdStm : * **identifier dot identifier** FunctionCall BlockStm : * **lbrace** StatementList **rbrace**	R93 Statement → 37 Decl → 38 IdStm → 39 IfStm → 40 SwitchStm → 131 DoWhileStm → 139 ForStm → 144 BreakStm → 145 ContinueStm → 146 WhileStm → 41 ReturnStm → 42 NoStm → 43 BlockStm → 44 Type → 45 **int**→ 20 **char** → 21 **boolean** → 22 **if** → 46 **switch** → 147 **dowhile** → 148 **for** → 149 **break** → 150 **continue** → 151 **while** → 47 **return** → 48 **semicolon** → 49 **identifier** → 50 **lbrace** → 51

Estado	Reglas	Transiciones
166	CaseClause : **case integer_literal colon** StatementList * StatementList : StatementList * Statement Statement : * Decl **semicolon** Statement : * IdStm **semicolon** Statement : * IfStm Statement : * SwitchStm Statement : * DoWhileStm Statement : * ForStm Statement : * BreakStm Statement : * ContinueStm Statement : * WhileStm Statement : * ReturnStm Statement : * NoStm Statement : * BlockStm Decl : * Type IdList Type : * **int** Type : * **char** Type : * **boolean** IfStm : * **if lparen** Expr **rparen** Statement IfStm : * **if lparen** Expr **rparen** Statement **else** Statement SwitchStm : * **switch lparen** Expr **rparen lbrace** ClauseList **rbrace** DoWhileStm : * **do** Statement **while lparen** Expr **rparen semicolon** ForStm : * **for lparen** ForInit **semicolon** ForCond **semicolon** ForUpdate **rparen** Statement BreakStm : * **break semicolon** ContinueStm : * **continue semicolon** WhileStm : * **while lparen** Expr **rparen** Statement ReturnStm : * **return** Expr **semicolon** ReturnStm : * **return semicolon** NoStm : * **semicolon** IdStm : * **identifier assign** Expr IdStm : * **identifier** FunctionCall IdStm : * **identifier dot identifier** FunctionCall BlockStm : * **lbrace** StatementList **rbrace**	R92 Statement → 37 Decl → 38 IdStm → 39 IfStm → 40 SwitchStm → 131 DoWhileStm → 139 ForStm → 144 BreakStm → 145 ContinueStm → 146 WhileStm → 41 ReturnStm → 42 NoStm → 43 BlockStm → 44 Type → 45 **int**→ 20 **char** → 21 **boolean** → 22 **if** → 46 **switch** → 147 **dowhile** → 148 **for** → 149 **break** → 150 **continue** → 151 **while** → 47 **return** → 48 **semicolon** → 49 **identifier** → 50 **lbrace** → 51
167	DoWhileStm : **do** Statement * **while lparen** Expr **rparen semicolon**	**while** → 168
168	DoWhileStm : **do** Statement **while** * **lparen** Expr **rparen semicolon**	**lparen** → 169

Estado	Reglas	Transiciones
169	DoWhileStm : **do** Statement **while** **lparen** * Expr **rparen** **semicolon** Expr : * AndExpr Expr : * Expr **or** AndExpr AndExpr : * RelExpr AndExpr : * AndExpr **and** RelExpr RelExpr : * SumExpr RelExpr : * SumExpr **eq** SumExpr RelExpr : * SumExpr **ne** SumExpr RelExpr : * SumExpr **gt** SumExpr RelExpr : * SumExpr **ge** SumExpr RelExpr : * SumExpr **lt** SumExpr RelExpr : * SumExpr **le** SumExpr SumExpr : * **not** ProdExpr SumExpr : * **minus** ProdExpr SumExpr : * **plus** ProdExpr SumExpr : * ProdExpr SumExpr : * SumExpr **minus** ProdExpr SumExpr : * SumExpr **plus** ProdExpr ProdExpr : * Factor ProdExpr : * ProdExpr **prod** Factor ProdExpr : * ProdExpr **div** Factor ProdExpr : * ProdExpr **mod** Factor Factor : * Literal Factor : * Reference Factor : * **lparen** Expr **rparen** Literal : * **integer_literal** Literal : * **char_literal** Literal : * **true** Literal : * **false** Reference : * **identifier** Reference : * **identifier** FunctionCall Reference : * **identifier** **dot** **identifier** FunctionCall	Expr → 170 AndExpr → 59 RelExpr → 60 SumExpr → 61 **not** → 62 **minus** → 63 **plus** → 64 PordExpr → 65 Factor → 66 Literal → 67 Reference → 68 **lparen** → 69 **integer_literal** → 70 **char_literal** → 71 **true** → 72 **false** → 73 **identifier** → 74
170	DoWhileStm : **do** Statement **while** **lparen** Expr * **rparen** **semicolon**	**rparen** → 171
171	DoWhileStm : **do** Statement **while** **lparen** Expr **rparen** * **semicolon**	**semicolon** → 172
172	DoWhileStm : **do** Statement **while** **lparen** Expr **rparen** **semicolon** *	R94

Estado	Reglas	Transiciones
173	ForStm : **for lparen** * ForInit **semicolon** ForCond **semicolon** ForUpdate **rparen** Statement ForInit : * Decl ForInit : * IdStmList ForInit : * Decl : * Type IdList Type : * **int** Type : * **char** Type : * **boolean** IdStmList : * IdStm IdStmList : * IdStmList **comma** IdStm IdStm : * **identifier** **assign** Expr IdStm : * **identifier** FunctionCall IdStm : * **identifier** **dot** **identifier** FunctionCall	ForInit → 174 Decl → 175 IdStmList → 176 R98 Type → 45 **int** → 20 **char** → 21 **boolean** → 22 IdStm → 177 **identifier** → 50
174	ForStm : **for lparen** ForInit * **semicolon** ForCond **semicolon** ForUpdate **rparen** Statement	**semicolon** → 178
175	ForInit : Decl *	R96
176	ForInit : IdStmList * IdStmList : IdStmList * **comma** IdStm	R97 **comma** → 179
177	IdStmList : IdStm *	R103

Estado	Reglas	Transiciones
178	ForStm : **for lparen** ForInit **semicolon** * ForCond **semicolon** ForUpdate **rparen** Statement ForCond : * Expr ForCond : * Expr : * AndExpr Expr : * Expr **or** AndExpr AndExpr : * RelExpr AndExpr : * AndExpr **and** RelExpr RelExpr : * SumExpr RelExpr : * SumExpr **eq** SumExpr RelExpr : * SumExpr **ne** SumExpr RelExpr : * SumExpr **gt** SumExpr RelExpr : * SumExpr **ge** SumExpr RelExpr : * SumExpr **lt** SumExpr RelExpr : * SumExpr **le** SumExpr SumExpr : * **not** ProdExpr SumExpr : * **minus** ProdExpr SumExpr : * **plus** ProdExpr SumExpr : * ProdExpr SumExpr : * SumExpr **minus** ProdExpr SumExpr : * SumExpr **plus** ProdExpr ProdExpr : * Factor ProdExpr : * ProdExpr **prod** Factor ProdExpr : * ProdExpr **div** Factor ProdExpr : * ProdExpr **mod** Factor Factor : * Literal Factor : * Reference Factor : * **lparen** Expr **rparen** Literal : * **integer_literal** Literal : * **char_literal** Literal : * **true** Literal : * **false** Reference : * **identifier** Reference : * **identifier** FunctionCall Reference : * **identifier dot identifier** FunctionCall	ForCond → 180 Expr → 181 R100 AndExpr → 59 RelExpr → 60 SumExpr → 61 **not** → 62 **minus** → 63 **plus** → 64 PordExpr → 65 Factor → 66 Literal → 67 Reference → 68 **lparen** → 69 **integer_literal** → 70 **char_literal** → 71 **true** → 72 **false** → 73 **identifier** → 74
179	IdStmList : IdStmList **comma** * IdStm IdStm : * **identifier** **assign** Expr IdStm : * **identifier** FunctionCall IdStm : * **identifier** **dot** **identifier** FunctionCall	IdStm → 182 **identifier** → 50
180	ForStm : **for lparen** ForInit **semicolon** ForCond * **semicolon** ForUpdate **rparen** Statement	**semicolon** → 183
181	ForCond : Expr *	R99
182	IdStmList : IdStmList **comma** IdStm *	R104
183	ForStm : **for lparen** ForInit **semicolon** ForCond **semicolon** * ForUpdate **rparen** Statement ForUpdate : * IdStmList ForUpdate : * IdStmList : * IdStm IdStmList : * IdStmList **comma** IdStm IdStm : * **identifier** **assign** Expr IdStm : * **identifier** FunctionCall IdStm : * **identifier** **dot** **identifier** FunctionCall	ForUpdate → 184 IdStmList → 185 R102 IdStm → 177 **identifier** → 50
184	ForStm : **for lparen** ForInit **semicolon** ForCond **semicolon** ForUpdate * **rparen** Statement	**rparen** → 186
185	ForUpdate : IdStmList * IdStmList : IdStmList * **comma** IdStm	R101 **comma** → 179

Estado	Reglas	Transiciones
186	ForStm : **for lparen** ForInit **semicolon** ForCond **semicolon** ForUpdate **rparen** * Statement Statement : * Decl **semicolon** Statement : * IdStm **semicolon** Statement : * IfStm Statement : * SwitchStm Statement : * DoWhileStm Statement : * ForStm Statement : * BreakStm Statement : * ContinueStm Statement : * WhileStm Statement : * ReturnStm Statement : * NoStm Statement : * BlockStm Decl : * Type IdList Type : * **int** Type : * **char** Type : * **boolean** IfStm : * **if lparen** Expr **rparen** Statement IfStm : * **if lparen** Expr **rparen** Statement **else** Statement SwitchStm : * **switch lparen** Expr **rparen lbrace** ClauseList **rbrace** DoWhileStm : * **do** Statement **while lparen** Expr **rparen semicolon** ForStm : * **for lparen** ForInit **semicolon** ForCond **semicolon** ForUpdate **rparen** Statement BreakStm : * **break semicolon** ContinueStm : * **continue semicolon** WhileStm : * **while lparen** Expr **rparen** Statement ReturnStm : * **return** Expr **semicolon** ReturnStm : * **return semicolon** NoStm : * **semicolon** IdStm : * **identifier assign** Expr IdStm : * **identifier** FunctionCall IdStm : * **identifier dot identifier** FunctionCall BlockStm : * **lbrace** StatementList **rbrace**	Statement → 187 Decl → 38 IdStm → 39 IfStm → 40 SwitchStm → 131 DoWhileStm → 139 ForStm → 144 BreakStm → 145 ContinueStm → 146 WhileStm → 41 ReturnStm → 42 NoStm → 43 BlockStm → 44 Type → 45 **int**→ 20 **char** → 21 **boolean** → 22 **if** → 46 **switch** → 147 **dowhile** → 148 **for** → 149 **break** → 150 **continue** → 151 **while** → 47 **return** → 48 **semicolon** → 49 **identifier** → 50 **lbrace** → 51
187	ForStm : **for lparen** ForInit **semicolon** ForCond **semicolon** ForUpdate **rparen** Statement *	R95
188	BreakStm : **break semicolon** *	R86
189	ContinueStm : **continue semicolon** *	R87

Cálculo de los conjuntos siguientes

Símbolo	Siguientes
CompilationUnit	EOF
ImportClauseList	import, library
ImportClause	import, library
LibraryDecl	EOF
FunctionList	rbrace, public, private
FunctionDecl	rbrace, public, private
Access	void, int, char, boolean
FunctionType	identifier
Type	identifier
ArgumentDecl	lbrace
ArgumentList	rparen, comma
Argument	rparen, comma
FunctionBody	rbrace, public, private
StatementList	rbrace, int, char, boolean, identifier, if, while, return, semicolon, lbrace, switch, do, for, break, continue, case, default
Statement	rbrace, int, char, boolean, identifier, if, while, return, semicolon, lbrace, else, switch, do, for, break, continue, case, default
Decl	semicolon
IdList	semicolon, comma
IfStm	rbrace, int, char, boolean, identifier, if, while, return, semicolon, lbrace, else, switch, do, for, break, continue, case, default
WhileStm	rbrace, int, char, boolean, identifier, if, while, return, semicolon, lbrace, else, switch, do, for, break, continue, case, default
ReturnStm	rbrace, int, char, boolean, identifier, if, while, return, semicolon, lbrace, else, switch, do, for, break, continue, case, default
NoStm	rbrace, int, char, boolean, identifier, if, while, return, semicolon, lbrace, else, switch, do, for, break, continue, case, default
IdStmList	comma, semicolon, rparen
IdStm	comma, semicolon, rparen
BlockStm	rbrace, int, char, boolean, identifier, if, while, return, semicolon, lbrace, else, switch, do, for, break, continue, case, default
Expr	comma, semicolon, rparen, or
AndExpr	comma, semicolon, rparen, or, and
RelExpr	comma, semicolon, rparen, or, and
SumExpr	comma, semicolon, rparen, or, and, eq, ne, gt, ge, lt, le, minus, plus
ProdExpr	comma, semicolon, rparen, or, and, eq, ne, gt, ge, lt, le, minus, plus, prod, div, mod

Factor	comma, semicolon, rparen, or, and, eq, ne, gt, ge, lt, le, minus, plus, prod, div, mod
Literal	comma, semicolon, rparen, or, and, eq, ne, gt, ge, lt, le, minus, plus, prod, div, mod
Reference	comma, semicolon, rparen, or, and, eq, ne, gt, ge, lt, le, minus, plus, prod, div, mod
FunctionCall	comma, semicolon, rparen, or, and, eq, ne, gt, ge, lt, le, minus, plus, prod, div, mod
ExprList	rparen, comma
ForStm	rbrace, int, char, boolean, identifier, if, while, return, semicolon, lbrace, else, switch, do, for, break, continue, case, default
ForUpdate	rparen
ForCond	semicolon
SwitchStm	rbrace, int, char, boolean, identifier, if, while, return, semicolon, lbrace, else, switch, do, for, break, continue, case, default
DoWhileStm	rbrace, int, char, boolean, identifier, if, while, return, semicolon, lbrace, else, switch, do, for, break, continue, case, default
ForInit	semicolon
ContinueStm	rbrace, int, char, boolean, identifier, if, while, return, semicolon, lbrace, else, switch, do, for, break, continue, case, default
BreakStm	rbrace, int, char, boolean, identifier, if, while, return, semicolon, lbrace, else, switch, do, for, break, continue, case, default
ClauseList	rbrace, case, default
CaseClause	rbrace, case, default
DefaultClause	rbrace, case, default

Descripción de los métodos programados en la clase TintoParser

En la clase TintoParse.java tenemos 2 métodos a modificar, el método **initActionTable()** y **initGotoTable()**:

En el primer método tenemos que incrementar **el tamaño del array** a la nueva **cantidad de estados** y **tokens** que tenemos.

En este método nos encargaremos de reconocer las transiciones a tipos terminales y aceptaciones de regla si procede. Para el avance haremos uso del **SHIFT**, y para reducir a la aceptación de la regla haremos uso del **REDUCE**.

```
*/
private void initActionTable() {
    //actionTable = new ActionElement[143][40]; // 143 estados, 40 tokens
    actionTable = new ActionElement[190][49]; // 190 estados, 49 tokens

    actionTable[0][IMPORT] = new ActionElement(ActionElement.REDUCE,3);
    actionTable[0][LIBRARY] = new ActionElement(ActionElement.REDUCE,3);

    actionTable[1][EOF] = new ActionElement(ActionElement.ACCEPT,0);

    actionTable[2][IMPORT] = new ActionElement(ActionElement.SHIFT,5);
    actionTable[2][LIBRARY] = new ActionElement(ActionElement.SHIFT,6);

    actionTable[3][EOF] = new ActionElement(ActionElement.REDUCE,1);

    actionTable[4][IMPORT] = new ActionElement(ActionElement.REDUCE,2);
    actionTable[4][LIBRARY] = new ActionElement(ActionElement.REDUCE,2);

    actionTable[5][IDENTIFIER] = new ActionElement(ActionElement.SHIFT,7);

    actionTable[6][IDENTIFIER] = new ActionElement(ActionElement.SHIFT,8);

    actionTable[7][SEMICOLON] = new ActionElement(ActionElement.SHIFT,9);

    actionTable[8][LBRACE] = new ActionElement(ActionElement.SHIFT,10);

    actionTable[9][IMPORT] = new ActionElement(ActionElement.REDUCE,4);
    actionTable[9][LIBRARY] = new ActionElement(ActionElement.REDUCE,4);
```

En el segundo método también tenemos que incrementar el **tamaño del array** a la nueva cantidad de **estados** y **símbolos no terminales** que tenemos.

En este método, nos encargaremos de reconocer las transiciones a **tipos no terminales**. Para ellos, le diremos en función de la entrada que tengamos a qué estado tenemos que pasar.

```
private void initGotoTable() {
    //gotoTable = new int[143][33];  // 143 estados, 33 s@mbolos no terminales
    gotoTable = new int[190][45];  // 190 estados, 45 s@mbolos no terminales

    gotoTable[0][S_COMPILATION_UNIT] = 1;
    gotoTable[0][S_IMPORT_CLAUSE_LIST] = 2;

    gotoTable[2][S_LIBRARY_DECL] = 3;
    gotoTable[2][S_IMPORT_CLAUSE] = 4;

    gotoTable[10][S_FUNCTION_LIST] = 11;

    gotoTable[11][S_FUNCTION_DECL] = 13;
    gotoTable[11][S_ACCESS] = 14;

    gotoTable[14][S_FUNCTION_TYPE] = 17;
    gotoTable[14][S_TYPE] = 18;


    gotoTable[23][S_ARGUMENT_DECL] = 24;

    gotoTable[24][S_FUNCTION_BODY] = 26;
```

Obviamente tanto los **tokens** como los **símbolos no terminales** hemos tenido que añadirlos a sus respectivos métodos.

En este caso los terminales a **TokenConstants.java** y los no terminales a **SymbolConstants.java**.

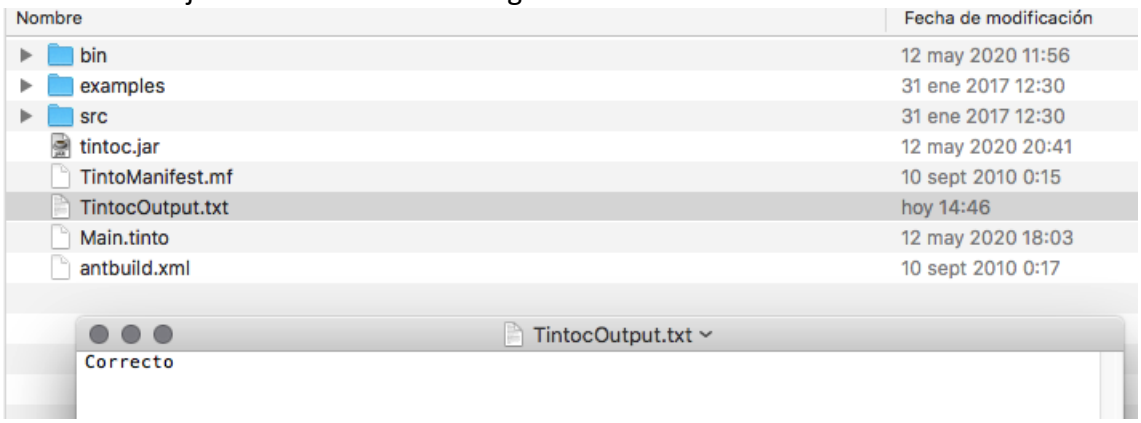
Todo el código está comentado donde hemos realizado las modificaciones para un correcto seguimiento del código.

Pruebas de funcionamiento

Como main de pruebas se ha usado el siguiente código:

```
import Console;

library Main {
  public void Main()
  {
       int b=1, p=1, c=9;

       switch(b) {
              case 0:
                     imprimir(b);
        imprimir(p);
                     break;
              case 1:
                     imprimir(c);
                     break;
              default:
                     imprimir(p);
       }

    if ( p== 0) {
      imprimir(p);
    } else {
      for(int i=0; i<10; i = i+1)
        imprimir(i);
    }

       for (int i=1; i<100; i = i+1)
       {
         if( esPrimo(i) ) {
        imprimir(i);
        imprimir(b);
      }
        else
        imprimir(i);
       }

    do {
      imprimir(c);
      imprimir(b);
    } while( c==9 );

    if( c == 9)
      break;
    else
      continue;
  }

  private void imprimir(int i)
  {
    Console.print(i);
    Console.print('\n');
  }

  /**
   * Verifica si un número es primo
   */
  private boolean esPrimo(int i)
  {
    int j = 2;
    while(j<i)
    {
      if(i%j == 0) return false;
      j = j+1;
    }
    return true;
  }
}
```

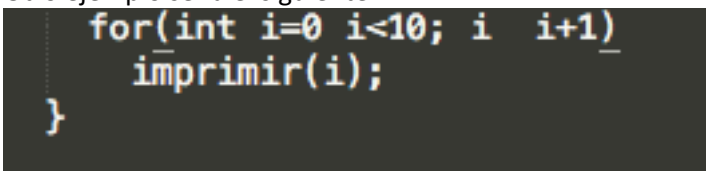
Al realizar la ejecución obtenemos lo siguiente:

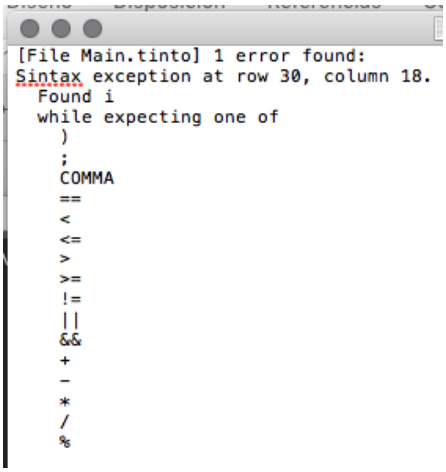


Esto quiere decir, que nuestro código ha sido reconocido sin ningún problema, si provocamos algún fallo intencionado, por ejemplo, eliminando una llave de alguna sentencia:

Hemos cambiado un } por un “ “ y así nos lo muestra.

Otro ejemplo sería el siguiente:





Por lo que podemos pensar sin lugar a duda que nuestro compilador está correctamente realizado.

Evidentemente, esta conclusión es después de pulir los primeros fallos que se generaron en la primera compilación.