GRAMMAR RULES AND SEMANTIC RULES

GROUP 13

ASHWIN KIRAN GODBOLE 2018B5A70423P

SAMARTH KRISHNA MURTHY 2018B2A70362P

GRAMMAR RULES

1. program -> otherFunctions mainFunction
2. mainFunction -> TK_MAIN stmts TK_END
3. otherFunctions -> function otherFunctions
4. otherFunctions -> eps
5. function -> TK_FUNID input_par output_par TK_SEM stmts TK_END
6. input_par -> TK_INPUT TK_PARAMETER TK_LIST TK_SQL parameter_list TK_SQR
7. output_par -> TK_OUTPUT TK_PARAMETER TK_LIST TK_SQL parameter_list TK_SQR
8. output_par -> eps
9. parameter_list -> dataType TK_ID remaining_list
10. dataType -> primitiveDatatype
11. dataType -> constructedDatatype
12. primitiveDatatype -> TK_INT
13. primitiveDatatype -> TK_REAL
14. constructedDatatype -> TK_RECORD TK_RUID
15. constructedDatatype -> TK_UNION TK_RUID
16. constructedDatatype -> TK_RUID
17. remaining_list -> TK_COMMA parameter_list
18. remaining_list -> eps
19. stmts -> typeDefinitions declarations otherStmts returnStmt
20. typeDefinitions -> actualOrRedefined typeDefinitions
21. typeDefinitions -> eps
22. actualOrRedefined -> typeDefinition
23. actualOrRedefined -> definetypestmt
24. typeDefinition -> TK_RECORD TK_RUID fieldDefinitions TK_ENDRECORD
25. typeDefinition -> TK_UNION TK_RUID fieldDefinitions TK_ENDUNION
26. fieldDefinitions -> fieldDefinition fieldDefinition moreFields
27. fieldDefinition -> TK_TYPE fieldtype TK_COLON TK_FIELDID TK_SEM
28. fieldtype -> primitiveDatatype
29. fieldtype -> TK_RUID
30. moreFields -> fieldDefinition moreFields
31. moreFields -> eps
32. declarations -> declaration declarations

33. declarations -> eps
34. declaration -> TK_TYPE dataType TK_COLON TK_ID global_or_not TK_SEM
35. global_or_not -> TK_COLON TK_GLOBAL
36. global_or_not -> eps
37. otherStmts -> stmt otherStmts
38. otherStmts -> eps
39. stmt -> assignmentStmt
40. stmt -> iterativeStmt
41. stmt -> conditionalStmt
42. stmt -> ioStmt
43. stmt -> funCallStmt
44. assignmentStmt -> singleOrRecId TK_ASSIGNOP arithmeticExpression TK_SEM
45. singleOrRecId -> TK_ID constructedVariable
46. constructedVariable -> oneExpansion moreExpansions
47. constructedVariable -> eps
48. oneExpansion -> TK_DOT TK_FIELDID
49. moreExpansions -> oneExpansion moreExpansions
50. moreExpansions -> eps
51. funCallStmt -> outputParameters TK_CALL TK_FUNID TK_WITH TK_PARAMETERS inputParameters TK_SEM
52. outputParameters -> TK_SQL idList TK_SQR TK_ASSIGNOP
53. outputParameters -> eps
54. inputParameters -> TK_SQL idList TK_SQR
55. iterativeStmt -> TK_WHILE TK_OP booleanExpression TK_CL stmt otherStmts TK_ENDWHILE
56. conditionalStmt -> TK_IF TK_OP booleanExpression TK_CL TK_THEN stmt otherStmts elsePart
57. elsePart -> TK_ELSE stmt otherStmts TK_ENDIF
58. elsePart -> TK_ENDIF
59. ioStmt -> TK_READ TK_OP var TK_CL TK_SEM
60. ioStmt -> TK_WRITE TK_OP var TK_CL TK_SEM
61. arithmeticExpression -> term expPrime
62. expPrime -> lowPrecedenceOperators term expPrime
63. expPrime -> eps
64. term -> factor termPrime
65. termPrime -> highPrecedenceOperator factor termPrime
66. termPrime -> eps
67. factor -> TK_OP arithmeticExpression TK_CL
68. factor -> var
69. highPrecedenceOperator -> TK_MUL
70. highPrecedenceOperator -> TK_DIV
71. lowPrecedenceOperators -> TK_PLUS
72. lowPrecedenceOperators -> TK_MINUS

73. booleanExpression -> TK_OP booleanExpression TK_CL logicalOp TK_OP booleanExpression TK_CL
74. booleanExpression -> var relationalOp var
75. booleanExpression -> TK_NOT TK_OP booleanExpression TK_CL
76. var -> singleOrRecId
77. var -> TK_NUM
78. var -> TK_RNUM
79. logicalOp -> TK_AND
80. logicalOp -> TK_OR
81. relationalOp -> TK_LT
82. relationalOp -> TK_LE
83. relationalOp -> TK_EQ
84. relationalOp -> TK_GT
85. relationalOp -> TK_GE
86. relationalOp -> TK_NE
87. returnStmt -> TK_RETURN optionalReturn TK_SEM
88. optionalReturn -> TK_SQL idList TK_SQR
89. optionalReturn -> eps
90. idList -> TK_ID more_ids
91. more_ids -> TK_COMMA idList
92. more_ids -> eps
93. definetypestmt -> TK_DEFINETYPE A TK_RUID TK_AS TK_RUID
94. A -> TK_RECORD
95. A -> TK_UNION

SEMANTIC RULES

1. program.node = mkNode(program, otherFunctions.node, mainFunction.node)
2. mainFunction.node = stmts.node
3. otherFunctions.node = mkNode(otherFunctions, function.node, otherFunctions1.node)
4. function.name = TK_FUNID.value
5. function.node = mkNode(function, input_par.node, output_par.node, stmts.node)
6. input_par.node = parameter_list.node
7. output_par.node = parameter_list.node
8. id.node = mkLeaf(TK_ID, TK_ID.entry)
9. addType(TK_ID.entry, dataType.type)
10. parameter_list.node = mkNode(parameter_list, id.node, remaining_list.node)
11. dataType.type = primitiveDatatype.type
12. dataType.type = constructedDatatype.type
13. primitiveDatatype.type = integer
14. primitiveDatatype.type = real
15. constructedDatatype.ctype = record

16. constructedDatatype.name = TK_RUID.value
17. constructedDatatype.ctype = record
18. constructedDatatype.name = TK_RUID.value
19. constructedDatatype.name = TK_RUID.value
20. remaining_list.node = parameter_list.node
21. stmts.node = mkNode(stmts, typeDefinitions.node, declarations.node, otherStmts.node, returnStmt.node)
22. typeDefinitions.node = mkNode(typeDefinitions, actualOrRedefined.node, typeDefinitions.node)
23. actualOrRedefined.node = typeDefinition.node
24. actualOrRedefined.node = definetypestmt.node
25. ruid.node = mkLeaf(TK_RUID, TK_RUID.value)
26. ruid.name = TK_RUID.value
27. ruid.ctype = record
28. typeDefinition.node = mkNode(typeDefinition, ruid.node, fieldDefinitions.node)
29. ruid.node = mkLeaf(TK_RUID, TK_RUID.value)
30. ruid.name = TK_RUID.value
31. ruid.ctype = union
32. typeDefinition.node = mkNode(typeDefinition, ruid.node, fieldDefinitions.node)
33. fieldDefinitions.node = mkNode(fieldDefinitions, fieldDefinition1.node, fieldDefinition2.node, moreFields.node)
34. fid.node = mkLeaf(TK_FIELDID, TK_FIELDID.entry)
35. addType(TK_FIELDID.entry, fieldtype.type)
36. fieldDefinition.node = fid.node
37. fieldtype.type = primitiveDatatype.type
38. fieldtype.type = TK_RUID.name
39. moreFields.node = mkNode(morefields, fieldDefinition.node, moreFields1.node)
40. declarations.node = mkNode(declarations, declaration.node, declarations1.node)
41. id.node = mkLeaf(TK_ID, TK_ID.entry)
42. addType(TK_ID.entry, dataType.type)
43. addGlobalStatus(TK_ID.entry, global_or_not.isGlobal)
44. declaration.node = id.node
45. global_or_not.isGlobal = true
46. global_or_not.isGlobal = false
47. otherStmts.node = mkNode(otherStmts, stmt.node, otherStmts1.node)
48. stmt.node = assignmentStmt.node
49. stmt.node = iterativeStmt.node
50. stmt.node = conditionalStmt.node
51. stmt.node = ioStmt.node
52. stmt.node = funCallStmt.node
53. assignmentStmt.node = mkNode(assignmentStmt, singleOrRecId.node, arithmeticExpression.node)
54. id.node = mkLeaf(TK_ID, TK_ID.entry)

55. singleOrRecId.node = mkNode(singleOrRecId, id.node, constructedVariable.node)
56. constructedVariable.node = mkNode(constructedVariable, oneExpansion.node, moreExpansions.node)
57. fid.node = mkLeaf(TK_FIELDID, TK_FIELDID.entry)
58. oneExpansion.node = fid.node
59. moreExpansions.node = mkNode(moreExpansions, oneExpansion.node, moreExpansions1.node)
60. funid.node = mkLeaf(TK_FUNID, TK_FUNID.value)
61. funCallStmt.node = mkNode(funCallStmt, outputParameters.node, funid.node, inputParameters.node)
62. outputParameters.node = idList.node
63. inputParameters.node = idList.node
64. iterativeStmt.node = mkNode(iterativeStmt, booleanExpression.node, stmt.node, otherStmts.node)
65. conditionalStmt.node = mkNode(conditionalStmt, booleanExpression.node, stmt.node, otherStmts.node, elsePart.node)
66. elsePart.node = mkNode(elsePart, stmt.node, otherStmts.node)
67. ioStmt.node = var.node
68. ioStmt.iop = read
69. ioStmt.node = var.node
70. ioStmt.iop = write
71. arithmeticExpression.node = mkNode(arithmeticExpression, term.node, expPrime.node)
72. expPrime.node = mkNode(expPrime, lowPrecedenceOperators.node, term.node, expPrime.node)
73. term.node = mkNode(term, factor.node, termPrime.node)
74. termPrime.node = mkNode(termPrime, highPrecedenceOperator.node, factor.node, termPrime.node)
75. factor.node = arithmeticExpression.node
76. factor.node = var.node
77. highPrecedenceOperator.op = MUL
78. highPrecedenceOperator.op = DIV
79. lowPrecedenceOperators.op = PLUS
80. lowPrecedenceOperators.op = MINUS
81. booleanExpression.op = logicalOp.op
82. booleanExpression.node = mkNode(booleanExpression, booleanExpression1.node, booleanExpression2.node)
83. booleanExpression.op = relationalOp.op
84. booleanExpression.node = mkNode(booleanExpression, var1.node, var2.node)
85. booleanExpression.op = NOT
86. booleanExpression.node = mkNode(booleanExpression, booleanExpression1.node)
87. var.node = singleOrRecId.node

88. var.type = integer
89. var.node = mkLeaf(NUM, num.value)
90. var.type = real
91. var.node = mkLeaf(RNUM,rnum.value)
92. logicalOp.op = AND
93. logicalOp.op = OR
94. relationalOp.op = LT
95. relationalOp.op = LE
96. relationalOp.op = EQ
97. relationalOp.op = GT
98. relationalOp.op = GE
99. relationalOp.op = NE
100. returnStmt.node = optionalReturn.node
101. optionalReturn.node = idList.node
102. id.node = mkLeaf(TK_ID, TK_ID.entry)
103. idList.node = mkNode(idList, id.node, more_ids.node)
104. more_ids.node = idList.node
105. ruid1.node = mkLeaf(TK_RUID1, TK_RUID1.value)
106. ruid2.node = mkLeaf(TK_RUID2, TK_RUID2.value)
107. definetypestmt.node = mkNode(definetypestmt, ruid1.node, ruid2.node)
108. A.ctype = record
109. A.ctype = union