Walter Orozco
Ricardo Chian

“Fase # 3”

➢ Gramática Original:

- Program → Decl+
- Decl → VariableDecl | FunctionDecl | ConstDecl | ClassDecl | InterfaceDecl
- VariableDecl → Variable ;
- Variable → Type ident
- ConstDecl → const ConstType ident;
- ConstType → int | double | bool | string
- Type → int | double | bool | string | ident | Type[]
- FunctionDecl → Type ident (Formals) StmtBlock | void ident (Formals) StmtBlock
- Formals → Variable , Formals | Variable
- ClassDecl → class ident < : ident > < , ident+ , > { Field* }
- Field → VariableDecl | FuncionDecl | ConstDecl
- interfaceDecl → interface ident { Prototype* }
- Prototype → Type ident (Formals) ; | void ident (Formals) ;
- StmtBlock → { VariableDecl* ConstDecl* Stmt* }
- Stmt → < Expr > ; | IfStmt | WhileStmt | ForStmt | BreakStmt | Return Stmt | PrintStmt | StmtBlock | CallStmt
- CallStmt → ident (Actuals) | ident . ident (Actuals)
- Actuals → Expr , Actuals | Expr
- IfStmt → if (Expr) Stmt < else Stmt >
- WhileStmt → while (Expr) Stmt
- ForStmt → for (Expr ; Expr ; Expr) Stmt
- ReturnStmt → return Expr ;
- BreakStmt → break ;
- PrintStmt → Console.Writeline(Expr* ,) ;
- Expr → Lvalue = Expr | Constant | LValue | this | (Expr) | Expr + Expr | Expr * Expr | Expr % Expr | - Expr | Expr < Expr | Expr <= Expr | Expr == Expr | Expr && Expr | ! Expr | New (ident)
- LValue → ident | Expr . ident
- Constant → intConstant | doubleConstant | boolConstant | stringConstant | null

Walter Orozco
Ricardo Chian

- Gramática Final Implementada:

0. Program’ → Program
1. Program → Decl Decl2
2. Decl2 → Decl Decl2
3. Decl2 → eps
4. Decl → VariableDecl
5. Decl → FunctionDecl
6. Decl → ConstDecl
7. Decl → ClassDecl
8. Decl → InterfaceDecl
9. VariableDecl → Variable ;
10. Variable → Type ident
11. ConstDecl → const ConstType ident ;
12. ConstType → int
13. ConstType → double
14. ConstType → bool
15. ConstType → string
16. Type → int
17. Type → double
18. Type → bool
19. Type → string
20. Type → ident
21. Type → Type [ ]
22. FunctionDecl → Type ident (Formals) StmtBlock
23. FunctionDecl → void ident (Formals) StmtBlock
24. Formals → Variable , Formals
25. Formals → Variable
26. ClassDecl → class ident ClassDecl2 ClassDecl3 { Field2 }
27. ClassDecl2 → : ident
28. ClassDecl2 → eps
29. ClassDecl3 → , ident ClassDecl3
30. ClassDecl3 → ident ClassDecl3
31. ClassDecl3 → eps
32. Field2 → Field Field2
33. Field2 → eps
34. Field → VariableDecl
35. Field → FuncionDecl
36. Field → ConstDecl
37. InterfaceDecl → interface ident { Prototype2 }
38. Prototype2 → Prototype Prototype2
39. Prototype2 → eps
40. Prototype → Type ident (Formals) ;

Walter Orozco
Ricardo Chian

41. Prototype → void ident (Formals) ;
42. StmtBlock → { VariableDecl2 ConstDecl2 Stmt2 }
43. VariableDecl2 → VariableDecl VariableDecl2
44. VariableDecl2 → eps
45. ConstDecl2 → ConstDecl ConstDecl2
46. ConstDecl2 → eps
47. Stmt2 → Stmt Stmt2
48. Stmt2 → eps
49. Stmt → Expr1 ;
50. Stmt → IfStmt
51. Stmt → WhileStmt
52. Stmt → ForStmt
53. Stmt → BreakStmt
54. Stmt → ReturnStmt
55. Stmt → PrintStmt
56. Stmt → StmtBlock
57. Stmt → CallStmt
58. CallStmt → ident (Actuals)
59. CallStmt → iden . ident (Actuals)
60. Actuals → Expr , Actuals
61. Actuals → Expr
62. Expr1 → Expr
63. Expr1 → eps
64. IfStmt → if (Expr) Stmt IfStmt2
65. IfStmt2 → else Stmt
66. IfStmt2 → eps
67. WhileStmt → while (Expr) Stmt
68. ForStmt → for (Expr ; Expr ; Expr) Stmt
69. ReturnStmt → return Expr ;
70. BreakStmt → break ;
71. PrintStmt → Console.Writeline(Expr Expr2) ;
72. Expr2 → , Expr Expr2
73. Expr2 → eps
74. Expr → Lvalue = Expr
75. Expr → Constant
76. Expr → LValue
77. Expr → this
78. Expr → (Expr)
79. Expr → Expr + Expr
80. Expr → Expr * Expr
81. Expr → Expr % Expr
82. Expr → - Expr
83. Expr → Expr < Expr

Walter Orozco
Ricardo Chian

84. Expr → Expr > Expr
85. Expr → Expr <= Expr
86. Expr → Expr >= Expr
87. Expr → Expr == Expr
88. Expr → Expr && Expr
89. Expr → ! Expr
90. Expr → New (ident)
91. LValue → ident
92. LValue → Expr . ident
93. Constant → intConstant
94. Constant → doubleConstant
95. Constant → boolConstant
96. Constant → stringConstant
97. Constant → null

Walter Orozco
Ricardo Chian

- Tabla de Reglas Semánticas:

No.	Producción	Regla Semántica
0	Program’ → Program	Program’.val = Program.val
1	Program → Decl Decl2	Program.val = Decl.val || Decl2.val
2	Decl2 → Decl Decl21	Decl2.val = Decl.val || Decl21.val
3	Decl2 → eps	Decl2.val = null
4	Decl → VariableDecl	Decl.val = VariableDecl.val
5	Decl → FunctionDecl	Decl.val = FunctionDecl.val
6	Decl → ConstDecl	Decl.val = ConstDecl.val
7	Decl → ClassDecl	Decl.val = ClassDecl.val
8	Decl → InterfaceDecl	Decl.val = InterfaceDecl.val
9	VariableDecl → Variable ;	VariableDecl.val = Variable.val || “;”

10	Variable → Type ident	Variable.val = Type.val || ident.val
		Ident.type = Type.type

11	ConstDecl → const ConstType ident ;	ConstDecl.val = “const” || ConstType.val || ident.val || “;”
		Ident.type =ConstType.type

12	ConstType → int	ConstType.type = int
		ConstType.val = int

13	ConstType → double	ConstType.type = double
		ConstType.val = double

14	ConstType → bool	ConstType.type = bool
		ConstType.val = bool

15	ConstType → string	ConstType.type = string
		ConstType.val = string

16	Type → int	Type.type = int
		Type.val = int

17	Type → double	Type.type = double
		Type.val = double

18	Type → bool	Type.type = bool
		Type.val = bool

19	Type → string	Type.type = string
		Type.val = string

20	Type → ident	Type.type = ident.type
		Type.val = ident.val

21	Type → Type2 [ ]	Type.type = Type2.type
		Type.val = Type2.val || “[ ]”

22	FunctionDecl → Type ident (Formals) StmtBlock	FunctionDecl.val = Type.val || ident.val || “(” || Formals.val || “)” || StmtBlock.val
23	FunctionDecl → void ident (Formals) StmtBlock	FunctionDecl.val = “void” || ident.val || “(” || Formals.val || “)” || StmtBlock.val
24	Formals → Variable , Formals2	Formals.val = Variable.val || “,” || Formals2.val
25	Formals → Variable	Formals.val = Variable.val
26	ClassDecl → class ident ClassDecl2 ClassDecl3 { Field2 }	ClassDecl.val = “class” || ident.val || ClassDecl2.val || ClassDecl3.val || “{” || Field2.val || “}”
27	ClassDecl2 → : ident	ClassDecl2.val = “:” || ident.val
28	ClassDecl2 → eps	ClassDecl2.val = null
29	ClassDecl3 → , ident ClassDecl3	ClassDecl3.val = “,” || ident.val || ClassDecl3.val
30	ClassDecl3 → ident ClassDecl3	ClassDecl3.val = ident.val || ClassDecl3.val

Walter Orozco
Ricardo Chian

31	ClassDecl3 → eps	ClassDecl3.val = null
32	Field2 → Field Field21	Field2.val = Field.val || Field21.val
33	Field2 → eps	Field2.val = null
34	Field → VariableDecl	Field.val = VariableDecl.val
35	Field → FuncionDecl	Field.val = FuncionDecl.val
36	Field → ConstDecl	Field.val = ConstDecl.val
37	InterfaceDecl → interface ident { Prototype2 }	InterfaceDecl.val = "interface" || ident.val || "{" || Prototype2.val || "}"
38	Prototype2 → Prototype Prototype21	Prototype2.val = Prototype.val || Prototype21.val
39	Prototype2 → eps	Prototype2.val = null
40	Prototype → Type ident (Formals) ;	Prototype.val = Type.val || ident.val || "(" || Formals.val || ") ;"
41	Prototype → void ident (Formals) ;	Prototype.val = "void" || ident.val || "(" || Formals.val || ") ;"
42	StmtBlock → { VariableDecl2 ConstDecl2 Stmt2 }	StmtBlock.val = "{" || VariableDecl2.val || ConstDecl2.val || Stmt2.val || "}"
43	VariableDecl2 → VariableDecl VariableDecl21	VariableDecl2.val = VariableDecl.val VariableDecl21.val
44	VariableDecl2 → eps	VariableDecl2.val = null
45	ConstDecl2 → ConstDecl ConstDecl21	ConstDecl2.val = ConstDecl.val ConstDecl21.val
46	ConstDecl2 → eps	ConstDecl2.val = null
47	Stmt2 → Stmt Stmt21	Stmt2.val = Stmt.val || Stmt21.val
48	Stmt2 → eps	Stmt2.val = null
49	Stmt → Expr1 ;	Stmt.val = Expr1.val || ";"
50	Stmt → IfStmt	Stmt.val = IfStmt.val
51	Stmt → WhileStmt	Stmt.val = WhileStmt.val
52	Stmt → ForStmt	Stmt.val = ForStmt.val
53	Stmt → BreakStmt	Stmt.val = BreakStmt.val
54	Stmt → ReturnStmt	Stmt.val = ReturnStmt.val
55	Stmt → PrintStmt	Stmt.val = PrintStmt.val
56	Stmt → StmtBlock	Stmt.val = StmtBlock.val
57	Stmt → CallStmt	Stmt.val = CallStmt.val
58	CallStmt → ident (Actuals)	CallStmt.val = ident.val || "(" || Actuals.val || ")"
59	CallStmt → iden . ident2 (Actuals)	CallStmt.val = ident.val || "." || ident2.val || "(" || Actuals.val || ")"
60	Actuals → Expr , Actuals	Actuals.val = Expr.val || "," || Actuals.val
61	Actuals → Expr	Actuals.val = Expr.val

62	Expr1 → Expr	Expr1.val = Expr.val
		Expr1.type = Expr.type

63	Expr1 → eps	Expr1.val = null
		Expr1.type = null

64	IfStmt → if (Expr) Stmt IfStmt2	
65	IfStmt2 → else Stmt	IfStmt2.val = "else" || Stmt.val

66	IfStmt2 → eps	IfStmt2.val = null
		IfStmt2.type = null

67	WhileStmt → while (Expr) Stmt	
68	ForStmt → for (Expr ; Expr ; Expr) Stmt	
69	ReturnStmt → return Expr ;	ReturnStmt.val = "return" || Expr.val || ";"

Walter Orozco
Ricardo Chian

70	BreakStmt → break ;	BreakStmt.val = “break ;”
71	PrintStmt → Console.Writeline(Expr Expr2)	PrintStmt.val = Expr.val || Expr2.val
72	Expr2 → , Expr3 Expr21	Expr2.val = “,” || Expr3.val || Expr21.val

73	Expr2 → eps	Expr2.val = null
		Expr2.type = null

74	Expr → Lvalue = Expr2	Expr.val = LValue.val
		Expr.type = LValue.type
		LValue.val = Expr2.val
		LValue.type = Expr2.type

75	Expr → Constant	Expr.val = Constant.val
		Expr.type = Constant.type

76	Expr → LValue	Expr.val = LValue.val
		Expr.type = LValue.type

77	Expr → this	Expr.val = this.val Expr.type = this.type

78	Expr → (Expr2)	If(Expr2.val != error && Expr3.val != error)
		{Expr.val = Expr.val
		Expr.type = Expr.type
		}
		Else{ Expr.val = error }

79	Expr → Expr2 + Expr3	If(Expr2.val != error && Expr3.val != error)
		{if(Expr2.type != Expr3.type){Convert(Expr2,Expr3)
		Expr.val = Expr2.val + Expr3.val
		Expr.type = Expr2.type}
		Else{ Expr.val = Expr2.val + Expr3.val Expr.type = Expr2.type
		}
		}
		Else{ Expr.val = error }

80	Expr → Expr2 * Expr3	If(Expr2.val != error && Expr3.val != error)
		{if(Expr2.type != Expr3.type){Convert(Expr2,Expr3)
		Expr.val = Expr2.val * Expr3.val
		Expr.type = Expr2.type
		}
		Else{ Expr.val = Expr2.val * Expr3.val
		Expr.type = Expr2.type
		}
		}
		Else{ Expr.val = error }

81	Expr → Expr2 % Expr3	If(Expr2.val != error && Expr3.val != error)
		{if(Expr2.type != Expr3.type){Convert(Expr2,Expr3)
		Expr.val = Expr2.val % Expr3.val
		Expr.type = Expr2.type
		}
		Else{ Expr.val = Expr2.val % Expr3.val Expr.type = Expr2.type
		}

Walter Orozco
Ricardo Chian

		}
		Else{ Expr.val = error }

82	Expr → - Expr2	If(Expr2.val != error)
		{Exp.val = "-" || Expr2.val
		Expr.type = Expr2.type}
		Else{ Expr.val = error }

83	Expr → Expr2 < Expr3	If(Expr2.val != error && Expr3.val != error)
		{if(Expr2.val < Expr3.val){Expr.val = true}
		Else{ Expr.val = false}
		}
		Else{ Expr.val = error }

84	Expr → Expr2 > Expr3	If(Expr2.val != error && Expr3.val != error)
		{if(Expr2.val > Expr3.val){Expr.val = true}
		Else{ Expr.val = false}
		}
		Else{ Expr.val = error }

85	Expr → Expr2 <= Expr3	If(Expr2.val != error && Expr3.val != error)
		{if(Expr2.val <= Expr3.val){Expr.val = true}
		Else{ Expr.val = false}
		}
		Else{ Expr.val = error }

86	Expr → Expr2 >= Expr3	If(Expr2.val != error && Expr3.val != error)
		{if(Expr2.val >= Expr3.val){Expr.val = true}
		Else{ Expr.val = false}
		}
		Else{ Expr.val = error }

87	Expr → Expr2 == Expr3	If(Expr2.val != error && Expr3.val != error)
		{if(Expr2.val == Expr3.val){Expr.val = true}
		Else{ Expr.val = false}
		}
		Else{ Expr.val = error }

88	Expr → Expr2 && Expr3	If(Expr2.val != error && Expr3.val != error)
		{if(Expr2.val == true && Expr3.val == true){Expr.val = true}
		Else{ Expr.val = false}
		}
		Else{ Expr.val = error }

89	Expr → ! Expr2	If(Expr2.type == bool){}
		Else{ error}

90	Expr → New (ident)	Expr.val = "New (" || ident.val || ")"

91	LValue → ident	LValue.val = ident.val
		LValue.type = ident.type

92	LValue → Expr . ident	LValue.val = Expr.val || "." || ident.val
93	Constant → intConstant	Constant.value = intConstant

		Constant.type = int

94	Constant → doubleConstant	Constant.value = doubleConstant
		Constant.type = double

95	Constant → boolConstant	Constant.value = boolConstant
		Constant.type = bool

96	Constant → stringConstant	Constant.value = stringConstant
		Constant.type = string

97	Constant → null	Constant.value = null
		Constant.type = null

- Variables:
    - Type : heredado
    - Value : sintetizado
- Procesos:
    - Convertir